PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Tesis de Doctorado
en Informática

Tree models : algorithms and information theoretic properties

Alvaro Martín

Orientador: Dr. Gadiel Seroussi

Supervisor: Dr. Alfredo Viola
Tribunal: Dr. Sergio Verdú y Dr. Frans Willems (Revisores)
Dr. Eduardo Canale, Dr. Alberto Pardo, y Dr. Gregory Randall

2009

Tree models : algorithms and information theoretic properties
Martín, Alvaro
ISSN 0797-6410
Tesis de Doctorado en Informática
Reporte Técnico RT 09-20
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República.
Montevideo, Uruguay, setiembre de 2009

A mi hija Camila,
y a mi esposa y compañera Mercedes,
que me ha apoyado con inagotable paciencia
durante estos años.

Contents

Agradecimientos

Mucha gente me ha acompañado durante mis estudios de doctorado. Siempre he encontrado el apoyo imprescindible para poder terminar esta tesis en mis dos amores, mi hija Camila y mi esposa Mercedes. A mis padres, Yvonne y Eduardo, les debo el gusto por la ciencia que siempre me han transmitido. Nada de esto hubiera comenzado de no ser por ellos. Mi hermana Ana, así como Teresa, Omar, Silvana, la Bisa, y los Raffo también han estado muy presentes durante todo este tiempo.

Quiero agradecer muy especialmente a Gadiel Seroussi y a Marcelo Weinberger, quienes me motivaron para embarcarme en este proyecto. Desde mi primera visita a los Laboratorios Hewlett-Packard ellos han representado una gran fuente de conocimientos y de inspiración. Sus consejos y nuestras discusiones han ido muchas veces más allá de lo estrictamente técnico o científico. Sin lugar a dudas les debo a ellos mucho de lo que he aprendido en estos últimos años.

Mi supervisor, Tuba, también ha sido de invalorable ayuda para mi. Mi más sincero agradecimiento por su fuerte apoyo y todos sus consejos.

Recuerdo mi visita a Palo Alto, California, como una experiencia realmente extraordinaria. Muchas gracias a Erik Ordentlich, Tsachy Weissman, Vinay Deolalikar, Ronny Roth, Neri Merhav, Giovanni Motta, y Krishnamurthy Viswanathan. También a Popi, Mathieu, Waled, y Quan.

También quiero agradecer a Guillermo Sapiro, Alberto Bartesaghi, y Facundo Memoli por haberme recibido maravillosamente en Minneapolis.

Siempre he sentido el apoyo de muchas personas del Instituto de Computación y del Área informática del Pedeciba, así como de Gregory Randall, Nacho, Fefo, y Marcelo. Muchas gracias a todos.

Finalmente, agradezco a muchos amigos y familiares, a quienes tantas veces dediqué tanto menos tiempo del que hubiese querido.

Álvaro Martín
Montevideo
Julio de 2009

Acknowledgments

Many people have gone along with me during my studies. I have always found the indispensable support to pursue this thesis in my two loves, my daughter Camila and my wife Mercedes. I owe to my parents, Yvonne and Eduardo, the like for science that they have always transmitted to me. None of this would have begun were it not for them. My sister Ana, as well as Teresa, Omar, Silvana, la Bisa, and los Raffo have also been there for me all this time.

Very special thanks go to Gadiel Seroussi and Marcelo Weinberger, who encouraged me to engage in this project. Since my first visit to the Hewlett-Packard Laboratories they have been a rich source of knowledge and inspiration. Their guidance and our discussions have gone many times beyond scientific or technical advice. I definitely owe to them much of what I have learned during the past years.

My supervisor, Tuba, has also been of invaluable help for me. My most sincere thanks for his strong support and his advise.

I remember the time visiting HP Labs in Palo Alto as an extraordinary experience. Thanks to Erik Ordentlich, Tsachy Weissman, Vinay Deolalikar, Ronny Roth, Neri Merhav, Giovanni Motta, and Krishnamurthy Viswanathan. Also to Popi, Mathieu, Waled, and Quan.

I would also like to thank Guillermo Sapiro, Alberto Bartesaghi, and Facundo Memoli, for the great time in Minneapolis.

I have always felt the support of many people from the Instituto de Computación and the Informatics area of Pedeciba, as well as Gregory Randall, Nacho, Fefo, and Marcelo. Thanks to all.

Finally, I thank many friends and family, with whom so many times I have spent much less time than I would have wished to.

Álvaro Martín
Montevideo
July, 2009

Summary of notations and abbreviations

General

$\mathbb{Z}, \mathbb{Z}_{\geq 0}, \mathbb{Z}_{>0}$	The integers, the nonnegative integers, and the positive integers, respectively, 15
$\mathbb{R}$	The real numbers, 15
$f(n) = o(g(n))$	$\lvert f(n)\rvert \leq \epsilon \lvert g(n)\rvert$ for all $\epsilon > 0$ and $n > n_0(\epsilon)$, 15
$f(n) = O(g(n))$	$\lvert f(n)\rvert \leq K \lvert g(n)\rvert$ for some $K > 0$ and $n > n_0$, 15
$f(n) = \Omega(g(n))$	$g(n) = O(f(n))$, 15
$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$ and $g(n) = O(f(n))$, 15
$H(X)$	The entropy of a random variable X, 15
$H(p)$	The entropy of a probability vector, or a probability distribution, p, 15
$h(\cdot)$	The binary entropy function, 15
$H(X\|Y)$	The conditional entropy of X given Y, 15
$I(X;Y)$	The mutual information between X and Y, 16
$D(P\|\|Q)$	The divergence between P and Q, 16
$\delta_{u,v}$	A function valued one if $u = v$ and zero otherwise, 52
$\mathbf{1}_{i,j}$	A matrix valued 1 in entry i, j, and zero everywhere else, 62
M_{*i}	Sum over the i-th column of the matrix M, 52
M_{i*}	Sum over the i-th row of the matrix M, 52

Strings

α	Alphabet size, 17

$\mathcal{A}$	Alphabet of symbols, 17
$\mathcal{A}^*, \mathcal{A}^+, \mathcal{A}^m$	Set of finite strings, positive-length strings, and length-m strings over $\mathcal{A}$, respectively, 17
\$	A distinguished symbol such that $\$ \notin \mathcal{A}$, 22
λ	empty string, 17
x_j^k	The substring $x_j x_{j+1} \dots x_k$, 17
x^k	The substring $x_1 x_2 \dots x_k$, 17
$\overline{x}$	The reverse string of x, 17
$\|x\|$	The length of x, 17
head(x)	The first symbol of x, 17
tail(x)	The substring $x_2 \dots x_{\|x\|}$, 17
$\preceq, \prec$	The prefix and proper prefix relation, 17

Context (and generalized context) trees

$u \in T$	u is a node of T, 21
$\textsc{word}(T)$	The set of words of T, 21
$\mathcal{I}(T)$	The set of internal nodes of T, 51
$\mathsf{depth}(T)$	The depth of T (valued zero for a single node tree), 51
$u \xrightarrow{w} v$	An edge labeled w going from node u to node v, 21
$\deg(u)$	The number of outgoing edges of a node u, 21
$\textsc{chld}_T(u)$	The set of children of a node u, 21
$\textsc{par}_T(u)$	The parent of a node u, 21
$G_T = (V_T, E_T)$	The state transition support graph of T, 71
T_{full}	The completion of a GCT T to a full tree, 23
T_{suf}	The FSM closure of T obtained by adding, as nodes, all the suffixes of nodes of T, 29
T_{c}	The minimal canonical extension of T, 71
T_N	The normalized presentation of T, 25
$\mathbf{N}(T)$	The set of GCTs with the same normalized presentation as T, 25

κ_T	The total (FSM) over-refinement of T, 95
κ_t	The (FSM) over-refinement of $t \in \mathcal{I}(T_{\mathrm{suf}}) \backslash \mathcal{I}(T_{\mathrm{c}})$, 95
$C_T(x)$	The canonical decomposition of x with respect to T, 21
$V_T(x)$	The first component of $C_T(x) = \langle r, u, v \rangle$, 22
S_T	The set of permanent states of T, 23
$S_T^{\$}$	The set of transient states of T, 23
S_T^{A}	The set of all states of T, 23
$\sigma_T(x)$	The tree-state function, 22
$s_0 \cdots s_n$	State sequence of a given string, 18
$n_s^{(a)}(x)$	The number of occurrences of symbol a in context $\overline{s}$ in x, 37
$n_s(x)$	The number of times a symbol of x occurs in context $\overline{s}$, 37
$N(x)$	State transition matrix of x^n, 51
$N_{\mathrm{c}}(x^n)$	State transition matrix of x^n with respect to T_{c}, 89

Probability assignments and coding

$\langle T, p \rangle$	A (generalized) context tree model with (generalized) context tree T and parameter p, 23
$\mathrm{P}_{\langle T, p_T \rangle}(x^n)$	Probability assigned to x^n by the model $\langle T, p_T \rangle$, 51
$\mathrm{P}_{\langle T, p_T \rangle}\{\cdot\}$	Probability under $\mathrm{P}_{\langle T, p_T \rangle}(\cdot)$ of an event that depends on a random sequence $X^n \in \mathcal{A}^n$, 51
$\hat{\mathrm{P}}_T(x^n)$	The maximum likelihood probability of x^n under T, 99
$\mathrm{E}_{\langle T, p_T \rangle}[\,\cdot\,]$	Expectation of a random variable with respect to $\mathrm{P}_{\langle T, p_T \rangle}(\cdot)$, 51
$\mathcal{H}$	The entropy rate of a source, 2
$\hat{\mathcal{H}}(x)$	The empirical entropy rate of x with respect to a given model structure, 6
B_P	The strings in the set B with positive P-probability, 18
$\mathcal{L}(T, x^n)$	The code length assigned to x^n under context tree T by a given code, 37

Type classes and universal type classes

$\mathcal{T}(T, x^n)$	The type class of x^n with respect to T, 51
$\mathcal{T}^*(T, x^n)$	The close-ended type class of x^n with respect to T, 53

T-class	A type class with respect to T, 51
T-class*	A close-ended type class with respect to T, 53
$\mathcal{N}_T$	The number of T-classes*, 86
$\mathcal{U}(x^n)$	The universal tree type class of x^n, 119
$\mathcal{M}(x^n)$	The universal Markov type class of x^n, 119

Context tree estimation

$K(T, x^n)$	Cost function to be minimized, 112
K_T	Penalization function increasing with $\|S_T\|$, 112
$f(n)$	A function increasing with n in the definition of $K(T, x^n)$, 112
$\hat{T}(x^n)$	Context tree estimate for x^n, 112

Specific notation for Chapter 3 and Appendix C

$\kappa(x^n, s)$	The code length assigned by the KT probability assignment to the symbols of x^n that occur in s, 38
$n(T)$	The number of times a permanent state of T is selected, 38
$\mathcal{L}_T^{\mathsf{KT}}(x^n)$	The code length assigned by the KT probability assignment conditioned on the states of T, 37
$\mathcal{C}(T)$	The code length of the natural code for T, 37
$\mathrm{ST}(x)$	The compact suffix tree of x, 40
$T(x^n)$	The context tree that minimizes $\mathcal{L}(T, x^n)$, 37
$T_F(x^n)$	An FSM closure of $T(x^n)$, 44
$T'(x^n)$	A GCT that is equivalent to $T(x^n)$ for the purpose of encoding x^n, 40
$\hat{T}'(x^n)$	The GCT obtained from $T(x^n)$ by deleting all the leaves, as well as nodes whose outgoing degree after deleting the leaves is one, 44
$\hat{T}'_F(x^n)$	An FSM closure of $\hat{T}'(x^n)$, 44
s_i	The state selected by x^i in $T_F(x^n)$, 44
$\hat{s}_i$	The state selected by x^i in $\hat{T}'_F(x^n)$, 44
z_i	The string such that $\hat{s}_i z_i$ is the longest prefix of $\overline{x^i}$ in $\textsc{word}(\hat{T}'_F(x))$, 45
b_i	The symbol that immediately follows $\hat{s}_i z_i$ in $\overline{x^i}$, 45

u_{i+1}	The string such that $x_{i+1}\hat{s}_i = \hat{s}_{i+1}u_{i+1}$, 46
$\tilde{T}'_{F_i}(x^n)$	The FSM GCT constructed adding $s_0 \cdots s_i$ to $\hat{T}'_F(x^n)$ and all necessary suffixes, 141
$\tilde{s}_i$	The state selected by x^i in $\tilde{T}'_{F_{i-1}}(x^n)$, 141
$\tilde{z}_i$	The string such that $\tilde{s}_i\tilde{z}_i$ is the longest prefix of $\overline{x^i}$ that is a word of $\tilde{T}'_{F_{i-1}}(x^n)$, 141
$\tilde{b}_i$	The symbol that immediately follows $\tilde{s}_i\tilde{z}_i$ in $\overline{x^i}$ and $\tilde{s}_i\tilde{z}_i x_{i-\lvert\tilde{s}_i\tilde{z}_i\rvert} \in T_F(x^n)$ (or λ), 141
$\tilde{u}_{i+1}$	The string such that $x_{i+1}\tilde{s}_i = \tilde{s}_{i+1}\tilde{u}_{i+1}$ (or λ), 143
$\{T_i\}$	The sequence of trees obtained by inserting $\overline{x^i}$ in T_{i-1} starting from $T_0 = T(x^n)$, 146
suf'_i	The longest prefix of $\overline{x^i}$ that is a word of T_{i-1}, 146
suf_i	The longest prefix of $\overline{x^i}$ that is a substring of $\overline{x^{i-1}}$, 146

Specific notation for Chapter 4 and Appendix E

U	Set of pseudo-states of a context tree, 58
ℓ_s	Length of the forced state sequence of s, 57
$\mu_1(s)\cdots\mu_{\ell_s}(s)$	Forced pseudo-state sequence of s, 57
$\nu_1(s)\cdots\nu_{\ell_s}(s)$	Forced state sequence of s, 57
$\rho(u)$	Parent of u, 58
$\Lambda(u)$	Descendants of u, 58
$\bar{\Lambda}(u)$	Proper descendants of u, 58
$J(x^n)$	Forced sequence parsing of x^n, 62
$t_0 \cdots t_r$	The ordered members of $J(x^n)$, 62
$\vec{j}$	Position of the last state that forces the transition $s_j \to s_{j+1}$, 62
j_Δ	Position of s_j in the forced state sequence $\nu_1(s_{\vec{j}})\cdots\nu_{\ell_{s_{\vec{j}}}}(s_{\vec{j}})$, 62
$\tau(u,a)$	The longest prefix of au in U, 62
τ_i	Shorthand for $\tau(s_{t_i}, x_{t_i+1})$, 64
$d(u,av)$	A matrix that redirects an edge (u,av) to $(u,\tau(u,a))$, 62

$\tilde{\Delta}^{+}(s)$	Forced pseudo-state transition matrix of $s \in S_T$, 60
$\tilde{\Delta}^{-}(s)$	Forced state transition matrix of $s \in S_T$, 60
$\tilde{\Theta}(s)$	Forced transition substitution matrix of $s \in S_T$, 60
$\Theta(s)$	Forced transition net substitution matrix of $s \in S_T$, 62
$K(x^n)$	Auxiliary matrix in the construction of $F(x^n)$, see Equation (4.10), 62
$G_K(x^n)$	A graph with incidence matrix $K(x^n)$, 60
$D(x^n)$	Auxiliary matrix in the construction of $F(x^n)$, see Equation (4.11), 63
$G_D(x^n)$	A graph with incidence matrix $D(x^n)$, 60
$B(x^n)$	Auxiliary matrix with context-dropping transitions for the construction of $F(x^n)$, see Equation (4.12), 63
$G_B(x^n)$	A graph with incidence matrix $B(x^n)$, 60
$F(x^n)$	Pseudo-state transition matrix of x^n , see Equation (4.13), 63
$\hat{F}(x^n)$	Normalized pseudo-state transition matrix of x^n, 67
$G_N(x^n)$	The state transition graph of x^n, 60
$G_F(x^n)$	The pseudo-state transition graph of x^n, 60
ω	Tagging function for the edges of a graph, 64
β_i	A sequence of context-dropping transitions from τ_i to the parent of $\mu_1(s_{t_{i+1}})$, 67
ς_i	A context-dropping sequence β_{i-1} followed by the forced pseudo-state sequence of s_{t_i}, 67
$G_T^{(\mathrm{F})} = (V_T^{(\mathrm{F})}, E_T^{(\mathrm{F})})$	The pseudo-state transition support graph of T, 82
$\Xi_T(x^n)$	Multinomial factor in the formula (4.15) for the size of $\mathcal{T}^*(T, x^n)$, 70
$\Xi_T^{\alpha}(x^n)$	Multinomial factor for symbol occurrence counts $\prod_s \binom{n_s}{n_s^{(a)} \cdots n_s^{(z)}}$ where $\mathcal{A} = \{a \cdots z\}$, 70
$B'_{\mu_1 b, \rho}$	A shorthand for $B'_{\mu_1(w)b, \rho'(\mu_1(w)b)}$, 74
$\Pi^{(\ell_w=1)}$	A factor in the formula of Lemma 4.22 , see Equation (4.27), 74
Π_{δ}	A factor in the formula of Lemma 4.22, 74
$\Pi^{(\ell_w>1)}$	A factor in the formula of Lemma 4.22 , see Equation (4.28), 74

M_b	A shorthand for $B'_{\mu_1 b,\rho} + N'_{*wb}$, 78
p_b	A shorthand for N'_{*wb}/M_b, ($p_b = 0$ if $M_b = 0$), 79
Q	A shorthand for $F_{\mu_1(w)*} + \delta_{s_n,\mu_1(w)}$, 78
q	A shorthand for $q = N_{*w}/Q$, ($q = 0$ if $Q = 0$), 79
$\mathcal{T}'(T, x^n)$	The type class of x^n with respect to T where the initial conditions are determined by transient states, 185
$\mathcal{T}'^*(T, x^n)$	The close-ended type class of x^n with respect to T where the initial conditions are determined by transient states, 186
$i_0(x^n)$	The last index in the state sequence of x^n where a transient state occurs, 185

Specific notation for Chapter 5

$n_{u,v}$	Number of times a transition from context $\overline{u}$ to context $\overline{v}$ occurs in x^n, 104
$x^0_{-\infty}$	A semi-infinite string that precedes x^n, 98
$\mathbf{i}(u)$	Indicator function valued one if x_1 occurs in context $\overline{u}$ and zero otherwise, 99
$\mathbf{f}(u)$	Indicator function valued one if $\overline{u}$ occurs at the end of x^n and zero otherwise, 99
δ	A generic constant in $\{-1, 0, 1\}$ to account for border adjustments, 99
$\mathcal{K}(T, x^n)$	The collection of counts $\{ n_s^{(a)} \}_{s \in S_T,\, a \in \mathcal{A}}$, 99
$\mathcal{K}_b(T, x^n)$	The collection of counts $\{n_s^{(a)}\}_{s \in S_T,\, a \in \mathcal{A} \setminus \{b\}}$, 100
$C_{w,a}\left(n_w^{(a)}\right)$	A code for a count $n_w^{(a)}$, 101
$\lfloor Z_{w,a} \rfloor^{\top}$	The unary part of the Golomb encoding of $\lfloor Z_{w,a} \rfloor$, 101
$\lfloor Z_{w,a} \rfloor^{\perp}$	The uniform part of the Golomb encoding of $\lfloor Z_{w,a} \rfloor$, 101
$z_{w,a}$	The estimation error of $n_w^{(a)}$, $n_w^{(a)} - \frac{n_s^{(a)}}{n_s} n_w$, 100
$Z_{w,a}$	The absolute value of the estimation error of $n_w^{(a)}$, $\lvert z_{w,a} \rvert$, 100
$\mathrm{sg}_{w,a}$	The sign of the estimation error of $n_w^{(a)}$, one if $z_{w,a} > 0$ and zero otherwise, 100
$C^{\star}_u(n_{\overline{u}})$	A code for a count $n_{\overline{u}}$, 106

$\lfloor Z_u \rfloor^{\top}$	The unary part of the Golomb encoding of $\lfloor Z_u \rfloor$, 106
$\lfloor Z_u \rfloor^{\perp}$	The uniform part of the Golomb encoding of $\lfloor Z_u \rfloor$, 106
z_u	The estimation error of $n_{\overline{u}}$, 106
Z_u	The absolute value of the estimation error of $n_{\overline{u}}$, $\lvert z_u \rvert$, 106
sg_u	The sign of the estimation error of $n_{\overline{u}}$, one if $z_u > 0$ and zero otherwise, 106
$T^{[m]}$	Truncation of T to depth m, 103
$T_{\mathbf{c}}^{[m]}$	The minimal canonical extension of $T^{[m]}$, 103
$S_{\mathbf{c}}^{[m]}$	The set of states of $T_{\mathbf{c}}^{[m]}$, 104
$S_{\mathbf{c}}^{[m]}(r)$	The set of states of $T_{\mathbf{c}}^{[m]}$ that are children of r, 106
$\sigma_{\mathbf{c}}^{[m]}(u)$	State selected by u in $T_{\mathbf{c}}^{[m]}$, 104
U'_{k+1}	The set of nodes of $T_{\mathbf{c}}^{[k+1]}$ that are not in the original context tree $T^{[k+1]}$ and are not in $T_{\mathbf{c}}^{[k]}$, 106
U_{k+1}	The set of parent nodes of elements of U'_{k+1}, 106
R_i	The set of parent nodes of states of $T_{\mathbf{c}}^{[i]}$, 107

Abbreviations

BWT	Burrows-Wheeler transform, 10
CPMF	Conditional probability mass function, 18
CTW	Context Tree Weighting, 9
FSM	Finite state machine, 1
GCT	Generalized context tree, 21
GCTM	generalized context tree model, 23
KT	Krichevsky-Trofimov, 37
LZ	Lempel–Ziv, 13
NML	Normalized Maximum Likelihood, 12
PPM	Prediction by Partial Matching, 10

Chapter 1

Introduction

1.1 Information sources and universal coding

Consider an *information source* that produces messages to be transmitted over a communication system. As described in Shannon's foundational paper [70], we can think of a discrete information source as generating a message, symbol by symbol, each belonging to a finite alphabet $\mathcal{A}$; we model this sequence of messages by means of a stochastic process. We will interchangeably use the terms *source* and *model*, to refer to the probabilistic modeling of an information source. Depending on the characteristics of the system, we may choose different classes of stochastic processes. In a *memoryless* source, each symbol is generated according to a fixed probability law; the symbols are independent and identically distributed (i.i.d.). When contiguous symbols are presumably related to each other, we may instead model the information source by means of a Markov chain.

In a *Markov* (or *finite-memory*) *source*, there exists an integer constant $m \geq 0$, such that for every index i the probability of the i-th symbol x_i is determined exclusively by the preceding m symbols $x_{i-m} \cdots x_{i-1}$. The minimum such m is the *order* of the Markov source. A more general setting associates the emission of symbols to state transitions of a Markov chain. Thus, for a Markov chain with a finite set of states S, and a transition probability matrix p, the source emits a symbol $f(u, v)$ whenever the Markov chain goes in a one-step transition from state u to state v, where f is a function $f : S \times S \to \mathcal{A}$. When for every state u, the symbols associated to positive probability state transitions departing from u are all different, i.e., $f(u, v) \neq f(u, v')$ for all $v \neq v'$ such that $p_{u,v} > 0$ and $p_{u,v'} > 0$, the source can be implemented with a *finite state machine* (FSM). The latter is comprised of a finite set of states S and a next-state function $g : S \times \mathcal{A} \to S$. A source of this kind is termed an *FSM source* and it is characterized by an underlying FSM, together with a probability distribution for the initial state and a set of conditional probability distributions on $\mathcal{A}$, one associated to each state of the FSM, of emitting a symbol given the current state. Notice that Markov sources are special cases of FSM sources. In general, the latter may yield *infinite-memory* processes, in the sense that no fixed number of past symbols may suffice to determine the current state of the Markov chain.

We point out that the terminology used for classes of sources is not always consistent in the literature. The term Markov source in [70] refers to the general setting in which a function $f(u, v)$ associates a symbol to each possible transition. The term FSM source has been used for example in [84] with a fixed initial state, while in [2], this is termed a *unifilar Markov source*. In [2], however, the emission of symbols is associated to states rather than transitions, and the initial state is selected according to a stationary distribution of the Markov chain.

What we have termed a Markov source, is a *unifilar finite-memory Markov source* in [2]. We will get into a more detailed discussion of stochastic modeling of information sources in Section 2.1 when we review finite-memory processes.

Now, suppose we want to encode the data generated by a source as a sequence of binary symbols to be transmitted over a communication channel. A *code*, $\mathcal{C} : \mathcal{A} \to \{0,1\}^\star$, maps each symbol of the source alphabet $\mathcal{A}$ to a *codeword*, which is a finite binary string. If no symbol is assigned a codeword that is a prefix of another codeword, the code is a *prefix code*. A code is *uniquely decodable* if any binary sequence that arises as the concatenation of codewords, $\mathcal{C}(x_1)\mathcal{C}(x_2)\cdots\mathcal{C}(x_n)$, uniquely determines the original sequence of source symbols $x_1, x_2 \cdots x_n$. Of course, a prefix code is uniquely decodable, although there are uniquely decodable codes that are not prefix codes.

The *entropy* of a source was introduced as a fundamental limit in *source coding* (or *data compression*) by Shannon in his seminal paper [70]. Indeed, for a random variable X, which takes values on a finite alphabet $\mathcal{A} = \{a_1 \ldots a_k\}$, with probabilities $\{p_1 \ldots p_k\}$, the *entropy* of X, $H(X) = -\sum p_i \log p_i$,[1] gives a lower bound on the expected number of bits required to describe an outcome of X. In other words, if each symbol a_i is encoded with a binary string of length l_i by a uniquely decodable code $\mathcal{C}$, then the expected length of $\mathcal{C}$ for X, $\sum_i p_i l_i$, is lower bounded by $H(X)$. This follows essentially from Kraft's inequality for prefix codes, extended to uniquely decodable codes by McMillan [52], which constrains the codeword lengths to satisfy $\sum 2^{-l_i} \leq 1$. Conversely, for any set of positive integers $\{l_i\}$ that satisfy Kraft's inequality there exists a prefix code with these codeword lengths (see, e.g., [12, Chapter 5]), which makes non-prefix codes of little interest among the class of uniquely decodable codes.

Notice that the minimum expected code length $H(X)$ is attained exactly if we can define a code with *ideal codeword lengths* $l_i = -\log p_i$. Although these lengths obviously satisfy Kraft's inequality, they may not be integer numbers. A Shannon code[2] [70] assigns a codeword of length $\lceil -\log p_i \rceil$ to each symbol a_i, and achieves an expected code length that exceeds the entropy of X by at most one bit. This code was applied in [70] to blocks $X^n = X_1 \ldots X_n$, of n consecutive symbols emitted by a source (regarded as "macro-symbols" in an *extended alphabet* $\mathcal{A}^n$), to obtain a per-symbol expected code length that satisfies

$$\frac{H(X^n)}{n} \leq \frac{\mathrm{E}\big[\,\mathcal{L}_{\mathcal{C}}(X^n)\,\big]}{n} < \frac{H(X^n)}{n} + \frac{1}{n}\,,$$

where $\mathrm{E}\big[\,\cdot\,\big]$ denotes expectation with respect to the source and $\mathcal{L}_{\mathcal{C}}(X^n)$ denotes the length of the codeword $\mathcal{C}(X^n)$. When $\{X_i\}$ represents a discrete stationary random process, $\frac{1}{n}H(X^n)$ has a limit $\mathcal{H}$, termed the *entropy rate* of the process [70]. $\mathcal{H}$ represents a lower bound on the expected per-symbol code length, which is at the same time asymptotically attainable, for example by a Shannon code on length-n blocks of symbols. Moreover, by means of an *arithmetic coder* [57], one can overcome the practical problems that arise from extending the alphabet size as n grows large, and approach in practice the entropy rate limit.

[1] Exponentials and logarithms are taken with respect to base 2.

[2] The Shannon code is also known as a Shannon-Fano code, as Fano developed independently a different construction for essentially the same code published in [26].

In general, a probability distribution P, suitable for modeling the data generation mechanism of the information source, is unknown a priori. When it is reasonable to assume that P belongs to a certain known class $\mathcal{P}$ (for example the class of all Markov sources of a fixed order m), we may seek a code that performs asymptotically well for *all* sources in the class simultaneously. Here, the term *code* refers in fact to a sequence of codes, $\{\mathcal{C}_n\}$, one for each input length n, and we are interested in large n. In the sequel we will use the term *code* to refer either to a specific mapping from a given finite alphabet to binary codewords, or to a sequence of such mappings, $\{\mathcal{C}_n\}$, $\mathcal{C}_n : \mathcal{A}^n \to \{0,1\}^*$.

For a sequence of n symbols $x^n \in \mathcal{A}^n$, the *pointwise redundancy* of a code is the difference between the codeword length for x^n and the ideal code length, $-\log P(x^n)$, where $P(x^n)$ is the probability assigned by the source to x^n. Thus, the pointwise redundancy is given by

$$R_{P,\mathcal{C}_n}(x^n) = \mathcal{L}_{\mathcal{C}_n}(x^n) + \log P(x^n)\,.$$

The expectation of $R_{P,\mathcal{C}_n}(X^n)$ with respect to P is the *expected redundancy* of the code for the given source, and it is given by

$$\bar{R}_{P,\mathcal{C}_n} = \mathrm{E}_P\big[\, R_{P,\mathcal{C}_n}(X^n)\,\big] = \mathrm{E}_P\big[\, \mathcal{L}_{\mathcal{C}_n}(X^n)\,\big] - H(X^n)\,.$$

Since the entropy $H(X^n)$ represents a lower bound on the mean length of any uniquely decodable code, the expected redundancy is non-negative. Given a code and a class $\mathcal{P}$ of distributions, the *worst-case expected redundancy* is defined as

$$\bar{R}_{\mathcal{C}_n} = \sup_{P\in\mathcal{P}} \bar{R}_{P,\mathcal{C}_n}\,,$$

and the *worst-case maximum redundancy* is defined as

$$R^*_{\mathcal{C}_n} = \sup_{P\in\mathcal{P}} \max_{x^n\in\mathcal{A}^n} R_{P,\mathcal{C}_n}(x^n)\,.$$

We say that a code, $\{\mathcal{C}_n\}$, is *universal* in the class $\mathcal{P}$ if the *normalized* expected redundancy, $\frac{1}{n}\bar{R}_{P,\mathcal{C}_n}$, vanishes for each $P \in \mathcal{P}$ as n goes to infinity. Universality may be *strong* or *weak*, depending on whether the convergence is uniform in the class, i.e., whether $\frac{1}{n}\bar{R}_{\mathcal{C}_n}$ vanishes, or not, respectively [18].[3] If the normalized *maximum pointwise redundancy*, defined as $\frac{1}{n}\max_{x^n\in\mathcal{A}^n} R_{P,\mathcal{C}_n}(x^n)$, converges to zero for each $P \in \mathcal{P}$, the code is *pointwise universal* in the class $\mathcal{P}$. Again, the universality is *strong* if $\frac{1}{n}R^*_{\mathcal{C}_n}$ vanishes as n grows. Clearly, pointwise universality implies universality (in expectation).

With the problem of optimally encoding a source under perfect knowledge of the probability distribution essentially solved, we can address universality by looking for *universal probability distributions* Q_n on length-n sequences,[4] which are simultaneously close to all

[3] The notion of universality we have defined corresponds to *minimax* universality in [18], where also *maximin* universality is defined. Later, however, it was shown that both concepts are equivalent [31, 64, 20].

[4] In rigor it is only required that the ideal code lengths $\{-\log Q_n(x^n)\}$, over all length-n sequences x^n, satisfy Kraft's inequality. Thus, Q_n does not need to sum up to unity.

the distributions P in the class, in the sense of yielding a vanishing normalized *divergence*[5], defined [43] as

$$\frac{1}{n}D(P||Q_n) = \frac{1}{n}\mathrm{E}_P\big[-\log Q_n(X_1\cdots X_n) + \log P(X_1\cdots X_n)\big]\,. \tag{1.1}$$

Notice that (1.1) is the penalty in per-symbol expected ideal code length incurred when assuming that the distribution is Q_n when it is actually P. In other words, the divergence $D(P||Q_n)$ is the expected redundancy of an ideal code for Q_n when the actual distribution is P. Notice also that, although a code attaining ideal length for Q_n only exists when $Q_n(x^n)$ is a power of two for all length-n sequences x^n, the codeword lengths of a Shannon code for Q_n are within one bit of the ideal codeword lengths and, thus, the normalized expected redundancy of this Shannon code vanishes if (1.1) does. A sequence of probability distributions $\{Q_n\}_{n>0}$ is called a *probability assignment*. We will formalize this definition later on. Thus, the problem of finding universal codes for classes of models is equivalent to that of finding universal probability assignments for them.

The idea of universal coding was first introduced by Kolmogorov [39], and developed later for several particular classes of processes (see e.g. [29, 30, 18, 19, 42]), showing that it is possible in fact to construct universal codes for several classes of interest. A natural question is, then, at what rate a universal code can approach the entropy rate $\mathcal{H}$. For a parametric class $\mathcal{P}$, Rissanen's lower bound [59, Theorem 1] "quantifies" the very intuitive idea that the optimal rate depends on the richness of $\mathcal{P}$. Specifically, consider a parametric class of distributions $\mathcal{P} = \{P_\theta\}_{\theta\in\Theta^k}$, where the parameter space Θ^k is a compact subset of $\mathbb{R}^k$, where $\mathbb{R}$ denotes the real numbers. It is shown in [59] that, under some mild conditions, for any universal code for $\mathcal{P}$, the normalized expected redundancy for a block of n symbols is lower-bounded by a *model cost* term of $\frac{k\log n}{2n}$ bits plus lower order terms, for *most* values of the class parameter θ. Previous works showed that the lower bound holds at least for one value of the model parameter [41, 79]. Notice that the model cost term grows with the dimension k of the parameter space. Thus, when it comes to modeling an information source, we are faced with a trade off. If we choose a class that is too rich, we may incur an unnecessarily slow convergence rate, while if the class is too simple, it may be impossible to accurately fit any probability distribution in the class to the actual data. This observation is the main motivation for the investigation of *tree models*, which offer economic parametrizations of Markov sources. In some cases of practical interest, tree models allow for a significant reduction in the dimension of the parameter space as compared to basic Markov sources.

In light of Rissanen's lower bound, we will say that a code is *universal with optimal convergence rate* for a parametric class $\mathcal{P}$, if for each fixed source in the class, the normalized expected redundancy vanishes as $\frac{k\log n}{2n}$ up to lower order terms, where k is the dimension of the parameter space. Somewhat surprising is that when the class $\mathcal{P}$ is a countable union of subclasses, i.e., $\mathcal{P} = \bigcup_i \mathcal{P}_i$, and there exists a strongly universal code in each subclass $\mathcal{P}_i$, it is possible to construct a *twice-universal code* [65], whose worst-case expected redundancy

[5] The divergence is also called Kullback-Leibler distance, relative entropy, information divergence, and cross entropy (see, e.g., [12]).

within each subclass $\mathcal{P}_i$ is asymptotically as small as that of the universal code for $\mathcal{P}_i$ that is optimal in the sense of minimizing the worst-case expected redundancy.

For a parametric class $\mathcal{P}_i$ of distributions with a k_i-dimensional parameter, under the assumptions of Rissanen's lower bound, the redundancy of any code is lower bounded by $\frac{k_i \log n}{2n}$ not only in a worst-case parameter scenario, but for most values of the parameter. This applies in particular to an optimal code for $\mathcal{P}_i$, and, thus, we will say that a code is *twice-universal* in the union of parametric classes $\mathcal{P} = \bigcup_i \mathcal{P}_i$, if for any source in $\mathcal{P}_i$, for any i, the normalized expected redundancy of the code vanishes as $\frac{k_i \log n}{2n}$ up to lower order terms. A typical example is the construction of twice-universal codes in the class $\mathcal{P}$ of Markov sources, which is comprised of the union of all subclasses $\mathcal{P}_i$, each composed of all Markov sources of order i. A twice-universal code for $\mathcal{P}$ would be one that attains a normalized expected redundancy with main term $\frac{k_i \log n}{2n}$ when applied on sequences emitted by sources of order i, where $k_i = |\mathcal{A}|^i(|\mathcal{A}| - 1)$ is the dimension of the source parameter.

There are many strategies that we can apply to find a (twice-)universal probability assignment within a given class. We mention here three classical approaches, which have important representatives in the class we are interested in, namely, the class of tree models. Probably one of the first ideas one could think of is a *"plug-in"* approach, where the parameter, or even the parameter dimension in a twice-universal setting, is estimated based on the data observed so far, and this estimation is used to assign a probability for the next symbol given the past, as if it were the "true" parameter. The appeal of this idea is that it intrinsically yields *sequential codes*, i.e., the probability assigned to a sequence x^n, which determines the code length for x^n, is the product of the n conditional probabilities given the past assigned *sequentially* to the symbols of the sequence. Thus, by using an arithmetic encoder, both the coding and the decoding stages can proceed as the symbols become available, independently of the data to come. A *mixture* approach is based on defining a probability distribution over length-n sequences, for each n, that is a weighted average of all the distributions in a class where we seek universality. The weighting is chosen so as to make the average close, in a divergence sense, to all the distributions in the reference class simultaneously. Depending on the weighting, it is sometimes possible to identify an expression for the conditional probability of the next symbol given the past, which is easy to evaluate (see, e.g., [88], or the survey in [53] and references therein). This yields, also in this case, a sequential coding algorithm. In fact, there are remarkable examples where a code that results from a mixture approach can be interpreted as a plug-in code (and viceversa). This is the case, for example, of the Krichevsky-Trofimov probability assignment [42], and also of a plug-in code based on the well known Laplace estimator. Finally, in situations where sequentiality is not of prevailing interest, one may consider a *two-pass* approach. Here, the encoder scans the whole input data in a first pass and determines a code, which is described to the decoder, and then used to actually encode the sequence in a second pass. The encoded data consists therefore of two parts: the second part encodes the input sequence itself, and the first part is a description of the code used in the second part. In a two-pass universal code, the expectation of the normalized length of the second part usually converges to the entropy rate of the source, while the normalized length of the first part vanishes.

As opposed to the classical stochastic approach for the study of data compression, in an *individual sequence setting*, the input data is not regarded as generated by any probabilistic source. The goal in this case is to define a code such that the code length for each input sequence x is close to the shortest codeword assigned to x by any *competing* encoder (or simply *competitor*) in a family $\mathcal{M}$ of limited resources (e.g., limited memory). If the family $\mathcal{M}$ is parameterized by a k-dimensional parameter, where the dimension here expresses the amount of resources, the term $\frac{k \log n}{2n}$ may be regarded as a determinist counterpart of Rissanen's lower bound. Although such a general lower bound for the individual sequence setting has not been established in full generality, it can be justified to some extent by the results in [83] for encoders implementable with finite state machines, and it is still regarded as the target benchmark for the convergence rate in this setting.

An example of a class of competitors is the family $\mathcal{M}_m$ of finite-memory encoders of order m, i.e., a competitor in this family is constrained to encode each symbol x_i with a codeword $f(x_{i-m} \cdots x_{i-1}, x_i)$ that, besides x_i, depends at most on the last m encoded symbols. This family can be parameterized by a parameter of dimension $k_m = (\alpha - 1)m^\alpha$. It is not difficult to see that the smallest per-symbol code length achieved by an encoder in $\mathcal{M}_m$, for any given sequence x, is lower bounded by the m-th order *empirical entropy rate* of x, $\hat{\mathcal{H}}(x) = -\frac{1}{n} \log \hat{P}(x)$, where $\hat{P}$ is the maximum-likelihood probability of x in the class of m-th order Markov models. We say that a code is *universal for individual sequences* with respect to $\mathcal{M}_m$ if it attains, for each x, a code length of $\hat{\mathcal{H}}(x) + o(1)$ bits. Since $\hat{P}$ minimizes $-\log P(x)$ among all P in the class of Markov sources of order m, it follows that for a universal code for individual sequences in $\mathcal{M}_m$, the normalized maximum pointwise redundancy in the class of Markov sources of order m vanishes and, therefore, the code is pointwise universal in a stochastic setting for this class. Moreover, if the excess over $\hat{\mathcal{H}}(x)$ in the individual sequence setting is of order $\frac{k_m \log n}{2n}$, the code also has optimal convergence rate in the stochastic setting.

For the family of competitors comprised of the countable union $\mathcal{M} = \bigcup_m \mathcal{M}_m$, we say that a code is *twice-universal for individual sequences* with respect to $\mathcal{M}$ if it is universal for individual sequences with respect to $\mathcal{M}_m$ with a convergence rate of order $\frac{k_m \log n}{2n}$ simultaneously for every m. Again, a twice-universal code for individual sequences in this family is also twice-universal in the class of Markov sources in a stochastic setting. These notions of universality and twice-universality for individual sequences in the families $\mathcal{M}_m$ and $\mathcal{M}$ extend straightforwardly to similar families of competitors related to tree models that we will define later on.

In the remaining sections of this chapter we outline the dissertation's main topics, which will be studied in depth in later chapters. We start in Section 1.2 by roughly describing *context trees* and how they are used to define tree models, our main subject of investigation. We also describe a generalization of the classical context trees, and connect them to FSMs through what we shall define as the *FSM closure* of a *generalized context tree*. In Section 1.3 we briefly describe some known source codes for tree models, and, in particular, a specific version of the so-called Context algorithm [58], which we will later study in detail from an algorithmic complexity point of view. In Section 1.4 we describe the *method of types* [15] and the particular characteristics that apply to tree models. This motivates the study of *enumer-*

ative codes [11] for tree sources, which we outline in Section 1.5. Some of the tools developed in the investigation of enumerative codes for tree sources will derive in a generalization of the method of types following the same idea as in [67]. This is introduced in Section 1.6 together with applications in simulation of individual sequences. Finally, in Section 1.7 we summarize the main contributions of this thesis and in Section 1.8 we introduce some basic definitions.

1.2 Tree models

In many practical applications, an information source can be well approximated by a model in which the conditional probability assigned to the next emitted symbol, given all the past, depends on a finite (bounded) *context*, where by context we mean any contiguous sequence of the most recently emitted past symbols. In other words, such an information source can be well approximated by a Markov source of some finite order m. Such a model can be fully parameterized for an alphabet of size α by the conditional probabilities of $\alpha - 1$ symbols in each state (m-vector of symbols) of the underlying Markov chain. Thus, in general, we require a parameter space of dimension $(\alpha - 1)\alpha^m$. It is often the case, however, that the actual context length required to determine the probability distribution of the next symbol varies depending on the context. For example, in many languages, the frequency of occurrence of the letter following a 'q' can be accurately estimated even ignoring the context preceding that 'q', whereas, in other situations, a longer context may greatly improve this estimation. In such a case, the dimension of the parameter space, which, if we were to use a fixed context length m would grow exponentially with m, can be dramatically reduced by lumping together equivalent states that yield identical conditional distributions (e.g., contexts ending in 'q' in our language example). The reduced models, first considered in [58], were termed *tree models* in [84], since they can be represented with a simple tree structure. Tree models have also been referred to as *variable length Markov chains* [8] in the statistics literature.

Roughly speaking, a tree model consists of a full α-ary *context tree*,[6] where α is the size of the source alphabet, and a set of conditional probability distributions on the alphabet, one associated with each leaf of the tree (the *states*). Each edge of the tree is labeled with a symbol from the source alphabet. The state selected by a string is determined by descending from the root, matching the labels of the edges with the symbols in the string, starting from the last symbol an advancing in reverse order, until a leaf is reached. Thus, in the context tree T_1 of Figure 1.1, all strings ending with symbol 0 select the same state, while for strings ending with symbol 1, it may be necessary to examine one, or even two more past symbols of the string in order to determine the state. A full parametrization of a binary Markov source of order three, which we term a *plain Markov model*, corresponds to an underlying context tree of depth three that is balanced, like T_3 in Figure 1.1. Notice that in this example, the number of states of T_3 is twice the number of states of T_1. Thus, since for a binary alphabet the number of states is equal to the dimension of the parameter space, in situations where T_1 is suitable for modeling an information source, the model cost term is reduced by a factor of

[6] We say that an α-ary tree T is *full* if every internal node of T has exactly α children; T is *full balanced* if it is full and all its leaves are at the same depth.

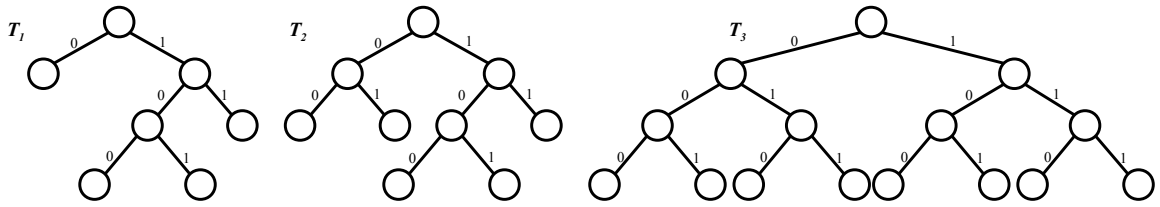

Figure 1.1: *Context trees over* $\mathcal{A} = \{0, 1\}$

1/2 as compared to a plain Markov model.

Although the savings in model size can be important for tree models as compared to plain Markov models, it may be possible to go even further in this direction if the structure underlying the model is not constrained to be a full context tree (see, e.g., [89, 80, 73, 68]). Indeed, we would ideally let the model class group the states of a Markov chain arbitrarily, if such a general scheme were computationally feasible. In Chapter 2 we define an extension of the class of tree models, named *generalized context tree models*. In this extended class the state selection mechanism is governed by *generalized context trees*, which need not be full. Generalized context trees will serve as a fundamental tool in the development of efficient encoding and decoding algorithms that we present in Chapter 3. Although our main motivation for extending the class of tree models is algorithmic, as we will discuss later, we formalize and discuss the class of generalized context tree models in detail on its own right, as this richer class offers potentially significant improvements in model fitting capability relative to the usual full-tree models.

As noted in [82], a context tree (and therefore also a generalized context tree) might not define a next-state function, i.e., the occurrence of a symbol in a given state does not necessarily determine the following state. For example, in the context tree T_1 of Figure 1.1, the state selected by a string ending with symbol 0, which "remembers" only one past symbol, does not allow the determination of the following state if the next symbol is 1. As we shall see, this results in algorithmic complexity issues in practical applications, and, furthermore, it causes tree models to deviate from plain Markov models with respect to some important theoretical properties, which consequently require, in some cases, a more intricate analysis. Tree models that do define a next-state function were termed FSMX in [60]; their characterization was explored in [69], where they are referred to as *FSM trees*, and in [88]. The *FSM closure* of a context tree [48, 69, 88], is the smallest FSM tree containing the context tree. All the tree models that are generated by a context tree T as the parameter is allowed to range over its valid domain can also be generated as FSM models based on the FSM closure of T.

In Chapter 2 we extend the definition of FSM closure to generalized context trees. We characterize this FSM closure and present an algorithm that builds it efficiently. Moreover, this algorithm constructs a mapping from the states in the FSM closure to the states in the generalized context tree, such that given the state selected by a string in the former, we get, in constant time, the state selected in the latter.

The efficient mapping from the states in the FSM closure to the states in the generalized

context tree will turn out to be very useful in a sequential coding algorithm. In such an algorithm, each symbol may be encoded using a code that depends on the state selected in a generalized context tree by the string seen so far, as well as statical information associated to that state, which is updated as the symbols of the source are consumed. Thus, to encode a string x, the encoder needs to determine the state selected by each prefix of x, in increasing order of length. In principle, determining each state involves descending from the root of the generalized context tree until a state is determined, which may become computationally expensive for a large tree. However, if the encoder is equipped with an FSM closure of the generalized context tree, it may efficiently determine the sequence of states selected by the prefixes of x in the FSM closure by applying the next-state function in each step, and use the constructed mapping to access the states actually selected in the original generalized context tree. The application of this idea is outlined in the next section, and studied in detail in Chapter 3. The context tree T_2 of Figure 1.1 is the FSM closure of T_1. Notice that the number of states (leaves), and, hence, the dimension of the parameter space, is larger for a tree model based on the FSM closure than for a tree model based on the original context tree T_1. Thus, in the mentioned coding application, accessing the states selected in the original context tree, rather than in the FSM closure, is vital to avoid increasing the model cost term of the code length.

1.3 Low complexity coding algorithms for tree sources

Coding of tree sources has been extensively studied in the framework of twice-universality, be it in the stochastic setting, or in the individual sequence setting. In the latter case, the competing encoders are constrained to encode each symbol based on the state selected by the preceding symbols in a fixed but arbitrary context tree. The starting point of this research is found in [58], where a plug-in type of algorithm named Context was introduced. The model estimator of Context was improved in [82], obtaining a universal code with optimal convergence rate in the subclass of FSM tree models, assuming a known bound on the depth of the underlying context tree. This was further developed in [84], removing the assumption of a bound on the depth of the context tree, as well as the condition of defining an FSM, thus giving a universal code with optimal convergence rate in the whole class of tree models.

Context Tree Weighting (CTW) [88, 87] is a sequential coding algorithm based on the mixture approach. The coding probability is obtained by mixing all the distributions over length-n sequences defined by tree models. The first version of CTW in [88] assumed a known bound on the depth of the underlying context tree of the source, an assumption that was removed in [87].

The idea of applying two-pass codes in a twice-universal framework was outlined in [65] for countable unions of parametric model classes, which includes the union of all tree models. The best model structure in the whole class (e.g., a context tree) is estimated and described to the decoder in a first pass, and then the data is encoded in a second pass with a universal code for the above best model structure. In [61], this approach is termed "semi-predictive" for the case in which the universal code used for the given model structure is sequential.

It is shown for both CTW [88, 87] and the semi-predictive version of Context with an appropriate probability assignment (see, e.g., [56, 90]), that for any tree model with a K-dimensional parameter, the normalized excess code length given by these codes on sequences of length n, over the empirical entropy rate of the tree model, is at most $(K \log n)/(2n)+O(K/n)$, for any K. Thus, these codes are twice-universal for individual sequences, and therefore also twice-universal in a stochastic sense.

Since much of the emphasis in the information-theoretic literature has been on sequentiality, the two-pass approach has not received much attention. On the other hand, probably due to the existence of efficient implementations, other data compression algorithms that are also based on context models but lack the above provable strong universality properties, are very popular though not necessarily sequential. This is the case of the algorithms based on the Burrows-Wheeler transform (BWT) [9] (see [23] and references therein). These algorithms can be viewed as coding based on a context tree (see [24] for an information-theoretic analysis), except that no attempt at context selection is made. Instead, the sequence is reordered in such a way that symbols occurring in "similar" contexts appear in nearby locations. Coding is often done by sub-optimal, simple methods in a second pass.

The popularity of BWT-based schemes suggests that, in many applications, sequentiality is not a fundamental requirement. Thus, in such cases, a low-complexity implementation of the semi-predictive approach is of interest. With this approach, a context tree, which stores all relevant statistical information of the sequence, is "pruned" to minimize code length [56], and described to the decoder in a first pass through the data. In a second pass, each symbol is assigned a conditional probability sequentially, conditioned on the state selected in the pruned context tree by the substring traversed so far. This assignment, in turn, is used for, e.g., arithmetic coding.

It is shown in [3] that the semi-predictive approach can be implemented in linear *encoding* time.[7] The first pass is solved by combining the dynamic programming ideas of [56] with the use of *compact suffix trees* [32] as in [45], which allows for efficiently collecting the relevant statistical information of the given sequence. Indeed, the use of compact suffix trees is also crucial for low complexity implementations of the BWT, and other context based algorithms such as Prediction by Partial Matching (PPM) [10] in its multiple variants (see, e.g., [23, 44]). The second pass of the semi-predictive approach, which involves sequential state transitioning as the symbols of the sequence are encoded, is addressed in [3] by the use of the BWT. Unfortunately, the BWT is not available during decoding to provide a constant transition time per symbol.[8]

In Chapter 3 we use the FSM closure tool developed in Chapter 2 to implement the second pass of the semi-predictive approach efficiently, thus, yielding a linear-time implementation of encoding *and decoding*, without recourse to the BWT. Notice that since the model optimization carried out in the first pass takes place only at the encoder, the semi-predictive approach is especially attractive in situations where the computational requirements are asymmetric

[7] Throughout, we will measure complexity by the number of *register-level* operations, defined as arithmetic and logic operations, address computations, and memory references, on operands of size $O(\log n)$.

[8] An alternative approach for efficient context transition, used for PPM in [23], is the use of suffix links. For the semi-predictive Context algorithm, again, this approach cannot be used directly at the decoder.

and a low complexity decoder is needed. For the implementation of the first pass, we make use of compact suffix trees, which are generally not full. This is facilitated by generalized context trees, which need not be full (so that states may be given by nodes other than the leaves), and the edges may be compacted (i.e., labeled by strings of length greater than one).

1.4 Type classes of tree models

Despite the extended use of tree models in data compression and other applications in information theory, important theoretical questions about these models remained unanswered. Some of these questions are addressed, for the first time, in this work. In Chapter 4, we extend the *method of types* [15] to tree models. In this method, the set of sequences of a given length n over a finite alphabet $\mathcal{A}$ is partitioned into *type classes*, where two sequences belong to the same class if and only if every probability distribution in a certain class $\mathcal{P}$ assigns both sequences the same probability.[9] For a parametric family $\mathcal{P}$, a type class comprises all the sequences that yield a given value for a sufficient statistic for $\mathcal{P}$.

In our case, $\mathcal{P}$ is the set of all tree models obtained from a fixed context tree as the model parameter, i.e., the vector of conditional probability distributions associated to the states, varies along its valid domain. A sufficient statistic in this case is the vector of state-conditioned empirical distributions for the given context tree. As a consequence, the partition of sequences into type classes depends exclusively on the underlying context tree, in the same way that, for example, type classes of Markov models of order m depend only on m.

Applications of the method of types in hypothesis testing, channel coding, source coding, rate-distortion theory, and other areas are surveyed in [13]. Beside memoryless models, type classes and their applications have also been studied for other cases, such as Markov models (e.g., [21, 14, 13, 35]) and FSM models (e.g. [83]).

For Markov and other FSM models, the size of the type class of a sequence of length n is given exactly by Whittle's formula [86], which was also derived, using different methods, in [6] and [34]. The latter shows that the problem is equivalent to counting Eulerian circuits in a graph, which can be done explicitly through the BEST Theorem [22] (named after de Bruijn, van Aardenne-Ehrenfest, Smith and Tutte). The proofs rely strongly on the defining property of FSMs, namely, a next-state function. As mentioned, however, context trees do not always define a next-state function, so the results for FSMs do not extend, in general, to trees. In this work we focus on basic properties of type classes for tree models, e.g., their number and size.

In Chapter 4 we derive an exact formula for the size of the type class of x^n with respect to a given context tree. The formula resembles, and generalizes, Whittle's formula, as the problem is also reduced in our case to one of counting Eulerian circuits in a directed graph, which is derived from the given context tree and the associated counts for x^n. The lack of a next-state function, and the loss of context occurring in the state transition sequences

[9] Type classes are defined in terms of empirical distributions for memoryless models in [15]. The more general notion of type class used here is introduced in [13] when extensions of the method of types to wider model families are considered.

of context trees, however, are major challenges in the derivation. The exact combinatorial formula obtained in the derivation is then analyzed to characterize the asymptotic behavior of the expected size of the type class for a random sequence emitted by a tree source, which, again, generalizes in a nontrivial fashion the corresponding behavior for FSMs.

We also study the number of type classes for sequences of length n induced by a given context tree, and we estimate the number of classes tightly, up to a multiplicative constant. This result also generalizes the corresponding result for FSMs, presented in [83] and attributed to N. Alon.

1.5 Enumerative coding

The general idea of *enumerative coding* is to encode a sequence of symbols by describing, with a uniform code,[10] its index within a given set S according to a predefined order. Although it had been applied for specific purposes before (e.g. [46, 66]), enumerative coding was presented as a systematic method in [11]. Just to give a concrete example, taken from [11], consider a sequence of length n emitted by a memoryless binary source with parameter p. We take S as the set of all binary length-n sequences where the number of occurrences of the symbol 1 lies in the interval $[np - m, np + m]$. It can be shown that for sufficiently large n, we can take m so as to make the probability of S arbitrarily close to unity, and the normalized code length, $\frac{\lceil \log |S| \rceil}{n}$, arbitrarily close to the entropy rate of the source. Thus, this examples illustrates an enumerative coding scheme that permits encoding and decoding with a compression rate asymptotically optimal and a negligible error probability (associated to the sequences that do not belong to S).

The method of types is particulary suitable for the application of enumerative coding, by means of a two-part code comprised of a preamble, which describes the type class to which the sequence at hand belongs, followed by an enumerative code of the sequence within its type class. Since all sequences in a type class are equiprobable, a uniform encoding of the index minimizes the expected length of the second part. Thus, the problem of assigning probabilities to sequences universally for the model of reference reduces to optimally assigning probabilities to type classes, i.e., to minimizing the expected length of the first part of the code. This is indeed the case for the *Normalized Maximum Likelihood* (NML) code [71, 62], which can be interpreted as a description of the type class, generated by assigning to it a probability proportional to its ML probability, followed by an identification of the sequences within its type class.

Implementing the NML code, however, is difficult even for the simplest model classes. Other universal methods, based, for example, on the Krichevskii-Trofimov sequential probability assignment [42], are computationally efficient, and also assign the same code length to all the sequences of a given type class. They do not, however, provide a separate and identifiable description of the type class. In Chapter 5 we will be interested in *enumerative codes* for

[10] A uniform code for a finite set S is any code with minimum expected code length under the uniform distribution over S. If $|S|$ is a power of two, all elements of S are encoded with equally long bit streams of length $\log |S|$. Otherwise, the code has words of length $\lceil \log |S| \rceil$ and $\lfloor \log |S| \rfloor$.

tree models that are universal with optimal convergence rate, and possess both qualities: they provide a separate description of the type class of the encoded sequence, and this description can be efficiently computed. By "efficient computation" we mean one whose encoding running time is polynomial in the length of the input sequence, and with code construction time that is also polynomial in the dimension of the parameter space of the model.[11]

For (non-curved) exponential families of probability distributions (see, e.g., [7]) satisfying some mild regularity conditions, most type classes have, to first approximation, the same ML probability [54, Appendix A]. This observation leads to universal enumerative codes with optimal convergence rate, for which uniform coding is used both for the set of type classes and for the set of sequences of each type class. In particular, for FSM models,[12] which are asymptotic exponential families [75, 76], such a code can be efficiently implemented.

In contrast to the FSM model case, a non-FSM tree model does not induce an exponential family of distributions even in an asymptotic sense [76], and it is possible to find type classes that have small probability for *any* choice of the model parameter. This suggests that while a uniform code for the set of type classes may be suboptimal, savings in code length might be recovered with a *non-uniform* code for the type classes.

In Chapter 5, we construct such a non-uniform code. This, together with the tools developed in Chapter 4, leads to an efficient universal enumerative code with optimal convergence rate for tree models. Furthermore, in the twice-universal setting, in which a context tree is not given and optimality is rather required for *any* possible tree model, we show that, by suitably estimating a context tree from the data, the sequences in the aforementioned "atypical" type classes for each given context tree would in fact estimate a different context tree. These type classes can thus be discarded from the coding space, leading to a twice-universal enumerative code in the class of tree models.

1.6 Universal tree type classes and simulation of individual sequences

In [67], the set of length n sequences is partitioned into so-called *universal type classes*, where two sequences belong to the same class if and only if their Lempel–Ziv (LZ) parsing [91] yields the same parsing tree. This partition extends in a sense the conventional notion of type class, as it is shown that any two sequences in the same class satisfy the following property.

P1 For any fixed integer j, the variational distance between the empirical distributions of j-tuples corresponding to the two sequences is a vanishing function of n.

[11] Notice that, since the number of type classes in the cases of interest, and in particular for tree models, is exponential in the dimension of the parameter space of the model, a construction of the NML code relying on the computation of the ML probability of each type class would be very inefficient.

[12] Although FSM models are not strictly exponential families (they are curved exponential families), conditioning on a fixed final state s does define an exponential family of probability distributions over the length-n sequences with final state s (see, e.g., [72]). Since all sequences in a FSM type class share the same final state, conditioning the probability of a type class on its final state, say s, amounts to dividing by the probability of observing state s at time n. This does not affect the mentioned asymptotic property of the ML probability of type classes.

We show in Chapter 5 that, if we use an appropriate context tree estimator, some statistics of the input sequence lie within relatively small well characterized ranges, which depend on the type class of the sequence with respect to the estimated context tree. This is exploited in Chapter 5, in a twice-universal enumerative code, to discard some type classes from the coding space. In Chapter 6 we observe that, by the same property, any two sequences that estimate the same context tree and belong to the same type class with respect to the estimated context tree must have similar statistics. Thus, we define a *universal tree type class* as the set of all sequences that estimate the same context tree and belong to the same type class with respect to the estimated context tree. A similar notion of *universal Markov type class* is defined in [47] using a plain Markov order estimator instead of a context tree estimator. We will see that both universal Markov type classes and universal tree type classes satisfy P1 with a similar convergence rate of the statistics as n goes to infinity, but universal tree type classes achieve a faster convergence rate as a function of the order.

Universal type classes (be it based on LZ, plain Markov order, or context trees) find application in the problem of simulation of individual sequences presented in [67]. Given a training sequence, x, the goal is to generate a simulated sequence, y, that is statistically similar to x in the sense of satisfying P1. The uncertainty on the simulated output given the training data should be as large as possible, so as to make the simulated sequence look as "original" as possible. In Chapter 6 we also investigate this application and compare the performance, in terms of uncertainty of the output given the input, of a simulator based on universal tree type classes against other simulation schemes that also satisfy P1.

1.7 Summary of main contributions

In Chapter 2 we define generalized context trees and characterize their FSM closures in Theorem 2.6. We present an efficient algorithm for the construction of the FSM closure in Section 2.4 and we analyze its computational complexity in Theorem 2.9. Generalized context trees are especially suitable for the application of compact suffix trees, which provide a valuable tool for model optimization in two-pass algorithms. Indeed, in Chapter 3 we exploit existing efficient algorithms for building compact suffix trees, the generalized context tree formalism, and the algorithm for constructing FSM closures to present the first linear-time algorithm for encoding/decoding with the semi-predictive version of Context.

In Chapter 4 we characterize type classes of tree models, thus extending the method of types to tree sources. We give a formula for the exact size of a type class in Theorem 4.15 and we derive the asymptotic behavior of the expected size of a type class in Theorem 4.18. The asymptotic number of type classes induced by a context tree is characterized in Theorem 4.33.

In Chapter 5 we apply the method of types in the investigation of universal and twice-universal enumerative codes in the class of tree sources, for which we apply the results of Chapter 4. We show that, in general, a uniform encoding of the type classes may be suboptimal. This motivates the definition in Section 5.2 and Section 5.4 of a family of codes for encoding symbol and string occurrence counts in a given context. These codes, which are interesting on their own, are used to implement a universal enumerative code with optimal

convergence rate in the class of tree sources in Section 5.4, which in turn is used in Section 5.5 to derive a twice-universal enumerative code in the same class.

In Chapter 6 we define universal type classes, in the spirit of [67], based on tree models, and we explore their application to simulation of individual sequences. The simulation scheme makes use of the algorithmic enumeration of the context tree type class developed in Chapter 4.

Most of the results presented in the following chapters have been published in joint works with Gadiel Seroussi and Marcelo Weinberger [48, 49, 50], and, in the case of some topics of Chapter 6, also with Neri Merhav [47]. The parts of these publications where the author of this thesis participated less actively were not included in the thesis and cited where needed. There are also some portions of this thesis that have not been published elsewhere. This includes all the derivations in Chapter 4, from which only the main results are stated in [49], as well as Theorem 6.1 for which a plain Markov version was presented in [47]. Also the results of Appendix E and the proofs of Lemma 5.2 and Lemma 5.8 are published here for the first time.

1.8 Basic definitions

We denote by $\mathbb{R}$ the set or real numbers. The integers, the nonnegative integers, and the positive integers are denoted $\mathbb{Z}$, $\mathbb{Z}_{\geq 0}$, and $\mathbb{Z}_{>0}$, respectively.

We use the standard asymptotic O-notation. If f is asymptotically dominated by g, i.e., for every positive ϵ,

$$|f(n)| \leq \epsilon |g(n)| \,, \quad \text{for all } n > n_0(\epsilon) \,,$$

then we write $f(n) = o(g(n))$. We write $f(n) = O(g(n))$ if and only if there exists a positive constant K such that

$$|f(n)| \leq K |g(n)| \,, \quad \text{for all } n > n_0 \,.$$

We also write $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$. Finally, $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

Let X be a random variable that takes values on a finite alphabet $\mathcal{A} = \{a_1 \dots a_k\}$ with probability distribution P, $P(a_i) = p_i$. We recall that the *entropy* of X, $H(x)$, is the expectation with respect to P of $-\log P(X)$, given by

$$H(X) = -\sum p_i \log p_i \,,$$

with the convention that $p_i \log p_i = 0$ if $p_i = 0$ and all logarithms are to base two unless specified otherwise. Since $H(X)$ depends only on P, we sometimes write simply $H(P)$, or $H(p)$, where p is the probability vector $p = (p_1 \dots p_k)$. In particular when the alphabet has two elements, $p = (q, 1-q)$, $H(p)$ takes the form

$$H(p) = h(q) = -q \log q - (1-q) \log(1-q) \,,$$

where the function $h : [0,1] \to [0,1]$ is called the *binary entropy function*.

For random variables X and Y, the *conditional entropy* of X given Y, denoted $H(X|Y)$, is the average, with respect to the distribution of Y, of the entropy of the conditional distribution of X given a specific value of Y.

For two probability distributions P, Q over the same alphabet $\mathcal{A} = \{a_1 \dots a_k\}$, the *divergence* between P and Q is

$$D(P||Q) = \sum_{i=1}^{k} P(a_i) \log \frac{P(a_i)}{Q(a_i)} ,$$

where $p \log p/q = 0$ if $p = 0$ and $p \log p/q = \infty$ if $q = 0$ and $p > 0$.

The *mutual information* between X and Y is

$$I(X;Y) = H(X) - H(X|Y) .$$

The mutual information $I(X;Y)$ is equal to the divergence between the joint probability distribution of X, Y and the distribution given by the product of the marginal distributions of X and Y.

This chapter contains material published in [48].

Chapter 2

Tree sources and FSM closures

We start this chapter with a review of finite-memory processes in Section 2.1, which also sets some terminology and notation for the rest of this dissertation. In particular, we concentrate on FSM models and tree models, both of which are formalisms for generating random processes. We generalize the class of tree models by introducing *generalized context trees* as a new state selection mechanism, and we define the concept of FSM closure for these trees, which is instrumental for the algorithms developed in Chapter 3.

2.1 Finite-memory processes and tree models

In this section, we review finite-memory processes and their parametrizations, particularly tree models. An important aspect emphasized in this review is the distinction between a process and its representations. We first introduce some notation. Let $\mathcal{A}$ be an alphabet of $\alpha \geq 2$ symbols, and let λ denote the empty string. As is customary, we let $\mathcal{A}^*$, $\mathcal{A}^+$, and $\mathcal{A}^m$, denote, respectively, the set of finite strings, the set of positive-length strings, and the set of strings of length $m \geq 0$ over $\mathcal{A}$. In the sequel, except when specifically stated, the variables a, b, c, and d will always represent symbols from $\mathcal{A}$, and r, s, t, u, v, w, x, y, and z will represent strings in $\mathcal{A}^*$. We use the notation u_j^k as shorthand for $u_j u_{j+1} \dots u_k$, $u_i \in A$, $j \leq i \leq k$, and extend it by defining $u_j^k = \lambda$ when $j > k$. Also, we omit the subscript when $j = 1$, i.e., $u^k = u_1^k$. For $u = u^k$, we let $|u| = k$ denote the length of u, $\overline{u} = u_k u_{k-1} \dots u_1$ its reverse string, head(u) its first symbol, u_1 (or λ if $k = 0$), and tail$(u) = u_2^k$ its longest proper suffix. For strings $u, v \in \mathcal{A}^*$, we denote by uv the concatenation of u and v. If u is a prefix (resp. proper prefix) of v, we write $u \preceq v$ (resp. $u \prec v$) Formally, we use the terms *string* and *sequence* interchangeably, but favor the latter in cases where the sequence length is presumed to be unbounded.

Following [63], we consider a (probability assignment) function P from $\mathcal{A}^*$ into the real interval $[0, 1]$ satisfying the conditions

(Q1) $P(\lambda) = 1$,

(Q2) $P(u) = \sum_{a \in \mathcal{A}} P(ua), \quad \forall u \in A^*$.

We will refer to P as a *string process*, or simply a *process* (the term *information source* is used in [63]). Notice that although the string process formalism is different from the usual setting of discrete time, discrete space random processes, all notions of interest in the conventional setting can be expressed very naturally with string processes. For example, assuming $P(x^n) \neq 0$, the function $P(a|x^n) \stackrel{\Delta}{=} P(x^n a)/P(x^n)$, $a \in \mathcal{A}$, is a *conditional probability mass function*

(CPMF) by **(Q2)**, and is naturally interpreted as the probability of the "next" symbol x_{n+1} being equal to a, conditioned on x^n.[1] The string process setting, on the other hand, is very natural when discussing universal coding schemes, which can be regarded as carefully crafted string processes [63].

One way of generating string processes is by use of a *recursive model* [63]. Specifically, given a set of *states* S, consider a *state function* $\sigma : \mathcal{A}^* \to S$ and a set of CPMFs $\{p(\cdot|s)\}_{s\in S}$. For an arbitrary sequence $x^n \in \mathcal{A}^n$, let the state sequence s_0^n be given by $s_i = \sigma(x^i)$, $0 \le i \le n$, and define the function P by

$$P(\lambda) = 1; \quad P(x^n) = \prod_{i=1}^{n} p(x_i|s_{i-1}), \quad n \ge 1\,. \tag{2.1}$$

Clearly, this assignment defines a string process. We say that the model, denoted $\langle \sigma, p\rangle$, generates the process P.[2] For any state s, and x^n such that $\sigma(x^n) = s$, we say that x^n *selects* s, and that s *accepts* x^n. A state is called *permanent* if it accepts arbitrarily long sequences; otherwise, the state is called *transient*. An important particular class of state functions considered, e.g., in [2], is defined through *finite state machines*. For our purposes, an FSM over $\mathcal{A}$ is given by a triple $\mathcal{F} = (S, f, s_0)$, where S is a finite set of states, $f : S \times \mathcal{A} \to S$ is a *next-state function*, and $s_0 \in S$ is the *initial state*. The state sequence s_0^n for x^n is recursively defined by $s_i = f(s_{i-1}, x_i)$. In classical probability theory, the state sequence corresponds to a Markov chain (cf., e.g., [28]). Notice, however, that our definition of permanent state is based solely on the state function, and is independent of the CPMFs associated with the states. Thus, this definition differs from the notion of a *recurrent* state in the theory of Markov chains, which depends on the CPMFs. It is possible to find CPMF assignments that will make a permanent state non-recurrent (provided that some conditional probabilities are set to zero). Our notion of permanent state corresponds to one for which there exists *some* assignment of CPMFs that makes the state recurrent in the classical sense.[3] Our transient states, on the other hand, are always non-recurrent in the classical sense, independently of the CPMFs. Notice that if $s' = f(s, a)$ and s is a permanent state then so must be s' (as it accepts strings of arbitrary length).

For a set of strings $B \subseteq \mathcal{A}^*$, and a process P, we define $B_P = \{u \in B \,|\, P(u) \ne 0\}$. A process P is a *Markov (or finite-memory) source*, if there exists a nonnegative integer m such that, for all $n \ge m$, $a \in \mathcal{A}$, and $x^n \in \mathcal{A}_P^n$, $P(a|x^n)$ satisfies

$$P(a|x^n) = P(a|x_{n-m+1}^n). \tag{2.2}$$

The minimum integer m for which the finite-memory property holds for P is referred to as the *order* of the process. Clearly, this property holds for m if P can be generated with a recursive model such that, for all $n \ge m$ and $x^n \in \mathcal{A}^n$, x^n selects the same state as x_{n-m+1}^n.

[1] When $P(x^n) = 0$, the numerator in the definition of $P(a|x^n)$ must also vanish by **(Q2)**, and the function is undefined.

[2] This model is termed *recursive* in [63] since, in full generality, σ is any recursive function on $\mathcal{A}^*$.

[3] In fact, all but a set of measure zero of the assignments will make a permanent state recurrent. In this sense, the structural model properties we will be interested in will be generally graph-theoretic or algebraic, will be required to hold for "some choice" of CPMFs, but will actually hold for "most choices."

Conversely, every finite-memory process P of order m can be generated by a "basic" FSM model, which corresponds to the notion of *plain Markov model* introduced in Chapter 1. Specifically, we let $S = \cup_{j=0}^{m} \mathcal{A}^j$, $s_0 = \lambda$, and for $a \in \mathcal{A}$ and $b_1^j \in \mathcal{A}^j$, $0 \le j \le m$, the next-state function is given by $f(b_1 b_2 \ldots b_j, a) = b_1 b_2 \ldots b_j a$ for $j < m$, and $f(b_1 b_2 \ldots b_m, a) = b_2 \ldots b_m a$. To complete the FSM model it suffices to select $p(a|b_1 b_2 \ldots b_j) = P(a|b_1^j)$ for all $b_1^j \in \mathcal{A}_P^j$, $0 \le j \le m$ (the choices when $P(b_1^j) = 0$ are inconsequential). On the other hand, not all FSM models define finite-memory processes [2]. Notice that the states corresponding to strings shorter than m symbols in the above FSM are transient, and their sole purpose is to accommodate arbitrary CPMFs $P(\cdot|x^n)$ for $n < m$ (these CPMFs are not constrained by (2.2)). As an alternative to transient states, a *particular* assignment for these strings is often obtained by letting $S = \mathcal{A}^m$ and assuming that s_0 is a given fixed state. In any case, for a given finite set S and arbitrary n, the computation in (2.1) involves a constant number of factors $p(x_i|s)$ for transient states s (and each transient state occurs at most once). Thus, the contribution of transient states to the ideal code length, $-\log P(x^n)$, is $O(1)$, and the properties of the process of most interest to us are determined by the permanent states (each carrying, in general, a parameter of dimension $\alpha - 1$). Transient states are just a "nuisance" that requires formal treatment, but has no impact on the main results.

The finite-memory property depends only on the probability assigned to sufficiently long sequences. To simplify the discussion, we will also constrain the choice of probabilities conditioned on short sequences by further requiring, for each string $v \in \mathcal{A}^*$, the following condition on the process P:

$$\text{if } P(a|uv) \text{ is independent of } u \; \forall u \in \mathcal{A}_P^+, \text{ then } P(a|uv) = P(a|v), \quad a \in \mathcal{A}\,. \tag{2.3}$$

The condition requires that probability assignments conditioned on short strings be consistent with the memory properties of longer strings, ruling out situations, for example, in which the order of the process is determined by the CPMFs of the transient states.[4]

For each particular finite-memory process of order m, other FSM model representations may involve less than α^m (permanent) states. A *tree model* (see, e.g., [58, 82, 84]) is another type of recursive model (not necessarily an FSM model) that may involve less states than the above "basic" FSM model. In a tree model, the state function is determined by a *context tree*. A context tree is a full α-ary tree,[5] where each edge is labeled with a symbol in $\mathcal{A}$, and each node with the string formed by concatenating the edge labels on the path from the root (labeled by λ) to the node. An example of a context tree over a binary alphabet is shown in Figure 2.1. The set of permanent states defined by a context tree is comprised of all the leaves of the full tree, while the internal nodes comprise the set of transient states. For a sufficiently long string x^n, the permanent state selected by x^n is the (unique) leaf in the tree that is a prefix of $\overline{x}^n$. When x^n is not long enough to determine a permanent state, then x^n selects the transient state labeled $\overline{x}^n$. Notice that the state selected by x^n, permanent or

[4] For example, consider a binary finite-memory process for which $P(0|u) = p$ for all strings $u \in \mathcal{A}^+$, and $P(0|\lambda) = q$. Clearly, whenever $q \neq p$, the condition (2.3) is not satisfied by this process and $m = 1$, whereas $m = 0$ for $q = p$. Thus, the value of q, given by the CPMF corresponding to a transient state, determines the order of the process, a situation avoided by requiring (2.3).

[5] A α-ary tree is full if and only if every internal node has exactly α children.

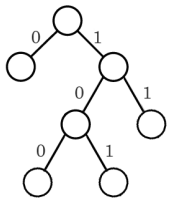

Figure 2.1: *Binary context tree T*

transient, is labeled $x_n x_{n-1} \dots x_{n-j}$, where the symbols are reversed relative to their order in the corresponding suffix of x^n. In general, any suffix of x^n will be called a *context* in which x_{n+1} occurs. To avoid ambiguity, we will use the notation $p(a|s)$ to denote conditioning on an abstract state s, and $P(a|x_{n-j}^n)$ to denote conditioning on an arbitrary suffix of x^n, which may or may not correspond to a state.

The "basic" FSM model representation is equivalent to a tree model in which all the leaves in the context tree have depth m. In a *minimal* tree model representation, the state $\sigma(x^n)$ for $x^n \in \mathcal{A}_P^n$ is determined by the smallest integer $\ell(x^n)$ such that $P(\cdot|ux_{n-\ell(x^n)+1}^n)$ is independent of $u \in \mathcal{A}^*$, $ux_{n-\ell(x^n)+1}^n \in \mathcal{A}_P^*$. Sets of "sibling" leaves $\{b_1 b_2 \dots b_{m-1} b \,|\, b \in A\}$ sharing the same CPMF in the original model can be merged into one state (leaf), represented by the parent node $b_1 b_2 \dots b_{m-1}$ (where (2.3) guarantees compatibility with the CPMF corresponding to the shorter state). The merging is repeated recursively whenever possible, seeking the shortest possible context that determines the CPMF, until any set of α sibling leaves contains at least two leaves with different associated CPMFs. A precise characterization of minimality is given in the more general setting of Section 2.2.

The minimal tree model might not be representable as an FSM model with the same number of states. For example, as noted in [82], in the binary context tree of Figure 2.1, the state following the emission of a 1 at state 0 in T could be either 100 or 101, and more past symbols are required to make the next-state determination than provided by the length-one context (which is nevertheless sufficient to determine the CPMF). The relation between these two classes of models will be the subject of Section 2.3.

2.2 Generalized context trees

In practice, the use of variable-length contexts often yields significant savings in model size compared to a plain Markov model. It is due to these savings that the theory and practice of tree models based on full context trees have received much attention in the literature, and efficient methods for model optimization have been developed (see, e.g., [58, 82, 84, 90]). There might be other opportunities for model size reduction, however, that are difficult to exploit using a full tree. Full tree models, for example, do not provide a mechanism for merging a proper subset of sibling leaves sharing a common CPMF into a single state.[6] In this section we present a more general class of tree models that could exploit some of these

[6] The compression algorithm of [45] leads in some cases to such merges, although the model is not formalized.

additional relations and provide a more economical parametrization of the process. Although this feature makes the general class interesting in itself, our main motivation in discussing it here is its use in auxiliary data structures in Chapter 3.

2.2.1 Terminology and notation

Consider a finite, rooted, ordered, and directed tree T (see, e.g., [37, 38] for tree terminology) with the following properties:

(**i**) Each edge is labeled with a string from $\mathcal{A}^+$.

(**ii**) Each node has one incoming edge, except for the *root* of the tree, which has none. Each node has at most α outgoing edges, which must be labeled with strings starting with different symbols from $\mathcal{A}$.

(**iii**) Each node is labeled with a finite string, obtained by concatenating the labels of the edges on the path from the root to the node. The root is labeled with λ.

For simplicity, we do not distinguish between nodes and their labels, and, for $w \in \mathcal{A}^*$, we use the expression "w is a node of T" as shorthand for "T has a node labeled with w." Furthermore, we identify T with its set of nodes, and we write, for instance, $u \in T$ when u is a node of T. We denote the number of outgoing edges of a node u by $\deg(u)$, and if aw is the label of an edge outgoing from u, we say that this edge is *in the direction* of a. If T has an edge labeled w, going from node u to node v, we write $u \xrightarrow{w} v$, and say that v is a *child* of u, and that u is the *parent* of v, denoted $u = \textsc{par}_T(v)$. The set of children of a node u is denoted $\textsc{chld}_T(u)$. A node v is a *descendant* of u if $u \preceq v$ (u is then an *ancestor* of v). An edge of T is said to be *atomic* if it is labeled with a single-letter string; otherwise it is said to be *composite*. If $u \xrightarrow{a} ua$ is an atomic edge, then ua is an *atomic child* of u. A tree T is atomic if every edge of T is atomic. If T is atomic and every internal node u of T has $\deg(u) = \alpha$, we say that T is *full* (this coincides with our previous definition of a conventional *full* tree). A string w is a *word* of T if it is a prefix of a node of T. The set of words of T will be denoted $\textsc{word}(T)$. Thus, by our convention of identifying the symbol T with its set of nodes, we have $T \subseteq \textsc{word}(T)$, with equality holding if and only if T is atomic.

The combinatorial structure just described has been widely used, under various guises and terminologies, as an underlying data structure for efficient string processing algorithms. The structure (or variants sharing many of its properties) has been referred to as an $\mathcal{A}^+$ *tree* [32], a *PATRICIA* tree [55, 38, 74], a *compact digital tree* [74], etc. It has found numerous applications, for instance, in string storing, searching and retrieval [38, 74], pattern matching [85, 51, 32], and in the mentioned works [44, 23, 3] related to data compression, to list just a few (we cite a few references that contain extensive bibliographies). To emphasize the application of our interest, we will refer to T as a *generalized context tree* (GCT). An example of a GCT over $\mathcal{A} = \{\texttt{a},\texttt{b},\texttt{c}\}$ is shown in Figure 2.2(A).

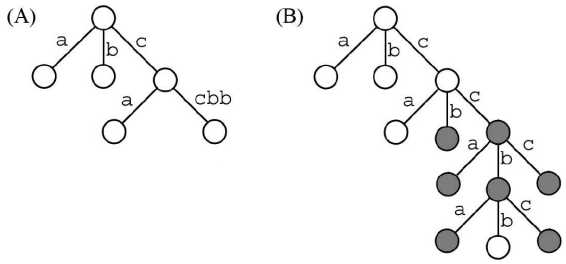

Figure 2.2: *A GCT T over* {a,b,c} *and the corresponding T_{full} (with added nodes in gray)*

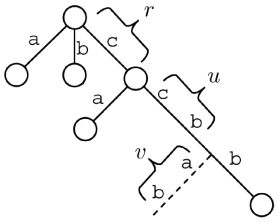

Figure 2.3: *Canonical decomposition of* ccbab $= \langle$c, cb, ab$\rangle$

2.2.2 Source definition

We next describe how a GCT defines the state function of a recursive model that generates a string process. For a GCT T, and an arbitrary string $y \in \mathcal{A}^*$, we define the *canonical decomposition* of y with respect to T as the triplet $C_T(y) = \langle r, u, v\rangle$ such that $r, u, v \in \mathcal{A}^*$, r is the longest prefix of y that is a node of T, ru is the longest prefix of y that is a word of T, and $y = ruv$. The decomposition is illustrated in Figure 2.3 with an example, taken over the GCT T of Figure 2.2(A).

Notice that v is the suffix of y that "falls off" the tree. In general, any, or all, of r, u and v may be null strings. A similar notion of *canonical reference* was defined in [32]. As we will often make separate reference to it, we will denote the first component, r, of $C_T(y)$ by $V_T(y)$.

Let \$ be a symbol such that $\$ \notin \mathcal{A}$. Given a sequence x^n and a GCT T, we define the *tree-state function* $\sigma_T : \mathcal{A}^* \to \mathcal{A}^* \cup \{\, w\$ \,|\, w \in \mathcal{A}^* \,\}$ as follows:

$$\sigma_T(x^n) = \begin{cases} V_T(\overline{x}^n) & \text{if } V_T(\overline{x}^n z) = V_T(\overline{x}^n) \quad \forall z \in \mathcal{A}^*, \\ \overline{x}^n\$ & \text{otherwise.} \end{cases} \tag{2.4}$$

Since T is finite, the first case of (2.4) must hold for sufficiently large n, making $\sigma_T(x^n)$ a node of T. For small values of n, the second case in (2.4) may hold. Thus, viewing σ_T as a state function, its permanent states are given by all the nodes s of T for which there exist arbitrarily long sequences $y^n \in \mathcal{A}^*$ satisfying $V_T(y^n) = s$, whereas its transient states are arbitrary words of T other than leaves, with the symbol \$ appended. By extension, we call these nodes and words, respectively, permanent and transient states of T. A transient state $u\$$ accepts only the single string $\overline{u}$, which is not long enough to "fall off" the tree or reach a

leaf. We denote the set of permanent states of T by S_T, the set of transient states of T by $S_T^{\$}$, and the set of all states of T by $S_T^{\mathrm{A}} = S_T \cup S_T^{\$}$.

As an example, for the GCT shown in Figure 2.2(A), we have $S_T = \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{ca},\mathtt{ccbb}\}$ and $S_T^{\$} = \{\lambda\mathtt{\$},\mathtt{c\$},\mathtt{cc\$},\mathtt{ccb\$}\}$. In this example, $\mathtt{c} \in S_T$ but we still have $\sigma_T(\mathtt{cc}) = \mathtt{cc\$}$, namely, the second case in (2.4) holds. The extra symbol \$ serves to distinguish states that would otherwise correspond to the same string from $\mathcal{A}^*$, e.g., c and c\$ in the example. A natural interpretation of this symbol, which will be more explicitly adopted in Chapter 3, is that of a conceptual marker preceding the first actual symbol of x^n.

The following lemma summarizes the above discussion, characterizing permanent and transient states by giving formal meaning to situations in which a sequence "falls off the tree."

2.1. Lemma. *A string s is a permanent state of a GCT T if and only if $s \in T$ and either $\deg(s) < \alpha$ or s has a composite outgoing edge. A string $w\$$ is a transient state of T if and only if $w \in \mathrm{WORD}(T)$ and w is not a leaf of T.*

When T is a full tree, the set of permanent states is identical to the set of leaves. For the full binary GCT of Figure 2.1, for example, we have $S_T = \{0, 100, 101, 11\}$ and $S_T^{\$} = \{\lambda\$, 1\$, 10\$\}$.

We call *generalized context tree model* (GCTM), and denote with $\langle T, p\rangle$ the recursive model defined by the state set S_T^{A}, the state function $\sigma_T(\cdot)$ of (2.4), and an associated set of CPMFs $\{p(\cdot|s)\}$, $s \in S_T^{\mathrm{A}}$. The probability assignment (2.1) generated by $\langle T, p\rangle$ clearly has finite-memory, with order m upper-bounded by the length of the longest word of T. In order to satisfy also (2.3), it suffices to require that if $s\$$ is a transient state such that all permanent states of the form $V_T(su)$, $u \in \mathcal{A}^*$, share the same CPMF, then this CPMF is also associated with $s\$$.

Remark. Our definitions are quite general in letting arbitrary words define transient states of the GCT, and allowing arbitrary CPMFs to be associated with these states, as long as (2.3) is satisfied. A popular convention is to use for a transient state the CPMF associated with the permanent state that would be selected had the sequence been preceded by as many copies of a fixed symbol as needed [58]. In the context of source coding, another reasonable convention is to assume that transient states are associated with uniform distributions.

Relation to full-tree models. The conventional full-tree models are a special case of GCTMs. For any model $\langle T, p\rangle$, the GCT T can be completed to a full context tree T_{full}, for which there exists a probability assignment p' such that $\langle T, p\rangle$ and $\langle T_{\mathrm{full}}, p'\rangle$ generate the same process, as follows. Let s be a permanent state of T, with associated CPMF $p(\cdot|s)$. Then,

a. if s is a leaf of T, then s is a state (leaf) of $S_{T_{\mathrm{full}}}$ and $p'(\cdot|s) = p(\cdot|s)$;

b. otherwise, for every $a \in \mathcal{A}$ such that s does not have an edge in the direction of a, sa is a state of $S_{T_{\mathrm{full}}}$ and $p'(\cdot|sa) = p(\cdot|s)$;

c. for every composite edge aw emanating from s, with $w = w_1^\ell$, $\ell \geq 1$, all the strings $saw_1^i c_i$, $0 \leq i < \ell$, $c_i \in \mathcal{A} \setminus \{w_{i+1}\}$, are states of $S_{T_{\mathrm{full}}}$, sharing the CPMF $p(\cdot|s)$.

d. $S_{T_{\mathrm{full}}}^{\$} = S_T^{\$}$, and for any $w\$ \in S_T^{\$}$ we have $p'(\cdot|w\$) = p(\cdot|w\$)$.

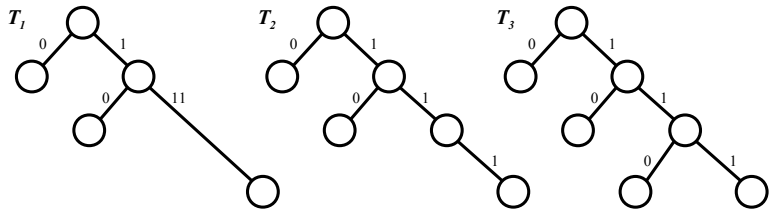

Figure 2.4: *Normalization of a GCT over a binary alphabet*

It is possible, therefore, for S_T to be significantly smaller than $S_{T_{\text{full}}}$, providing a more economical parametrization of the process. In other words, a minimal model in the full-tree sub-class may still be reducible in the GCTM class. Part (B) of Figure 2.2 shows the underlying full tree T_{full} corresponding to the GCT in Part (A) of the same figure. In the example, we have $|S_T| = 5$ and $|S_{T_{\text{full}}}| = 9$. Later on in this section, we characterize minimal representations in the GCTM class. However, the current state of the art in modeling algorithms does not allow us to efficiently optimize code length in this class. Thus, we cannot take advantage of the additional flexibility. The GCT extension will be used in our case as an algorithmic tool for dealing with suffix trees that may not be full, in order to achieve the complexity results of Chapter 3. The code length optimized, however, will still correspond to the sub-class of full-tree models.

2.2.3 Normal generalized context trees

We next present a partition of the set of GCTs into equivalence classes. This partition simplifies the derivation of further results. Given a GCT T, we say that a node $v \in T$ is a *pseudo-leaf* if $\deg(v) \leq 1$ (the case $\deg(v) = 0$ corresponds to a leaf). For example, the node 11 is a pseudo-leaf of T_2 in Figure 2.4. We say that v is a *phantom node* of T if $v \notin T$, and $v = ua$, where $a \in \mathcal{A}$, $u \in T$, and for every $b \in \mathcal{A} \setminus \{a\}$, $ub \in T$. By Lemma 2.1, $u \in S_T$. In Figure 2.4, $v = 11$ is a phantom node of T_1. If we add ua as a node to T (by either adding or splitting an edge), it becomes a pseudo-leaf and a permanent state accepting the same set of strings previously accepted by u, which, again by Lemma 2.1, ceases to be a permanent state. In Figure 2.4, T_2 is obtained from T_1 by adding the phantom node 11 as a node, which becomes a pseudo-leaf in T_2. Since the set of words of the GCT that are not leaves remains unchanged, so does the set of transient states. Thus, the sets of states of the two GCTs are in one-to-one correspondence. Moreover, for a GCTM $\langle T, p\rangle$, if we also associate with the added node ua the CPMF $p(\cdot|u)$, then the new GCTM generates the same process as the original one. Thus, a GCT T with a phantom node ua is indistinguishable, from the point of view of the properties of interest to us, from $T \cup \{ua\}$.

We call the operation of replacing a phantom node of a GCT with the actual node a *normalization* step, and we call a GCT without phantom nodes *normal*. For $\alpha = 2$, normalization might be a two-step process, in that replacing a phantom node with the actual node by splitting a composite edge labeled with a string *of length two*, creates another phantom

node, which in turn needs to be replaced by adding a leaf to the new node. This is illustrated in Figure 2.4, where the addition of the pseudo-leaf 11 in T_2 creates the phantom node 110, which is added to finally obtain the normal GCT T_3. Clearly, this situation does not occur for $\alpha > 2$. One can also take an "unnormalization" step by eliminating a pseudo-leaf from a full set of sibling nodes. Again, in the binary case, this step could create another "unnormalizable" pseudo-leaf. Notice that a full context tree is always normal.

The normalization/unnormalization operations define a partition of the set of all α-ary GCTs into classes, where two GCTs belong to the same class if and only if one can be obtained from the other through a finite sequence of normalization/unnormalization operations. Let $\mathbf{N}(T)$ denote the class of T in this partition. Clearly, $\mathbf{N}(T)$ contains one and only one normal GCT T_N, which we call the *normalized presentation* of T. In Figure 2.4, $\{T_1, T_2, T_3\}$ is a class and T_3 is the normalized presentation of T_i for $i = 1, 2, 3$. The GCT T_N can be obtained from T by replacing each phantom node with an actual node (and, in the binary case, possibly adding leaves as noted, so that no phantom nodes are left). Also, note that $T_N = \bigcup_{T' \in \mathbf{N}(T)} T'$.

2.2.4 Minimal generalized context tree models

A GCTM $\langle T, p \rangle$ is said to be *minimal* if no other GCTM $\langle T', p' \rangle$ generates the same process and has a smaller number of permanent states.[7] To characterize minimality, we start with the conventional sub-class of full-tree models, for which the characterization is simple and well known (see, e.g., [84]). For completeness, we state and show a proof of this characterization in Lemma 2.2 below. We say that a GCT T' is an *extension* of a GCT T if it contains all the nodes of T.

2.2. Lemma. *A full-tree model $\langle T, p \rangle$ with $\mathcal{A}^* = \mathcal{A}_P^*$ is minimal if and only if there is no set of α sibling leaves of T sharing the same CPMF. Moreover, if $\langle T, p \rangle$ is minimal, and $\langle T', p' \rangle$ generates the same process, where T' is also a full context tree, then T' is an extension of T.*

Proof. The necessity of the minimality condition is straightforward, since sets of sibling leaves with identical CPMFs can always be merged, reducing the number of states (constraint (2.3) guarantees that transient CPMFs do not impede the merging). Assume the condition holds, and $\langle T', p' \rangle$ generates the same process as $\langle T, p \rangle$, with T' full. Assume u is a node in $T \setminus T'$. Then, there is a leaf $u' \in T'$ such that $u' \prec u$, and there is a full set of sibling leaves of T that descend from u'. But, since $u' \in S_{T'}$, $\mathcal{A}_P^* = \mathcal{A}^*$, and both tree models generate the same process, these leaves of T must be associated with the same CPMF that is associated with u' in T', contradicting the assumed condition. Thus, we must have $T \subseteq T'$, which also establishes the minimality of T. □

The situation is far more complex for the GCTM class. Since the characterization of minimal GCTMs is not needed for the results in the sections to follow, its discussion and proof are deferred to Appendix A.

[7] While we emphasize the permanent states because they determine the lasting statistics of the source, it can be shown that a minimal GCTM is also minimal in its number of transient states.

2.3 FSM closures of generalized context trees

FSMs and GCTs are combinatorial mechanisms used for process generation, providing the state function σ of a recursive model. For a GCT T, σ is given by the tree-state function σ_T and $s_0 = \lambda\$$, whereas for FSMs, σ is recursively defined by the next-state function, starting from an initial state s_0. The class of FSM models properly includes finite-memory processes (see, e.g., [2]). However, as shown by the example in Figure 2.1, a minimal tree model representation of a finite-memory process might have fewer states than an FSM model representation of the same process. In this section, we study the relation between these two process-generating mechanisms and we define the FSM closure of a GCT. FSM closures of full context trees have already been considered in [69, 88]; here, we target the broader family of GCTs, and we present an efficient algorithm for constructing FSM closures in the broader setting.

2.3.1 Refinements

We now study structural relations between recursive models that generate the same process, and develop tools that will prove useful in investigating the FSM closure of a GCT. Let σ and σ' be state functions taking values in state sets S and S', respectively. We say that σ' is a refinement of σ if there exists a *refinement function* $g : S' \to S$ such that for all sequences x^n, if $\sigma'(x^n) = s'$ and $\sigma(x^n) = s$, then $g(s') = s$. This notion of refinement was presented in [27] for FSMs, and is used also in [81]. We will loosely identify state functions with the mechanisms defining them and say, e.g., that an FSM $\mathcal{F}$ is a refinement of a GCT T.

We now focus on refinement relations between GCTs, using the partition, defined in Section 2.2.3, of the set of all α-ary GCTs into equivalence classes of GCTs sharing a common normalized presentation. Lemma 2.3 below is an obvious consequence of our discussion on normalization.

2.3. Lemma. *If $\mathbf{N}(T^*) = \mathbf{N}(T)$ then there exists a one-to-one refinement mapping between T^* and T.*

Next, we relate the notion of refinement more directly to the combinatorial structure of a GCT.

2.4. Lemma. *Let T and T' be GCTs. T' is a refinement of T if and only if T'_N is an extension of T, where T'_N is the normalized presentation of T'.*

Proof. Assume first that $T \subseteq T'_N$. Consider a sequence of transformations in which we start with T, and we add one node of $T'_N \setminus T$ at a time, until we obtain T'_N. Let T^* denote a generic GCT obtained at an intermediate step of this process. Since $T' \in \mathbf{N}(T'_N)$, by transitivity of the refinement and Lemma 2.3, it suffices to prove that every addition step in this process produces a refinement T^*_r of T^*.

The addition of a node v can be the result of either adding an outgoing edge to a node u of T^* in a direction in which u did not have an edge, or splitting an outgoing composite edge

of u, inserting v. Clearly, in either case, $u \in S_{T^*}$, $\{v\} = S_{T_r^*} \setminus S_{T^*}$, and $S_{T^*} \setminus S_{T_r^*}$ is either $\{u\}$ or empty. Thus, a refinement function g^* is defined such that $g^*(v) = u$ and $g^*(z) = z$ for all $z \in S_{T_r^*} \setminus \{v\}$. As for the transient states, $S^{\$}_{T_r^*} = S^{\$}_{T^*}$ (thus defining an identity mapping), unless v is a leaf of T_r^*, in which case $S^{\$}_{T_r^*} = S^{\$}_{T^*} \cup \{uw\$: uw \prec v, w \in \mathcal{A}^*\}$. Clearly, in the latter case, $g^*(uw\$) = u$. Hence, T'_N is a refinement of T.

Assume now that T' is a refinement of T. Then, by transitivity of the refinement and Lemma 2.3, there exists a refinement function $g : T'_N \to T$. If $T \not\subseteq T'_N$, there exists a node $w \in T \setminus T'_N$. Clearly, $w \neq \lambda$, so we assume $w = ua$ for some $u \in \mathcal{A}^*$, $a \in \mathcal{A}$. Since $ua \notin T'_N$, for some $y \in \mathcal{A}^*$, we have $\sigma_{T'_N}(ya\overline{u}) = u' \preceq u$, and $\sigma_T(ya\overline{u}) = g(u') \succeq ua$. Write $ua = u'bv$, $b \in \mathcal{A}$. We claim that for all $d \in \mathcal{A} \setminus \{b\}$, we must have $u'd \in T'_N$. Otherwise, if $u'd \notin T'_N$, by Lemma 2.1, we would have $\sigma_{T'_N}(zd\overline{u}') = u'$ for some z, and $\sigma_T(zd\overline{u}') \not\succeq ua$, a contradiction to our previous determination of $g(u')$. Now, since T'_N is normal, we must also have $u'b \in T'_N$, for otherwise $u'b$ would be a phantom node of T'_N. Thus, we have a contradiction to the fact that $V_{T'_N}(u'bv) = u'$. Therefore, $T \subseteq T'_N$. □

Remarks

(a) It follows from Lemma 2.4 that the notions of refinement and extension coincide for normal GCTs, and, thus, for all full context trees. Lemma 2.4 also implies that the sufficient condition given in Lemma 2.3 for the existence of a one-to-one refinement mapping between two GCTs is also necessary.

(b) The notions of refinement and minimality were related in [81] for FSM models. An FSM model is minimal if no FSM model with fewer states can generate the same process. It is shown in [81, Lemma 1] that if $\langle \mathcal{F}, p\rangle$ and $\langle \mathcal{F}', p'\rangle$ generate the same process, and $\mathcal{F}$ is minimal, then $\mathcal{F}'$ is a refinement of $\mathcal{F}$. While an analogous result holds for full-tree models (see Lemma 2.2), and for ternary normal GCTMs (see Theorem A.1 in Appendix A), it is interesting to notice that this property does not hold, in general, for GCTMs with $\alpha \neq 3$, as shown by the examples, given in Appendix A, of multiple minimal GCTMs of the same process.[8]

2.3.2 Definition and properties of FSM closures

We say that an FSM $\mathcal{F}$ is an *FSM closure* of a GCT T if it is a refinement of T with a minimal number of permanent states. As in the definition of a minimal GCTM, we adopt the number of permanent states in $\mathcal{F}$ as the relevant measure of minimality. However, it is shown in [48] that, in fact, there exists an FSM closure of T that is *also* minimal in the stronger sense of having a minimal (total) number of states.

We say that a GCT T has the *FSM property* if it defines a next-state function $f : S_T^{\mathrm{A}} \times \mathcal{A} \to$

[8] The underlying GCTs of these multiple minimal GCTMs, however, may be proper refinements of the FSM in a minimal FSM model (not derived from a GCT). This is the case in the example given in Appendix A for $\alpha = 4$, where it is easy to see that the process admits a minimal FSM model with two recurrent states (and one transient state).

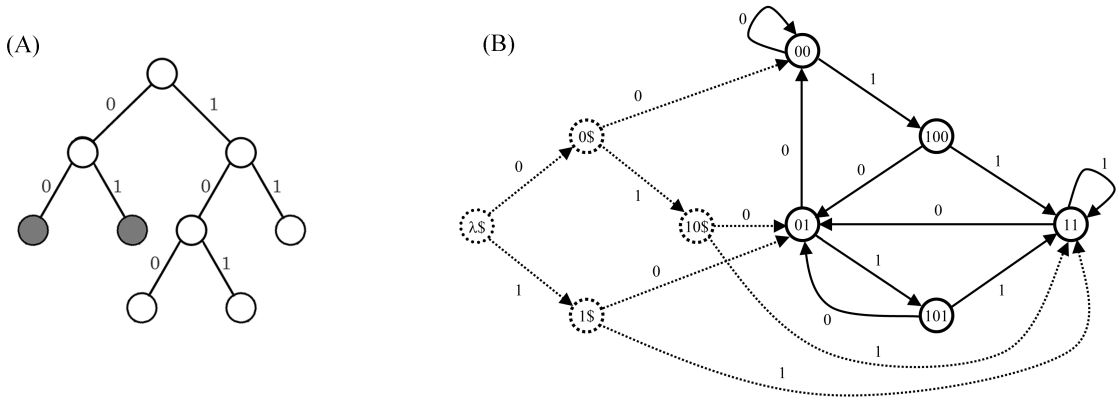

Figure 2.5: *FSM closure T_F of binary GCT T and corresponding finite state machine*

S_T^{A} such that, for any sequence x^{n+1}, we have

$$\sigma_T(x^{n+1}) = f(\sigma_T(x^n), x_{n+1}).$$

For brevity, when T has the FSM property we say that "T is FSM," and we do not distinguish between T and the corresponding FSM. Clearly, if T is FSM then it is also an FSM closure of T. The FSM property facilitates the implementation of GCTMs, due to the recursive form of the next-state function. However, as discussed in Section 2.1 and exemplified in Figure 2.1, a GCT may not be FSM. In such cases, its FSM closure allows for an efficient implementation of state transitions as it was exemplified in Section 1.2.

Figure 2.5(A) shows a GCT T_F with the FSM property that is an FSM closure of the GCT T of Figure 2.1. New nodes added to T are dark. Figure 2.5(B) shows the finite state machine associated with T_F. Transient states and their transitions are shown with dashed lines. Although in this simple example one can derive T_F from T fairly simply, this is not the case in general. Indeed, the number of permanent states of an FSM closure is, in the worst case, quadratic in the number of permanent states of the original GCT [48].

Next, we give a sufficient condition for a GCT to have the FSM property.

2.5. Lemma. *Let T be a GCT. If for every permanent state $s \in S_T$, the suffix* tail(s) *is a node of T, then T is FSM and the next-state function f satisfies, for all $a \in \mathcal{A}$, $f(s,a) = V_T(as)$.*

Proof. We show that a next-state function f can be defined for T. Let $s \in S_T$, and let $w\overline{s}$ be a string accepted by s, $w \in \mathcal{A}^*$. For any $a \in \mathcal{A}$ we have $as\overline{w} \notin \textsc{word}(T)$, for otherwise $as\overline{w}$ is a prefix of a permanent state and, by the assumption of the lemma, we would have $s\overline{w} \in \textsc{word}(T)$ implying that $w\overline{s}$ selects a transient state. Thus, $w\overline{s}a$ selects a permanent state $r = V_T(as\overline{w})$. Clearly, $r \preceq as$, for otherwise $s \prec \mathrm{tail}(r)$ and, by the lemma assumption, $\mathrm{tail}(r) \in T$, implying that $w\overline{s}$ would not have selected s. Therefore, $r = V_T(as)$, and we can define the state transition $f(s,a) = V_T(as)$.

For a transient state $z = u\$$ of T, if $au \in \textsc{word}(T)$ then we define $f(z,a) = az$. Otherwise, au selects the permanent state $V_T(au)$, and we can define $f(z,a) = V_T(au)$. The

next-state function is now defined for all states $s \in S_T^{\mathrm{A}}$, and, hence, T is FSM. □

In order to unambiguously determine the state s selected by a string x^j, given that the previous state is $s' = \sigma_T(x^{j-1})$, and given the symbol x_j, we must be able to guarantee, at least, that $s \preceq \overline{x^j}$. The condition of Lemma 2.5 implies that tail(s) must be a prefix of s'. Hence, since s' is in turn a prefix of $\overline{x^{j-1}}$, the occurrences of x_j in state s' indeed implies that $s \preceq \overline{x^j}$. On the other hand, if tail(s) is not a node of T, s' may be a proper prefix of tail(s). Notice however that if tail(s) is a phantom node of T, this can only happen if tail(s) is a prefix of $\overline{x^{j-1}}$, and we can still determine that $s \preceq \overline{x^j}$ when x_j occurs in s'. Thus, we can not claim, in general, that the condition of Lemma 2.5 is also necessary, except if T is normal. Theorem 2.6 below fully characterizes GCTs having the FSM property.

2.6. THEOREM. *A GCT T with normal presentation T_N is FSM if and only if every suffix of a node of T_N is a node of T_N.*

Proof. By Lemma 2.3, we can assume without loss of generality that T is normal. If every suffix of a node of T is a node of T, then T is FSM by Lemma 2.5. Suppose now that v is a node of T but tail(v) is not. Then, there exists a node u, a symbol a, and a string w, such that $uaw = \mathrm{tail}(v)$, and, since T is normal, there also exists a symbol $b \neq a$, and strings y, z, such that $V_T(uawy) = V_T(ubz) = u$. Now, with $c = \mathrm{head}(v)$, the string $\overline{cuawy} = \overline{vy}$ selects a state s such that $v \preceq s$, and the string $cubz$ selects a state s' that must be different from s. Thus, the occurrence of the symbol c in state u does not uniquely determine the next state in T. □

Define T_{suf} as the GCT obtained from a GCT T by adding, as nodes, all the suffixes of nodes of T. Notice that the addition of a node may cause a composite edge to split. Thus, T_{suf} might contain nodes that are added to satisfy structural constraints of the tree, rather than directly as suffixes of nodes of T. For example, if w is a node of T with an outgoing edge uav, and the construction calls for adding the node $wubv'$, where $a \neq b$, then the edge $w \xrightarrow{uav} wuav$ is split as $w \xrightarrow{u} wu \xrightarrow{av} wuav$, and the new node $wubv'$ is added as a child of wu. The GCTs in Figures 2.1 and 2.5(A) satisfy $T_F = T_{\mathrm{suf}}$. The suffix 00 of state 100 of T is not a node of T, and therefore T does not satisfy the sufficient condition of Lemma 2.5.

By Lemma 2.3 T_{suf} is FSM, and since normalization does not increase the number of permanent states, it follows from Theorem 2.6 that T_{suf} is a refinement of T with the least number of permanent states among all GCTs. It is conceivable, however, that an FSM refinement that is not constrained to having an underlying GCT structure (namely, one that does not correspond to a GCT with the FSM property), might have fewer permanent states. It can be shown though this not to be the case [48].

2.7. THEOREM. *[48] Let T be a GCT. Then, T_{suf} is an FSM closure of T.*

Remarks

(a) It is shown in [48] that the permanent state sets of any two FSM closures of a GCT T are in one-to-one correspondence, which extends to the state sequences followed by any

string. Thus, all FSM closures are essentially equivalent, differing possibly only in the transient states, which are of little interest to us. Therefore, we will henceforth refer to T_{suf} as *the* FSM closure of T.

(b) If T is atomic, then T is FSM if and only if every substring of a node of T_N is a node of T_N, since in such a tree every prefix of a node is a node.

(c) It is readily verified that if T is full, so is T_{suf}.

(d) By Theorem 2.6, for any normal GCT T_F that is an FSM refinement of T we have $T \subseteq T_{\text{suf}} \subseteq T_F$. Thus, if T_{suf} is normal, it is the *only* normal GCT which is an FSM closure of T. This property holds in particular for full context trees.

2.4 A linear-time algorithm for constructing FSM closures

We present an algorithm that constructs the FSM closure T_{suf} of an arbitrary GCT T, together with the associated next-state function. We will prove that the algorithm runs in time that is linear in the sum of the lengths of the strings that label edges of T and in the total number of nodes in T_{suf}.

The algorithm starts with a representation of T, and adds the necessary nodes and edges to construct T_{suf}. At any time during the computation, we denote by T' the intermediate GCT in existence at the time. Thus, T' evolves from T to T_{suf}. When referring to canonical decompositions $C_{T'}$, we mean the decomposition with respect to the instantaneous state of T' at the time of the reference.

The algorithm is presented in the form of a main routine **`MakeFSM`**, and three subroutines, whose functions are broadly described as follows:

- **`Verify(`**w**`)`**: Receives a node w of T' as input, and verifies that the suffix $\text{tail}(w)$ is in T', adding it if necessary together with the FSM transition $f(\text{tail}(w), \text{head}(w)) = w$. The entire (evolving) tree is traversed and verified through recursive calls to this subroutine. Clearly, the condition verified by **`Verify`** is necessary and sufficient (if applied recursively to all the nodes) for the constructed tree to be T_{suf}.

- **`Insert(`**r**`,`**u**`,`**v**`)`**: Receives a node r of T', and strings u, v. Inserts, if necessary, new nodes ru and ruv, doing necessary edge splits and additions.

- **`PropagateTransitions(`**F**`,`**w**`)`**: For a function $F : \mathcal{A} \to T_{\text{suf}}$ adds to the description of the FSM associated with T_{suf} a set of state transitions of the form $f(w, a) = F(a)$, originating from w, for all $a \in \mathcal{A}$ such that $f(w, a)$ was not defined by **`Verify`**.

The routines maintain the following data arrays:

- **`Tail[`**w**`]`**: A pointer from the node in the tree containing w to the node containing $\text{tail}(w)$, which allows the algorithm to jump from w to its suffix in constant time. These *suffix links* [32] are essential to the efficient implementation of the algorithm.

- **`Traversed[`**w**`,`**a**`]`**: A flag indicating whether an attempt was made to traverse an edge starting from node w in the direction of a. Initially set to *false* for all $a \in \mathcal{A}$ for nodes $w \in T$ as well as for new nodes as they are created.

- **`Transitions[`**w**`]`**: A function mapping $\mathcal{A}$ into $T' \cup \{\perp\}$, where $\perp$ denotes an undefined state. The function lists the FSM transitions from state w. The list of transitions is initially set to $\perp$ for all $a \in \mathcal{A}$.

- **`Origin[`**w**`]`**: The original node in T that w descends from. Initially, **`Origin(`**w**`)`**$=w$. This array connects the states of the constructed FSM closure to the original states of T.

- **`Children[`**w**`]`**: The list of children of a node w. Maintained as part of the representation of T'.

The routines are listed in Figure 2.6. We initially omit implementation details, in order to establish functional correctness. Some of the implementation details are essential for analyzing the complexity of the algorithm, and will be provided when we pursue that analysis.

2.8. PROPOSITION. **`MakeFSM`** *constructs* T_{suf}*, and the state transitions between permanent states of the associated FSM.*[9]

Proof. We say that **`Verify`** *visits* a node t of T' whenever the subroutine is invoked with t as its argument. First, we observe that **`Verify`** visits each node at most once. Clearly, the invocation from Step 1 of **`MakeFSM`** (see Figure 2.6) is not repeated. When **`Verify`** is recursively invoked from its Step 16, the edge leading to the visited node is marked as "traversed," and the node is never visited again from that step. Invocations from steps 7 and 9 visit nodes that have just been created in a call to **`Insert`**, and whose incoming edges are already marked as traversed. Therefore, **`Verify`** never revisits a node. Notice also that when new nodes are inserted in the tree (Step 4 of **`Verify`**), the string associated with the new node is shorter than one that already existed in the tree. It follows from the finiteness of the initial tree T that the total number of nodes inserted is finite, and, thus, the recursion sequence of **`Verify`** is finite and **`MakeFSM`** terminates. On the other hand, notice that new nodes that are created are either visited immediately (steps 7 and 9), or their incoming edges were marked as "not traversed." Hence, since the loop in Step 12 recursively traverses all edges outgoing from the current node that had not been traversed (which is done in a conventional pre-order tree traversal recursion), every node of the final tree T' is visited exactly once. We now claim that when the algorithm terminates, $T' = T_{\text{suf}}$. To prove the claim, observe that in Step 1, **`Verify`** extracts the suffix $x = \text{tail}(w)$ of its argument. In Step 2, the canonical decomposition of x is computed, The first component, r, of this decomposition, corresponds to a prefix of x that is already in the tree. Step 4 constructs the parts of x that were missing. Therefore, a call to **`Verify(`**w**`)`** guarantees that $\text{tail}(w)$ will be a node of the constructed tree. Since the algorithm starts with T, and it only adds suffixes of nodes that were already in the tree, every

[9] In addition, also the transitions involving transient states that are nodes of T_{suf} are constructed.

```
MakeFSM
```

```
1.  Verify(λ)
2.  PropagateTransitions({ (a, λ) | a ∈ A }, λ)
```

```
Verify(w)
```

```
1.  Set c = head(w), x = tail(w)
2.  Compute ⟨r, u, v⟩ = C_{T'}(x)
3.  If u ≠ λ or v ≠ λ
4.      Insert(r,u,v)
5.      If u ≠ λ
6.          If Traversed[r, head(u)]
7.              Verify(ru)
8.      Else If v ≠ λ and Traversed[r, head(v)]
9.          Verify(rv)
10. Set Tail[w] = pointer to node x
11. Set Transitions[x](c) = w
12. For a ∈ A
13.     If not Traversed[w, a]
14.         Set Traversed[w,a] = true
15.         If w has an edge az in the direction of a
16.             Verify(waz)
```

```
Insert(r, u, v)
```

```
1.  If u == λ
2.      Add r --v--> rv to T'
3.      Set Origin(rv) = Origin(r)
4.  Else
5.      Split r --uy--> ruy into r --u--> ru --y--> ruy
6.      Set Origin(ru) = Origin(r)
7.      Set Traversed[ru, head(y)] = Traversed[r, head(u)]
8.      If v ≠ λ
9.          Add ru --v--> ruv to T'
10.         Set Origin(ruv) = Origin(ru)
```

```
PropagateTransitions(F,w)
```

```
1.  For a ∈ A
2.      If Transitions[w](a) =⊥
3.          Set Transitions[w](a) = F(a)
4.  For v in Children[w]
5.      PropagateTransitions(Transitions[w], v)
```

Figure 2.6: *Algorithm for computing* T_{suf}

node of T' is either a suffix of a node of T, or a node inserted to allow a bifurcation (e.g., if 001 and 01 are suffixes, a node must exist at 0, even if it is not a suffix). Finally, since all the nodes of T' are visited, every suffix of a node of T is a node of T'. Hence, upon termination of **MakeFSM**, $T' = T_{\text{suf}}$.

Transitions of the FSM associated with T_{suf}, of the form $f(x, c) = cx$, are constructed in Step 11 of **Verify**. Transitions of the form $f(x, c) = u \prec cx$ are added in the subroutine **PropagateTransitions**. Overall, this process exhausts all transitions between permanent states. In addition, also the transitions involving transient states that are nodes of T_{suf} are constructed. □

A key supporting structure generated by the algorithm is the array **Tail** of suffix links, constructed in Step 10. A comment is in place here about the apparent redundancy between Step 1 and Step 10, both of which seem to "compute" the longest proper suffix, x, of w. In Step 1, we read the symbols of x *as a substring of* w, a pointer to which we get as input to **Verify**. In Step 10, after possibly having built it, we have access to a *pointer to the node labeled* x in T'. That pointer is then stored in the array **Tail** for use in later stages of the algorithm. Note that w and x, as nodes, could be located in very different parts of T'.

An example of the workings of the algorithm is presented in Figure 2.7. Figure 2.7(A) (excluding the dashed arrow) shows a non-FSM GCT T over the alphabet $\mathcal{A} = \{\texttt{a},\texttt{b},\texttt{c}\}$. Figures 2.7(A), 2.7(B), and 2.7(C) present the tree T', and the suffix links created, after each iteration of the loop in Step 12 of **Verify**(λ) (namely, one iteration for each child of λ in T). Nodes added to T' in each iteration are light grayed, switching to dark in later iterations. Nothing changes in the first iteration, except for the addition of the suffix link from node `a`, which is verified in this iteration, to the root. In the second iteration, processing the branch in the direction of `b` leads to the verification of nodes `ba`, `baa` and `bac`. The tails of the latter two (`aa` and `ac`, respectively) were not previously in the tree, and are thus added by **Verify** via calls to **Insert**. Since node `a` has already been verified, **Traversed[a,**a**]** is *true* for all $a \in \mathcal{A}$ and **Verify** is recursively called for the inserted nodes. No further insertions are required, and suffix links are defined for the new nodes as shown in Figure 2.7(B). Execution for the branch in the direction of `c` proceeds in a similar way, though in this case, recursive verification of some new nodes leads to further insertions. For instance **Verify**(`cbacb`) leads to successive creations of nodes `bacb`, `acb`, `cb` and `b`, the latter two causing the split of previously traversed edges. Notice how the search for the longest proper suffix of an inserted node during its verification is helped by following the suffix link of its parent. For example, during verification of `bacb`, we can start the search for `acb` directly at node `ac` by following the suffix link from `bac` in Figure 2.7(B).

Complexity analysis. Most individual steps of the algorithm listed in Figure 2.6 can be executed in constant time per node visited, assuming strings associated with edges in the input tree T are efficiently represented, e.g., following the suffix tree methods surveyed in [32], the string v in an edge $u \xrightarrow{v} uv$ of T is defined by a pair of pointers to some memory buffer where the actual symbols are held. Thus, for example, a substring of v can be defined and "copied" somewhere else by manipulating the pointers in constant time. Notice that any new

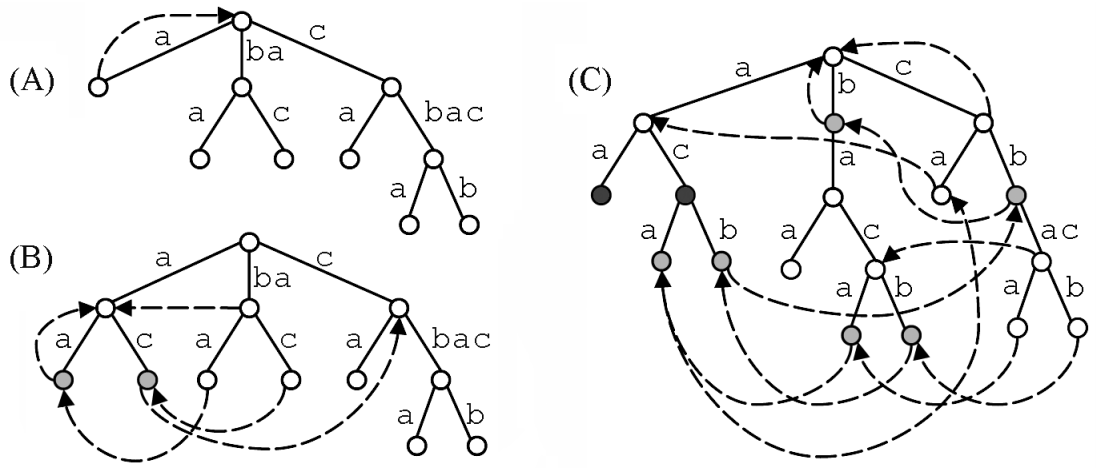

Figure 2.7: *FSM closure (C) and intermediate stages (A,B), with suffix links*

edges inserted by the algorithm are labeled with substrings of previously existing labels, so no additional memory buffer space is needed.

The exception to the statement above is Step 2 of **`Verify`**, namely, the computation of $C_{T'}(x)$. In principle, the step calls for a string comparison seeming to require symbol by symbol access, and time proportional to $|x|$, which could lead, in the worst case, to total execution time quadratic in the total length of strings in the tree. However, the suffix links available at that point in the execution of the algorithm can be used to improve the efficiency of this computation.

The shallowest layer of executions of the loop in Step 12 iterates over the children of the root λ. It is readily verified, by observing Figure 2.6, that except for that layer, every time **`Verify(`**w**`)`** is invoked, w is a node of the form $w = auv$, w's parent is $au \neq \lambda$, and au was previously verified and has a suffix link pointing to node u. Then, we can compute $C_{T'}(uv)$ starting from node u, reading individual symbols of v (which we access by reading the appropriate part of w as a string), and traversing the tree until we find the first prefix of uv that is not a word of T'. When the node being verified is $bv \in \text{CHLD}_{T'}(\lambda)$, we take $u = \lambda$, and proceed in a similar fashion with v. In any case, the number of operations is proportional to $|v|$ rather than $|x|$.

Further savings are possible when the invocation **`Verify(`**ru**`)`** is made from Step 7. For this case, we show that we can count on $x' = \text{tail}(ru)$ being a word of T', and we only need to find its location in the tree to verify whether it is a node, or an edge must be split to create one. To see this, recall that we get to Step 7 after a call to **`Insert`** which split an edge $r \xrightarrow{uy} ruy$. Also, since Traversed$[r, \text{head}(u)]$ must be true, the node ruy must have been visited either from Step 16 of **`Verify`** or from Step 7 or 9 immediately after creation. This guarantees that $\text{tail}(ruy)$ is a node of T' and so $x' = \text{tail}(ru)$ is a word of T'. Now, to find x' in the tree, we can start from node $r' = \text{tail}(r)$, and traverse in the direction of string u, advancing by full edges of the tree, and making comparisons only at the nodes to determine the direction of the next edge. Each time an edge is traversed, we advance in u by the same

number of symbols as the length of the edge, until we exhaust the symbols of u. In this case, therefore, the cost of the computation is proportional to the number of nodes we encounter in the path from r' to $r'u$, rather than the number of symbols $|u|$. We will refer to this case as the *fast mode* of **`Verify`**.

The following theorem bounds the running time of **`MakeFSM`**. Its proof is based on the observations above, and an analysis of the various configurations arising during the recursive sequence of invocations of routine **`Verify`**. The full proof is deferred to Appendix B.

2.9. THEOREM. *Let $N_E = \sum_e |e|$, where the sum is taken over labels e of edges of T, and let $N' = |T_{\text{suf}}|$, the number of nodes in T_{suf}. Then,* **`MakeFSM`** *runs in time $O(N_E + N')$.*

This chapter contains material published in [48].

Chapter 3
Linear-time twice-universal coding of tree sources

In this chapter, we apply the results of Section 2.3 to the implementation, in linear encoding/decoding time, of the semi-predictive approach to twice-universal coding in the class of full-tree models. The GCT extension is used as an algorithmic tool in the implementation. We start by reviewing the semi-predictive approach to twice-universal coding.

3.1 The semi-predictive approach

The basic idea of a two-pass approach to twice-universality for countable union of models is given in [65]. The case in which the second pass implements a sequential code is referred to as *semi-predictive* in [61], and the target class is specialized to full-tree models in [56]. Thus, the twice-universal code presented in this subsection (both for individual sequences and in a probabilistic setting) is known (see, e.g., [90]), and we re-derive it here for completeness.

For an individual sequence x^n, the coding scheme searches, in a first pass through the data, for the full context tree $T(x^n)$ that minimizes the code length

$$\mathcal{L}(T, x^n) = \mathcal{L}_T^{\mathsf{KT}}(x^n) + \mathcal{C}(T) \tag{3.1}$$

over all full context trees T of any size, where $\mathcal{L}_T^{\mathsf{KT}}(x^n)$ denotes the code length assigned by the *Krichevsky-Trofimov* (KT) sequential probability assignment [42] conditioned on the states of T (with a uniform distribution assigned to symbols occurring in transient states), and $\mathcal{C}(T)$ denotes the cost of encoding T using a *natural* code defined, e.g., in [56, 88]. With a natural code, a full tree is encoded with one bit per node, specifying whether the node is a leaf or internal.[1] As is well known for full α-ary trees (see, e.g., [37, p. 595]), $|S_T| = ((\alpha-1)|T|+1)/\alpha$, or, equivalently,

$$\mathcal{C}(T) = |T| = \frac{\alpha|S_T|-1}{\alpha-1}\,. \tag{3.2}$$

To specify $\mathcal{L}_T^{\mathsf{KT}}(x^n)$, let $n_s^{(a)}(x^j)$ denote the number of occurrences of symbol a in context $\overline{s}$ in x^j, namely,

$$n_s^{(a)}(x^j) = \left|\{\, i \,:\, |s| \le i < j,\ x_{i-|s|+1}^{i} = \overline{s},\ x_{i+1} = a \,\}\right|.$$

Here, $\overline{s}$ is an arbitrary string. When s is a permanent state of a full context tree T, $n_s^{(a)}(x^j)$ is the number of occurrences of a in state s. Define also $n_s(x^j) = \sum_{a\in\mathcal{A}} n_s^{(a)}(x^j)$, the number

[1] For $\alpha > 2$, the natural code is a redundant representation of the tree, as it can be seen that, asymptotically in the number of nodes, an efficient representation requires only $h(1/\alpha)$ bits per node, with $h(\cdot)$ denoting binary entropy. The natural code is used here for simplicity, as it does not affect universality or other asymptotic properties of interest.

of times a symbol of x^j occurs in (arbitrary) context $\overline{s}$, i.e., the number of occurrences of context $\overline{s}$ in x^{j-1} (or occurrences of state s, if $s \in S_T$). Then, upon observing x^j, the KT probability assignment takes the form

$$p_{j+1}(x_{j+1} = a|\sigma_T(x^j) = s) = \frac{2n_s^{(a)}(x^j) + 1}{2n_s(x^j) + \alpha}\,.$$

Furthermore, let

$$n(T) = \sum_{s \in S_T} n_s(x^n)\,.$$

Notice that when using the context tree T, $n - n(T)$ is the number of symbols that are coded in a transient state with a uniform distribution. Consequently,

$$\mathcal{L}_T^{\mathsf{KT}}(x^n) = (n - n(T)) \log \alpha + \sum_{s \in S_T} \kappa(x^n, s), \tag{3.3}$$

where $\kappa(x^n, s)$ denotes the code length assigned by the KT probability assignment to the symbols of x^n that occur in state s. By [42], we have

$$\kappa(x^n, s) \stackrel{\triangle}{=} \log \frac{\Gamma\left(n_s(x^n) + \frac{\alpha}{2}\right)\Gamma\left(\frac{1}{2}\right)^{\alpha}}{\Gamma\left(\frac{\alpha}{2}\right)\prod_{a \in \mathcal{A}}\Gamma\left(n_s^{(a)}(x^n) + \frac{1}{2}\right)}\,. \tag{3.4}$$

In a second pass, after finding and describing the optimal context tree $T(x^n)$, the algorithm uses the KT probability assignment to encode the data conditioned on $T(x^n)$. Clearly, upon decoding $T(x^n)$, the decoder can decode the data in a single pass. Due to the properties of the KT probability assignment [42] used at each state, with an arithmetic coder of sufficient precision, this scheme is twice-universal in the sense that, for any sequence x^n, and *any* number M of states, it achieves a per-symbol code length

$$\frac{1}{n}\mathcal{L}(x^n) = \frac{1}{n}\min_T \mathcal{L}(T, x^n) \leq \hat{\mathcal{H}}_M(x^n) + \frac{M(\alpha - 1)}{2}\frac{\log n}{n} + O\left(\frac{M}{n}\right), \tag{3.5}$$

where $\hat{\mathcal{H}}_M(x^n)$ denotes the minimum, over all context trees with M (permanent) states, of the empirical entropy rate of x^n with respect to the context tree. Indeed, by [42], the code length assigned by the KT probability assignment to the symbols of x^n that occur in a state s is,

$$\kappa(x^n, s) \leq n_s \hat{\mathcal{H}}(x^n|s) + \frac{(\alpha - 1)}{2}\log n_s + O(1),$$

where $\hat{\mathcal{H}}(x^n|s)$ denotes the memoryless empirical entropy of the subsequence of x^n formed by the symbols that occur in state s. Thus, assuming that x^n is sufficiently long to select at least one permanent state of an arbitrary context tree T, so that $n(T) > 0$, we have

$$\sum_{s \in S_T} \kappa(x^n, s) \leq n(T) \sum_{s \in S_T} \frac{n_s}{n(T)} \hat{\mathcal{H}}(x^n|s) + \sum_{s \in S_T}\left(\frac{(\alpha - 1)}{2}\log n_s + O(1)\right),$$

Encode(x^n)

1. //First pass:
2. Compute $\mathsf{ST}(\overline{x^{n-1}}\$)$, compact suffix tree of $\overline{x^{n-1}}\$$
3. Compute $T'(x^n)$ s.t. $T'_{\mathtt{full}}(x^n) = T(x^n)$ by pruning $\mathsf{ST}(\overline{x^{n-1}}\$)$ and making edges leading to leaves atomic
4. Encode $T'_{\mathtt{full}}(x^n)$
5. //Second pass:
6. Compute $T'_F(x^n) = T'_{\mathrm{suf}}(x^n)$
7. Set $s = \lambda$
8. For i in $1..n$
9. Encode x_i using statistics located in array Origin[s] of $T'_F(x^n)$
10. Set s = Transitions[s](x_i) using $T'_F(x^n)$

Figure 3.1: *Coding algorithm*

or, denoting by $\hat{\mathcal{H}}_T(x^n)$ the empirical entropy rate defined by x^n for a model with underlying context tree T,

$$\sum_{s \in S_T} \kappa(x^n, s) \leq n\hat{\mathcal{H}}_T(x^n) + \frac{(\alpha-1)}{2}|S_T| \log n + O(|S_T|),$$

which is of course still valid even if $n(T) = 0$. Notice that the number $n - n(T)$ of symbols that are coded in transient states of T is no larger than the depth of T, which, in turn, is at most the number, $(|S_T| - 1)/(\alpha - 1)$, of internal nodes of T. Therefore, by (3.3),

$$\mathcal{L}_T^{\mathsf{KT}}(x^n) \leq n\hat{\mathcal{H}}_T(x^n) + \frac{(\alpha-1)}{2}|S_T| \log n + O(|S_T|).$$

Since also $\mathcal{C}(T) = O(|S_T|)$ by (3.2), we finally get, from (3.1),

$$\mathcal{L}(T, x^n) \leq n\hat{\mathcal{H}}_T(x^n) + \frac{(\alpha-1)}{2}|S_T| \log n + O(|S_T|).$$

The inequality in (3.5) then follows since, in the first pass, the encoder selects the context tree $T = T(x^n)$ that minimizes $\mathcal{L}(T, x^n)$. This establishes the twice-universality in the individual sequence setting. Twice-universality in a probabilistic setting also follows, by taking expectation with respect to the true model in (3.5), and noticing that the expected empirical entropy rate is upper-bounded by the entropy rate by the definition of empirical entropy rate.

3.2 An efficient algorithm: complexity analysis

We first present the linear-time algorithm for the encoding stage. The algorithm is detailed below and summarized in Figure 3.1.

3.2.1 First encoding pass: finding the optimal tree

A procedure for finding the optimal full context tree $T(x^n)$ in linear time is described in [3]. The procedure described below is similar in that it is also based on the application (pioneered in [44]) of suffix tree techniques, and on the tree pruning ideas of [56]. Nevertheless, our procedure is given in terms of the GCT formalism.

First, notice that all the nodes in $T(x^n)$ correspond to strings that actually occurred as substrings of $\overline{x}^n$, except for those leaves that are added to complete a full tree, for otherwise $\mathcal{C}(T(x^n))$ could have been made shorter without affecting $\mathcal{L}^{\mathsf{KT}}_{T(x^n)}(x^n)$. Let $T'(x^n)$ denote the GCT that is obtained from $T(x^n)$ by deleting all leaves that *did not* occur as substrings of $\overline{x}^n$, as well as any node u such that $\deg(u) = 1$ after deleting those leaves, except if $\overline{u} \preceq x^n$. This exception guarantees that all transient states emitting symbols of x^n take the form $u\$$, where $u \in T'(x^n)$.[2] Moreover, even though internal nodes of $T'(x^n)$ may now become permanent states, the only permanent states that will emit symbols of x^n are leaves. Thus, for the purpose of coding x^n, $T'(x^n)$ is equivalent to $T(x^n)$. Clearly, we have $T'_{\text{full}}(x^n) = T(x^n)$.

Now, consider the *compact suffix tree* (see, e.g., [32]) $\mathsf{ST}(\overline{x^{n-1}}\$)$ of $\overline{x^{n-1}}\$$ where, as introduced in Section 2.1, \$ denotes a special symbol that is conceptually assumed to precede x_1. The leaves of $\mathsf{ST}(\overline{x^{n-1}}\$)$ are given by all strings of the form $\overline{x^j}\$$, $0 \le j < n$, and v is an internal node of $\mathsf{ST}(\overline{x^{n-1}}\$)$ if and only if there exist two different symbols $a, b \in \mathcal{A} \cup \{\$\}$ such that both $a\overline{v}$ and $b\overline{v}$ are sub-strings of $\$x^{n-1}$ (thus, $\deg(v) > 1$). The last symbol of the string labeling the incoming edge of any leaf of $\mathsf{ST}(\overline{x^{n-1}}\$)$ is \$ (on the other hand, $T(x^n)$ is an α-ary tree, and \$ is just a symbol appended to its transient states). The use of this symbol in $\mathsf{ST}(\overline{x^{n-1}}\$)$ guarantees that the mentioned nodes u of $T'(x^n)$ with $\deg(u) = 1$ belong also to $\mathsf{ST}(\overline{x^{n-1}}\$)$. Thus, by the definition of $\mathsf{ST}(\overline{x^{n-1}}\$)$ and $T'(x^n)$, all internal nodes of $T'(x^n)$ are also internal nodes of $\mathsf{ST}(\overline{x^{n-1}}\$)$. Moreover, the leaves of $T'(x^n)$ are words of $\mathsf{ST}(\overline{x^{n-1}}\$)$, but notice that since the incoming edges corresponding to these leaves must be atomic (for otherwise the description of $T(x^n)$ could be shortened without affecting $\mathcal{L}^{\mathsf{KT}}_{T(x^n)}(x^n)$), it may be the case that a leaf of $T'(x^n)$ is not a node of $\mathsf{ST}(\overline{x^{n-1}}\$)$. Thus, one can obtain $T'(x^n)$ by "pruning" $\mathsf{ST}(\overline{x^{n-1}}\$)$ and possibly shortening incoming edges of the resulting leaves to make them atomic (Step 3 in Figure 3.1).

The algorithm that derives $T'(x^n)$ by pruning $\mathsf{ST}(\overline{x^{n-1}}\$)$ is based on the observation that, by recursively assigning costs to sub-trees, an optimal context tree consists of optimal sub-trees, and can be obtained by dynamic programming. This observation was first made in [56] and is used also in [3]. It should be noticed, however, that the formulation in [56] is simplified by the fact that the context tree to be pruned has bounded depth and is atomic. In our case, to assign the costs consider a given GCT T' with all incoming edges of leaves atomic, and a sequence x^n such that for all j, $0 \le j < n$, $\sigma_{T'}(x^j) = \sigma_{T'_{\text{full}}}(x^j)$, and $\overline{x^j} \in T'$ whenever $\overline{x^j} \in \text{WORD}(T')$ (as observed, only sequences emitted from leaves of T', and transient states, are relevant to the discussion). Let u be a node of T', and let u' be the prefix of u of length

[2] We recall that, in full generality, transient states correspond to all the words of the GCT except for the leaves, and not just to those words corresponding to nodes.

$|u'| = |\text{PAR}_{T'}(u)| + 1$ (or $u' = \lambda$ if $u = \lambda$). We associate to u a natural code cost defined by

$$\mathcal{C}_u(T') = \begin{cases} \alpha(|u| - |u'|) + 1 + (\alpha - \deg(u)) & \text{if } u \text{ is internal,} \\ 1 & \text{if } u \text{ is a leaf.} \end{cases} \tag{3.6}$$

The term $\alpha(|u| - |u'|)$ gives the number of nodes of T'_{full} in the set

$$\{va : u' \preceq v \prec u, a \in \{\lambda\} \cup \mathcal{A}\} \setminus \{u\},$$

i.e., nodes that are descendants of u' but not of u. These nodes are added to T'_{full} in place of the possibly composite edge $\text{PAR}_{T'}(u) \longrightarrow u$ to make T'_{full} a full tree. Notice that, since all incoming edges of leaves are atomic in T', $\alpha(|u| - |u'|)$ is zero when u is a leaf. When u is an internal node, the term $(\alpha - \deg(u))$ additionally gives the number of atomic children ua of u in T'_{full} that are not words of T', i.e., the number of leaves added in T'_{full} as children of u to make T'_{full} a full tree. Thus, the sum of $\mathcal{C}_u(T')$ over all nodes u of T' gives the number of nodes in T'_{full}. Hence, by (3.2),

$$\mathcal{C}(T'_{\text{full}}) = \sum_{u \in T'} \mathcal{C}_u(T'). \tag{3.7}$$

We associate to the sub-tree rooted at u the cost $K_{T'}(u)$ recursively defined by

$$K_{T'}(u) = \mathcal{C}_u(T') + \begin{cases} \delta_u \log \alpha + \sum_{w \in \text{CHLD}_{T'}(u)} K_{T'}(w) & \text{if } u \text{ is internal,} \\ \kappa(x^n, u) & \text{if } u \text{ is a leaf,} \end{cases} \tag{3.8}$$

where $\delta_u = 1$ if $\overline{u} \preceq x$, and $\delta_u = 0$ otherwise. Since all transient states selected by x^j in T' as j runs through $0 \le j < n$ are nodes of T', the sum of the term $\delta_u \log \alpha$ along all internal nodes of T' accounts for all symbols coded in transient states. Thus, by (3.1), (3.3), and (3.7) we have $K_{T'}(\lambda) = \mathcal{L}(T'_{\text{full}}, x^n)$. Thus, Equation (3.8) can be used as the basis of a dynamic programming minimization procedure. Specifically, we associate to each node u of the compact suffix tree $\mathsf{ST}(\overline{x^{n-1}}\$)$ a cost defined by

$$K(u) = 1 + \log \alpha\,, \quad \text{if } u \text{ is a leaf,} \tag{3.9}$$

and, for an internal node u,

$$K(u) = \min \left(\alpha(|u| - |u'| + 1) + 1 + \sum_{w \in \text{CHLD}_{\mathsf{ST}(\overline{x^{n-1}}\$)}(u)} (K(w) - 1)\,,\ 1 + \kappa(x^n, u) \right). \tag{3.10}$$

Recall that in $T'(x^n)$, the compacted version of the optimal context tree $T(x^n)$, all incoming edges to leaves are atomic, for otherwise the description of $T(x^n)$ could be shortened without affecting $\mathcal{L}^{\mathsf{KT}}_{T(x^n)}(x^n)$. Hence, for a leaf u of $\mathsf{ST}(\overline{x^{n-1}}\$)$ with $u = vaw\$$, $v = \text{PAR}_{\mathsf{ST}(\overline{x^{n-1}}\$)}(u)$, $a \in \mathcal{A}$, and $|w| > 0$, the edge $v \xrightarrow{aw} u$ must be shortened in the process of pruning $\mathsf{ST}(\overline{x^{n-1}}\$)$ to get $T'(x^n)$, making $u' = va$ a leaf. Thus, (3.9) is consistent with (3.8) since $\mathcal{C}_{u'}(T') = 1$, and (3.4) yields $\kappa(x^n, u') = \kappa(x^n, u) = \log \alpha$ for only one symbol occurs in context $\overline{u'}$ by the

definition of $\mathsf{ST}(\overline{x^{n-1}}\$)$. For each internal node u of $\mathsf{ST}(\overline{x^{n-1}}\$)$, we take in (3.10) the minimum of the sum of the costs of the optimal sub-trees rooted at all its children, and the cost of making u' a leaf, where again we let u' be the prefix of u of length $|u'| = |\mathrm{PAR}_{\mathsf{ST}(\overline{x^{n-1}}\$)}(u)| + 1$ (or $u' = \lambda$ if $u = \lambda$). Notice that the term $\delta_u \log \alpha$ of (3.8) is incorporated into the summation over all children in (3.10) since, due to the use of the special symbol \$ in $\mathsf{ST}(\overline{x^{n-1}}\$)$, we have $u\$ \in \mathrm{CHLD}_{\mathsf{ST}(\overline{x^{n-1}}\$)}(u)$ in case $\overline{u} \preceq x$ (therefore, u may have up to $\alpha + 1$ children). Also notice that the subtraction of 1 from $K(w)$ in the summation accounts for the term $-\deg(u)$ in (3.6).

A dynamic programming algorithm that minimizes $K(\lambda)$, thus minimizing $\mathcal{L}(T'_{\mathrm{full}}, x^n)$, is now straightforward from (3.9) and (3.10). Equivalently, we can minimize $K' = K - 1$, which takes the simpler form

$$K'(u) = \log \alpha\,, \quad \text{if } u \text{ is a leaf,} \tag{3.11}$$

and, for an internal node u,

$$K'(u) = \min \left(\alpha(|u| - |u'| + 1) + \sum_{w \in \mathrm{CHLD}_{\mathsf{ST}(\overline{x^{n-1}}\$)}(u)} K'(w)\,,\ \kappa(x^n, u) \right). \tag{3.12}$$

Specifically, in a post-order traversal [37] of $\mathsf{ST}(\overline{x^{n-1}}\$)$, we shorten incoming composite edges of leaves of $\mathsf{ST}(\overline{x^{n-1}}\$)$, and, for each internal node u, we evaluate (3.12) making u' a leaf in case the minimum is achieved by the second argument. The information required for the pruning decisions at each node u consists of the children costs $K'(w)$, which have already been computed at the time of visiting u, and by (3.4), the set of counts $\{n_u^{(a)}(x^n)\}_{a \in \mathcal{A}}$. Clearly, these counts are obtained recursively as the sum of the corresponding counts over all children of u, and can be collected in the same post-order traversal of $\mathsf{ST}(\overline{x^{n-1}}\$)$. The recursion starts from the leaves $u\$$ of $\mathsf{ST}(\overline{x^{n-1}}\$)$, for which the symbol a that follows $\$\overline{u}$ in x^n can be recorded during the suffix tree construction and associated to the leaf.

Summarizing, the computational cost of finding $T'(x^n)$, which for the purpose of coding x is equivalent to $T(x^n)$, is given by the cost of the following operations:

(**i**) Building the (compact) suffix tree $\mathsf{ST}(\overline{x^{n-1}}\$)$, and some associated data structures;

(**ii**) Pruning $\mathsf{ST}(\overline{x^{n-1}}\$)$ in a post-order traversal, computing the costs $K'(u)$ for all nodes u, and possible inserting new nodes as leaves of $T'(x^n)$.

It is well known (see, e.g., [32]) that the computational cost of building $\mathsf{ST}(\overline{x^{n-1}}\$)$ is $O(n)$. The adaptation of the generic suffix tree algorithms to building also some additional *ad hoc* structures (e.g., associating an emitted symbol with each leaf of the tree) is straightforward and does not affect the complexity. It is shown in [3, Theorem 2] that the computation of each $\kappa(x^n, u)$ can be performed in registers of size $O(\log n)$ in a constant number of operations, and that this precision is sufficient for preserving the validity of (3.5). Since, by definition, $\mathsf{ST}(\overline{x^{n-1}}\$)$ has n leaves, it has $O(n)$ nodes when represented as a compact tree. Thus, the whole pruning process takes $O(n)$ operations, since the insertion of additional nodes as possible leaves of $T'(x^n)$ clearly does not affect the linearity.

3.2.2 Second pass: sequential encoding

After encoding $T(x^n)$ with a natural code (which can be specified recursively with a pre-order traversal of the tree [37]), the encoder makes a second pass through the data that involves, for each j, finding $\sigma_{T(x^n)}(x^j)$ and arithmetic encoding x_{j+1} using the corresponding KT probability assignment. As observed in [3, Corollary 1], even though $|T(x^n)|$ may be significantly larger than $|T'(x^n)|$ (since $\mathsf{ST}(\overline{x^{n-1}}\$)$ is a compact tree), it is still $O(n)$, for otherwise $T(x^n)$ would not have emerged as the optimal context tree in (3.1) (think, e.g., of the context tree $\{\lambda\}$, for which $\mathcal{C}(\{\lambda\}) = 1$ and $\mathcal{L}^{\mathsf{KT}}_{\{\lambda\}}(x^n) < n \log \alpha + o(n)$). Therefore, $T(x^n)$ can be described in linear time. As for arithmetic coding once $\sigma_{T(x^n)}(x^j)$ is determined, it is shown in [82] that, again, performing a constant number of arithmetic operations per symbol in registers of size $O(\log n)$ guarantees a precision that will not affect the validity of (3.5). Thus, we focus on the determination of the state.

In [3], the BWT of $\$x^n$ facilitates the transition between states in constant time. Alternatively, notice that the algorithms that construct $\mathsf{ST}(\overline{x^{n-1}}\$)$ in linear time can also maintain pointers (so-called *suffix links*, see, e.g., [32]) between each leaf $au\$$ of $\mathsf{ST}(\overline{x^{n-1}}\$)$, and the leaf $u\$$, as suffixes are inserted in length order. If each leaf $u\$$ is in turn linked to the corresponding state $\sigma_{T(x^n)}(\overline{u})$, then each state transition can be done in constant time. Clearly, these links can be created with an additional traversal of $\mathsf{ST}(\overline{x^{n-1}}\$)$ in linear time. These methods, however, require either the BWT of $\$x^n$, or the suffix tree $\mathsf{ST}(\overline{x^{n-1}}\$)$, none of which are, in principle, available to the decoder. Thus, we propose an alternative linear-time method, based on the FSM closure of $T'(x^n)$, that can be employed also at the decoding side.

Specifically, before starting the second pass, the encoder builds an FSM closure $T'_F(x^n)$ of $T'(x^n)$ (without loss of generality, $T'_{\mathrm{suf}}(x^n)$), using the algorithm **`MakeFSM`** of Section 2.4.[3] For every permanent state w of $T'_F(x^n)$, and every symbol $c \in \mathcal{A}$, the encoder then has access to the next-state transition $f(w, c)$ via the mapping **`Transitions[`**w**`]`** constructed during the execution of **`MakeFSM`** (see Section 2.4). This mapping also provides the state transitions for all transient states that are associated with nodes. Since, by the definition of $T'(x^n)$, all the transient states that are actually visited with x^n are indeed associated with nodes, it follows that, starting from the root (which is used as the context of x_1), the encoder can make each transition between states of $T'_F(x^n)$ in constant time. In addition, the link **`Origin[`**w**`]** provides access to the state of $T'(x^n)$ that is being refined by w, which accumulates the relevant statistics for the KT probability assignment (loop starting at Step 8 in Figure 3.1). These statistics are possibly shared with other states of $T'_F(x^n)$. The following corollary to Theorem 2.9 establishes the linear time complexity of the proposed encoder.

3.1. Corollary. *`MakeFSM`*$(T'(x^n))$ *runs in time* $O(n)$.

Proof. By Theorem 2.9, it suffices to prove that both the sum of the lengths of the strings that label edges of $T'(x^n)$ and the number of nodes of $T'_F(x^n)$ are $O(n)$. The former is clearly

[3] Notice that while only states associated with leaves actually occur in the sequence of permanent states of $T'(x^n)$ determined by x^n, this is no longer the case with $T'_F(x^n)$, for which we take full advantage of the GCT formalism.

upper-bounded by $|T(x^n)|$, which was already observed to be $O(n)$ at the beginning of Section 3.2.2. As for $T'_F(x^n)$, let $T''(x^n)$ denote the GCT obtained by deleting from $T'(x^n)$ those nodes that are not in $\mathsf{ST}(\overline{x^{n-1}}\$)$ (and the corresponding incoming edges). By the definition of a (compact) suffix tree, $u \in \mathsf{ST}(\overline{x^{n-1}}\$)$ if and only if either u is a suffix of $\overline{x^{n-1}}\$$, or there exist $a, b \in \mathcal{A} \cup \{\$\}$, $a \neq b$, such that both ua and ub are sub-strings of $\overline{x^{n-1}}\$$. Thus, every suffix of a node of $\mathsf{ST}(\overline{x^{n-1}}\$)$ is also a node of $\mathsf{ST}(\overline{x^{n-1}}\$)$. Since $T''(x^n) \subseteq \mathsf{ST}(\overline{x^{n-1}}\$)$, and $T''_F(x^n)$ is formed by adding as nodes all the suffixes of the nodes of $T''(x^n)$, it follows that the added nodes are in $\mathsf{ST}(\overline{x^{n-1}}\$)$, and therefore we also have $T''_F(x^n) \subseteq \mathsf{ST}(\overline{x^{n-1}}\$)$. Consequently, $|T''_F(x^n)| = O(n)$. Now, in addition to the nodes of $T''_F(x^n)$, $T'_F(x^n)$ includes all the suffixes of the nodes of $T'(x^n)$ that are not in $\mathsf{ST}(\overline{x^{n-1}}\$)$. As observed in the description of the pruning step in Section 3.2.1, these nodes can only be leaves of $T'(x^n)$, and all corresponding incoming edges must have length one. Thus, these leaves take the form wa, where $w \in T''(x^n)$ (and, hence, $w \in T''_F(x^n)$) and $a \in \mathcal{A}$. Thus, the corresponding suffixes take the form va, where $v \in T''_F(x^n)$, so that the number of additional nodes of $T'_F(x^n)$ cannot be larger than $\alpha|T''_F(x^n)| = O(n)$. □

3.2.3 Decoding

If the decoder had access to $T'(x^n)$ as it starts scanning the compressed bit-stream, the decoding stage could proceed in rather the same way as the encoding. That is, the decoder could build an FSM closure of $T'(x^n)$, $T'_F(x^n)$, and proceed to decode the symbols of x^n sequentially, using the statistics in $T'(x^n)$ and efficiently transiting between states by means of the FSM $T'_F(x^n)$. However, only $T(x^n)$ has been described and, as shown in [48], its FSM closure might, in principle, have a super-linear number of nodes. Of course, a modified encoder can describe $T'(x^n)$ (by simply specifying, for every node of $T(x^n)$, whether it is also a node of $T'(x^n)$), without affecting the validity of (3.5). This modified code is still twice-universal, and a linear-time implementation of the decoder follows trivially from reversing some of the operations at the encoder. However, it is not necessary to penalize the code length to preserve linear time complexity. Next, we present a decoder that has access to $T(x^n)$ only, but requires a more elaborate analysis.

Assume $T(x^n) \neq \{\lambda\}$ (for otherwise $T'(x^n)$ would also be known), and let $\hat{T}'(x^n)$ denote the GCT obtained from $T(x^n)$ by deleting all the leaves, as well as nodes whose outgoing degree after deleting the leaves is one. Clearly, $\hat{T}'_{\text{full}}(x^n) \subseteq T(x^n)$, and $|\hat{T}'(x^n)| \leq |T'|$ for any T' such that $T'_{\text{full}} = T(x^n)$. Thus, $\hat{T}'(x^n) \subseteq T'(x^n)$. The decoder starts by building an FSM closure $\hat{T}'_F(x^n)$ of $\hat{T}'(x^n)$, which, by the proof of Corollary 3.1, can be done in linear time (again, we assume $\hat{T}'_F(x^n) = \hat{T}'_{\text{suf}}(x^n)$). Then, the key idea is to relate, for every i, $0 \leq i < n$, the state $\hat{s}_i \stackrel{\triangle}{=} \sigma_{\hat{T}'_F(x^n)}(x^i)$ to the state selected in the FSM closure of $T(x^n)$, $s_i \stackrel{\triangle}{=} \sigma_{T_F(x^n)}(x^i)$ (again, $T_F(x^n) = T_{\text{suf}}(x^n)$), which is the state needed by the decoder, and to show that the linkage between the two states can be executed without compromising the overall linear complexity. Figure 3.2 illustrates some of the decoding operations. Figure 3.2(A) shows a context tree $T(x^n)$, and the leaves (grayed) and internal nodes (dark) deleted to obtain $\hat{T}'(x^n)$. The latter GCT is shown in Figure 3.2(B), which also illustrates the relation between

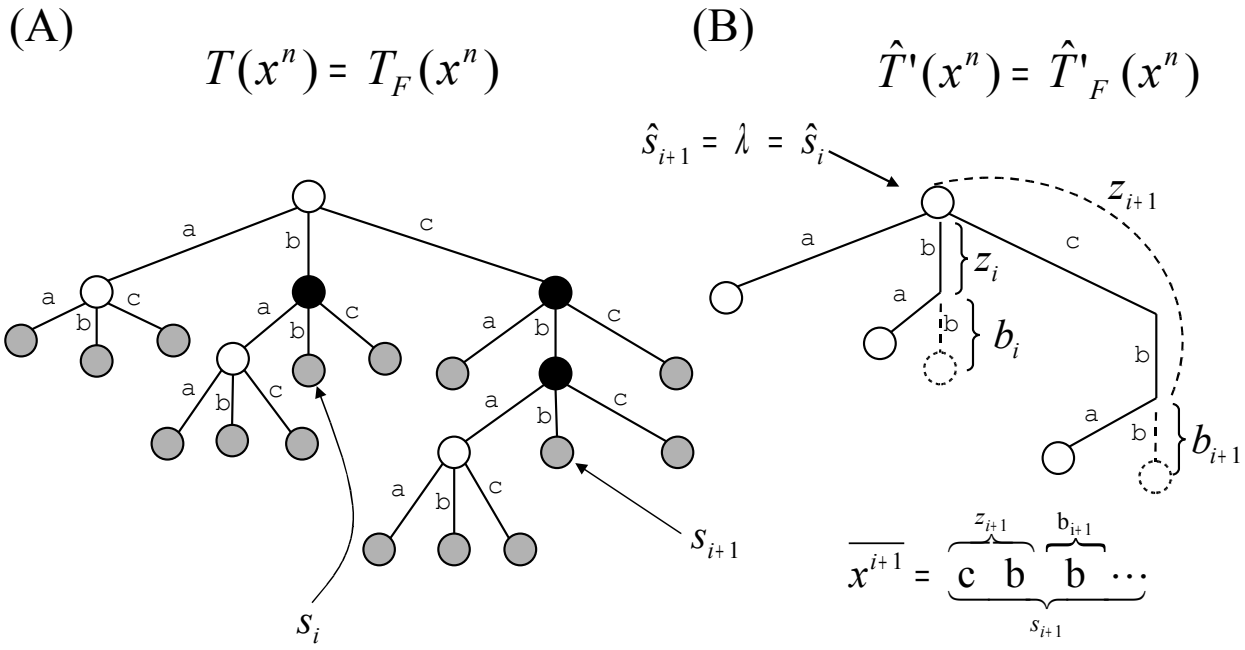

Figure 3.2: *Decoding tree*

$\hat{s}_i$ and s_i. In this particular case, both $T(x^n)$ and $\hat{T}'(x^n)$ are FSM.

We start by establishing a key connection between $\hat{T}'_F(x^n)$ and $T_F(x^n)$.

3.2. Lemma. *The following relations hold for $\hat{T}'_F(x^n)$ and $T_F(x^n)$.*

(i) *If $u \in \hat{T}'_F(x^n)$ then $ua \in T_F(x^n)$ for all $a \in \mathcal{A}$.*

(ii) *If $u \in \text{WORD}(\hat{T}'_F(x^n))$ and $ua \notin \text{WORD}(\hat{T}'_F(x^n))$ for some $a \in \mathcal{A}$, then ua is a leaf of $T_F(x^n)$.*

Proof. If $u \in \hat{T}'_F(x^n)$, by construction of the closure, there exists a string v such that $vu \in \hat{T}'(x^n)$ and therefore $vua \in T(x^n)$ for all $a \in \mathcal{A}$, thus $ua \in T_F(x^n)$ for all $a \in \mathcal{A}$.

If $u \in \text{WORD}(\hat{T}'_F(x^n))$ then there exists a string y such that $uy \in \hat{T}'_F(x^n)$ and therefore $uya \in T_F(x^n)$. Since $T(x^n)$ is full, so is $T_F(x^n)$ and $uya \in T_F(x^n)$ implies $ua \in T_F(x^n)$. Let $v \in \mathcal{A}^*$ $uav \in T_F(x^n)$. By construction of the closure there exists $w \in \mathcal{A}^*$ such that $wuav \in T(x^n)$. If $v \neq \lambda$, $wua \in \text{WORD}(\hat{T}'(x^n))$, $wuav' \in \hat{T}'(x^n)$ for some v' and $uav' \in \hat{T}'_F(x^n)$, a contradiction. □

The relation between $\hat{s}_i$ and s_i is given by Lemma 3.3 below, for which we remove the symbols \$ from transient states, and define z_i such that $\hat{s}_i z_i$ is the longest prefix of $\overline{x^i}$ in $\text{WORD}(\hat{T}'_F(x))$. Furthermore, define $b_i \stackrel{\triangle}{=} x_{i-|\hat{s}_i z_i|}$ in case $|\hat{s}_i z_i| < i$, or $b_i = \lambda$ otherwise (these definitions are illustrated in Figure 3.2).

3.3. Lemma. *For every i, $0 \leq i < n$, we have $s_i = \hat{s}_i z_i b_i$.*

Proof. Since $T_F(x^n)$ is a full tree, it suffices to show that $\hat{s}_i z_i b_i \in \text{WORD}(T_F(x))$, and that either $b_i = \lambda$, or $\hat{s}_i z_i b_i c \notin \text{WORD}(T_F(x))$ for any $c \in \mathcal{A}$. To prove the first claim, observe that, by definition, $\hat{s}_i z_i \in \text{WORD}(\hat{T}'_F(x))$, so that there exists a string y such that $\hat{s}_i z_i y \in \hat{T}'_F(x^n)$. The claim follows from Lemma 3.2 by which $\hat{s}_i z_i y a \in T_F(x^n)$ for every $a \in \mathcal{A}$, and since $T_F(x^n)$ is full, $\hat{s}_i z_i b \in T_F(x^n)$ for every $b \in \mathcal{A}$. As for the second claim, assume that $\hat{s}_i z_i b_i c \in \text{WORD}(T_F(x))$ for some $c \in \mathcal{A}$. Since $T_F(x^n)$ is full, this assumption implies that $\hat{s}_i z_i b_i c \in T_F(x^n)$ for all $c \in \mathcal{A}$, so that there exists a string v for which $v\hat{s}_i z_i b_i c \in T(x^n)$. Thus, $v\hat{s}_i z_i b_i \in \text{WORD}(\hat{T}'(x^n))$, implying that $v\hat{s}_i z_i b_i w \in \hat{T}'(x^n)$ for some string w, and furthermore that $\hat{s}_i z_i b_i w \in \hat{T}'_F(x^n)$. Since, by definition, $\hat{s}_i z_i$ is the longest prefix of $\overline{x^i}$ that is a word of $\hat{T}'_F(x^n)$, we must then have $b_i = \lambda$. □

Next, we show that, in fact, it is not necessary to revisit the decoded sequence for all the symbols in z_i in order to determine s_i. Observe that, by the FSM property of $\hat{T}'_F(x^n)$, $\hat{s}_{i+1} \preceq x_{i+1}\hat{s}_i$. Removing again the symbols \$ from transient states, define u_{i+1} to be the string satisfying $x_{i+1}\hat{s}_i = \hat{s}_{i+1}u_{i+1}$.

3.4. LEMMA. *If* $\hat{s}_{i+1}u_{i+1}\text{head}(z_i) \in \text{WORD}(\hat{T}'_F(x))$, *then* $z_{i+1} = u_{i+1}z_i$.

Proof. By the definition of z_i and u_i, there exist sequences t and v such that

$$\overline{x^{i+1}} = x_{i+1}\hat{s}_i z_i b_i t = \hat{s}_{i+1} z_{i+1} v = \hat{s}_{i+1} u_{i+1} z_i b_i t\,. \tag{3.13}$$

Furthermore, by the assumption of the lemma and the definition of z_{i+1}, we have $u_{i+1}\text{head}(z_i) \preceq z_{i+1}$; we show that we also have $z_{i+1} \preceq u_{i+1}z_i$. Otherwise, by (3.13) and the definition of z_{i+1}, $\hat{s}_{i+1}u_{i+1}z_i b_i$ would be a word of $\hat{T}'_F(x^n)$ with $b_i \neq \lambda$, implying the existence of a string y such that $\hat{s}_{i+1}u_{i+1}z_i b_i y \in \hat{T}'_F(x^n)$. Thus, since $\hat{T}'_F(x^n)$ is FSM, $\hat{s}_i z_i b_i y$ is also a node of $\hat{T}'_F(x^n)$ by (3.13), so that $\hat{s}_i z_i b_i \in \text{WORD}(\hat{T}'_F(x))$, in contradiction with the definition of z_i. Now, if $z_i = \lambda$, the proof is complete. If $z_i \neq \lambda$, then we have $z_{i+1} = u_{i+1}z'_i$, where z'_i is a non-empty prefix of z_i; define z''_i by $z_i = z'_i z''_i$. The proof is complete if we show that $z''_i = \lambda$. Since $\hat{s}_{i+1}z_{i+1} \in \text{WORD}(\hat{T}'_F(x))$ by the definition of z_{i+1}, there exists a string w such that $\hat{s}_{i+1}u_{i+1}z'_i w$ is a node of $\hat{T}'_F(x^n)$ and, by the FSM property of $\hat{T}'_F(x^n)$, so is $\hat{s}_i z'_i w$. Moreover, $w \neq \lambda$, for otherwise $\hat{s}_i z'_i$ would be a node of $\hat{T}'_F(x^n)$, and thus $\hat{s}_i$ would not be the state at time i. If $z''_i \neq \lambda$, we have $\text{head}(w) \neq \text{head}(z''_i)$, for otherwise $\hat{s}_{i+1}z_{i+1}\text{head}(z''_i) \in \text{WORD}(\hat{T}'_F(x))$, contradicting the definition of z_{i+1}. It follows that $\hat{s}_i z'_i \text{head}(w)$ and $\hat{s}_i z'_i \text{head}(z''_i)$ are two different words in $\hat{T}'_F(x^n)$, making $\hat{s}_i z'_i$ a node, a contradiction. □

Given $\hat{s}_i$, $|u_i|$, and $|z_{i-1}|$ (starting with $z_0 = \lambda$), we can recursively determine $|z_i|$ by checking decoded symbols and descending $\hat{T}'_F(x^n)$, starting from $\hat{s}_i$, in the direction $x_{i-|\hat{s}_i|}$, $x_{i-|\hat{s}_i|-1}$, $\cdots, x_{i-|\hat{s}_i|-|u_i|-1}$. If, at some point, the concatenated string is not a word of $\hat{T}'_F(x^n)$, by definition, we have determined $|z_i|$. Otherwise, $\hat{s}_i u_i \text{head}(z_{i-1}) \in \text{WORD}(\hat{T}'_F(x))$, and, by Lemma 3.4, $|z_i| = |u_i| + |z_{i-1}|$. Thus, the determination of $|z_i|$ requires at most $|u_i| + 1$ comparisons. Since $|u_i| = |\hat{s}_{i-1}| - |\hat{s}_i| + 1$, we need at most n comparisons along x^n.

Now, given $\hat{s}_i$, $|z_i|$, $\text{head}(z_i)$, and b_i, it is easy to determine $\sigma_{T(x^n)}(x^i)$ (which contains the decoding statistics) in constant time per input symbol by defining an additional data

structure. Specifically, for every internal node u of $T(x^n)$ that is also a node of $\hat{T}'_F(x^n)$, consider the set A_u of symbols for which u has an edge of $\hat{T}'(x^n)$ in their direction, and let $u(a)$ denote the edge of $\hat{T}'_F(x^n)$ in the direction of $a \in A_u$. For every j, $1 \leq j < |u(a)|$, let $v_{u,a}(j)$ denote the node of $T(x^n)$ obtained by concatenating j symbols of $u(a)$ to u. A data structure **Jump[u]**, linking u with each $v_{u,a}(j)$, can be built in $O(|T(x^n)|)$ total time for all relevant nodes, e.g., by initially setting up the data structure for the nodes of $\hat{T}'(x^n)$, and then updating it as edges of $\hat{T}'(x^n)$ are split by **MakeFSM**. In addition, for a node w of $\hat{T}'_F(x^n)$ that is not a node of $T(x^n)$, the initialization of **Origin[w]** in **MakeFSM** can readily be modified so that it points to its ancestor in $T(x^n)$, rather to an ancestor in $\hat{T}'(x^n)$. Equipped with the data structures **Jump[u]**, and based on Lemmas 3.3 and 3.4, we can determine $\sigma_{T(x^n)}(x^i)$ for $z_i \neq \lambda$ as follows:

- If $\hat{s}_i$ is an internal node of $T(x^n)$ and $\text{head}(z_i) \in A_{\hat{s}_i}$, then $\hat{s}_i z_i$ is given by $v_{\hat{s}_i,\text{head}(z_i)}(|z_i|)$ (by definition, $|z_i| < |\hat{s}_i(\text{head}(z_i))|$), and $\sigma_{T(x^n)}(x^i) = s_i = \hat{s}_i z_i b_i$ (Lemma 3.3).
- If $\hat{s}_i$ is an internal node of $T(x^n)$ and $\text{head}(z_i) \notin A_{\hat{s}_i}$, then $\sigma_{T(x^n)}(x^i) = \hat{s}_i \text{head}(z_i)$.
- If $\hat{s}_i$ is a leaf of $T(x^n)$, then $\sigma_{T(x^n)}(x^i) = \hat{s}_i$.
- If $\hat{s}_i$ is not a node of $T(x^n)$, then **Origin[$\hat{s}_i$]** points to $\sigma_{T(x^n)}(x^i)$.

When $z_i = \lambda$, $\sigma_{T(x^n)}(x^i) = \hat{s}_i$ if $\hat{s}_i$ is a node of $T(x^n)$, and **Origin[$\hat{s}_i$]** points to $\sigma_{T(x^n)}(x^i)$ otherwise.

The implementation of the semi-predictive approach to Context algorithm outlined in this section is denoted **SPContextFSM**. The following theorem summarizes our discussion.

3.5. THEOREM. *SPContextFSM encodes and decodes any sequence x^n in time $O(n)$.*

Remarks

(a) The **SPContextFSM** decoder does not explicitly obtain $T'(x^n)$, the pruned GCT given by the set of nodes of $T(x^n)$ that actually occurred as substrings of $\overline{x}^n$. Since, after decoding x^n, the decoder can determine $T'(x^n)$, a plausible approach to linear-time decoding would be to obtain $T'_F(x^n)$ adaptively, starting from $\hat{T}'_F(x^n)$ and adding the missing nodes as new words of $T(x^n)$ are decoded. Thus, $\hat{T}'_F(x^n)$ would grow on the fly "as needed." We show in Section C.1 of Appendix C that such a procedure can indeed be implemented in linear time without recourse to additional complex data structures (e.g., **Jump[u]**). However, the description of **SPContextFSM** is simpler.

(b) **SPContextFSM** is presented as an application of the concept of FSM closure, solving the open problem of linear-time decoding. In Section C.2 of Appendix C, we present yet another solution to this problem, this time without recourse to the FSM closure. This approach will typically require more storage space than **SPContextFSM** at the decoder. The idea in this case is to extend $T(x^n)$ on the fly with the suffix tree of the string decoded so far. To this end, the suffix tree of $\overline{x^{i-1}}\$$ is built in an "anti-sequential" manner as in [23], i.e., the suffix tree of $\overline{x^j}\$$ is available at each step j, $0 < j < i$.

This chapter contains material published in [49].

Chapter 4

Type classes of tree models

In this chapter, we characterize type classes of tree models. We recall that two strings, x^n and y^n, belong to the same type class with respect to a tree model $\langle T, p_T \rangle$, if and only if they are assigned equal probabilities regardless of the model parameter, i.e., for every value of the conditional probability mass functions $p_T(\cdot|\cdot)$ associated to the states of T (with appropriate conventions for the initial states of the process). It is not difficult to see that this condition holds if and only if x^n and y^n have the same symbol counts $n_s^{(a)}$ (see Section 3.1), for all states s of T, and all symbols $a \in \mathcal{A}$. Thus, in the case of tree models, as in other cases of interest, the notion of type defined in probabilistic terms admits a combinatorial characterization. Notice that this characterization depends on the underlying context tree T, but not on the model parameter.

We investigate both the size of type classes and the number of type classes induced by a context tree. To obtain the exact size of a type class, we follow a graph theoretic approach similar to the derivation of Whittle's formula for FSMs in [34], establishing a one-to-one correspondence between the set of strings in a type class and the set of Eulerian circuits in a certain graph derived from the context tree. For general context trees, however, the problem turns out to be far more involved than the FSM case, due to the lack of a next-state function.

An important property of type classes for FSM models is that their size behaves asymptotically as $\exp(n\hat{\mathcal{H}}_{\mathcal{F}}(x^n))$ for every sequence x^n, where $\hat{\mathcal{H}}_{\mathcal{F}}(x^n)$ is the empirical entropy rate of x^n with respect to the given FSM, $\mathcal{F}$, underlying the model. This property does not extend, in general, to tree models. For example, the empirical entropy rate of the sequence $x^n = 001\,001 \cdots 001$ with respect to the context tree T_1 of Figure 4.1 is $\hat{\mathcal{H}}_{T_1}(x^n) = \frac{2}{3}h(\frac{1}{2})$ where h is the binary entropy function. The factor $h(\frac{1}{2})$ in $\hat{\mathcal{H}}_{T_1}(x^n)$ arises from state 0, where half of the occurring symbols are 0 and half are 1 (we write $h(\frac{1}{2})$ unevaluated to emphasize this fact). On the other hand, in order to reach state $s_i = 100$, we must have $s_{i-2} = s_{i-1} = 0$. Therefore, to preserve conditional counts, the state sequence must follow the fixed cycle $0{\to}0{\to}100{\to}0{\to}0{\to}100{\to}\cdots$, and the type class of x^n is just $\{x^n\}$. Although this is an extreme example, restrictions on state transitions such as the one in this example may, in general, rule out many of the state sequences that could be obtained by picking the next symbol at each state freely, according to the prescribed counts. Such freedom is already limited to some extent in the FSM case, as expressed by the cofactor that multiplies the multinomial factors in Whittle's formula [86]. This cofactor, however, is polynomial in n, and thus negligible with respect to the main factor of the formula. As shown in the example, the reduction may be far more significant in the case of tree models, where the restrictions are more intricate.

In Section 4.2 we derive the mentioned exact formula for the size of the type class of x^n with respect to a given context tree, which generalizes Whittle's formula. As mentioned,

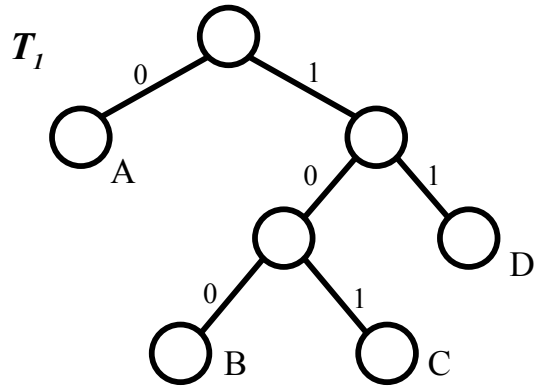

Figure 4.1: *A context tree T_1 over $\mathcal{A} = \{0, 1\}$*

the problem is also reduced in our case to one of counting Eulerian circuits in a directed graph. Although the constructed graph contains some of the elements of the FSM closure, it is different from it, and, in fact, it generally does not correspond to a conventional FSM (in general, the type class of a sequence with respect to the FSM closure of a context tree is smaller than the type class with respect to the context tree). The difficulty in the construction stems precisely from the lack of a next-state function, and the loss of context occurring in the state transition sequences of context trees.

In Section 4.3 we present an asymptotic upper bound for the expected logarithm of the size of the type class of a string with respect to a tree model $\langle T, p_T \rangle$. Later, in Chapter 5, we will use a coding argument to show that this bound is tight, in the sense that this upper bound differs from a lower bound by $O(1)$.

In Section 4.4, we study the number of type classes for sequences of length n induced by a given context tree, and we estimate the number of classes tightly, up to a multiplicative constant. In this case also, we generalize the known result for the FSM case, which is stated in [83] and attributed to N. Alon.

Finally, in Section 4.5, we compare, for a tree model $\langle T, p_T \rangle$, the average size of the type classes with respect to T and to its FSM closure T_{suf}, and the number of type classes induced by both context trees. We define the (FSM) *over-refinement* of T, κ_T, which depends exclusively on the structure of the context tree T. When comparing the expected logarithm of the size of the type classes in T and T_{suf} of a random sequence, it turns out that the terms of order $\log n$ differ by a factor of κ_T. We also show that there is, asymptotically, a factor of n^{κ_T} more type classes in T_{suf} than in T.

The characterization of the type classes of tree models enables applications of these models in enumerative coding and universal simulation that we explore later in Chapter 5 and Chapter 6, respectively.

4.1 Preliminaries

We denote by $\mathcal{I}(T)$ the set of internal nodes of a tree T, and by $\mathsf{depth}(T)$ the depth of T, i.e., $\mathsf{depth}(T) = \max\{|u| : u \in T\}$. In this chapter, except when explicitly stated, we consider a fixed context tree T. Thus, we omit the dependence on T from most of the notation we introduce next. We will also assume that T is nontrivial, i.e., $|T| > 1$. Notice that for $|T| = 1$ the problem reduces to a memoryless setting, which is well understood [15]. For simplicity we fix an *initial state* s_0 of maximal length, i.e., a leaf of maximal depth in T. Thus, a sequence x^n determines a state sequence $s_0(x^n)$, $s_1(x^n)$, $\cdots$, $s_n(x^n)$, defined as $s_i(x^n) = \sigma_T(\overline{s_0}x^i)$. Notice that, by imposing maximal length on s_0, and regarding x^n as preceded by $\overline{s_0}$, we can guarantee that all states $s_i(x^n)$ in the sequence are permanent. We then refer to the permanent states of T simply as *the* states of T. In Appendix E we extend the results of this chapter to the more general setting in which the initial conditions are determined by transient states of the context tree. In this case, the exact formula for the size of a type class is slightly modified, although the asymptotic behavior of the expected size of a type class and the number of type classes remain the same.

We omit the argument x^n of s_i, and of other objects of the form $f(x^n)$, when clear from the context. A state s_i is itself a string; we use the notation $(s_i)_j^k$ when indexing its symbols. For a tree model $\langle T, p_T\rangle$ the set of CPMFs associated to the states of T define a probability assignment $\mathrm{P}_{\langle T,p_T\rangle}(\cdot)$ via (2.1). Specifically,

$$\mathrm{P}_{\langle T,p_T\rangle}(\lambda) = 1; \quad \mathrm{P}_{\langle T,p_T\rangle}(x^n) = \prod_{i=1}^{n} p_T(x_i|s_{i-1}), \quad n \geq 1\,. \tag{4.1}$$

For each $n \geq 0$, the assignment (4.1) defines a probability distribution on $\mathcal{A}^n$. To avoid confusion, we reserve the notation $\mathrm{P}_{\langle T,p_T\rangle}\{\cdot\}$ to refer to the probability of an event that depends on a random sequence $X^n \in \mathcal{A}^n$ drawn with probability $\mathrm{P}_{\langle T,p_T\rangle}(\cdot)$, where n will be clear from the context. For example we write $\mathrm{P}_{\langle T,p_T\rangle}\left\{n_s^{(a)} > 0\right\}$ to denote the probability $\sum_{x^n : n_s^{(a)}(x^n)>0} \mathrm{P}_{\langle T,p_T\rangle}(x^n)$. Throughout we use capital letters to denote a random sequence emitted by a tree source, e.g. X^n, and lowercase letters to denote specific sequences, e.g. x^n. We denote by $\mathrm{E}_{\langle T,p_T\rangle}[\,\cdot\,]$ the expectation of a random variable with respect to $\mathrm{P}_{\langle T,p_T\rangle}(\cdot)$. From (4.1), we see that the type class of x^n with respect to T, denoted $\mathcal{T}(T, x^n)$ or simply $\mathcal{T}(x^n)$ when T is clear from the context, takes the form

$$\mathcal{T}(T, x^n) \;=\; \left\{ y^n \in \mathcal{A}^n : n_s^{(a)}(y^n) \,=\, n_s^{(a)}(x^n)\ \forall s \in S_T, a \in \mathcal{A} \right\},$$

where, with a slight abuse on the notation of Chapter 3, here we let $n_s^{(a)}(x^n)$ denote the number of occurrences of a in state s in x^n, regarding x^n as preceded by $\overline{s_0}$ for the purpose of selecting states. We refer to $\mathcal{T}(T, x^n)$ as the *T-class* of x^n, as a shorthand for the type class of x^n with respect to T.

The *state transition matrix* of x^n, denoted $N(x^n)$, is an $|S_T|\times|S_T|$ matrix, with row and column index set S_T, and entries

$$N_{s,t} = |\{i : 1 \leq i \leq n, s_{i-1} = s, s_i = t\}|, \quad s, t \in S_T,$$

namely, $N_{s,t}$ is the number of transitions from s to t in the state sequence of x^n. For a matrix M we define $M_{i*} = \sum_j M_{i,j}$ and $M_{*j} = \sum_i M_{i,j}$. Thus, N_{s*} is n_s, the number of symbols in x^n that occur in state s, and N_{*s} is the total number of times s is transitioned into in the state sequence of x^n.

Since in the state sequence of any given string, x^n, every state is entered as many times as it is exited, with the only exception of the initial and final states, then N satisfies the *flow conservation equations*, i.e.,

$$N_{*s}(x^n) + \delta_{s,s_0(x^n)} = N_{s*}(x^n) + \delta_{s,s_n(x^n)}\,, \quad \text{for all } s \in S_T\,,$$

where we use the Kronecker delta notation: $\delta_{u,v} = 1$ when $u = v$, and zero otherwise. The following lemma characterizes the support of $N(x^n)$.

4.1. Lemma. *Let s, t be arbitrary states of T. There exists a string x^n such that $N_{s,t}(x^n) > 0$ if and only if $s \preceq \mathrm{tail}(t)$ or $\mathrm{tail}(t) \prec s$.*

Proof. Consider two consecutive states $s = s_{i-1}(x^n)$, $t = s_i(x^n)$ in the state sequence of an arbitrary string x^n. Then, we have $t \preceq \overline{x^i} s_0 = x_i \overline{x^{i-1}} s_0$, and, therefore, $\mathrm{tail}(t) \preceq \overline{x^{i-1}} s_0$. Since also $s \preceq \overline{x^{i-1}} s_0$, we must have $s \preceq \mathrm{tail}(t)$ or $\mathrm{tail}(t) \prec s$. Thus, if $N_{s,t}(x^n)$ is nonzero, then s and t must satisfy $s \preceq \mathrm{tail}(t)$ or $\mathrm{tail}(t) \prec s$.

Suppose now that $s \preceq \mathrm{tail}(t)$ and let x^n be a string such that $t \prec \overline{x^n}$. Then, we have $s_n = t$ and, also, $s_{n-1} = s$, since $s \preceq \mathrm{tail}(t)$ and $\mathrm{tail}(t) \prec \overline{x^{n-1}}$. Thus, we have $N_{s,t}(x^n) > 0$. If, on the other hand, $\mathrm{tail}(t) \prec s$, then we take x^n such that $\mathrm{head}(t)s \prec \overline{x^n}$. Since $\mathrm{tail}(t) \prec s$, we have $t \prec \overline{x^n}$ and, hence, $s_n = t$. By the definition of x^n we also have $s_{n-1} = s$ and therefore $N_{s,t}(x^n) > 0$. □

Objects N_{FS}, $(n_{\mathrm{FS}})_s^{(a)}$, and $\mathcal{T}_{\mathrm{FS}}$ can be defined for arbitrary FSMs, in analogy to the definitions of N, $n_s^{(a)}$, and $\mathcal{T}$ for context trees above. In this case, the existence of a next-state function implies directly that counting sequences in $\mathcal{T}_{\mathrm{FS}}(x^n)$ is equivalent to counting state sequences[1] with the same initial state and transition counts as x^n. This property is essential in the various derivations of Whittle's formula for FSMs [86, 6, 34, 35]. Later, we will show that a correspondence between sequences in $\mathcal{T}(x^n)$ and state transition counts exists also for context trees. This connection can be stated in a simpler form if we do it for *close-ended type classes*, where the close-ended type class of x^n with respect to a context tree T, or simply the *T-class** of x^n, is defined as

$$\mathcal{T}^*(T, x^n) = \{\, y^n \in \mathcal{T}(T, x^n) : s_n(x^n) = s_n(y^n) \} \,.$$

[1] Special care must be taken when different symbols lead to the same state transition, which makes the two counts differ in a binomial coefficient factor. For instance, suppose that a, b are two (and the only two) different symbols such that an FSM in state s transitions to state t when either a or b occur. Given the state sequence of x^n, there are $\binom{(n_{\mathrm{FS}})_s^{(a)} + (n_{\mathrm{FS}})_s^{(b)}}{(n_{\mathrm{FS}})_s^{(a)}}$ ways of labeling the transitions $s \to t$ with $(n_{\mathrm{FS}})_s^{(a)}$ symbols a and $(n_{\mathrm{FS}})_s^{(b)}$ symbols b, all of which give rise to strings in the same type class. Since both counts $(n_{\mathrm{FS}})_s^{(a)}$ and $(n_{\mathrm{FS}})_s^{(b)}$ are known given the type class, this brings no difficulty in computing the size of the class. Notice that in context trees, whether FSM or not, different symbols always lead to different states, and this case does not arise.

Notice that the above equality condition on the final states is always satisfied for sequences in the same type class in FSMs, by the flow conservation equations and the existence of a next-state function. Next, we argue that we incur no loss of generality if we confine our attention to close-ended type classes. We start by showing the following relation between the final states of any two sequences in the same T-class.

4.2. LEMMA. *Let T be a context tree and let x^n, y^n be sequences in the same T-class. If $s_{n-1}(x^n)$ and x_n determine $s_n(x^n)$, then we have $s_n(x^n) = s_n(y^n)$. Otherwise, we must have $s_{n-1}(x^n) = s_{n-1}(y^n)$, and $x_n = y_n$.*

Proof. We first show that if $s_{n-1}(x^n)$ and x_n determine $s_n(x^n)$, i.e., if $\mathrm{tail}(s_n(x^n)) \in T$, then $s_n(y^n) = s_n(x^n)$. Consider a state $s = au$, with $a \in \mathcal{A}$, $u \in \mathcal{A}^*$, such that $u = \mathrm{tail}(s) \in T$. If in the state sequence of an arbitrary string z we have $s_i = s = au$, then $u \preceq \overline{z^{i-1}} s_0$. Thus, the symbol $z_i = a$ is emitted from a state s_{i-1} of the form uv. Conversely, if symbol a is emitted from a state s_{i-1} of the form uv, then $au \preceq \overline{z^i} s_0$, and therefore $s_i = au = s$. Hence,

$$N_{*au}(x^n) = \sum_{uv \in S_T} n_{uv}^{(a)}(x^n) = \sum_{uv \in S_T} n_{uv}^{(a)}(y^n) = N_{*au}(y^n)\,.$$

Also, by the flow conservation equations on the transition matrices, we have

$$\begin{aligned} N_{*au}(x^n) &= N_{au*}(x^n) + \delta_{au,s_n(x^n)} - \delta_{au,s_0} \\ N_{*au}(y^n) &= N_{au*}(y^n) + \delta_{au,s_n(y^n)} - \delta_{au,s_0}\,. \end{aligned}$$

Since the total number of symbol emitted from au in x^n and y^n is the same, $N_{au*}(x^n) = N_{au*}(y^n)$, and therefore $\delta_{au,s_n(x^n)} = \delta_{au,s_n(y^n)}$, which proves the claim.

As a consequence of the claim above, if $s_n(x^n) \neq s_n(y^n)$, we must have $s_n(x^n) = auv$ with $a \in \mathcal{A}$, $u \in S_T$, and $v \in \mathcal{A}^+$. Now, if in the state sequence of an arbitrary string z we have $s_i = auw$ for some $w \in \mathcal{A}^+$, then $u \preceq \overline{z^{i-1}} s_0$. Thus, the symbol $z_i = a$ is emitted from the state u. Conversely, if symbol a is emitted from state $s_{i-1} = u$, then $au \preceq \overline{z^i} s_0$, and therefore s_i must have the form auw with $w \in \mathcal{A}^+$. Hence,

$$\begin{aligned} n_u^{(a)}(x^n) &= \sum_{auw \in S_T} N_{*auw}(x^n) \\ &= \sum_{auw \in S_T} N_{auw*}(x^n) + \delta_{auw,s_n(x^n)} - \delta_{auw,s_0}\,, \end{aligned}$$

and also,

$$n_u^{(a)}(y^n) = \sum_{auw \in S_T} N_{auw*}(y^n) + \delta_{auw,s_n(y^n)} - \delta_{auw,s_0}\,.$$

Since $n_u^{(a)}(y^n) = n_u^{(a)}(x^n)$, and the total number of symbols emitted from auw in x^n and y^n is the same, we have,

$$\sum_{auw \in S_T} \delta_{auw,s_n(y^n)} = \sum_{auw \in S_T} \delta_{auw,s_n(x^n)} = 1\,.$$

We conclude that the final state of y^n has the form auw. Thus, $s_{n-1}(x^n) = s_{n-1}(y^n) = u$, and also $x_n = y_n = a$, as claimed. □

The following corollary to Lemma 4.2 connects $\mathcal{T}^*(x^n)$ with $\mathcal{T}(x^n)$.

4.3. Corollary. *For a context tree T and a sequence x^n, we have $\mathcal{T}(x^n) = \mathcal{T}^*(x^n)$ if $s_{n-1}(x^n)$ and x_n determine $s_n(x^n)$, and $\mathcal{T}(x^n) = \{y^{n-1}x_n : y^{n-1} \in \mathcal{T}^*(x^{n-1})\}$ otherwise.*

Proof. If $s_{n-1}(x^n)$ and x_n determine $s_n(x^n)$, then $\mathcal{T}(x^n) = \mathcal{T}^*(x^n)$ by Lemma 4.2 and the definition of $\mathcal{T}^*(x^n)$. Suppose now that $s_{n-1}(x^n)$ and x_n do not determine $s_n(x^n)$, and consider the subsequence x^{n-1}. For all states s and all symbols a, we have

$$n_s^{(a)}(x^{n-1}) = n_s^{(a)}(x^n) - \delta_{s,s_{n-1}(x^n)}\delta_{a,x_n}\,.$$

Similarly, for an arbitrary string $z^n \in \mathcal{T}(x^n)$, we have

$$n_s^{(a)}(z^{n-1}) = n_s^{(a)}(z^n) - \delta_{s,s_{n-1}(z^n)}\delta_{a,z_n}\,.$$

Since we must have $s_{n-1}(z^n) = s_{n-1}(x^n)$ and $z_n = x_n$ by Lemma 4.2, we conclude that $z^{n-1} \in \mathcal{T}^*(x^{n-1})$. Thus, the string z^n belongs to the set $\{y^{n-1}x_n : y^{n-1} \in \mathcal{T}^*(x^{n-1})\}$. Conversely, if we append the symbol x_n to the end of a string $y^{n-1} \in \mathcal{T}^*(x^{n-1})$, we get

$$n_s^{(a)}(y^{n-1}x_n) = n_s^{(a)}(y^{n-1}) + \delta_{s,s_{n-1}(y^{n-1})}\delta_{a,x_n}, \quad \text{for all } s \in S_T, a \in \mathcal{A}\,.$$

Since $n_s^{(a)}(y^{n-1}) = n_s^{(a)}(x^{n-1})$ and $s_{n-1}(y^{n-1}) = s_{n-1}(x^{n-1})$ by the definition of $\mathcal{T}^*(x^{n-1})$, we conclude that $n_s^{(a)}(y^{n-1}x_n) = n_s^{(a)}(x^n)$ for all states s and all symbols a, and, therefore, $y^{n-1}x_n \in \mathcal{T}(x^n)$. □

Based on Corollary 4.3, we now show that if an enumeration scheme[2] for T-classes* is available, we can use it to implement an enumeration scheme for T-classes. Suppose we are given an enumeration scheme for T-classes* of any length and let $g^*(x^n)$ be the index assigned to x^n within $\mathcal{T}^*(x^n)$ by this scheme. We can associate to each sequence x^n an index within $\mathcal{T}(x^n)$, $g(x^n)$, as follows: if $\mathrm{tail}(s_n(x^n)) \in T$, we take $g(x^n) = g^*(x^n)$, otherwise, we take $g(x^n) = g^*(x^{n-1})$. Notice that, by Corollary 4.3, in the former case we have $\mathcal{T}(x^n) = \mathcal{T}^*(x^n)$ since s_n is determined by $s_{n-1}(x^n)$ and x_n. In the latter case, s_n is not determined by $s_{n-1}(x^n)$ and x_n, and, again by Corollary 4.3, $\mathcal{T}(x^n)$ is obtained from $\mathcal{T}^*(x^{n-1})$ by appending the symbol x_n to the end of each sequence of $\mathcal{T}^*(x^{n-1})$. Therefore, in this case, $\mathcal{T}(x^n)$ and $\mathcal{T}^*(x^{n-1})$ are the same size. Thus, the scheme above assigns indexes in the range $0 \cdots |\mathcal{T}(x^n)| - 1$ and covers all integers in this interval as x^n varies in a type class. The following lemma shows that the problem of enumerating sequences in $\mathcal{T}(x^n)$ is solved by the above construction of $g(x^n)$ as soon as we know how to enumerate sequences in $\mathcal{T}^*(x^n)$.

[2]By enumeration we mean any one-to-one mapping between sequences in $\mathcal{T}^*(x^n)$ and indexes in the range $0 \cdots |\mathcal{T}^*(x^n)| - 1$. We will be interested in algorithmic implementations of the mapping with time and memory complexity that are polynomial in n (the size of $\mathcal{T}^*(x^n)$ is in general exponential in n).

4.4. Lemma. *Given the T-class of x^n together with the index $g(x^n)$, we can recover x^n by computing a single evaluation of the inverse function of g^* and, additionally, performing $O(1)$ operations on registers of length $O(n)$.*

Proof. We will give a sequence of $O(1)$ operations required to determine the argument for an invocation of the inverse function of g^* and using this invocation to obtain x^n.

If the (as yet unknown) final state of x^n is determined by $s_{n-1}(x^n)$ and x_n, then $s_n(x^n)$ is of the form $s_n(x^n) = au$ with $a \in \mathcal{A}$ and $u \in T$. For each state au of this form we can determine wether $s_n(x^n) = au$ by means of the flow conservation equations. Specifically, we can compute the number of entries to state au as $\sum_{uv \in S_T} n_{uv}^{(a)}$, and the number of exits from au as $\sum_{b \in \mathcal{A}} n_{au}^{(b)}$. If we find the final state to be of the form $s_n(x^n) = au$ with $a \in \mathcal{A}$ and $u \in T$, we then get x^n by applying the inverse function of g^* to the T-class* determined by the given T-class and the found final state. Otherwise, the final state of x^n is not determined by $s_{n-1}(x^n)$ and x_n, and, therefore, it must be of the form $s_n(x^n) = auv$ with $a \in \mathcal{A}$, $u \in S_T$, and $v \in \mathcal{A}^+$.

For each internal node au of T with $u \in S_T$ we can determine wether the final state $s_n(x^n)$ is of the form $s_n(x^n) = auv$ by applying the flow conservation equations to the states in the set

$$W = \{auv : v \in \mathcal{A}^*, auv \in S_T\}\,.$$

Specifically, we compare the number of entries to states in W with the number of exits from states in W. The former can be computed as $n_u^{(a)}$, and the latter as $\sum_{s \in W} \sum_{b \in \mathcal{A}} n_s^{(b)}$. If we find that

$$n_u^{(a)} + \sum_{s \in W} \delta_{w,s_0} \neq \sum_{s \in W} \sum_{b \in \mathcal{A}} n_s^{(b)}\,,$$

then $s_n(x^n)$ belongs to W by the flow conservation equations. We can then determine $\mathcal{T}^*(x^{n-1})$ from $\mathcal{T}(x^n)$ by decrementing $n_u^{(a)}$ by one, and letting u be the final state of x^{n-1}. After recovering x^{n-1} from its index within $\mathcal{T}^*(x^{n-1})$ applying the inverse function of g^*, we get x^n by appending the symbol a to the end of x^{n-1}. □

In view of Corollary 4.3 and by Lemma 4.4, in the sequel we focus on the enumeration of $\mathcal{T}^*(x^n)$ rather than $\mathcal{T}(x^n)$. Lemma 4.5 below establishes a fundamental relation between sequences in $\mathcal{T}^*(x^n)$ and state transition counts for arbitrary context trees. This result is well known for FSMs and, as mentioned, it is essential in several derivations of Whittle's formula [86, 6, 34, 35].

4.5. Lemma. *For sequences $x^n, y^n \in \mathcal{A}^n$, and a context tree T, we have $y^n \in \mathcal{T}^*(x^n)$ if and only if $N(y^n) = N(x^n)$.*

Proof. Clearly, since a state transition $s_{i-1} \to s_i$ uniquely determines the symbol $x_i = \text{head}(s_i)$ that occurs in state s_{i-1}, the equality $N(y^n) = N(x^n)$ implies that $y^n \in \mathcal{T}(x^n)$. Moreover, by the flow conservation equations, the final state of a sequence is uniquely determined by the initial state s_0 and the state transition matrix. Thus, if $N(y^n) = N(x^n)$, then we must have $s_n(y^n) = s_n(x^n)$ and therefore $y^n \in \mathcal{T}^*(x^n)$.

Suppose now that $y^n \in \mathcal{T}^*(x^n)$. By Lemma 4.1, we only need to show that $N_{s,t}(y^n) = N_{s,t}(x^n)$ for all $s, t \in S_T$ such that $s \preceq \text{tail}(t)$ or $\text{tail}(t) \prec s$. Consider first the case $\text{tail}(t) \prec s$, i.e., $t = au$ and $s = uv$ with $a \in \mathcal{A}$, $u \in \mathcal{A}^*$, and $v \in \mathcal{A}^+$. Since the occurrence of symbol a in state uv uniquely determines the next state au, we have,

$$N_{uv,au}(x^n) = n_{uv}^{(a)}(x^n) = n_{uv}^{(a)}(y^n) = N_{uv,au}(y^n) \,.$$

Consider now the case $s \preceq \text{tail}(t)$, i.e., $s = u$ and $t = auv$ with $a \in \mathcal{A}$, and $u, v \in \mathcal{A}^*$. Since $\sigma_T(\overline{uv}) = u$, state auv can only be reached via state u. Hence, $N_{u,auv}(x^n) = N_{*auv}(x^n)$, and by the flow conservation equations on the transition matrices, we have,

$$N_{u,auv}(x^n) = N_{auv*}(x^n) + \delta_{auv,s_n(x^n)} - \delta_{auv,s_0} \,.$$

Now, since $y^n \in \mathcal{T}^*(x^n)$, the total number of symbols emitted from auv in x^n and y^n is the same, and also $\delta_{auv,s_n(x^n)} = \delta_{auv,s_n(y^n)}$. Thus, we have

$$N_{u,auv}(x^n) = N_{auv*}(y^n) + \delta_{auv,s_n(y^n)} - \delta_{auv,s_0} \,. \tag{4.2}$$

By the flow conservation equations, the right hand side of (4.2) equals $N_{*auv}(y^n)$. Thus, we get $N_{u,auv}(x^n) = N_{*auv}(y^n)$, and the proof is completed by recalling that $N_{*auv}(y^n) = N_{u,auv}(y^n)$, since state auv can only be reached via state u. □

Unfortunately, Lemma 4.5 does not allow an enumeration of $\mathcal{T}^*(x^n)$ by enumerating state sequences with transition counts compatible with $N(x^n)$. Unlike in the FSM case, some such state sequences may not correspond to any symbol sequence.

4.6. Example. Consider the context tree of Figure 4.1, where, for clarity, we use the labels A,B,C,D in lieu of $0, 100, 101, 11$, respectively. The sequence $x^n = 001101$, with initial state B, defines a state sequence BAABDAC (see Figure 4.2(N)), which, taking A,B,C,D as the order for rows and columns, yields the transition matrix

$$N(x^n) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{4.3}$$

The state sequence BABDAAC is compatible with $N(x^n)$, but it does not correspond to any symbol sequence, since the state sequence BA can only be followed by states A or C, even though transitions from A to B may be valid (and required) elsewhere in the sequence. The difficulty in determining the type class size for context trees lies in accounting for these restrictions on state transition sequences, which stems from the possible loss of context in some of the states.

To derive some of the main results in the remainder of this chapter, we rely on notions from graph theory, which we define next, following, loosely, reference [5]. A (directed) graph is a pair $G = (V, E)$ where V is a finite set of *vertices* and E is a *multi-set* of *edges*, which are pairs in $V \times V$. Thus, we allow multiple parallel, distinguishable, directed edges between

the same pair of nodes; when the multiplicity of every edge in G is at most one, we refer to G as a *1-graph*. We call u the *source* of $e = (u, v)$, and v its *destination*. Both u and v are called *endpoints* of e. The *incidence matrix* of G is a $|V|\times|V|$ matrix M, with rows and columns indexed by elements of V, and where $M_{u,v}$ is the multiplicity of (u, v) in E. A *path* is a sequence of edges $e_1, e_2 \cdots e_m$, where the destination of e_i equals the source of e_{i+1}. A path is *closed* if it starts and ends at the same vertex and it is *simple* if $e_i \neq e_j$ for $i \neq j$ (more than one of the distinguishable copies of an edge may occur in the path, but no copy may occur more than once). A *circuit* is a closed simple path. A path is called *Eulerian* when it is simple and it exhausts the multi-set of edges of the graph. An *unlabeled path* is an equivalence class in the equivalence relation on paths induced by considering all the copies of an edge as equivalent (indistinguishable). We denote an unlabeled path by a sequence of vertices. An unlabeled path is called Eulerian if it is the equivalence class of an Eulerian path.

4.2 The size of a type class

In the following definitions we present some basic components of the construction that leads to a precise formula for $|\mathcal{T}^*(x^n)|$. Consider a state $s \in S_T$ with $|s| = r$. By our assumption of T being nontrivial we have $r > 0$. We define

$$\ell_s = \max\{\, j : \forall i,\, 1 \le i \le j,\ \exists s' \in S_T \text{ s.t. } s' \preceq (s)_i^r \,\}\,. \tag{4.4}$$

In other words, ℓ_s is the smallest integer such that the suffix $(s)_{\ell_s+1}^r$ is an internal node of T. Since $(s)_1^r = s \in S_T$, clearly $\ell_s \ge 1$. Let $\mu_i(s) = (s)_{\ell_s-i+1}^r$, i.e., we decompose s as

$$s = (s)_1^{\ell_s-i}\mu_i(s), \quad 1 \le i \le \ell_s\,. \tag{4.5}$$

For $1 \le i \le \ell_s$ let $\nu_i(s)$ be the unique state in S_T such that $\nu_i(s) \preceq \mu_i(s)$ (such a state exists by the definition of ℓ_s). Clearly, $\nu_1(s)\cdots\nu_{\ell_s}(s) = s$ is a *forced state sequence* that *must* be followed to reach s. Moreover, $\nu_1(s)\cdots\nu_{\ell_s}(s)$ is the longest forced state sequence that ends at s, since by the definition of ℓ_s, we know that $\mathrm{tail}(\mu_1(s))$ is an internal node of T. Indeed, for $1 \le i \le \ell_s$ we get, from (4.4),

$$\mathrm{tail}(\mu_i(s)) \in \mathcal{I}(T) \quad \text{iff } i = 1\,. \tag{4.6}$$

If a state s' appears in the sequence $\nu_1(s)\cdots\nu_{\ell_s}(s)$, its forced state sequence is a subsequence of $\nu_1(s)\cdots\nu_{\ell_s}(s)$. More specifically, we show the relations of Lemma 4.7 below.

4.7. Lemma. *Let $s, s' \in S_T$, $1 \le j \le \ell_s$, and $1 \le i \le \ell_{s'}$.*

(**i**) *If $s' \preceq \mu_j(s)$, then $1 \le i + j - \ell_{s'} \le \ell_s$. In particular with $i = 1$ we get $j \ge \ell_{s'}$.*

(**ii**) *If $s' = \nu_j(s)$, then $\nu_i(s') = \nu_{i+j-\ell_{s'}}(s)$.*

(**iii**) *If $s' = \mu_j(s)$, then $\ell_{s'} = j$, and $\mu_i(s') = \mu_i(s)$.*

Proof. The fact that $i+j-\ell_{s'} \le \ell_s$ follows directly from the assumptions since $i+j-\ell_{s'} \le \ell_{s'}+\ell_s-\ell_{s'}=\ell_s$. Thus, in order to show Part (i), we only need to prove that $1 \le i+j-\ell_{s'}$. We have from (4.5) that $s=(s)_1^{\ell_s-j}\mu_j(s)$, which, when $s' \preceq \mu_j(s)$, yields

$$s=(s)_1^{\ell_s-j}s'z, \quad \text{for some } z\in\mathcal{A}^*\,, \tag{4.7}$$

where $z=\lambda$ if $s'=\mu_j(s)$.

We claim that for all h such that $1\le h\le \ell_s+\ell_{s'}-j$, there exists a state $v\in S_T$ such that $v\preceq(s)_h^r$ where $r=|s|$. If $1\le h\le \ell_s-j$, we know from the definition of ℓ_s that there exists a state $v\in S_T$, such that $v\preceq(s)_h^r$. On the other hand, for $\ell_s-j<h\le\ell_s+\ell_{s'}-j$, we have $1\le h-\ell_s+j\le\ell_{s'}$. Hence, there exists a state $v\in S_T$, such that $v\preceq(s')_{h-\ell_s+j}^{r'}$, where $r'=|s'|$. Since $h>\ell_s-j$, we also have by (4.7) that $(s')_{h-\ell_s+j}^{r'}=(s)_h^{r'}$. Thus, $v\preceq(s)_h^{r'}\preceq(s)_h^r$. We conclude that for all $1\le h\le\ell_s+\ell_{s'}-j$, there exists a state $v\in S_T$, such that $v\preceq(s)_h^r$ as claimed. By the definition of ℓ_s, we then have, $\ell_s+\ell_{s'}-j\le\ell_s$. Thus, $j\ge\ell_{s'}$, and therefore $i+j-\ell_{s'}\ge 1$ as claimed in Part (i).

Now, the assumption $s'\preceq\mu_j(s)$ of Part (i) is fulfilled in both Part (iii) and Part (ii). Thus, in any case, (4.7) is valid, and (4.5) applied to s' gives,

$$\begin{aligned} s &= (s)_1^{\ell_s-j}(s')_1^{\ell_{s'}-i}\mu_i(s')z \\ &= (s)_1^{\ell_s-(i+j-\ell_{s'})}\mu_i(s')z\,, \end{aligned}$$

with $z=\lambda$ in the case of Part (iii). By Part (i), we can also apply (4.5) to s with $(i+j-\ell_{s'})$ in the role of i, to obtain

$$s=(s)_1^{\ell_s-(i+j-\ell_{s'})}\mu_{i+j-\ell_{s'}}(s)\,.$$

Hence, $\mu_i(s')z=\mu_{i+j-\ell_{s'}}(s)$. The proof of (ii) is completed by noting that the strings $\overline{\mu_i(s')}$, and $\overline{\mu_{i+j-\ell_{s'}}(s)}$, which are by definition sufficiently long to select a state in the context tree, must select the same state. In the case of (iii), we have $z=\lambda$, which yields $\mu_i(s')=\mu_{i+j-\ell_{s'}}(s)$. In particular for $i=1$, we get from (4.6) that also $1+j-\ell_{s'}=1$. Thus, $\ell_{s'}=j$, and $\mu_i(s')=\mu_i(s)$ as claimed. □

We define the set of *pseudo-states* of T as

$$U=\{\mu_i(s): s\in S_T, 1\le i\le\ell_s\}.$$

Since $\mu_{\ell_s}(s)=s$, we have $S_T\subseteq U$. The sets of *descendants* and *proper descendants* of $u\in\mathcal{A}^*$ are defined, respectively, as

$$\Lambda(u)=\{u'\in U: u\preceq u'\}, \quad \text{and} \quad \bar{\Lambda}(u)=\Lambda(u)\setminus\{u\}\,.$$

For a string $u=sz$ with $s\in S_T$ and $z\in\mathcal{A}^+$ (i.e., $u\notin T$), the *parent* of u, denoted $\rho(u)$, is defined as the longest string $v\in U$ such that u is a proper descendant of v. In the application of the definitions of parent and descendants the string u will be, in most cases, a pseudo-state of T. These general definitions, however, will be occasionally more convenient.

4.8. Example. In T_1 of Figure 4.1, state $s = 100$ has $\ell_s = 3$, $\mu_1(s) = 0$, $\mu_2(s) = 00$, $\mu_3(s) = 100$ and $\nu_1(s) = 0$, $\nu_2(s) = 0$, $\nu_3(s) = 100$. State $t = 101$ has $\ell_t = 2$, $\mu_1(t) = 01$, $\mu_2(t) = 101$ and $\nu_1(t) = 0$, $\nu_2(t) = 101$. We have $U = S_T \cup \{00, 01\}$, and state 0 is the parent of its proper descendants 00 and 01.

In much the same way that every sequence, x^n, determines a state sequence, x^n will also determine a unique *pseudo-state sequence* over U. The pseudo-states in $U \setminus S_T$ will provide enough of the context lost by their parents to make counting sequences in $\mathcal{T}^*(x^n)$ equivalent to counting pseudo-state sequences that are consistent with a *pseudo-state transition matrix*, $F(x^n)$.

In the pseudo-state sequence of x^n, every state s will be reached via a sequence of transitions $\mu_1(s) \cdots \mu_{\ell_s}(s)$, which we call the *forced pseudo-state sequence of* s. The *entry point* for this forced pseudo-state sequence, $\mu_1(s)$, will provide the minimally necessary context to reach s, guaranteeing that every transition in the forced pseudo-state sequence is valid (i.e., corresponds to a symbol sequence) independently of previous transitions. Thus, it will be possible to permute transitions when enumerating $\mathcal{T}^*(x^n)$, without generating invalid sequences (see Example 4.9 below). To maintain the mentioned minimally necessary context, the pseudo-state sequence of x^n may contain transitions that "drop" context. These transitions will not be associated with any symbol of x^n and, thus, the pseudo-state sequence may be longer than n.

4.9. Example. The sequence $x^n = 001101$ of Example 4.6, which gives the state sequence BAABDAC, would yield a pseudo-state sequence

$$\gamma = \text{B}{\to}01\overset{\lambda}{\to}\text{A}{\to}00{\to}\text{B}{\to}\text{D}{\to}01{\to}\text{C}\,,$$

where λ represents a context-dropping transition. Notice that in γ the state B is preceded by 00, which in turn is preceded by A. This pseudo-state sequence, A→00→B, is associated to the emission of the string 01 starting from A, which causes the state sequence AAB. Thus, no permutation of the pseudo-state transitions of γ generates the invalid sequence BAB mentioned in Example 4.6. Other permutations of pseudo-state transitions, however, give valid strings in the type class of x^n. For example, the sequence $y^n = 100101$ is obtained by the following permutation of the pseudo-state transitions of γ

$$\gamma' = \text{B}{\to}\text{D}{\to}01\overset{\lambda}{\to}\text{A}{\to}00{\to}\text{B}{\to}01{\to}\text{C}\,.$$

Notice that the pseudo-state 01 provides the minimally necessary context to uniquely determine the next state C when symbol 1 occurs, as in the last transition of γ. On the other hand, the occurrence of symbol 0 in state A or in any of its descendant pseudo-states, 00 and 01, will lead to pseudo-state 00. Thus, in this case, the pseudo-state 01 bears an unnecessarily long context, which is dropped in the transition labeled λ in γ and γ'.

For the sequence x^n in the above example, the matrix F, with rows and columns indexed

by $U = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, 00, 01\}$ in this order, would be given by

$$F(x^n) = \begin{pmatrix} 0&0&0&0&1&0 \\ 0&0&0&1&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&1 \\ 0&1&0&0&0&0 \\ 1&0&1&0&0&0 \end{pmatrix}. \tag{4.8}$$

The general construction of the matrix $F(x^n)$ is described in the sequel. The idea resembles the construction of T_{suf}, the FSM closure of T (see Section 2.3.2), where all the suffixes of strings in S_T are added to T. This maps *all* the transitions from problematic states to new, deeper states, ensuring unambiguous next-state determination from every state with every symbol. However, for some symbols, the deeper state in the FSM closure may provide unneeded extra context. For example, the FSM closure of the context tree of Figure 4.1 would add the states 00 and 01, which are necessary to allow next-state determination if a symbol 1 occurs in their context, but not when the symbol is 0. In general, this excessive context may cause undercounting of $\mathcal{T}^*(x^n)$. Therefore, in general, $F(x^n)$ will *not* be the state transition matrix of x^n with respect to T_{suf}.

We will interpret $F(x^n)$ as the incidence matrix of a *pseudo-state transition graph*, $G_F(x^n) = (U, E_\mu)$. Thus, a pseudo-state sequence compatible with $F(x^n)$ can be regarded as an Eulerian unlabeled path in $G_F(x^n)$, and we will count sequences in $\mathcal{T}^*(x^n)$ by counting such Eulerian unlabeled paths. Thus, we will think of each pair of consecutive pseudo-states in an unlabeled path in $G_F(x^n)$ as a "transition" in a pseudo-state sequence. We also say that a graph with incidence matrix $N(x^n)$ is a *state transition graph* of x^n, and we denote it by $G_N(x^n)$. Whittle's formula for FSMs was re-derived in [34] by counting Eulerian unlabeled path in $G_N(x^n)$. As we observed in Example 4.6, however, for an arbitrary context tree, some of the state sequences associated to Eulerian unlabeled path in $G_N(x^n)$ may not correspond to any sequence of symbols.

The construction of $F(x^n)$ is described next through a sequence of transformations applied to matrices. We start from $N(x^n)$ and calculate the intermediate matrices $K(x^n)$, $D(x^n)$, and $B(x^n)$, which we will define, to finally obtain $F(x^n)$. This construction may be regarded as a sequence of transformations applied to graphs, starting from $G_N(x^n)$ and ending with the pseudo-state transition graph, $G_F(x^n)$. Thus, we will interpret each of the matrices K, D, and B as the incidence matrix of a graph, which we denote $G_K(x^n)$, $G_D(x^n)$, and $G_B(x^n)$, respectively. Under this graph interpretation, each of the transformations that we apply to a matrix in the construction of $F(x^n)$ may be regarded as the redirection of some edges and eventually the addition of new edges to the corresponding graph. These edges, in turn, are interpreted as pseudo-state transitions when we follow a path. In particular, the state sequence of x^n, $s_0 \cdots s_n$, forms an Eulerian unlabeled path in $G_N(x^n)$, γ, and taking any path in the equivalence class of γ, i.e., fixing the order in which the edges are traversed, we can think of a redirection of an edge as a redirection of one of the transitions in the state sequence of x^n. We will adopt this kind of interpretation in some of the examples that illustrate the construction of $F(x^n)$.

Let $\tilde{\Delta}^+(s)$, $\tilde{\Delta}^-(s)$ and $\tilde{\Theta}(s)$ be integer matrices of dimensions $|U|\times|U|$, with rows and

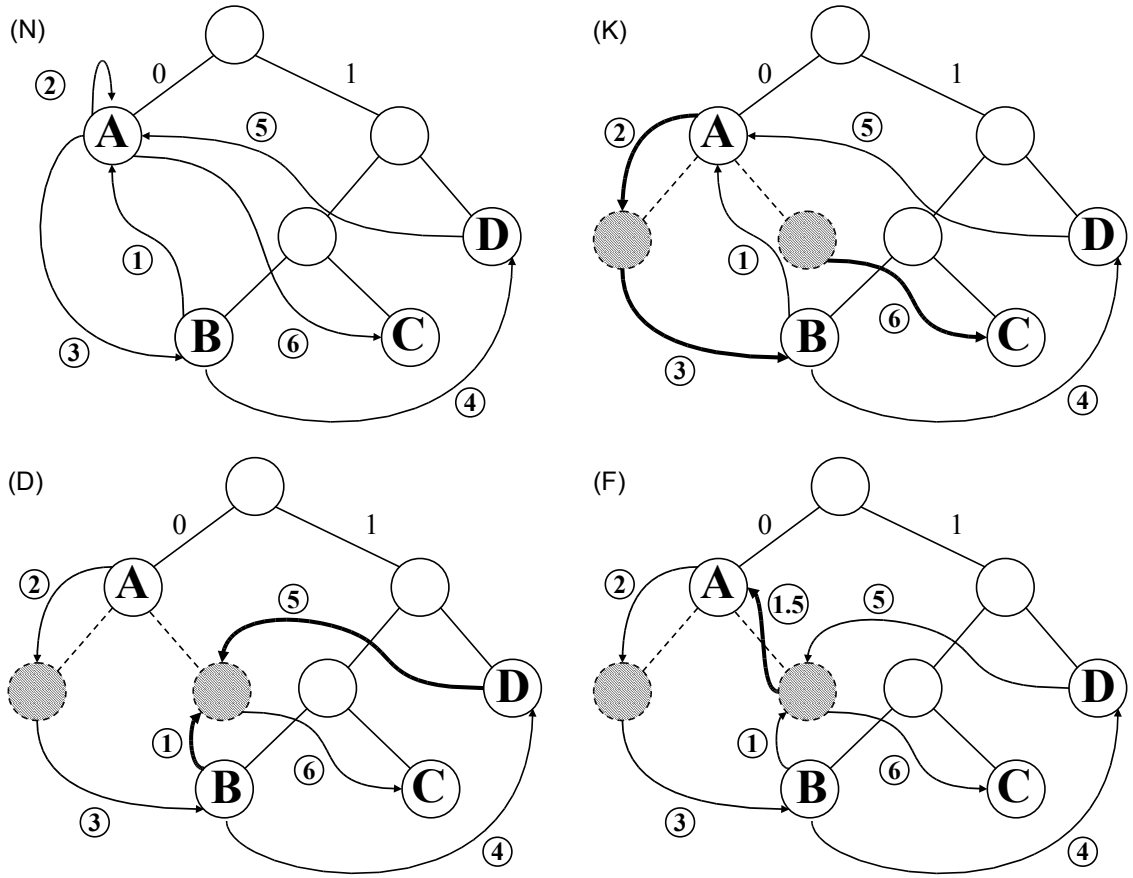

Figure 4.2: *(N) State sequence of* $x^n = 001101$ *in* $G_N(x^n)$ *super-unposed on the tree of Figure 4.1; (K) the same transitions after addition of pseudo-states and redirection in* $G_K(x^n)$*; (D) the same transitions after redirection in* $G_D(x^n)$*; (F) addition of transitions from* $G_B(x^n)$ *to obtain* $G_F(x^n)$*. Transitions affected in each case are highlighted.*

columns indexed by elements of U, and defined, for $u, v \in U$, by

$$\begin{aligned}
\tilde{\Delta}^+(s)_{u,v} &= |\{\, t : 1 \le t < \ell_s,\, \mu_t(s) = u,\, \mu_{t+1}(s) = v \,\}|\,, \\
\tilde{\Delta}^-(s)_{u,v} &= |\{\, t : 1 \le t < \ell_s,\, \nu_t(s) = u,\, \nu_{t+1}(s) = v \,\}|\,, \\
\tilde{\Theta}(s) &= \tilde{\Delta}^+(s) - \tilde{\Delta}^-(s)\,.
\end{aligned}$$

Taking $\tilde{\Theta}(s)$ as a correction to be added to the incidence matrix of a graph, we can interpret the correction as replacing edges of the form $\nu_i(s) \to \nu_{i+1}(s)$, which correspond to forced state sequences, with edges of the form $\mu_i(s) \to \mu_{i+1}(s)$ between elements of U, which correspond to forced pseudo-state sequences. Such substitution is expressed in $\tilde{\Theta}(s)$ by incrementing the entry $(\mu_i(s), \mu_{i+1}(s))$, and decrementing the entry $(\nu_i(s), \nu_{i+1}(s))$ by means of $\tilde{\Delta}^+(s)$ and $\tilde{\Delta}^-(s)$, respectively.

To apply the outlined corrections, we first extend the state transition matrix N to a matrix

$N^{(\mu)}$ of dimensions $|U|\times|U|$, by appending all-zero rows and all-zero columns. We define

$$\Theta(s) = \tilde{\Theta}(s) - \sum_{t=1}^{\ell_s-1} \Theta(\nu_t(s))\,, \tag{4.9}$$

and

$$K(x^n) = N^{(\mu)}(x^n) + \sum_{s\in S_T} N^{(\mu)}_{*s}(x^n)\Theta(s)\,. \tag{4.10}$$

The recursion in (4.9) is well defined, since $\nu_t(s)$ is strictly shorter than s. The term $-\sum_{t=1}^{\ell_s-1}\Theta(\nu_t(s))$ compensates for overcounts in (4.10), and ensures that every forced transition in the state sequence of x^n is taken care of by one and only one state, namely, the last state in the sequence that forces that transition. More precisely, we define the *forced sequence parsing* of x^n, $J(x^n)$, as the set of positions j in the state sequence of x^n such that $s_j{\to}s_{j+1}$ is not a forced transition of any subsequent state. Specifically, we let $t_0 < t_1 < \cdots < t_r$ be the indexes in

$$J = \{t_i\} = \{j : 0 \le j \le n, j \le h - \ell_{s_h} \text{ for all } h > j, h \le n\}\,.$$

For $0 \le j \le n$ we define $\vec{\jmath} = \min\{h \in J : h \ge j\}$. Notice that $n \in J$ by definition, and therefore $\vec{\jmath}$ is well defined. As we shall prove later on, when j is not itself in J, $\vec{\jmath}$ gives the position of the last state that forces the transition $s_j{\to}s_{j+1}$. The index $j_\Delta = j - \vec{\jmath} + \ell_{s_{\vec{\jmath}}}$ gives the position of s_j in the forced state sequence $\nu_1(s_{\vec{\jmath}})\cdots\nu_{\ell_{s_{\vec{\jmath}}}}(s_{\vec{\jmath}})$.

4.10. Example. The state sequence BAABDAC of the string $x^n = 001101$ of Example 4.6, is illustrated in Figure 4.2(N). The transitions labeled 2 and 3 in Figure 4.2(N), namely, $s_1 \to s_2$ and $s_2 \to s_3$, are forced transitions of the state $s_3 = \mathrm{B}$. Similarly, the transition labeled 6 in Figure 4.2(N), namely, $s_5 \to s_6$, is a forced transition of state $s_6 = \mathrm{C}$. Thus, we have $J = \{0, 3, 4, 6\}$. For $j = 1$ we have $\vec{\jmath} = 3$ and $j_\Delta = 1$, while for $j = 2$ we also have $\vec{\jmath} = 3$ but $j_\Delta = 2$. In the graph $G_N(x^n)$, the edges corresponding to the forced state sequences of s_3 and s_6 are replaced in $G_K(x^n)$ by edges corresponding to the forced pseudo-state sequences of s_3 and s_6, respectively, as shown in Figure 4.2(K).

Notice that the sources of possible illegal state sequences are transitions of the form $s_j{\to}s_{j+1}$, with $s_{j+1} = as_jv$ for some $v \in \mathcal{A}^+$, which may occur at indexes $j \notin J$. All edges of the form (s_j, s_{j+1}) in G_N corresponding to such transitions have been replaced in G_K by edges of the form $s_jw{\to}as_jw$, with $s_jw = \mu_i(s_{\vec{\jmath}})$, and $as_jw = \mu_{i+1}(s_{\vec{\jmath}})$. In the replacing edge, the source vertex has enough context to guarantee that the corresponding transition is valid.

In an unlabeled path over G_K, every state with $\ell_s > 1$ is accessed via its forced pseudo-state sequence $\mu_1(s)\cdots\mu_{\ell_s}(s)$. However, edges corresponding to state transitions that do not belong to a forced state sequence (and therefore lose context by the definition of ℓ_s), are not redirected in G_K. This is the case, for example, of transitions 1, 4, and 5 in Figure 4.2(N) that remain the same in Figure 4.2(K). As a consequence, entry point pseudo-states of the form $\mu_1(s)$ with $s \in S_T$ do not have incoming edges in G_K except, possibly, if $\mu_1(s) \in S_T$ (see, e.g., pseudo-state 01 in Figure 4.2(K), where we have $01 = \mu_1(s)$ for $s = \mathrm{C}$). The next adjustment redirects these edges associated to context-losing transitions.

For $u \in S_T$, $a \in \mathcal{A}$ such that au is not an internal node of T, let $\tau(u,a)$ denote the longest prefix of au in U. Let $\mathbf{1}_{i,j}$ denote a $|U|\times|U|$ matrix valued 1 in entry i,j, and zero everywhere else. For $av \in S_T$, we define the $|U|\times|U|$ matrix $d(u,av) = \mathbf{1}_{u,\tau(u,a)} - \mathbf{1}_{u,av}$ if $v \preceq u$, or zero otherwise. The matrix $d(u,av)$ effects the redirection of an edge associated to the transition caused by a in state u to the new destination $\tau(u,a)$, which preserves as much context as possible. Thus, we define

$$D(x^n) = K(x^n) + \sum_{u,v\in S_T} K_{u,v}(x^n) d(u,v)\,. \tag{4.11}$$

4.11. Example. Figure 4.2(D) shows the effect of applying (4.11) to the edges in the graph G_K of Figure 4.2(K). For u = B and $a = 0$ in the context tree of Figure 4.2, we have $\tau(u,a) = 01$. Thus, for u = B and v = A, $d(u,v)$ takes the edge associated to Transition 1 in Figure 4.2(K) to the edge associated to Transition 1 in Figure 4.2(D). Similarly, for u = D and v = A, $d(u,v)$ takes the edge associated to Transition 5 in Figure 4.2(K) to the edge associated to Transition 5 in Figure 4.2(D).

We can interpret the transformations given by (4.10) and (4.11), which take N to D, as redirections of state transitions in the original state sequence of x^n. This gives a sequence of edges of $G_D(x^n)$, denoted γ_D, which is not necessarily a path, since the destination of an edge may not coincide with the source of the next edge. This is illustrated in Figure 4.2(D), where the source of the edge associated to Transition 2 is different from the destination of the edge associated to Transition 1. In fact, state A, which is the entry point for the forced pseudo-state sequence of B, does not have any incoming edge. On the other hand, there are two edges entering the pseudo-state 01, namely, the edges corresponding to Transition 1 and Transition 5, but only Transition 6 exits the pseudo-state 01. This reflects the fact that as we scan the string $x^n = 001101$, there are two instants i in which the context[3] 10 is a suffix of $\overline{s_0}x^i$, namely, $i = 1$ and $i = 5$. Thus, in either of these two times, the emission of symbol 1 would generate a valid transition to state C. However, state C is only accessed once, and, hence, we can drop context in one of these two times in which pseudo-state 01 is entered.

We next define the matrix $B(x^n)$, which incorporates context-dropping transitions of the form $(u,\rho(u))$, where we recall that $\rho(u)$ is the parent of u. Specifically,

$$B_{u,v}(x^n) = \begin{cases} \sum_{w\in\Lambda(u)} \big(D_{*w}(x^n) - D_{w*}(x^n)\big), & u \in U\backslash S_T, v = \rho(u), \\ 0, & \text{otherwise.} \end{cases} \tag{4.12}$$

Finally,

$$F(x^n) = D(x^n) + B(x^n)\,. \tag{4.13}$$

[3]Recall that states and pseudo-states are labeled with symbols in reverse order of appearance in x^n.

4.12. Example. In Figure 4.2(D) there are two incoming edges into pseudo-state 01 and only one outgoing edge from 01. Thus, we have $B_{u,v} = 1$ for $u = 01$ and $v = \rho(u) = \mathrm{A}$, which is represented by the edge labeled 1.5 in Figure 4.2(F). Notice that the pseudo-state sequence γ of Example 4.9 can be identified as a path in the graph G_F represented in Figure 4.2(F) by following the order $1, 1.5, 2 \cdots 6$. The pseudo-state sequence γ' of Example 4.9 arises following the edges in the order $4, 5, 1.5, 2, 3, 1, 6$.

Lemma 4.13 below establishes a set of important properties of the matrix F, and the forced sequence parsing $J(x^n) = \{t_0, t_1 \cdots t_r\}$. To simplify notation, we denote $\tau_i = \tau(s_{t_i}, x_{t_i+1})$.

4.13. Lemma. *For every sequence x^n we have:*

(i) *For all $0 < h \leq n$, and all j such that $h > j$, and $j > h - \ell_{s_h}$, we have $s_j = \nu_i(s_h)$ where $i = \ell_{s_h} + j - h$ satisfies $1 \leq i < \ell_{s_h}$. We also have $|s_j| < |s_h|$, and $\ell_{s_h} - \ell_{s_j} \geq h - j$.*

(ii) *For all $0 \leq j \leq n$ we have $1 \leq j_\Delta \leq \ell_{s_{\vec{j}}}$, $\mu_{j_\Delta}(s_{\vec{j}}) \preceq \overline{x^j} s_0$ and therefore $s_j = \nu_{j_\Delta}(s_{\vec{j}})$.*

(iii) *$t_0 = 0$, $t_r = n$ and $t_i = t_{i+1} - \ell_{s_{t_{i+1}}}$ for $0 \leq i < r$.*

(iv) *$\mu_1(s_{t_{i+1}}) \prec x_{t_i+1} s_{t_i}$ for $0 \leq i < r$, and therefore $\tau_i \in \Lambda(\mu_1(s_{t_{i+1}}))$ for $0 \leq i < r$.*

(v) $N^{(\mu)} = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^-(s_{t_{i+1}})$.

(vi) $K = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^+(s_{t_{i+1}})$.

(vii) $D = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, \tau_i} + \tilde{\Delta}^+(s_{t_{i+1}})$.

(viii) $B_{u,\rho(u)} = \sum_{i=0}^{r-1} \sum_{w \in \Lambda(u)} \delta_{w,\tau_i} - \delta_{w,\mu_1(s_{t_{i+1}})}$.

(ix) *$F_{i,j} \geq 0$ for all i, j and $F_{*i} + \delta_{s_0,i} = F_{i*} + \delta_{s_n,i}$.*

(x) *$\sum_{i,j} D_{i,j} = \sum_{i,j} K_{i,j} = n$, and for all $u \in U$, we have $F_{u*} \leq n$ and $F_{*u} \leq n$.*

(xi) *If $v \prec u$ and $F_{u,v} > 0$, then $v = \rho(u)$, $F_{u,v} = B_{u,v}$, and tail$(u) \in T$.*

The proof of Lemma 4.13 is technical but rather straightforward, and it is deferred to Appendix D.

Notice that Lemma 4.13(ix) validates our interpretation of F as the incidence matrix of the pseudo-state transition graph $G_F(x^n)$ and, moreover, this part of the lemma also states that F satisfies the flow conservation equations. We next develop tools that will allow us to count strings in $\mathcal{T}^*(x^n)$ by counting Eulerian unlabeled paths in $G_F(x^n)$. To this end we will establish a connection between symbol sequences and paths in graphs. Moreover, in Section 4.4, we will relate the number of close-ended type classes induced by a context tree to the number of different pseudo-state transition graphs $G_F(x^n)$ that arise as x^n varies in $\mathcal{A}^n$. Lemma 4.14 below gives the fundamental relations between sequences and paths that will serve our purpose. To map Eulerian paths to sequences, we tag each edge $e \in E$ of a graph $G = (U, E)$, by means of the *tagging function* $\omega : E \to \mathcal{A}^*$, defined for $e = (u, v)$ as $\omega(e) = \lambda$ if $v \prec u$, or $\omega(e) = \mathrm{head}(v)$ otherwise. We extend ω to paths γ in G by concatenating the

tags of the edges encountered as the path is traversed in order. Furthermore, since $\omega(e)$ depends only on the source u and the destination v of e, we also extend ω to unlabeled paths by defining $\omega(\gamma)$ as the result of applying ω to any path in the equivalence class of γ, where we recall from the definition of unlabeled path that this equivalence class on paths is induced by considering all the copies of an edge equivalent (indistinguishable). For instance, in Example 4.9, transitions in G_F not tagged with λ are tagged with consecutive symbols of x^n.

4.14. Lemma. *Let G and G' be graphs with the same set of vertices U, and edges (u, v) of the following forms:*

- *$u = \mu_i(s), v = \mu_{i+1}(s)$ for some $s \in S_T$, $1 \leq i < \ell_s$.*
- *$u \in S_T$ and $v = \tau(u, b)$ for some $b \in \mathcal{A}$.*
- *$v = \rho(u)$ and tail$(u) \in T$.*

(**i**) *Let $\zeta = u_0, u_1 \cdots u_h$ be an unlabeled path in G starting from $u_0 = s_0$. Then $\overline{u_h}$ is a suffix of $\overline{s_0}\omega(\zeta)$.*

(**ii**) *Let $\gamma = u_0, u_1 \cdots u_m$ and $\gamma' = u'_0, u'_1 \cdots u'_{m'}$ be Eulerian unlabeled paths from s_0 to $s \in S_T$ in G and G' respectively. If $\omega(\gamma) = \omega(\gamma')$, then $m = m'$, and $u_i = u'_i$ for all $1 \leq i \leq m$.*

Proof.

We prove Part (i) by induction on h. For $h = 0$ the claim is clearly true. Assume its also true for $h - 1$, and consider the last edge $e = (u_{h-1}, u_h)$. If $u_h = \rho(u_{h-1})$, then $\omega(e) = \lambda$. Thus, $\overline{u_h}$, which is a suffix of $\overline{u_{h-1}}$, is also a suffix of $\overline{s_0}\omega(\zeta)$. If $u_{h-1} = \mu_i(s)$ and $u_h = \mu_{i+1}(s)$, or if $u_{h-1} \in S_T$ and $u_h = \tau(u_{h-1}, b)$, then $u_h = bz$ for some $b \in \mathcal{A}$, $z \preceq u_{h-1}$. Hence, $\omega(e) = b$ and $\overline{u_h} = \overline{z}b$ is a suffix of $\overline{s_0}\omega(\zeta)$.

For Part (ii) first we show that if $u_i = u'_i$ for all $i \leq \min\{m, m'\}$, then we must have $m = m'$. Suppose on the contrary that, with no loss of generality, $m > m'$. Then, since all vertices coincide up to index m', we have $\omega(u_0 \cdots u_{m'}) = \omega(u'_0 \cdots u'_{m'}) = \omega(\gamma')$. Thus, since $\omega(\gamma) = \omega(\gamma')$, we must have $\omega(u_{m'} \cdots u_m) = \lambda$. But $u_{m'} = s \in S_T$ by the assumptions, and therefore no edge labeled with λ, of the form $(u, \rho(u))$, can follow from s. We conclude that $m = m'$.

Now suppose the claim of Part (ii) is not true. Then, by the claim we have just proved, u_i and u'_i must be different for some $i \leq \min\{m, m'\}$. Let j be the minimum such index,

$$j = \min\{i : u_i \neq u'_i\}\,.$$

Since $u_0 = u'_0$ for both γ and γ' start at s_0, the index j satisfies $0 < j \leq \min\{m, m'\}$. We claim that one of the edges $e = (u_{j-1}, u_j)$, $e' = (u'_{j-1}, u'_j)$ must be of the form $(u, \rho(u))$. If they were both of type $(u, \tau(u, b))$, or both of type $(\mu_i(s), \mu_{i+1}(s))$, then u_j and u'_j would coincide for $\omega(\gamma) = \omega(\gamma')$. If one edge were of type $(u, \tau(u, b))$, and the other of type $(\mu_i(s), \mu_{i+1}(s))$ with $\mu_i(s) = u$, then u_j and u'_j would also coincide, since we must have $\tau(u, b) = \mu_{i+1}(s)$ by

definition of τ. Thus, one of the edges must be of the form $(u, \rho(u))$. Suppose, without loss of generality, that $u_j = \rho(u_{j-1})$, and let

$$r' = \min\{i : i \geq j, u'_i \in S_T\}\,.$$

Notice that r' is well defined, for $j \leq m'$ and $u'_{m'} \in S_T$. We claim that the edges (u'_{i-1}, u'_i) are of the form $(\mu_h(s), \mu_{h+1}(s))$ for $i = j \cdots r'$, and we prove it by induction on i. For $i = j$, we have $u'_{j-1} = u_{j-1}$ and $u_j = \rho(u_{j-1})$. Thus, $u'_{j-1} \notin S_T$, and therefore the edge is not of the form $(u, \tau(u, b))$. Also by definition of j, we know that $u'_j \neq u_j = \rho(u_{j-1})$, and therefore the edge must have the form $(\mu_h(s), \mu_{h+1}(s))$. For $i > j$, assume that the edges (u'_{k-1}, u'_k) are of the form $(\mu_h(s), \mu_{h+1}(s))$ for $k = j \cdots i-1$. By definition of r', we know that $u'_{i-1} \notin S_T$. Hence, the edge (u'_{i-1}, u'_i) is not of the form $(u, \tau(u, b))$. If $u'_i = \rho(u'_{i-1})$, then $\mathrm{tail}(u'_{i-1}) \in T$. By our inductive assumption, (u'_{i-2}, u'_{i-1}) has the form $(\mu_h(s), \mu_{h+1}(s))$, and therefore $u'_{i-2} = \mathrm{tail}(u'_{i-1})$, which belongs to T. When $i - 2 \geq j$, this contradicts the definition of r', and for $i = j + 1$ it contradicts the fact that $u_j = \rho(u'_{j-1})$. We conclude that the edges (u'_{i-1}, u'_i) are of the form $(\mu_h(s), \mu_{h+1}(s))$ for $i = j \cdots r'$ as claimed. As a result, since $r' \geq j$, the string $\omega(u'_0 \cdots u'_{r'})$ is not shorter than $\omega(u'_0 \cdots u'_j)$, which in turn is strictly longer than $\omega(u_0 \cdots u_j)$ for $u_j = \rho(u_{j-1})$. Hence, the index

$$r = \min\{i : \omega(u_0 \cdots u_i) = \omega(u'_0 \cdots u'_{r'})\}\,,$$

is strictly larger than j. We claim that $|u_r| < |u'_{r'}|$. By our definition of the tagging function ω, and our assumptions on the edges of G and G', we have that for each edge (u, v), we have $|v| \leq |u| + 1$, and the equality can only hold when $|\omega((u, v))| = 1$. Hence, we have

$$|u_{i+1}| - |u_i| \leq |\omega(u_0 \cdots u_{i+1})| - |\omega(u_0 \cdots u_i)|\,,$$

and summing from $i = j$ to $r - 1$, we obtain

$$|u_r| \leq |u_j| + |\omega(u_0 \cdots u_r)| - |\omega(u_0 \cdots u_j)|\,.$$

Since $u_j = \rho(u_{j-1})$, we have $\omega(u_0 \cdots u_j) = \omega(u_0 \cdots u_{j-1})$, and therefore,

$$|u_r| \leq |u_j| + |\omega(u_0 \cdots u_r)| - |\omega(u_0 \cdots u_{j-1})|\,.$$

By the definition of r, we have $\omega(u_0 \cdots u_r) = \omega(u'_0 \cdots u'_{r'})$, and by the definition of j, we have $\omega(u_0 \cdots u_{j-1}) = \omega(u'_0 \cdots u'_{j-1})$. Hence,

$$|u_r| \leq |u_j| + |\omega(u'_0 \cdots u'_{r'})| - |\omega(u'_0 \cdots u'_{j-1})|\,. \tag{4.14}$$

Since the edges (u'_{i-1}, u'_i) are of the form $(\mu_h(s), \mu_{h+1}(s))$ for $i = j \cdots r'$, every such edge contributes one symbol to $\omega(u'_0 \cdots u'_{r'})$ and takes the source to a destination that is one symbol longer. Therefore,

$$|\omega(u'_0 \cdots u'_{r'})| - |\omega(u'_0 \cdots u'_{j-1})| = |u'_{r'}| - |u'_{j-1}|\,.$$

Thus, from (4.14), we get

$$|u_r| \leq |u_j| + |u'_{r'}| - |u'_{j-1}|\,,$$

and since $u'_{j-1} = u_{j-1}$ by definition of j, we get

$$|u_r| \leq |u_j| + |u'_{r'}| - |u_{j-1}| \,.$$

Finally, since $u_j = \rho(u_{j-1})$, we have $|u_j| < |u_{j-1}|$ and, therefore, we get $|u_r| < |u'_{r'}|$ as claimed.

As proved in Part (i), $\overline{u_r}$ and $\overline{u'_{r'}}$ are both suffixes of $\omega(u_0 \cdots u_r) = \omega(u'_0 \cdots u'_{r'})$. Since $|u_r| < |u'_{r'}|$, we must have $u_r \prec u'_{r'}$, which is a contradiction since $u'_{r'} \in S_T$. □

Notice that by parts (vii) and (xi) of Lemma 4.13, any pseudo-state transition graph $G = G_F$ satisfies the assumptions of Lemma 4.14, since the incidence matrix of G_F is of the form $F(x^n)$ by definition.

Part (i) of Lemma 4.14 lets us keep track of the states selected in T by $\omega(\zeta)$ as we traverse un unlabeled path ζ in G_F. Indeed, since $\overline{u_h}$ is a suffix of $\overline{s_0}\omega(\zeta)$, $\overline{s_0}\omega(\zeta)$ selects the unique state s that is a prefix of u_h. By means of this property, we will construct an Eulerian unlabeled path ξ, such that $\omega(\xi) = x^n$. Furthermore, we will show that the application of the tagging function ω to all Eulerian unlabeled paths from s_0 to s_n in $G_F(x^n)$ yields sequences with the same state transitions matrix, which are therefore in $\mathcal{T}^*(x^n)$. Moreover, by Part (ii) of Lemma 4.14, every such Eulerian unlabeled path defines a unique sequence. These properties, together with the BEST Theorem [22], will allow us to derive the main result of this section, given in Theorem 4.15 below.

We need some definitions to state the theorem. Recalling that, by Lemma 4.13(iv), $\tau_i \in \Lambda(\mu_1(s_{t_{i+1}}))$, we define sequences β_i of elements of U, such that β_i is empty for $\tau_i = \mu_1(s_{t_{i+1}})$, and otherwise $\beta_i = v_0 v_1 \cdots v_m$ where $v_0 = \tau_i$, $\rho(v_m) = \mu_1(s_{t_{i+1}})$, and $v_j = \rho(v_{j-1})$ for $j = 1 \cdots m$. Moreover, for $0 < i \leq r$, where $t_r = n$ is the largest element of J, we define $\varsigma_i = \beta_{i-1}\mu_1(s_{t_i})\mu_2(s_{t_i}) \cdots \mu_{\ell_{s_{t_i}}}(s_{t_i})$. Finally, we define the normalized $|U| \times |U|$ matrix $\hat{F}$ as $\hat{F}_{i,j} = F_{i,j}/F_{i*}$ if $F_{i*} > 0$ and $\hat{F}_{i,j} = 0$ otherwise.

4.15. THEOREM. *Let x^n be a sequence in $\mathcal{A}^n$.*

(i) *$\xi = s_0\varsigma_1 \cdots \varsigma_r$ is an Eulerian unlabeled path from s_0 to s_n in G_F such that $\omega(\xi) = x^n$.*

(ii) *The function ω defines a one-to-one correspondence between the set of Eulerian unlabeled paths from s_0 to s_n in G_F and the sequences in $\mathcal{T}^*(x^n)$.*

(iii) *Let M denote the cofactor of entry (s_n, s_0) in $I - \hat{F}$. Then, $M \leq 1$ and*

$$|\mathcal{T}^*(x^n)| = M \frac{\prod_i F_{i*}!}{\prod_{i,j} F_{i,j}!} \,. \tag{4.15}$$

Proof. We start by Part (i). By Lemma 4.13(ii), if there exists indeed an unlabeled path ξ in G_F as defined in (i), we must have $\omega(\xi) = x^n$. We will prove that ξ is an Eulerian unlabeled path in G_F. Let $v_0 \cdots v_m = \xi$, and let $N(\xi)$ be the $|U| \times |U|$ matrix, indexed with elements from U, defined as $N_{u,v}(\xi) = |\{j = 1 \cdots m : v_{j-1} = u, v_j = v\}|$ for all $u, v \in U$. Notice that $N(\xi) = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i},\tau_i} + \tilde{\Delta}^+(s_{t_{i+1}}) + \tilde{B}$, where $\tilde{B}$ is the matrix of transitions of the subsequences β_i in ξ. Thus, by Lemma 4.13(vii), $N(\xi) = D + \tilde{B}$, and we must show that $\tilde{B} = B$ in order to prove that $N(\xi) = F$. By definition of β_i, $\tilde{B}_{u,v} > 0$ implies $v = \rho(u)$.

Moreover, by construction, $\tilde{B}_{u,v} = |\{i = 0 \cdots r-1 : \tau_i \in \Lambda(u), \mu_1(s_{t_{i+1}}) \notin \Lambda(u)\}|$. On the other hand, by Lemma 4.13(viii),

$$B_{u,v} = \sum_{i=0}^{r-1} \sum_{w \in \Lambda(u)} \delta_{w,\tau_i} - \delta_{w,\mu_1(s_{t_{i+1}})} \,.$$

Since $\tau_i \in \Lambda(\mu_1(s_{t_{i+1}}))$, the terms corresponding to indices i such that $\mu_1(s_{t_{i+1}}) \in \Lambda(u)$ are zero, and therefore, $B_{u,v} = |\{i = 0 \cdots r-1 : \tau_i \in \Lambda(u), \mu_1(s_{t_{i+1}}) \notin \Lambda(u)\}| = \tilde{B}_{u,v}$. We conclude that $N(\xi) = F$, and consequently ξ is an Eulerian unlabeled path in G_F.

We now turn to Part (ii). We first notice that, since $F(x^n)$ is defined exclusively as a function of the matrix $N(x^n)$, and by Lemma 4.5 we have $N(y^n) = N(x^n)$ for all $y^n \in \mathcal{T}^*(x^n)$, Part (i) implies that there exists an Eulerian unlabeled path γ such that $\omega(\gamma) = y^n$ for all $y^n \in \mathcal{T}^*(x^n)$, since x^n is arbitrary.

Now, let $\gamma = u_0, u_1 \cdots u_m$ be an arbitrary Eulerian unlabeled path from $u_0 = s_0$ to $u_m = s_n$ in $G_F(x^n)$. We will show that $y^n = \omega(\gamma)$ belongs to $\mathcal{T}^*(x^n)$, by showing that y^n and x^n share the same state transition matrix. Since edges of the form $(u, \rho(u))$ do not contribute with symbols to $\omega(\gamma)$ by the definitions of ω and ρ, we only need to take care of edges that come from positive entries in $D(x^n)$. We recall from Lemma 4.13(vii) that,

$$D(x^n) = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i},\tau_i} + \tilde{\Delta}^+(s_{t_{i+1}}) \,. \tag{4.16}$$

Let $\tilde{\gamma} = e_1 \cdots e_m$ be an arbitrary Eulerian path in the equivalence class of the unlabeled path γ, i.e., each edge e_h is of the form $e_h = (u_{h-1}, u_h)$, we distinguish multiple copies of an edge, and $e_h \neq e_j$ for $h \neq j$. To each index i in the summation of (4.16), we associate a subset Γ_i of the edges of $\tilde{\gamma}$ of the following form:

$$\Gamma_i = \{\, (s_{t_i}, \tau_i) \,\} \cup \left\{ (\mu_i(s_{t_{i+1}}), \mu_{i+1}(s_{t_{i+1}})) : i = 1 \cdots \ell_{s_{t_{i+1}}} - 1 \right\} . \tag{4.17}$$

We assign different copies of the multiple distinguishable edges of $G_F(x^n)$ to different sets Γ_i, so that these sets Γ_i are disjoint. Notice that we can do this disjoint assignment of edges of $\tilde{\gamma}$ to the sets Γ_i by (4.16) and the definitions of Γ_i and $\tilde{\Delta}^+$.

Consider an arbitrary edge of $\tilde{\gamma}$, $e_h = (u_{h-1}, u_h)$. We define l_{e_h} as the length of the prefix of y^n generated by the subpath $e_1 \cdots e_h$ of $\tilde{\gamma}$, i.e., $l_{e_h} = |\omega(e_1 \cdots e_h)|$. Thus, we have $y^{l_{e_h}} = \omega(e_1 \cdots e_h)$. Notice that if $e_h = (u_{h-1}, u_h)$ belongs to one of the sets Γ_i, then $u_h \neq \rho(u_{h-1})$ by (4.17) and, therefore, $\omega(e_1 \cdots e_h)$ is one symbol longer than $\omega(e_1 \cdots e_{h-1})$. As a consequence, we have

$$l_e \neq l_{e'} \,, \quad \forall\, e, e' \in \bigcup_i \Gamma_i \,, e \neq e' \,. \tag{4.18}$$

Hence, we associate to each edge e in $\bigcup_i \Gamma_i$ a *unique* integer l_e, $0 < l_e \leq n$, which we will interpret as an index in the state sequence of y^n. Moreover, we claim that for each $i = 0 \cdots r-1$, we can label the edges in Γ_i with integers, $\{0, 1, \cdots \ell_{s_{t_{i+1}}} - 1\}$, in such a way that for the edge labeled 0, $e \in \Gamma_i$, the state transition $s_{l_e-1}(y^n) \to s_{l_e}(y^n)$ has the form

$s_{t_i} \to s_{t_i+1}$, and for each $j = 1 \cdots \ell_{s_{t_{i+1}}} - 1$, the edge labeled j, $e' \in \Gamma_i$, is such that the state transition $s_{l_{e'}-1}(y^n) \to s_{l_{e'}}(y^n)$ has the form $\nu_j(s_{t_{i+1}}) \to \nu_{j+1}(s_{t_{i+1}})$. We then get, from (4.17) and the definition of $\tilde{\Delta}^-$, that

$$N^{(\mu)}(\omega(\tilde{\gamma})) = \sum_{i=0}^{r-1} \left(\mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^-(s_{t_{i+1}}) \right) . \tag{4.19}$$

To prove the claim, consider an edge of $\tilde{\gamma}$, $e_h = (u_{h-1}, u_h) \in \Gamma_i$, of the form $u_{h-1} = s_{t_i}$, $u_h = \tau_i$. By Lemma 4.14, $\overline{s_{t_i}}$ is a suffix of $\overline{s_0} y^{l_e - 1}$, thus $\sigma_T(y^{l_e-1}) = s_{t_i}$. Also $\overline{\tau_i}$ is a suffix of $\overline{s_0} y^{l_e}$ and, by definition of τ, we have $s_{t_i+1} \preceq \tau_i$. Hence, we have $\sigma_T(y^{l_e}) = s_{t_i+1}$. Therefore, the state transition $s_{l_e-1}(y^n) \to s_{l_e}(y^n)$ has the form $s_{t_i} \to s_{t_i+1}$. Similarly, for an edge of $\tilde{\gamma}$, $e_h = (u_{h-1}, u_h) \in \Gamma_i$, of the form $u_{h-1} = \mu_i(s_{t_{i+1}})$, $u_h = \mu_{i+1}(s_{t_{i+1}})$, we get $\sigma_T(y^{l_e-1}) = \nu_i(s_{t_{i+1}})$ and $\sigma_T(y^{l_e}) = \nu_{i+1}(s_{t_{i+1}})$. Therefore, the state transition $s_{l_e-1}(y^n) \to s_{l_e}(y^n)$ has the form $\nu_i(s_{t_{i+1}}) \to \nu_{i+1}(s_{t_{i+1}})$. The claim is proved and (4.19) follows.

Now, since $\omega(\gamma) = \omega(\tilde{\gamma})$ by the definition of ω, we get, from (4.19),

$$\begin{aligned} N^{(\mu)}(\omega(\gamma)) &= \sum_{i=0}^{r-1} \left(\mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^-(s_{t_{i+1}}) \right) \\ &= N^{(\mu)}(x^n) , \end{aligned}$$

where the last equality comes from Lemma 4.13(v). Hence, we get $\omega(\gamma) \in \mathcal{T}^*(x^n)$, by Lemma 4.5.

Up to this point we have shown that every sequence $y^n \in \mathcal{T}^*(x^n)$ can be generated by applying ω to some Eulerian unlabeled path, and that for every Eulerian unlabeled path, γ, we have $\omega(\gamma) \in \mathcal{T}^*(x^n)$. Finally, we notice that if γ and γ' are different Eulerian unlabeled paths from s_0 to s_n, then $\omega(\gamma) \neq \omega(\gamma')$ by Lemma 4.14. Hence, ω defines a one-to-one correspondence between the set of Eulerian unlabeled paths from s_0 to s_n in G_F, and the strings in $\mathcal{T}^*(x^n)$.

As for Part (iii), we first notice that we can assume that there are no isolated vertices in G_F. If there existed a vertex u with no incoming nor outgoing edge, the matrix $I - \hat{F}$ would be valued $(I - \hat{F})_{u,j} = (I - \hat{F})_{j,u} = \delta_{u,j}$, and we could eliminate row and column u while the cofactor M remains unchanged.

Let $G'_F = (V, E \cup \{e'\})$ be the graph obtained from G_F by adding an edge $e' = (s_n, s_0)$, and let $F' = F + \mathbf{1}_{s_n, s_0}$ be the incidence matrix of G'_F. Let C be the number of distinct Eulerian circuits in G'_F, where two circuits are considered equal if one can be obtained from the other by a cyclic permutation of the edges. By the BEST Theorem [22], C can be computed as

$$C = M' \prod_i (F'_{i*} - 1)! ,$$

where M' is any cofactor of the matrix L' defined as $L'_{u,v} = \delta_{u,v} F'_{u*} - F'_{u,v}$ (all cofactors of L' are equal for all its columns and rows sum up to zero). The matrix L, defined as $L_{u,v} = \delta_{u,v} F_{u*} - F_{u,v}$, differs only in row s_n from L'. Thus, M' is also equal to the $(s_n, s_0)th$

cofactor of L. Dividing each row i of L by F_{i*} we obtain $I - \hat{F}$, and the cofactor gets divided by $\prod_{i \neq s_n} F_{i*}$. Thus,

$$C = (1/F'_{s_n *}) M \prod_i F'_{i*}! = M \prod_i F_{i*}! \,.$$

The number Q of Eulerian paths from s_0 to s_n in G_F is equal to the number of Eulerian circuits in G'_F from s_0 to s_0 where e' is the last edge, i.e. $Q = C$ since cyclic permutations of the edges are not relevant for calculating C. The proof of (4.15) is completed by noticing that $\prod_{i,j} F_{i,j}!$ is the number of representatives in the equivalence class of an Eulerian unlabeled path. The inequality $M \leq 1$ follows from the trivial bound $\prod_i F_{i*}! \Big/ \prod_{i,j} F_{i,j}!$ on the number of unlabeled paths. □

When T is FSM, all suffixes of a state s are nodes of T by Theorem 2.6. Hence, we have $\tilde{\Delta}^+(s) = \tilde{\Delta}^-(s)\ \forall s \in S_T$, $U = S_T$, and $F = N$. Thus, in this case, (4.15) reduces to Whittle's formula.

Theorem 4.15 shows that the one-to-one correspondence between Eulerian paths and sequences of Part (ii) can be implemented straightforwardly. Indeed, given x^n, one can compute $J(x^n)$ in linear time and then compute ξ such that $\omega(\xi) = x^n$, easily, as defined in Part (i). Of course computing $\omega(\gamma)$ given an Eulerian unlabeled path γ can also be done in time proportional to the length of γ, which by Lemma 4.13(x) is linear in n. Hence, enumerating sequences in $\mathcal{T}^*(x^n)$ is equivalent to enumerating Eulerian unlabeled paths in G_F. The latter, in turn, can be done efficiently (polynomial in n) by recursive application of the formula in [22]. Thus, Theorem 4.15 yields an efficient algorithmic enumeration of $\mathcal{T}^*(x^n)$, and hence also of $\mathcal{T}(x^n)$. We include a more detailed description of an enumeration algorithm and show its polynomial complexity in Appendix F. This enumeration has applications in universal data compression and universal simulation, which we explore in Chapter 5 and Chapter 6, respectively.

4.16. Example. A direct application of (4.15) with the matrix F of (4.8) yields $|\mathcal{T}^*(x^n)| = 2$, and indeed, by direct enumeration of Eulerian unlabeled paths in G_F (see Figure 4.2(F)), we obtain

$$\mathcal{T}^*(x^n) = \{001101, 100101\} \,.$$

4.3 The expected size of a type class

In this section we study the asymptotic behavior of the expectation of $\log |\mathcal{T}(X^n)|$ with respect to a tree source. The analysis will be based on the formula (4.15) presented in Theorem 4.15. We denote by $\Xi_T(x^n)$ the multinomial factor in (4.15) computed with respect to T, i.e.,

$$\Xi_T(x^n) = \frac{\prod_i F(x^n)_{i*}!}{\prod_{i,j} F(x^n)_{i,j}!} \,. \tag{4.20}$$

For an FSM context tree T we have $F = N$ and due to the existence of a next-state function we have $\prod_v F_{s,v}! = \prod_{a\in\mathcal{A}} n_s^{(a)}!$. Hence, $\Xi_T(x^n)$ simplifies to

$$\Xi_T(x^n) = \Xi_T^{\alpha}(x^n) \triangleq \frac{\prod_s n_s!}{\prod_{s,a} n_s^{(a)}!}\,. \tag{4.21}$$

In this case, an application of Stirling's formula for factorials gives straightforwardly a well known connection between $\log \Xi_T(x^n)$ and the empirical entropy rate of x^n with respect to T, as we will show later, for completeness, in the derivation of Lemma 4.27. A similar relation is not apparent directly from (4.15) in general, as pseudo-state transition counts $F_{u,v}$ depend on symbol counts through a sequence of transformations of N. Hence, we will follow a different approach.

Next, we introduce some definitions that we need to state the main result of this section. These definitions will be also used extensively in the rest of this chapter and in Chapter 5. We say that a state s of T is *forgetful* if $as \in \mathcal{I}(T)$ for *all* $a \in \mathcal{A}$, namely, s loses context for every possible transition. A context tree with no such state is called *canonical*. If a context tree is not canonical, we can take a forgetful state s, and make a *single refinement* of s, where by single refinement we mean the extension of T with a full complement of children of s. Proceeding sequentially until no forgetful states remain, the context tree can be brought to canonical form in a finite number of such refinement steps. We show in Lemma 4.17 below that this extension, called the *minimal canonical extension* of T, is unique for each context tree T and we denote it T_{c}.

4.17. Lemma. *Let T' and $\tilde{T}$ be two refinements of T, each obtained by a finite sequence of single refinement steps on forgetful states. If T' and $\tilde{T}$ are both canonical, then $T' = \tilde{T}$.*

Proof. Suppose $T' \neq \tilde{T}$ and, without loss of generality, suppose $T'\backslash\tilde{T}$ is not empty. Consider a sequence of context trees $T'_0, T'_1 \cdots T'_r$, where $T'_0 = T$, $T'_r = T'$, and T'_i is obtained by a single refinement of a forgetful state v_i of T'_{i-1}. Let v_j be the first node in the sequence $v_1 \cdots v_r$ that is not refined in $\tilde{T}$. By definition of v_j we have $T'_{j-1} \subset \tilde{T}$. The node v_j must be a leaf of $\tilde{T}$ for otherwise its parent would appear before it in the sequence $v_1 \cdots v_r$. Moreover, since v_j is forgetful in T'_{j-1}, and $T'_{j-1} \subset \tilde{T}$, v_j is also a forgetful state of $\tilde{T}$. Hence, $\tilde{T}$ is not canonical. □

We now present the main result of this section, for which we introduce the *state transition support graph*, G_T, of a context tree T. We define $G_T = (V_T, E_T)$ as the 1-graph with $V_T = S_T$ and

$$E_T = \{\, (u,v) \in S_T^2 \,:\, u \preceq \mathrm{tail}(v) \text{ or } \mathrm{tail}(v) \prec u \,\}\,. \tag{4.22}$$

The incidence matrix of G_T is the support of the state transition matrix $N(x^n)$ by Lemma 4.1.

4.18. Theorem. *Let $\langle T, p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$ and all conditional probabilities positive, and let T_c be the minimal canonical extension of T. Then,*

$$E_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(X^n)|\big] \le n\mathcal{H} - \frac{|E_{T_c}| - |V_{T_c}|}{2}\log n + O(1)\,. \tag{4.23}$$

When T is FSM, it is also canonical, i.e. $T = T_c$, and $|E_{T_c}|-|V_{T_c}| = (\alpha-1)|S_T|$, in agreement with known results for FSMs, which we will show later in Lemma 4.27 for completeness. If T is not FSM, however, the factor $|E_{T_c}|-|V_{T_c}|$ is larger. This larger factor will be essential for showing the optimal convergence rate of the enumerative code that we will present in Chapter 5. Moreover, in Chapter 5 we will derive a different proof of Theorem 4.18, based on coding arguments, and illustrating the use of some of the tools developed in that chapter. The results in Chapter 5 will also allow us to establish the asymptotic tightness of the upper bound (4.23). The proof presented in this section ties more directly to the exact formula (4.15), and to the combinatorial properties of the minimal canonical extension.

Before we start with the asymptotic expectation analysis of the formula (4.15), we explore some connections between T-classes and T_c-classes*, which we will use in the proof of Theorem 4.18 and also in Section 4.4. In particular, we will show that an arbitrary context tree T induces the same type classes as its minimal canonical extension T_c. Thus, we can define an equivalence relation between context trees in which T and T' are equivalent if and only if $T_c = T'_c$. Then, all context trees in the same equivalence class induce the same type classes, which is expressed in the fact that the bound in (4.23) depends on parameters of T_c rather than T.

Throughout we consider a fixed common initial state of maximal depth for T and T_c, as well as for any context tree T' such that $T \subset T' \subset T_c$. Notice that all maximal depth states are shared by T and T_c, since $|s| < \mathsf{depth}(T)-1$ for any forgetful state s. We start by studying the effect of refining a forgetful state of T on the type classes induced by T.

4.19. Lemma. *Let s be a forgetful state of T, and let T' be the context tree obtained from T by a single refinement of s. If two sequences belong to the same T-class*, they belong to the same T'-class.*

Proof. Consider arbitrary symbols $a, b \in \mathcal{A}$. We will show that the transition matrix N of a string x^n with respect to T determines the symbol count $n_{sb}^{(a)}$. Since s is forgetful, as is an internal node of T and, therefore, $asb \in T$. Let $asbu$ be a state of T with $u \in \mathcal{A}^*$. Notice that $asbu$ is also a state of T' for $asbu \neq s$. Since $\mathrm{tail}(asbu)$ is not an internal node of T nor of T', we have $N_{*asbu} = N_{s,asbu}$, and $N'_{*asbu} = N'_{sb,asbu}$, where N' denotes the state transition matrix with respect to T'. Now, we must have $N_{*asbu} = N'_{*asbu}$, since both N_{*asbu} and N'_{*asbu} are equal to the number of occurrences of the pattern $\overline{asbu}$ in the sequence x^n, which is fixed. Hence, we get $N'_{sb,asbu} = N_{s,asbu}$. Summing over all $u \in \mathcal{A}^*$ such that $asbu$ is a state of T, we obtain $n_{sb}^{(a)}$.

If $y^n \in \mathcal{T}^*(T, x^n)$, by Lemma 4.5, we know that $N(y^n) = N(x^n)$, which, as we have proved, implies that $n_{sb}^{(a)}(y^n) = n_{sb}^{(a)}(x^n)$ for all symbols $a, b \in \mathcal{A}$. Since clearly also $n_t^{(a)}(y^n) = n_t^{(a)}(x^n)$ for all states $t \neq s$ and all symbols a, we conclude that $y^n \in \mathcal{T}(T', x^n)$. □

The following corollary is an immediate consequence of Lemma 4.19 and the sequential construction of T_c.

4.20. Corollary. *If x^n and y^n are sequences in the same T-class and they share the same final state with respect to T_c, then they belong to the same T_c-class*.*

Finally, we derive the claimed relation between the type classes induced by T and T_{c}.

4.21. Corollary. *Two sequences x^n and y^n belong to the same T-class if and only if they belong to the same T_c-class.*

Proof. We partition the sequences y^n in $\mathcal{T}(T, x^n)$ according to its final state with respect to T_{c}, which we denote $s_n(T_{\mathrm{c}}, y^n)$. Specifically, for $s \in S_{T_{\mathrm{c}}}$ we define $\mathcal{T}(T, x^n, s) = \{y^n \in \mathcal{T}(T, x^n) : s_n(T_{\mathrm{c}}, y^n) = s\}$. Thus,

$$\mathcal{T}(T, x^n) = \bigcup_{s \in S_{T_{\mathrm{c}}}} \mathcal{T}(T, x^n, s)\,. \tag{4.24}$$

By Corollary 4.20, $\mathcal{T}(T, x^n, s)$ is a subset of $\mathcal{T}(T_{\mathrm{c}}, x^n, s)$. But since T_{c} is a refinement of T, also $\mathcal{T}(T_{\mathrm{c}}, x^n, s) \subset \mathcal{T}(T, x^n, s)$. Hence, $\mathcal{T}(T_{\mathrm{c}}, x^n, s) = \mathcal{T}(T, x^n, s)$, and (4.24) becomes

$$\mathcal{T}(T, x^n) = \bigcup_{s \in S_{T_{\mathrm{c}}}} \mathcal{T}(T_{\mathrm{c}}, x^n, s) = \mathcal{T}(T_{\mathrm{c}}, x^n)\,.$$

□

In view of Corollary 4.21, in the sequel we assume without loss of generality that the context tree for which we analyze the formula (4.15) of Theorem 4.15 is canonical.

We will follow an incremental approach to obtain (4.23). For a given context tree T, we consider a *refinement sequence* of context trees $T_1 \cdots T_m$, where T_1 is a single-node context tree, $T_m = T$, and, for $i > 1$, T_i is obtained from T_{i-1} by a single refinement of a state $w \in S_{T_{i-1}}$ with $|w| \geq \mathsf{depth}(T_{i-1}) - 1$. It is readily verified that such a refinement sequence exists for any context tree T. We will decompose $\Xi_T(x^n)$, from (4.20), as

$$\Xi_T = \Xi_{T_1} \prod_{i=2}^{m} \Xi_{T_i} / \Xi_{T_{i-1}}$$

and we will keep track of the the asymptotic contribution of each factor $(\Xi_{T_i}/\Xi_{T_{i-1}})$ to the expected logarithm of Ξ_T. This decomposition is suitable for a clean separation of Ξ_T^{α} (defined in (4.21)), from which the entropy rate will arise in (4.23) by means of Stirling's formula, from factors that contribute terms of order $\log n$. Our analysis will be based on the formula (4.15) for the size of $\mathcal{T}^*(x^n)$. As we will show, the expected logarithms of $|\mathcal{T}^*(x^n)|$ and $|\mathcal{T}(x^n)|$ differ by $O(1)$, which makes them equivalent for the purpose of proving the asymptotic result of Theorem 4.18. Moreover, while the cofactor M in (4.15) is essential for the exact result of Theorem 4.15, we will be able to neglect this cofactor in the asymptotic analysis. Thus, since in (4.15) we have $M \leq 1$, by Theorem 4.15, we get

$$\log |\mathcal{T}^*(x^n)| \leq \log \Xi_T(x^n) \tag{4.25}$$

and, therefore, our goal will be to upper-bound the expectation of $\log \Xi_T(X^n)$.

The base building block for the derivation of (4.23) in this section is the following lemma, which will allow for the analysis of the factors $\Xi_{T_i}/\Xi_{T_{i-1}}$ in the mentioned incremental approach.

4.22. LEMMA. *Let T' be a context tree obtained from T by a single refinement of a state w with $|w| \geq \mathsf{depth}(T)-1$. Let $s_0 \in S_T$ and $s'_0 \in S_{T'}$ be initial states of maximal depth in T and T' respectively such that $s_0 \preceq s'_0$. Let $n_s^{(a)}$, U, $\rho(\cdot)$, ℓ_s, N, B, and F, be the objects defined in Section 4.1 and Section 4.2 for the context tree T, and let $n'^{(a)}_s$, U', $\rho'(\cdot)$, ℓ'_s, N', B', and F' be the corresponding objects[4] for T', all with respect to the given sequence x^n and the initial states s_0 and s'_0 respectively. Define*

$$\Pi = \frac{\Xi_{T'}}{\Xi_T} = \frac{\prod_i F'_{i*}! \Big/ \prod_{i,j} F'_{i,j}!}{\prod_i F_{i*}! \Big/ \prod_{i,j} F_{i,j}!}\,, \tag{4.26}$$

Define also

$$\Pi^{(\ell_w=1)} = \frac{\prod_{b\in\mathcal{A}} N'_{wb*}! \Big/ \prod_{a,b\in\mathcal{A}} n'^{(a)}_{wb}!}{N_{w*}! \Big/ \prod_{a\in\mathcal{A}} n_w^{(a)}!}\,, \tag{4.27}$$

and, letting $B'_{\mu_1 b,\rho}$ be a shorthand for $B'_{\mu_1(w)b,\rho'(\mu_1(w)b)}$,

$$\Pi^{(\ell_w>1)} = \frac{\prod_{b\in\mathcal{A}} \binom{B'_{\mu_1 b,\rho}+N'_{*wb}}{N'_{*wb}}}{\binom{F_{\mu_1(w)*}}{N_{*w}}} \cdot \Pi_\delta\,, \tag{4.28}$$

where

$$\Pi_\delta = \frac{\prod_{i\in L} \max\left\{1, F_{*\mu_i(w)}\delta_{s_n,\mu_i(w)}\right\}}{\prod_{i\in L} \max\left\{1, F'_{*\mu_i(w)}\delta_{s_n,\mu_i(w)}\right\}} \tag{4.29}$$

for some subset $L \subseteq \{1,2,\cdots,\ell_w-1\}$, taking, with a slight abuse of notation, $F'_{*\mu_i(w)} = 0$ if $\mu_i(w) \notin U'$.

Then, we have $F_{\mu_1(w)*} \geq N_{*w}$, so that (4.28) is well-defined, and Π satisfies

$$\Pi = \begin{cases} \Pi^{(\ell_w=1)}\,, & \text{if } \ell_w = 1\,, \\ \Pi^{(\ell_w=1)}\Pi^{(\ell_w>1)}\,, & \text{if } \ell_w > 1\,. \end{cases} \tag{4.30}$$

Moreover, in the case $\ell_w > 1$, the summation $\Sigma = \sum_{b\in\mathcal{A}} B'_{\mu_1 b,\rho}$ satisfies

$$\Sigma - \delta_{s_n,\mu_1(w)} = F_{\mu_1(w)*} - N_{*w}\,. \tag{4.31}$$

Imagine that we sequentially construct an Eulerian unlabeled path in the pseudo-state transition graph, $G_F(x^n)$, by selecting in each step one edge departing from the current pseudo-state, and following the selected edge so that its destination becomes the new current pseudo-state. The multinomial factor Ξ_T gives the number of choices in this construction if we assume that any such choice leads to a complete Eulerian unlabeled path in G_F (the proportion of valid choices is determined by the cofactor in (4.15) that we will neglect by (4.25)). Since $|w| \geq \mathsf{depth}(T)-1$, the refined state w, as well as its descendants in T', uniquely determine a next state for every symbol of the alphabet. Moreover, also due to the fact that $|w| \geq$

[4] A count $n'^{(a)}_s$ may differ by 1 from $n_s^{(a)}$ if $s_0 \neq s'_0$.

$\mathsf{depth}(T)-1$, no pseudo-state descends from w in T and no pseudo-state descends from wb in T', with $b \in \mathcal{A}$. Thus, the source of each of these state transitions, which depart from w or one of its descendants in T', is not replaced in the construction of F and F' by a different pseudo-state with more context. Hence, in relation to the mapping between strings and unlabeled paths in G_F of Theorem 4.15, each symbol in the alphabet is associated in state w with a transition to a different pseudo-state. The same is true for the descendants of w in T', which yields the factor $\Pi^{(\ell_w=1)}$ in (4.30). When $\ell_w > 1$, w is always accessed through its forced pseudo-state sequence $\mu_1(w) \to \mu_2(w) \cdots w$. The factor $\binom{F_{\mu_1(w)*}}{N_{*w}}$ in (4.28) represents the number of choices, when $\mu_1(w)$ is the current pseudo-state in our construction of an Eulerian unlabeled path, in selecting w out of any of the remaining states with a forced pseudo-state sequence entry point equal to $\mu_1(w)$, or equal to any ancestor of $\mu_1(w)$ (which are accessible from $\mu_1(w)$ via context-dropping transitions). When w is refined in T', this choice needs to be made at one of the higher context pseudo-states $\mu_1(w)b$ for each refining state wb. This is represented by the factors $\binom{B'_{\mu_1 b,\rho}+N'_{*wb}}{N'_{*wb}}$, which give, when $\mu_1(w)b$ is the current pseudo-state, the number of choices in selecting wb or taking a context-dropping transition to feed any of the remaining states with entry point $\mu_1(w)$, or an ancestor of it. The factor Π_δ accounts for adjustments in the application of the flow conservation equations due to the possible difference between the initial and final state. Appendix D contains the full proof of Lemma 4.22, which formalizes, in detail, this outline.

4.23. Example. Consider a context tree T with states $S_T = \{0, 10, 11\}$, and let T' be the context tree T_1 in Figure 4.1. T' is obtained from T by a single refinement of $w = 10$. We have $\ell_w = 2$, $\mu_1(w) = 0$, and $\mu_2(w) = 10$. Let $s_0 = 10$, $s'_0 = 100$, and $x^n = 001\,001 \cdots 001$. In T, which is FSM, we have $\Xi_T = \Xi_T^\alpha = \binom{2n/3}{n/3}$, as state 0 occurs $2n/3$ times, and it emits $n/3$ zeros, and $n/3$ ones. In T', pseudo-states $\mu_1(w)0 = 00$ and $\mu_1(w)1 = 01$ are added, and the unlabeled path γ' that generates x^n in $G_{F'}$, repeats $n/3$ times the cycle $100 \to 01 \to 0 \to 00 \to 100 \cdots$. Thus, for T', $\Xi'_T = 1$, in agreement with the discussion at the beginning of the chapter. Indeed, looking at γ' we see that $B'_{00,0} = 0$, and therefore the binomial coefficient for $b = 0$ in (4.28) is equal to 1. Since $N'_{*101} = 0$, the binomial coefficient for $b = 1$ in (4.28) is also equal to 1. On the other hand, $\binom{F_{\mu_1(w)*}}{N_{*w}}$ equals $\binom{N_{0*}}{N_{*10}} = \binom{2n/3}{n/3}$, which cancels the factor $\binom{N'_{0*}}{n'^{(1)}_0}$ of $\Xi_T^{\alpha\prime}$. Notice that the empirical entropy rates of x^n with respect to T and T' are $\hat{\mathcal{H}}_T(x^n) = \hat{\mathcal{H}}_{T'}(x^n) = \frac{2}{3}h(\frac{1}{2})$, where the factor $h(\frac{1}{2})$ arises from state 0, where half of the occurring symbols are 0 and half are 1. In T, which is FSM, $\Xi_T = \Xi_T^\alpha \approx \exp(n\hat{\mathcal{H}}_T(x^n))$ as it follows from applying Stirling's formula[5] to $\binom{2n/3}{n/3}$. Although the subsequence of x^n that occurs in the refined state $w = 10$ of T has memoryless empirical entropy equal to zero, its refinement introduces stronger state dependencies in T', as expressed by the factor $\binom{F_{\mu_1(w)*}}{N_{*w}}^{-1}$. In this example, these restrictions cancel the apparent degree of freedom in state 0.

It is readily verified that if T is FSM, all context trees in a refinement sequence for T are FSM. In this case, if T_i is obtained from T_{i-1} by refining a state w, we must have $\ell_w = 1$ in T_{i-1}, for otherwise the suffix $\mathrm{tail}(wb)$ of the state $wb \in S_{T_i}$ would not belong to

[5] See Lemma 4.25 below.

T_i, and T_i would not be FSM. Hence, successively applying Lemma 4.22 along the refinement sequence, the first line of (4.30) is always selected, and we end up with $\Xi_T(x^n) = \Xi_T^\alpha(x^n)$ as expected. When T is not FSM, however, additional factors arise from (4.28), as characterized by Lemma 4.24 below.

4.24. Lemma. *For any context tree T and all sequences x^n,*

$$\Xi_T(x^n) = \Xi_T^\alpha(x^n) \cdot \prod_{i=1}^{R} \Pi_i \,, \tag{4.32}$$

where each factor Π_i has the form of (4.28) applied to consecutive context trees of a refinement sequence for T, and $R = (|E_T| - |V_T|)/(\alpha - 1) - |S_T|$.

Proof. Let $T_1 \cdots T_m$ be a refinement sequence for T. We write $\Xi_T = \Xi_{T_1} \prod_{i=2}^m \Xi_{T_i}/\Xi_{T_{i-1}}$ and we apply Lemma 4.22 to each quotient $\Xi_{T_i}/\Xi_{T_{i-1}}$. Since clearly $\Xi_{T_1} = \Xi_{T_1}^\alpha$, the factors $\Pi^{(\ell_w=1)}$ of (4.30) cancel along the product for nodes w that are internal in T. Hence, $\Xi_T = \Xi_T^\alpha \cdot \prod_{i=1}^R \Pi_i$ where Π_i has the form of (4.28), and R is the number of times a context tree T_i is obtained from T_{i-1} by refining a state w of T_{i-1} with $\ell_w > 1$. Let $E_i = |E_{T_i}|$. We have

$$|E_T| - |V_T| = E_1 + \sum_{i=2}^{m} (E_i - E_{i-1}) - |S_T| \,. \tag{4.33}$$

Suppose T_i is obtained from T_{i-1} by a single refinement of a state w. In order to determine the difference between E_{T_i} and $E_{T_{i-1}}$, we need to analyze edges departing from w in $E_{T_{i-1}}$ and its descendants in E_{T_i}, as well as edges of the form (s, w) in $E_{T_{i-1}}$, or (s, wb) in E_{T_i}. Since $|w| \geq \mathsf{depth}(T_{i-1}) - 1$, w uniquely determines a next state for every symbol. Thus, there are α edges departing from w in $E_{T_{i-1}}$ and also α edges departing from each of the α children of w in E_{T_i}. Hence, edges departing from w in T_{i-1}, and its descendants in T_i, contribute with $\alpha^2 - \alpha = \alpha(\alpha - 1)$ to the term $E_i - E_{i-1}$. We now consider states $s \neq w$, for which there exists an edge (s, w) in $E_{T_{i-1}}$, or (s, wb) in E_{T_i}. From (4.22) we see that this amounts to considering states $s \neq w$ with $s \preceq \mathrm{tail}(w)$, or $\mathrm{tail}(w) \prec s$. Indeed, $s \preceq \mathrm{tail}(wb)$ implies that either $s \preceq \mathrm{tail}(w)$ or $s = \mathrm{tail}(w)b$ in which case $\mathrm{tail}(w) \prec s$. If on the other hand $\mathrm{tail}(wb) \prec s$, then also $\mathrm{tail}(w) \prec s$. Suppose $\ell_w = 1$ in T_{i-1}, and therefore $\mathrm{tail}(w)$ is an internal node of T_{i-1}. In this case, all states $s \neq w$ of T_{i-1} that descend from $\mathrm{tail}(w)$, uniquely determine a next state for the symbol $\mathrm{head}(w)$, both in T_{i-1} and in T_i. Hence, edges departing from these states make no contribution to $E_i - E_{i-1}$, and in this case we have $E_i - E_{i-1} = \alpha(\alpha - 1)$. If on the other hand $\ell_w > 1$, $\mathrm{tail}(w)$ is not an internal node of T_{i-1}. There exists a unique state s of T_{i-1} that satisfies $s \preceq \mathrm{tail}(w)$, and it is of course different from w. The edge (s, w) of $E_{T_{i-1}}$ is replaced by α edges $\{(s, wb) : b \in \mathcal{A}\}$ in E_{T_i}, yielding a contribution of $\alpha - 1$ to the term $E_i - E_{i-1}$. Hence, in this case, $E_i - E_{i-1} = \alpha(\alpha - 1) + \alpha - 1$. Therefore, from (4.33)

$$|E_T| - |V_T| = E_1 + (m-1)\alpha(\alpha - 1) + R(\alpha - 1) - |S_T| \,,$$

and since $E_1 = \alpha$,

$$|E_T| - |V_T| = \alpha(1 + (m-1)(\alpha - 1)) + R(\alpha - 1) - |S_T| \,.$$

Now, by construction of a refinement sequence, $m-1$ equals the number of internal nodes of T, and since T is a full tree, $|S_T| = 1 + (m-1)(\alpha-1)$. Thus,

$$|E_T| - |V_T| = (\alpha-1)|S_T| + R(\alpha-1)\,.$$

□

The following lemma not only yields a connection between $\Xi_T^\alpha(x^n)$ and the empirical entropy rate of x^n with respect to T, but it also will help us to analyze the combinatorial coefficients of the factors Π_i of Lemma 4.24, which have the form of (4.28). Lemma 4.25 is derived with standard methods based on Stirling's formula for factorials. The proof is omitted.

4.25. LEMMA. *Let $m = m_1 + m_2 + \cdots m_k$ with $k \geq 1$, and $m_i > 0$ for all $1 \leq i \leq k$, and let p be the probability vector $p = (\frac{m_1}{m}, \cdots \frac{m_k}{m})$. Then,*

$$mh(p) - \frac{k-1}{2}\log m - O(1) \leq \log \binom{m}{m_1 \cdots m_k} \leq mh(p) - \frac{k-1}{2}\log m - \frac{1}{2}\sum_{i=1}^{k}\log\frac{m_i}{m}\,,$$

where we let $\binom{m}{m_1\cdots m_k} = 1$ if $k = 1$, and we recall that $h(p)$ denotes the binary entropy function.

Lemma 4.25 applied to $\Xi_T^\alpha(x^n)$ gives

$$\log \Xi_T^\alpha \leq \sum_{s:N_{s*}>0} N_{s*}\hat{\mathcal{H}}(x^n|s) - \frac{k_s-1}{2}\log N_{s*} - \frac{1}{2}\sum_{a\in\mathcal{A}}(1-\delta_{n_s^{(a)},0})\log\frac{n_s^{(a)}}{N_{s*}}\,, \tag{4.34}$$

where $\hat{\mathcal{H}}(x^n|s)$ denotes the memoryless empirical entropy of the subsequence of x^n formed by the symbols that occur in state s, and k_s denotes the number of symbols a such that $n_s^{(a)} > 0$. In (4.34) and in the sequel, we follow the convention that if $\delta_{u,v} = 0$ in an expression of the form $\delta_{u,v}f(x^n)$, then the whole expression is valued zero, regardless of $f(x^n)$. Thus, for example, a term $(1-\delta_{n_s^{(a)},0})\log\frac{n_s^{(a)}}{N_{s*}}$ in (4.34) equals zero if $n_s^{(a)} = 0$. Now,

$$-\frac{k_s-1}{2}\log N_{s*} = -\frac{k_s-1}{2}(\log n + \log\frac{N_{s*}}{n}) \leq -\frac{k_s-1}{2}\log n - \frac{1}{2}\sum_{a\in\mathcal{A}}(1-\delta_{n_s^{(a)},0})\log\frac{N_{s*}}{n}\,,$$

where the inequality follows from the fact that $k_s = \sum_{a\in\mathcal{A}}(1-\delta_{n_s^{(a)},0})$. Thus, combining with (4.34), we get

$$\log \Xi_T^\alpha \leq \sum_{s:N_{s*}>0} N_{s*}\hat{\mathcal{H}}(x^n|s) - \frac{k_s-1}{2}\log n - \frac{1}{2}\sum_{a\in\mathcal{A}}(1-\delta_{n_s^{(a)},0})\log\frac{n_s^{(a)}}{n}\,. \tag{4.35}$$

Adding and subtracting $\frac{1}{2}(\alpha-k_s)\log n$, (4.35) becomes

$$\log \Xi_T^\alpha \leq \sum_{s:N_{s*}>0} N_{s*}\hat{\mathcal{H}}(x^n|s) - \frac{\alpha-1}{2}\log n + \frac{1}{2}\sum_{a\in\mathcal{A}}\delta_{n_s^{(a)},0}\log n - \frac{1}{2}\sum_{a\in\mathcal{A}}(1-\delta_{n_s^{(a)},0})\log\frac{n_s^{(a)}}{n}\,. \tag{4.36}$$

For a state s with $N_{s*} = 0$, we have $n_s^{(a)} = 0$ for all $a \in \mathcal{A}$. Thus, $\sum_{a\in\mathcal{A}} \delta_{n_s^{(a)},0} \log n > (\alpha-1)\log n$, and we can further bound $\log \Xi_T^\alpha$ by summing over all states in S_T, rather than only those with $N_{s*} > 0$. We then obtain

$$\log \Xi_T^\alpha \le n\hat{\mathcal{H}}_T(x^n) - \frac{\alpha-1}{2}|S_T|\log n + \frac{1}{2}\sum_{s,a}\delta_{n_s^{(a)},0}\log n - \frac{1}{2}\sum_{s,a}(1-\delta_{n_s^{(a)},0})\log\frac{n_s^{(a)}}{n}\,, \quad (4.37)$$

where we recall that $\hat{\mathcal{H}}_T(x^n)$ is the empirical entropy rate of x^n with respect to T.

When all conditional probabilities in states of T are positive, both summations of (4.37) are shown to be constant bounded in expectation by Lemma 4.26 below. The lemma is essentially proved, with a different notation, in the Appendix of [60].

4.26. Lemma. *Let $\langle T, p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$ and all conditional probabilities positive, and let $y \in \mathcal{A}^+$ be an arbitrary finite string. Then, there exists a constant C such that*

$$E_{\langle T,p_T\rangle}\big[\,\delta_{n_y,0}\log n\,\big] < C\,,$$

and

$$E_{\langle T,p_T\rangle}\big[\,-(1-\delta_{n_y,0})\log\frac{n_y}{n}\,\big] < C\,.$$

The following lemma is an immediate consequence of (4.37), Lemma 4.26, and the fact that the expectation of the empirical entropy rate is upper-bounded by the entropy rate.

4.27. Lemma. *Let $\langle T, p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$ and all conditional probabilities positive. Then,*

$$E_{\langle T,p_T\rangle}\big[\log \Xi_T^\alpha\big] \le n\mathcal{H} - \frac{\alpha-1}{2}|S_T|\log n + O(1).$$

Lemma 4.27 allows us to start putting in place the components of the bound (4.23), which, as we recall from (4.25) and its discussion, reduces essentially to bounding the expectation of $\log \Xi_T(X^n)$. By Lemma 4.24, we have

$$\log \Xi_T(x^n) = \log \Xi_T^\alpha(x^n) + \sum_{i=1}^{R}\log \Pi_i\,, \quad (4.38)$$

where Π_i has the form of (4.28) applied to consecutive context trees of a refinement sequence for T, and $R = (|E_T| - |V_T|)/(\alpha-1) - |S_T|$. Taking expectations, Lemma 4.27 accounts for the first term on the right hand side of (4.38), which gives rise to the entropy rate term of (4.23) and contributes a factor of $\frac{\alpha-1}{2}|S_T|$ to the negative term of order $\log n$ in (4.23). To take care of the summation in the second term on the right hand side of (4.38), we apply Lemma 4.25 to the binomial coefficients of (4.28), obtaining Lemma 4.28 below. The lemma shows that for each index i in the summation of (4.38), $\log \Pi_i$ contributes a term $-\frac{\alpha-1}{2}\log n$ plus terms that we will upper-bound in expectation by means of Lemma 4.26, which will finally yield (4.23).

To state the lemma we define M_b as a shorthand notation for $M_b = B'_{\mu_1 b,\rho} + N'_{*wb}$, where we recall that $B'_{\mu_1 b,\rho} = B'_{\mu_1(w)b,\rho'(\mu_1(w)b)}$, and we also define $Q = F_{\mu_1(w)*} + \delta_{s_n,\mu_1(w)}$.

4.28. Lemma. *Under the same assumptions of Lemma 4.22, we have*

$$\begin{aligned}\log \Pi^{(\ell_w>1)} \;\le\; & -\frac{\alpha-1}{2}\log n + O(1) - \delta_{s_n,\mu_1(w)} \log \frac{Q-N_{*w}}{n} - \sum_{i=1}^{\ell_w-1} (1-\delta_{F'_{*\mu_i(w)},0}) \log \frac{F'_{*\mu_i(w)}}{n} \\ & + \sum_b \left(\frac{1}{2}\delta_{N'_{*wb},0}\log n + \frac{1}{2}\delta_{N'_{*wb},M_b}\log n - \frac{1}{2}(1-\delta_{M_b,0})\log\frac{M_b}{n} \right. \\ & \left. -\frac{1}{2}(1-\delta_{N'_{*wb},0})\log\frac{N'_{*wb}}{n} - \frac{1}{2}(1-\delta_{N'_{*wb},M_b})\log\frac{M_b-N'_{*wb}}{n} \right). \end{aligned} \tag{4.39}$$

Proof.

Let $p_b = N'_{*wb}/M_b$ if $M_b > 0$, and $p_b = 0$ otherwise. For $B'_{\mu_1 b,\rho} > 0$, and $N'_{*wb} > 0$, we have

$$\log \binom{M_b}{N'_{*wb}} \le M_b h(p_b) - \frac{1}{2}\log M_b - \frac{1}{2}\log\frac{N'_{*wb}}{M_b} - \frac{1}{2}\log\frac{M_b-N'_{*wb}}{M_b}\,, \tag{4.40}$$

which, replacing $\log M_b$ by $\log\frac{M_b}{n} + \log n$ becomes,

$$\log \binom{M_b}{N'_{*wb}} \le M_b h(p_b) - \frac{1}{2}\log\frac{M_b}{n} - \frac{1}{2}\log n - \frac{1}{2}\log\frac{N'_{*wb}}{M_b} - \frac{1}{2}\log\frac{M_b-N'_{*wb}}{M_b}\,. \tag{4.41}$$

If one or both of $B'_{\mu_1 b,\rho}$ and N'_{*wb} are zero, then the left hand side of (4.41) vanishes, and also $h(p_b) = 0$. We add the terms $\frac{1}{2}\delta_{N'_{*wb},0}\log n + \frac{1}{2}\delta_{N'_{*wb},M_b}\log n$ to compensate the negative term $-\frac{1}{2}\log n$ and get (4.42) below, which is valid for any value of $B'_{\mu_1 b,\rho}$ and N'_{*wb}.

$$\begin{aligned}\log \binom{M_b}{N'_{*wb}} \;\le\; & M_b h(p_b) - \frac{1}{2}\log n + \frac{1}{2}\delta_{N'_{*wb},0}\log n + \frac{1}{2}\delta_{N'_{*wb},M_b}\log n - \frac{1}{2}(1-\delta_{M_b,0})\log\frac{M_b}{n} \\ & - \frac{1}{2}(1-\delta_{N'_{*wb},0})\log\frac{N'_{*wb}}{M_b} - \frac{1}{2}(1-\delta_{N'_{*wb},M_b})\log\frac{M_b-N'_{*wb}}{M_b}\,. \end{aligned} \tag{4.42}$$

By Lemma 4.13(x), we know that $\log n \ge \log M_b + O(1)$. Hence, we write

$$-\log\frac{N'_{*wb}}{M_b} = -\log\frac{N'_{*wb}}{n} - \log\frac{n}{M_b}\,,$$

where the last term, $-\log\frac{n}{M_b}$, is $O(1)$. Similarly, the term $\frac{M_b-N'_{*wb}}{M_b}$ is $\frac{M_b-N'_{*wb}}{n} + O(1)$. Therefore, (4.42) becomes

$$\begin{aligned}\log \binom{M_b}{N'_{*wb}} \;\le\; & M_b h(p_b) - \frac{1}{2}\log n + O(1) \\ & + \frac{1}{2}\delta_{N'_{*wb},0}\log n + \frac{1}{2}\delta_{N'_{*wb},M_b}\log n - \frac{1}{2}(1-\delta_{M_b,0})\log\frac{M_b}{n} \\ & -\frac{1}{2}(1-\delta_{N'_{*wb},0})\log\frac{N'_{*wb}}{n} - \frac{1}{2}(1-\delta_{N'_{*wb},M_b})\log\frac{M_b-N'_{*wb}}{n}\,. \end{aligned} \tag{4.43}$$

We now turn to the denominator $\binom{F_{\mu_1(w)*}}{N_{*w}}$ of (4.28). We recall the shorthand notation Q defined as $Q = F_{\mu_1(w)*} + \delta_{s_n,\mu_1(w)}$. Then, the logarithm of $\binom{F_{\mu_1(w)*}}{N_{*w}}$ is,

$$\log \binom{F_{\mu_1(w)*}}{N_{*w}} = \log \binom{Q}{N_{*w}} + \delta_{s_n,\mu_1(w)} \log\frac{Q-N_{*w}}{Q}\,, \quad N_{*w} < Q\,. \tag{4.44}$$

Defining $q = N_{*w}/Q$ if $Q > 0$, and $q = 0$ otherwise, Lemma 4.25 yields, for $0 < N_{*w} < Q$,

$$\log \binom{F_{\mu_1(w)*}}{N_{*w}} \geq Qh(q) - \frac{1}{2}\log Q - O(1) + \delta_{s_n,\mu_1(w)} \log \frac{Q - N_{*w}}{Q}\,. \tag{4.45}$$

By Lemma 4.22, we know that $N_{*w} \leq F_{\mu_1(w)*}$ and, therefore, we have $N_{*w} \leq Q$, by the definition of Q. Since in the special case $0 < N_{*w} = Q$ the last term of (4.45) is zero if $\delta_{s_n,\mu_1(w)} = 0$ and $-\infty$ otherwise, (4.45) is in fact valid for $0 < N_{*w} \leq Q$.

If one or both of N_{*w} and Q is zero, then the left hand side of (4.45) vanishes, and also $h(q) = 0$. The last term of (4.45) also vanishes in this case, since for $Q = 0$ we must have $\delta_{s_n,\mu_1(w)} = 0$ by the definition of Q, and for $N_{*w} = 0$ with $Q > 0$ we have $\log \frac{Q-N_{*w}}{Q} = 0$. Thus, canceling the term $-\frac{1}{2}\log Q$ when $Q = 0$ with a factor $(1 - \delta_{Q,0})$, we get the following equation, which is valid for any value of N_{*w} and Q.

$$\log \binom{F_{\mu_1(w)*}}{N_{*w}} \geq Qh(q) - (1 - \delta_{Q,0})\frac{1}{2}\log Q - O(1) + \delta_{s_n,\mu_1(w)} \log \frac{Q - N_{*w}}{Q}\,. \tag{4.46}$$

Since $Q = F_{\mu_1(w)*} + \delta_{s_n,\mu_1(w)}$ equals $F_{*\mu_1(w)}$ by Lemma 4.13(ix), we know that $\log n \geq \log Q$ by Part (x) of the same lemma. Therefore, for $n > 0$ we have

$$\log \binom{F_{\mu_1(w)*}}{N_{*w}} \geq Qh(q) - \frac{1}{2}\log n - O(1) + \delta_{s_n,\mu_1(w)} \log \frac{Q - N_{*w}}{n}\,. \tag{4.47}$$

Notice that, since $N_{*w} = \sum_b N'_{*wb}$, we have $Q = \sum_b M_b$ by Lemma 4.25. Hence, for $Q > 0$, we have $q = \sum_b p_b \frac{M_b}{Q}$, and we get, by Jensen's inequality,

$$Qh(q) \geq \sum_b M_b h(p_b)\,, \quad Q > 0\,. \tag{4.48}$$

If $Q = 0$, then $M_b = 0$ for all $b \in \mathcal{A}$, and the inequality (4.48) still holds (with equality). Therefore, summing (4.43) over all $b \in \mathcal{A}$ and subtracting (4.47) we get, from (4.28),

$$\begin{aligned} \log \Pi^{(\ell_w > 1)} \;\leq\; & -\frac{\alpha - 1}{2}\log n + O(1) + \log \Pi_\delta - \delta_{s_n,\mu_1(w)} \log \frac{Q - N_{*w}}{n} \\ & + \sum_b \left(\frac{1}{2}\delta_{N'_{*wb},0} \log n + \frac{1}{2}\delta_{N'_{*wb},M_b} \log n - \frac{1}{2}(1 - \delta_{M_b,0}) \log \frac{M_b}{n} \right. \\ & \left. - \frac{1}{2}(1 - \delta_{N'_{*wb},0}) \log \frac{N'_{*wb}}{n} - \frac{1}{2}(1 - \delta_{N'_{*wb},M_b}) \log \frac{M_b - N'_{*wb}}{n} \right)\,. \end{aligned} \tag{4.49}$$

As for Π_δ, defined in (4.29), we have

$$\log \Pi_\delta = \sum_{i \in L} \delta_{s_n,\mu_i(w)} \log \frac{F_{*\mu_i(w)}}{F'_{*\mu_i(w)}}\,, \tag{4.50}$$

and by Lemma 4.13(x), we get

$$\log \Pi_\delta \leq \sum_{i \in L} \delta_{s_n,\mu_i(w)} \log \frac{n}{F'_{*\mu_i(w)}}\,. \tag{4.51}$$

Notice that $s_n = \mu_i(w)$ implies that $s'_n = \mu_i(w) \neq s'_0$, for $|\mu_i(w)| < |w|$, and this in turn implies that $F'_{*\mu_i(w)} > 0$, by Lemma 4.13(ix). Hence, we get

$$\log \Pi_\delta \leq - \sum_{i=1}^{\ell_w - 1} (1 - \delta_{F'_{*\mu_i(w)},0}) \log \frac{F'_{*\mu_i(w)}}{n} \,, \tag{4.52}$$

and substituting in (4.49) we get (4.39). □

Remark. From (4.37) and (4.38), recalling that $R = (|E_T| - |V_T|)/(\alpha - 1) - |S_T|$ in (4.38), we obtain, by (4.39),

$$\log |\mathcal{T}^*(x^n)| \leq \log \Xi_T(x^n) \leq n\hat{\mathcal{H}}_T(x^n) - \frac{|E_T| - |V_T|}{2} \log n + f(\mathcal{T}^*(x^n)) \,, \tag{4.53}$$

where the first inequality comes from (4.25), and $f(\mathcal{T}^*(x^n))$ is a summation of terms of the form $\delta_{z,z'} \log n$, coming from (4.39) and (4.37), with $\delta_{z,z'}$ representing a zero/one valued condition on certain counts of the matrix F. Equation (4.53) gives an upper bound on $\log |\mathcal{T}^*(x^n)|$ valid for every individual sequence. While this bound is tight to the main term $n\hat{\mathcal{H}}_T(x^n)$ for FSM trees, we have already observed that, in general, $\log |\mathcal{T}^*(x^n)|$ may be much smaller than $n\hat{\mathcal{H}}_T(x^n)$ for some sequences. This gap between the right hand side of (4.53) and $\log |\mathcal{T}^*(x^n)|$ is explained by the difference between both sides of (4.48), respectively $\sum_b M_b h(p_b)$ and $Qh(q)$, which come from taking logarithms and applying Lemma 4.25 to the numerator and denominator of the quotient that defines $\Pi^{(\ell_w > 1)}$ in (4.28). As observed in Example 4.23, the denominator in (4.28) may be large. Equation (4.39), and thus (4.53), could be sharpened by including on the right hand side a term of the form $\sum_b M_b h(p_b) - Qh(q)$, which was neglected in Lemma 4.28. Although, as we shall see, a term of this form has no effect on the main terms of the expectation of $\log |\mathcal{T}^*(x^n)|$, for individual sequences, the magnitude of a negative term of the form $-Qh(q)$ may be comparable to $n\hat{\mathcal{H}}_T(x^n)$, as we saw in Example 4.23. Besides the intuition for $\Pi^{(\ell_w > 1)}$ given in our discussion following Lemma 4.22 in terms of the pseudo-state transition graph, we currently have no obvious interpretation of the difference $\sum_b M_b h(p_b) - Qh(q)$ in terms of symbol occurrence counts of x^n in the states of T. It is not clear whether there exists a more elementary asymptotic expression for the logarithm of the type class size for individual sequences, and we leave it as an open question.

If for each term of (4.39) of the form $\delta_{z,0} \log n$, and $-(1 - \delta_{z,0}) \log \frac{z}{n}$, there existed a pattern string y such that $z \geq n_y(x^n)$, then we could apply Lemma 4.26 to upper-bound the expectation of these terms by a constant, for tree models with all conditional probabilities positive. We will show that this condition indeed holds for canonical context trees. We point out that this is not necessarily true, in general, and here we make full use of Corollary 4.21, by which we assume without loss of generality that the context tree for which we analyze the formula (4.15) of Theorem 4.15 is canonical. Consider a forgetful state s of T, i.e., $as \in \mathcal{I}(T)$ for all $a \in \mathcal{A}$. Clearly, for all $j < n$ such that $s_j = s$, we have $j \notin J(x^n)$ for $\ell_{s_{j+1}} > 1$. Also, since $as \notin U$ for all $a \in \mathcal{A}$, we know that $\tilde{\Delta}^+(t)_{s*} = 0$ for all states t. Hence, by Lemma 4.13(vii), we have $D_{s*} = 0$. Since also $B_{s*} = 0$ by definition, for $s \in S_T$, we conclude

that $F_{s*} = 0$. Thus, by Lemma 4.13(ix), noting that $s_0 \neq s$ for s_0 is of maximal depth, we get $F_{*s} = \delta_{s_n,s}$. As a consequence, if $\rho(u) = s$, then $B_{u,\rho(u)}(x^n) \leq 1$ for all strings x^n. Therefore, if a context tree is not canonical we may find situation in which, for example, $M_b - N'_{*wb} \leq 1$ for all strings x^n.

For a canonical context tree T and pseudo-states u, v we show next that either $F_{u,v}(x^n) = 0$ for all strings x^n, or there exists a pattern string y such that $F_{u,v}(x^n) \geq n_y(x^n)$. We define the *pseudo-state transition support graph*, $G_T^{(\mathrm{F})} = (V_T^{(\mathrm{F})}, E_T^{(\mathrm{F})})$, for an arbitrary context tree T, as follows: take $V_T^{(\mathrm{F})} = U$, and $E_T^{(\mathrm{F})}$ comprised of all the edges (u, v) of the form

- $u = \mu_i(s), v = \mu_{i+1}(s)$ for some $s \in S_T$, $1 \leq i < \ell_s$.
- $u \in S_T$ and $v = \tau(u, b)$ for some $b \in \mathcal{A}$.
- $v = \rho(u)$ and $\mathrm{tail}(u) \in T$.

4.29. Lemma. *If T is canonical, then for every $(u, v) \in E_T^{(F)}$ there exists a fixed string y such that $F_{u,v}(x^n) \geq n_y(x^n)$.*

The proof is deferred to Appendix D. Notice that whenever $F_{u,v}(x^n)$ is positive, (u, v) must be an edge of $E_T^{(\mathrm{F})}$ by parts (vii) and (xi) of Lemma 4.13. Thus, the incidence matrix of the graph $G_T^{(\mathrm{F})}$ is the support matrix of F when T is canonical. Lemma 4.29, together with Lemma 4.28, allows now for bounding the expected logarithm of the factors Π_i in Lemma 4.24.

4.30. Lemma. *Let $\langle \tilde{T}, p_{\tilde{T}} \rangle$ be a tree source with all conditional probabilities positive. Let T' and T be consecutive context trees in a refinement sequence for $\tilde{T}$, such that T' is obtained from T by a single refinement of a state w with $\ell_w > 1$ in T. Then, if T and T' are canonical, the factor $\Pi^{(\ell_w > 1)}$ in (4.28) satisfies*

$$E_{\langle \tilde{T}, p_{\tilde{T}} \rangle} \left[\log \Pi^{(\ell_w > 1)} \right] \leq -\frac{\alpha - 1}{2} \log n + O(1) .$$

Proof.

The terms $\delta_{N'_{*wb},0} \log n$ and $-(1 - \delta_{N'_{*wb},0}) \log \frac{N'_{*wb}}{n}$ in (4.39), have constant expectation by Lemma 4.26. Also, since $M_b \geq N'_{*wb}$ by the definition of M_b, the term $-(1 - \delta_{M_b,0}) \log \frac{M_b}{n}$ is bounded from above by $-(1 - \delta_{N'_{*wb},0}) \log \frac{N'_{*wb}}{n} + \delta_{N'_{*wb},0} \log n$, where the second term accounts for those cases in which $N'_{*wb} = 0$ but $M_b > 0$. Hence, the term $-(1 - \delta_{M_b,0}) \log \frac{M_b}{n}$ in (4.39) has also constant bounded expectation.

Notice that $\mu_1(w)b \in U'$ for all $b \in \mathcal{A}$, as it belongs to the forced pseudo-state sequence of the state wb. Also $\mathrm{tail}(\mu_1(w)b) \in T'$ since $\mathrm{tail}(\mu_1(w))$ is an internal node of T by (4.6). Hence, $(\mu_1(w)b, \rho'(\mu_1(w)b)) \in E_{T'}^{(\mathrm{F})}$, and if T' is canonical, by Lemma 4.29, there exists a pattern string z_b such that

$$B'_{\mu_1 b,\rho} \geq n_{z_b}(x^n) .$$

Then, recalling that $M_b = B'_{\mu_1 b,\rho} + N'_{*wb}$, we see that $\delta_{N'_{*wb},M_b} = 1$ implies $B'_{\mu_1 b,\rho} = 0$, which in turn implies $\delta_{n_{z_b},0} = 1$. Hence, the expectation of the term $\delta_{N'_{*wb},M_b} \log n$ in (4.39) is $O(1)$,

since it is bounded from above by $\delta_{n_{z_b},0} \log n$. Also the term $-(1-\delta_{N'_{*wb},M_b}) \log \frac{M_b - N'_{*wb}}{n}$, from (4.39), is bounded as

$$-(1-\delta_{N'_{*wb},M_b}) \log \frac{M_b - N'_{*wb}}{n} \le -(1-\delta_{n_{z_b},0}) \log \frac{n_{z_b}}{n} + \delta_{n_{z_b},0} \log n \,,$$

where the last term accounts for those cases in which $n_{z_b} = 0$ but $B'_{\mu_1 b,\rho} > 0$. As a consequence, the term $-(1-\delta_{N'_{*wb},M_b}) \log \frac{M_b - N'_{*wb}}{n}$ is also constant bounded in expectation.

We now analyze the term $-\delta_{s_n,\mu_1(w)} \log \frac{Q-N_{*w}}{n}$, from (4.39). When $\mu_1(w) \notin U'$, $\mu_1(w)$ is not a state of T and therefore $\delta_{s_n,\mu_1(w)}$ is constantly equal to zero. If on the other hand $\mu_1(w) \in U'$, we recall that $Q = F_{\mu_1(w)*} + \delta_{s_n,\mu_1(w)}$ and, by Lemma 4.22, we have

$$Q - N_{*w} = \Sigma = \sum_{b \in \mathcal{A}} B'_{\mu_1 b,\rho} \,.$$

If $s_n = \mu_1(w)$, also $s'_n = \mu_1(w)$ for $\mu_1(w) \neq w$. Hence, $F'_{*\mu_1(w)} > 0$ by Lemma 4.13(ix). Since $\mathrm{tail}(\mu_1(w))$ is an internal node of T by (4.6), and a fortiori it is an internal node of T', $\tilde{\Delta'}^{+}(s)_{*\mu_1(w)} = 0$ for all states s of T'. Moreover, $\mu_1(w)$ has a full complement of children, $\{\mu_1(w)b : b \in \mathcal{A}\}$, in U', and therefore there is no state s of T' such that $\tau'(s,a) = \mu_1(w)$. As a consequence, by Lemma 4.13(vii), we have $D'_{*\mu_1(w)} = 0$ and therefore $F'_{*\mu_1(w)} = B'_{*\mu_1(w)} = \Sigma$. We conclude that if $s_n = \mu_1(w)$, then $\Sigma > 0$. Thus, we have

$$-\delta_{s_n,\mu_1(w)} \log \frac{Q - N_{*w}}{n} \le -(1-\delta_{\Sigma,0}) \log \frac{\Sigma}{n} \,.$$

Fixing any symbol $b \in \mathcal{A}$, and taking z_b such that $B'_{\mu_1 b,\rho} \ge n_{z_b}(x^n)$, we further bound

$$-(1-\delta_{\Sigma,0}) \log \frac{\Sigma}{n} \le -(1-\delta_{n_{z_b},0}) \log \frac{n_{z_b}}{n} + \delta_{n_{z_b},0} \log n \,,$$

where the second term accounts for those cases in which $n_{z_b} = 0$ but $\Sigma > 0$. Hence, the term $-\delta_{s_n,\mu_1(w)} \log \frac{Q-N_{*w}}{n}$ in (4.39) has constant bounded expectation.

It remains to bound the expectation of the term of (4.39) given by

$$-\sum_{i=1}^{\ell_w - 1} (1-\delta_{F'_{*\mu_i(w)},0}) \log \frac{F'_{*\mu_i(w)}}{n} \,. \tag{4.54}$$

Consider a pseudo-state $v = \mu_i(w)$ of T with $i < \ell_w$. If v is not a pseudo-state of T', the term $(1-\delta_{F'_{*\mu_i(w)},0}) \log \frac{F'_{*\mu_i(w)}}{n}$ of (4.54) is constantly equal to zero. If $v \in U'$, we claim that there exists $v' \in U'$ such that $(v',v) \in E^{(\mathrm{F})}_{T'}$. Since $v \in U'$, there exists a state t of T' such that $v = \mu_j(t)$ for some $1 \le j \le \ell_t$. Let $u = \mathrm{tail}(v)$. If $u \notin T'$, then $u = \mu_{j-1}(t)$ and $(u,v) \in E^{(\mathrm{F})}_{T'}$. If otherwise $u \in T'$, let s be a state of T' such that $u \preceq s$, and let $a = \mathrm{head}(v)$. Then, either $\tau'(s,a) = v$, in which case $(s,v) \in E^{(\mathrm{F})}_{T'}$, or $\tau'(s,a) \in \bar{\Lambda}(v)$, in which case there exists v' such that $v \prec v' \preceq \tau'(s,a)$ and $\rho'(v') = v$. Since $\mathrm{tail}(v') \preceq s$, we have $(v',v) \in E^{(\mathrm{F})}_{T'}$. The claim is proved, and therefore, by Lemma 4.29, there exists a pattern string y_i such that $F'_{*\mu_i(w)} \ge n_{y_i}(x^n)$. Hence, $(1-\delta_{F'_{*\mu_i(w)},0}) \log \frac{F'_{*\mu_i(w)}}{n}$ is bounded from above

by $(1-\delta_{n_{y_i},0})\log\frac{n_{y_i}}{n}+\delta_{n_{y_i},0}\log n$, where the second term accounts for those cases in which $n_{y_i}=0$ but $F'_{*\mu_i(w)}>0$. □

Lemmas 4.24, 4.27, and 4.30 yield the following bound on the expectation of $\log\Xi_T$, valid for a canonical context tree.

4.31. Lemma. *Let $\langle T,p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$, such that T is canonical and all conditional probabilities are positive. Then,*

$$E_{\langle T,p_T\rangle}\big[\log\Xi_T\big]\le n\mathcal{H}-\frac{|E_T|-|V_T|}{2}\log n+O(1)\,. \tag{4.55}$$

Proof. By Lemma 4.24 and Lemma 4.27,

$$\mathrm{E}_{\langle T,p_T\rangle}\big[\log\Xi_T\big]\le n\mathcal{H}-\frac{\alpha-1}{2}|S_T|\log n+O(1)+\sum_{i=1}^{R}\mathrm{E}_{\langle T,p_T\rangle}\big[\log\Pi_i\big]\,,$$

where $R=(|E_T|-|V_T|)/(\alpha-1)-|S_T|$, and each Π_i has the form of (4.28) applied to consecutive context trees of a refinement sequence for T. By Lemma 4.30, it then suffices to show that all subtrees in a refinement sequence for T are canonical. Indeed, in this case, $\mathrm{E}_{\langle T,p_T\rangle}\big[\log\Pi_i\big]\le-\frac{\alpha-1}{2}\log n+O(1)$ and therefore

$$\mathrm{E}_{\langle T,p_T\rangle}\big[\log\Xi_T\big]\le n\mathcal{H}-\frac{\alpha-1}{2}|S_T|\log n-R\frac{\alpha-1}{2}\log n+O(1)\,,$$

from which (4.55) follows.

Suppose a context tree T_i of a refinement sequence for T is not canonical. Then, there exists a state s of T_i such that as is an internal node of T_i for every symbol a. Thus, we have $\mathsf{depth}(T_i)>|s|+1$, and a fortiori, $\mathsf{depth}(T_j)>|s|+1$ for all $j>i$. Hence, s is never refined after step i of the refinement sequence. This implies that s is also a state of T, leading to a contradiction since T is canonical. □

The right hand side of (4.55) in Lemma 4.31 already gives an upper bound on $\mathrm{E}_{\langle T,p_T\rangle}\big[\log|\mathcal{T}^*(X^n)|\big]$ for T canonical, as $M\le 1$ in the formula (4.15) of Theorem 4.15. We next show that imposing a fixed final state on the strings of $\mathcal{T}(x^n)$, as in the definition of $\mathcal{T}^*(x^n)$, does not significantly diminish the size of $\mathcal{T}(x^n)$. Therefore, $\log|\mathcal{T}(x^n)|$ and $\log|\mathcal{T}^*(x^n)|$ have asymptotically the same expectation. The proof amounts to bounding $|\mathcal{T}(x^n)|$ by $\Xi_T(x^n)$ affected by a factor $\pi(x^n)$, whose logarithm is shown to be constant bounded in expectation. Once again we apply Lemma 4.29 to bound the expectation of terms of the form $-\log\frac{z}{n}$.

4.32. Lemma. *Let $\langle T,p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$, such that T is canonical and all conditional probabilities are positive. Then,*

$$E_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(X^n)|\big]\le n\mathcal{H}-\frac{|E_T|-|V_T|}{2}\log n+O(1)\,. \tag{4.56}$$

Proof. We claim that $|\mathcal{T}(x^n)| \le \pi(x^n)\Xi_T(x^n)$, with $\pi(x^n) = O\left(\prod_u \frac{n}{F_{u*}+1} \prod_{u,v} \frac{n}{F_{u,v}+1}\right)$, where the indexes u and v take values in all pseudo-states such that $F_{u,v}(y^n)$ is positive for some string y^n. By Corollary 4.3, $\mathcal{T}(x^n) = \mathcal{T}^*(x^n)$ for all strings x^n such hat $s_{n-1}(x^n)$ and x_n determine $s_n(x^n)$. Hence, $|\mathcal{T}(x^n)| \le \Xi_T(x^n)$ for all such strings, as $M \le 1$ in (4.15). Otherwise, let x^n be a string with final state $s_n(x^n) = s$, such that $s_{n-1}(x^n)$ and x_n do not determine $s_n(x^n)$. Again by Corollary 4.3 we have $|\mathcal{T}(x^n)| = |\mathcal{T}^*(x^{n-1})| \le \Xi_T(x^{n-1})$. Now, $N(x^n) - N(x^{n-1}) = \mathbf{1}_{u,s_n}$ and, by (4.10), $K(x^n) - K(x^{n-1}) = \mathbf{1}_{u,s_n} + \Theta(s_n)$. Also since $d(u, s_n) = 0$ by definition, we get from (4.11),

$$D(x^n) - D(x^{n-1}) = \mathbf{1}_{u,s_n} + \Theta(s_n) + \sum_{t,w \in S_T} \Theta(s_n)_{t,w} d(t,w)\,.$$

Hence, given the final state $s = s_n(x^n)$, the difference $D(x^n) - D(x^{n-1})$ is a constant independent of n and of x^n. From the definition of B in (4.12), we see that the difference $B(x^n) - B(x^{n-1})$ is determined by $D(x^n) - D(x^{n-1})$, and therefore it is also constant for the given final state s. As a consequence, for each state s such that $\mathrm{tail}(s) \notin T$ (i.e., $s_{n-1}(x^n)$ and x_n do not determine $s_n(x^n)$ if $s_n(x^n) = s$), there exists a constant matrix ε, such that $F(x^{n-1}) = F(x^n) + \varepsilon$ for all strings x^n with final state s. Then, denoting $F = F(x^n)$, we have

$$\begin{aligned}\frac{\Xi_T(x^{n-1})}{\Xi_T(x^n)} &= \frac{\prod_i (F_{i*}+\varepsilon_{i*})!}{\prod_{i,j}(F_{i,j}+\varepsilon_{i,j})!} \Bigg/ \frac{\prod_i F_{i*}!}{\prod_{i,j} F_{i,j}!} \\ &= \frac{\prod_i (F_{i*}+\varepsilon_{i*})!}{\prod_i F_{i*}!} \cdot \frac{\prod_{i,j} F_{i,j}!}{\prod_{i,j}(F_{i,j}+\varepsilon_{i,j})!}\,.\end{aligned}$$

Defining $\Phi^+ = \{i : \varepsilon_{i*} > 0\}$, $\Phi^- = \{i : \varepsilon_{i*} \le 0\}$, $\Omega^+ = \{(i,j) : \varepsilon_{i,j} > 0\}$, and $\Omega^- = \{(i,j) : \varepsilon_{i,j} \le 0\}$, we can write

$$\frac{\Xi_T(x^{n-1})}{\Xi_T(x^n)} = \frac{\prod_{i\in\Phi^+}\prod_{k=1}^{\varepsilon_{i*}}(F_{i*}+k)}{\prod_{i\in\Phi^-}\prod_{k=0}^{-\varepsilon_{i*}-1}(F_{i*}-k)} \cdot \frac{\prod_{(i,j)\in\Omega^-}\prod_{k=0}^{-\varepsilon_{i,j}-1}(F_{i,j}-k)}{\prod_{(i,j)\in\Omega^+}\prod_{k=1}^{\varepsilon_{i,j}}(F_{i,j}+k)}\,. \tag{4.57}$$

For each i, there are $-\sum_{(i,j)\in\Omega^-}\varepsilon_{i,j}$ factors of the form $F_{i,j} - k$ in the numerator, and $\sum_{(i,j)\in\Omega^+}\varepsilon_{i,j}$ factors of the form $F_{i,j} + k$ in the denominator. Since $\varepsilon_{i*} = \sum_{(i,j)\in\Omega^-}\varepsilon_{i,j} + \sum_{(i,j)\in\Omega^+}\varepsilon_{i,j}$, the total number of factors in the numerator of (4.57) equals the total number of factors in the denominator. We then have,

$$\frac{\Xi_T(x^{n-1})}{\Xi_T(x^n)} = \frac{\prod_{i\in\Phi^+}\prod_{k=1}^{\varepsilon_{i*}}(F_{i*}+k)/n}{\prod_{i\in\Phi^-}\prod_{k=0}^{-\varepsilon_{i*}-1}(F_{i*}-k)/n} \cdot \frac{\prod_{(i,j)\in\Omega^-}\prod_{k=0}^{-\varepsilon_{i,j}-1}(F_{i,j}-k)/n}{\prod_{(i,j)\in\Omega^+}\prod_{k=1}^{\varepsilon_{i,j}}(F_{i,j}+k)/n}\,.$$

By Lemma 4.13(x), $F_{t*} \le n$ for all $t \in U$, and of course also $F_{t*} + \varepsilon_{t*} = F_{t*}(x^{n-1}) \le n$. Hence, all factors $F_{i*} \pm k$ and $F_{i,j} \pm k$, are bounded from above by n, and therefore,

$$\frac{\Xi_T(x^{n-1})}{\Xi_T(x^n)} \le \prod_{i\in\Phi^-}\prod_{k=0}^{-\varepsilon_{i*}-1}\frac{n}{F_{i*}-k} \cdot \prod_{(i,j)\in\Omega^+}\prod_{k=1}^{\varepsilon_{i,j}}\frac{n}{F_{i,j}+k}\,,$$

which we can further bound by

$$\frac{\Xi_T(x^{n-1})}{\Xi_T(x^n)} \leq \prod_{i\in\Phi^-} -\varepsilon_{i*}\frac{n}{F_{i*}+\varepsilon_{i*}+1} \cdot \prod_{(i,j)\in\Omega^+} \varepsilon_{i,j}\frac{n}{F_{i,j}+1}\,.$$

The factor $\frac{n}{F_{i*}+\varepsilon_{i*}+1}$ is $\frac{n}{F_{i*}+1}\frac{F_{i*}+1}{F_{i*}+\varepsilon_{i*}+1}$, and the latter is bounded from above by $\frac{n}{F_{i*}+1}(-\varepsilon_{i*}+1)$. Hence, we get

$$\Xi_T(x^{n-1})/\Xi_T(x^n) = O\left(\prod_{i\in\Phi^-}\frac{n}{F_{i*}+1}\prod_{(i,j)\in\Omega^+}\frac{n}{F_{i,j}+1}\right).$$

The claim then follows since this is valid for all possible final states, and there are finitely many of them.

Now, since T is canonical, Lemma 4.29 states that for each u, v such that $F_{u,v}(y^n)$ is positive for some string y^n, there exists a string z such that $F_{u,v}(x^n) \geq n_z(x^n)$. As a consequence,

$$\log\frac{n}{F_{u*}(x^n)+1} \leq \log\frac{n}{F_{u,v}(x^n)+1} \leq \log\frac{n}{n_z(x^n)+1}\,,$$

which has constant bounded expectation by Lemma 4.26. □

Lemma 4.32 and Corollary 4.21 yield Theorem 4.18.

4.4 The number of type classes

We study the number of type classes induced on $\mathcal{A}^n$ by a context tree T, i.e., the number of T-classes. The main result of this section establishes a formula that is asymptotically tight up to multiplication by a constant. We can equivalently address the problem of counting the number of T-classes*, denoted $\mathcal{N}_T$, since, by the definition of close-ended type class, every T-class is subdivided in up to a constant number of T-classes*, one for each possible final state. The following theorem presents the main result of the section.

4.33. Theorem. *Let T be a context tree, and let T_c be the minimal canonical extension of T. Then, $\mathcal{N}_T = \Theta\left(n^{|E_{T_c}|-|V_{T_c}|}\right)$.*

Once again, when T is FSM, $T = T_c$, and $|E_{T_c}|-|V_{T_c}| = (\alpha-1)|S_T|$, in agreement with known results for FSMs [83].

We develop some needed tools, based on graph theory, which will be used in the proof of Theorem 4.33. Again, we loosely follow [5]. Consider a graph $G = (V, E)$. A *chain* is an alternating sequence of vertices and edges $v_1, e_1, v_2, e_2 \cdots v_m, e_m, v_{m+1}$ satisfying either $e_i = (v_i, v_{i+1})$ or $e_i = (v_{i+1}, v_i)$. We recall that we allow multiple parallel, distinguishable, directed edges between the same pair of nodes. A chain is *closed* if $v_1 = v_{m+1}$, it is *simple* if $e_i \neq e_j$ for $i \neq j$, and it is *elementary* if all vertices are different except possibly for v_1 and v_{m+1} that may coincide. The number of edges in a chain is called the *length* of the chain. A *cycle* is a closed simple chain. Notice that a path, as defined at the end of Section 4.1, corresponds

to a chain where every edge is traversed in the forward direction, i.e., $e_i = (v_i, v_{i+1})$ for all $i = 1 \cdots m$. Similarly, a circuit is a cycle where every edge is traversed in the forward direction.

We say that a graph is *connected* if any two vertices, u, w, can be joined with a chain $u = v_1, e_1, v_2, e_2 \cdots v_m, e_m, v_{m+1} = w$. We say that graph is *strongly connected* if for any two vertices, u, w, there exists a path from u to v and also a path from v to u. A graph is a *tree* if it is connected and has no cycles.[6] A *spanning tree* of $G = (V, E)$ is a tree, $G' = (V, E')$, with the same set of vertices as G and with $E' \subset E$. If a tree has a vertex, u, such that there exists a path from v to u for all vertices v, then u is called a *sink* of the tree (a tree can have at most one sink).

We associate to a chain in G, $\gamma = v_1, e_1 \cdots v_m, e_m, v_{m+1}$, a vector $\zeta(\gamma)$ from $\mathbb{Z}^{|E|}$ indexed with elements from E defined as

$$\zeta(\gamma)_e = |\{i = 1 \cdots m : e = (v_i, v_{i+1})\}| - |\{i = 1 \cdots m : e = (v_{i+1}, v_i)\}| .$$

The subspace of $\mathbb{R}^{|E|}$ spanned by $\{\zeta(\gamma) : \gamma \text{ is a cycle }\}$ is called the *cycle space* of G and it is known to have dimension $|E| - |V| + 1$ for strongly connected graphs [5]. A *circuit basis* for the cycle space is a basis formed by vectors $\zeta(c_i)$ where every c_i is an elementary circuit. We will make use of the following result from [5].

4.34. Proposition. *[5] Every strongly connected graph has a circuit basis.*

Let $G = (V, E)$ be a 1-graph. We recall that $\mathbb{Z}$, $\mathbb{Z}_{\geq 0}$, and $\mathbb{Z}_{>0}$ denote the integers, the nonnegative integers, and the positive integers, respectively. An *assignment of counters* for G is a vector in $\mathbb{Z}_{\geq 0}^{|E|}$, indexed by elements of E. The assignment η is said to be *cyclic* if and only if it satisfies the flow conservation equations $\sum_{e=(u,v)} \eta_e = \sum_{e=(v,w)} \eta_e$ for all $v \in V$; it is said to be *connected* if eliminating edges e with $\eta_e = 0$ from E results in a strongly connected graph. Every state sequence $s_0(x^n), s_1(x^n) \cdots s_n(x^n)$, with $s_n = s_0$ determines a closed path in the state transition support graph G_T. Therefore, in this case, $\eta_{(u,v)} = N(x^n)_{u,v} \forall (u, v) \in E_T$ is a cyclic assignment of counters for G_T, where we recall that $N(x^n)$ denotes the state transition matrix of x^n. Furthermore, if all states are visited in the state sequence, the assignment of counters is also connected. The key idea in the proof of Theorem 4.33 is to establish a correspondence between cyclic connected assignment of counters in a graph, and T_c-classes* with $s_n = s_0$ (the condition $s_n = s_0$ can then be removed easily without affecting the asymptotics of the theorem). To establish the correspondence, we need a few more definitions.

For a 1-graph $G = (V, E)$, a *weight function* ψ assigns a nonnegative integer weight to each edge of G. Notice that weight functions and assignment of counters are the same kind of objects as both assign a nonnegative integer to each edge of a 1-graph. We make the distinction, however, since they are intended for conceptually different purposes. We will regard weight functions as fixed objects that depend only on the context tree T. On the other

[6] Since the direction of an edge is not relevant for the construction of chain or a cycle, these notions of connected graph and tree coincide with the usual definitions for non directed graphs. Our derivations, however, will still be based on directed graphs.

hand, we will associate assignments of counters satisfying certain properties to T_c-classes*, and we will bound the number of such assignments. We extend a weight function ψ to paths by defining the weight of a path as the sum of the weights of its edges. We also define the weight of an assignment of counters η as $\psi(\eta) = \sum_{e\in E} \psi(e)\eta_e$.

The following lemma will be the main tool for the proof of Theorem 4.33.

4.35. Lemma. *Let $G = (V, E)$ be a strongly connected 1-graph and ψ a weight function for G such that G has no circuits of weight zero and at least one circuit of weight one. Then,*

(**i**) *The number of cyclic connected assignments of counters of weight n for G is bounded from below, for large n, by $Cn^{|E|-|V|}$ where C depends only on G and ψ.*

(**ii**) *A cyclic assignment of counters η, of weight n, is fully determined by the values η_e for a set E^* of $|E| - |V|$ edges. In particular E^* can be chosen to be any set of the form $E^* = E\setminus(\{a^*\} \cup T^*)$ where a^* is the only edge of weight one in a circuit of weight one, and T^* is a spanning tree with sink at the source of a^*.*

Proof. We first prove Part (i). Let $C = \{\zeta(c_1)\cdots\zeta(c_{|E|-|V|+1})\}$ be a circuit basis for the cycle space of G, and let c' be a circuit of G of weight one. Notice that only one edge of c' has weight one, and the rest, if any, have weight zero. Hence, c' must be elementary, as there are no circuits of weight zero in G. Since $\zeta(c')$ belongs to the cycle space of G, there exists a non trivial linear combination $\zeta(c') = \sum_{i=1}^{|E|-|V|+1} \alpha_i\zeta(c_i)$, $\alpha_i \in \mathbb{R}$. Taking i such that $\alpha_i \neq 0$, and replacing $\zeta(c_i)$ by $\zeta(c')$ in C, the spanned subspace of $\mathbb{R}^{|E|}$ does not change. Hence, we can assume without loss of generality that c_1 is an elementary circuit of G of weight one.

Consider arbitrary vertices $u, v \in V$. Since G is strongly connected, there exists a circuit γ that passes through u and v. As $\zeta(\gamma)$ belongs to the cycle space of G, there exists a linear combination $\zeta(\gamma) = \sum_{i=1}^{|E|-|V|+1} \alpha_i\zeta(c_i)$, with $\alpha_i \in \mathbb{R}$. Thus, the set of edges of γ is a subset of the union of the set of edges of all circuits c_i. Hence, since u and v are arbitrary, any linear combination $\sum_{i=1}^{|E|-|V|+1} k_i\zeta(c_i)$ with $k_i \in \mathbb{Z}_{>0}$, generates a cyclic assignments of counters for G that is connected. Moreover, since C is a basis of the cycle space, different linear combinations generate different cyclic connected assignments of counters for G. Thus, there are at least as many cyclic connected assignments of counters of weight no grater than n, as compositions of $\lfloor n/\max\psi(c_i)\rfloor$ in $|E| - |V| + 1$ positive summands. Each assignment η obtained in this way can be completed to an assignment η' of weight n adding copies of c_1, which is a fixed circuit of weight 1. Specifically, if $\eta = \sum_i k_i\zeta(c_i)$, we take

$$\eta' = (k_1 + n - \psi(\eta))\zeta(c_1) + \sum_{i>1} k_i\zeta(c_i)\,.$$

Now, since we have $\sum_i k_i = \lfloor n/\max\psi(c_i)\rfloor$, k_1 is determined by the remaining coefficients $k_2\cdots k_{|E|-|V|+1}$, and this in turn completely determines η'. This way, different linear combinations with $\sum_i k_i = \lfloor n/\max\psi(c_i)\rfloor$ generate different connected assignments of counters of weight n. Thus, we have at least as many cyclic connected assignments of counters of weight n, as compositions of $m = \lfloor n/\max\psi(c_i)\rfloor$ in $|E| - |V| + 1$ positive summands. The proof of Part (i) is completed by recalling that the number of such compositions is $\binom{m-1}{|E|-|V|} = \Omega\left(m^{|E|-|V|}\right)$.

We now turn to Part (ii). Let $\gamma = e_1 e_2 \cdots e_r$ be an elementary circuit of weight 1 in G with $\psi(e_r) = 1$, and $\psi(e_i) = 0$ for $i = 1 \cdots r-1$. Let T be a spanning tree of G with a sink at the source of e_r, with a set of edges $E_T \supset \{e_1 \cdots e_{r-1}\}$. Such T can be constructed in $|V| - r$ steps, starting with the set of edges $\{e_1 \cdots e_{r-1}\}$ together with its r different adjacent vertices, and adding an edge e in each step, in such a way that the destination of e is in T, but the source is not yet in T. We will show that the values of η_e for $e \in E_T \cup \{e_r\}$ can be computed from the remaining values, which we regard as given. Let V_γ be the set of vertices of γ, and for $v \in V$, let $d(v)$ be the distance in T from v to V_γ, i.e., the length of the unique path in T from v to a vertex of V_γ. For each $v \in V \setminus V_\gamma$ we will show how to compute η_{e_v} for the unique edge e_v, with source v, which belongs to E_T. We take all the vertices $v \in V \setminus V_\gamma$ in decreasing order of $d(v)$. For each edge $e = (u, v)$ with destination v, either $e \notin E_T$, or $d(u) = d(v) + 1$. In any case, the value η_e is known, as either it is given, or it has already been computed. Since the value η_e is known for all edges $e \neq e_v$ with source v, we can compute η_{e_v} from the flow conservation equation $\sum_{e=(u,v)} \eta_e = \sum_{e=(v,w)} \eta_e$. After finishing this process, we finally get to know η_e for all edges e except for those in γ, the unique circuit of $E_T \cup \{e_r\}$. We can now compute η_{e_r} as $n - \sum_{e \in E \setminus \{e_1 \cdots e_r\}} \eta_e \psi(e)$ and continue calculating η_e for $e = e_1 \cdots e_{r-1}$, using the flow conservation equation. □

Since T_c is a refinement of T, we have $\mathcal{N}_T \leq \mathcal{N}_{T_\mathrm{c}}$. By Lemma 4.5, determining $\mathcal{N}_{T_\mathrm{c}}$ is in turn equivalent to counting the number of $|S_{T_\mathrm{c}}| \times |S_{T_\mathrm{c}}|$ matrices M for which there exists a sequence x^n with $N_\mathrm{c}(x^n) = M$, where $N_\mathrm{c}(x^n)$ is the state transition matrix of x^n with respect to T_c. We will use Part (ii) of Lemma 4.35, applied to the state transition support graph of T_c, $G = G_{T_\mathrm{c}}$, to bound the number of such matrices M for which there exists a sequence x^n with $N_\mathrm{c}(x^n) = M$ *and* $s_n(x^n) = s_0$. We will then show that the condition $s_n(x^n) = s_0$ can be removed without affecting the asymptotic result. This will prove the relation $\mathcal{N}_T = O(n^{|E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|})$, required by the claim in Theorem 4.33. The proof will be constructive, yielding an algorithm that recovers N_c given a well characterized set of $|E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|$ of its entries and the final state s_n.

As for the relation $n^{|E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|} = O(\mathcal{N}_T)$, also claimed in the theorem, recall that by Corollary 4.21, T and T_c define the same type classes. Hence, each T-class* with final state s, is partitioned into a constant number of T_c-classes*, one for each possible final state su that refines s in T_c. As a consequence, we have $\mathcal{N}_{T_\mathrm{c}} = O(\mathcal{N}_T)$, and therefore it suffices to show that $n^{|E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|} = O(\mathcal{N}_{T_\mathrm{c}})$. However, this is more involved than simply applying Lemma 4.35 to G_{T_c}, since some closed paths in G_{T_c} may not correspond to a valid state sequence (in analogy to Example 4.6). Instead, in Lemma 4.36 below, we apply Lemma 4.35 to the pseudo-state transition support graph of T_c, $G = G^{(\mathrm{F})}_{T_\mathrm{c}}$, defined in Section 4.3, just before Lemma 4.29. In the application of Lemma 4.35 we take $\psi(e) = |\omega(e)|$, where ω is the tagging function of Theorem 4.15, and we are interested in assignments of counters of weight n to $G^{(\mathrm{F})}_{T_\mathrm{c}}$. The connection between the set of cyclic connected assignments of counters in $G^{(\mathrm{F})}_{T_\mathrm{c}}$ and the set of T_c-classes*, is given by the tagging function ω, and Lemma 4.14. This line of reasoning is formalized in the following lemma.

4.36. Lemma. *Consider a canonical context tree T with pseudo-state transition support graph*

$G_T^{(F)} = (V_T^{(F)}, E_T^{(F)})$. The number of T-classes for sequences of length n with initial state s_0, and a final state s_n, is bounded from below for large n by $Cn^{|E_T^{(F)}|-|V_T^{(F)}|}$ where C is a positive constant that depends only on T.*

Proof. We first show that $G_T^{(\mathrm{F})}$ is strongly connected. By Lemma 4.29, for each edge $e = (u, v)$ of $G_T^{(\mathrm{F})}$ there exists a pattern string y_e, such that $F_{u,v}(y^m) > n_{y_e}(y^m)$. Hence, taking y^m as the concatenation of the patterns[7] $\overline{y_e}$ for all edges of $G_T^{(\mathrm{F})}$, we know that $F_{u,v}(y^m\overline{s_0}) > 0$ for all $(u, v) \in E_T^{(\mathrm{F})}$. By Theorem 4.15, we have $y^m\overline{s_0} = \omega(\gamma)$ where γ is an Eulerian path from s_0 to s_0 (i.e., a circuit) in $G_F(y^m\overline{s_0})$. Now, if $u \in U$ is a pseudo-state, by the definition of $G_T^{(\mathrm{F})}$, u is the endpoint of some edge in $G_T^{(\mathrm{F})}$. Thus, since $F_{u,v}(y^m\overline{s_0}) > 0$ for all $(u, v) \in E_T^{(\mathrm{F})}$, γ visits all pseudo-states of U, and therefore $G_F(y^m\overline{s_0})$ is strongly connected. Now, by parts (vii) and (xi) of Lemma 4.13, if there exists an edge (u, v) in $G_F(y^m\overline{s_0})$, i.e., $F_{u,v}(y^m\overline{s_0}) > 0$, then $(u, v) \in E_T^{(\mathrm{F})}$. Hence, $G_T^{(\mathrm{F})}$ is also strongly connected as claimed.

Let ψ be a weight function for $G_T^{(\mathrm{F})}$ defined as $\psi(e) = |\omega(e)|$. By the definitions of $G_T^{(\mathrm{F})}$ and ψ, there are no circuits of weight zero in $G_T^{(\mathrm{F})}$. Also for $b \in \mathcal{A}$ and $s = \sigma_T(bbb\cdots)$ there is a circuit of weight 1 starting from s, for either $\tau(s, b) = s$, or $\tau(s, b) = bs$. In the former case (s, e_1, s) is a circuit of weight 1. In the latter case, (s, e_1, bs, e_2, s) is a circuit of weight 1, as $\mathrm{tail}(bs) = s \in T$, and therefore $(bs, \rho(bs) = s)$ is an edge of $E_T^{(\mathrm{F})}$. We are then under the assumptions of Lemma 4.35.

Let Φ be the set of connected assignments of counters of weight n for $G_T^{(\mathrm{F})}$, such that

$$\sum_{e=(u,v)\in E_T^{(\mathrm{F})}} \eta_e + \delta_{s_0,v} = \sum_{e=(v,w)\in E_T^{(\mathrm{F})}} \eta_e + \delta_{s_n,v}\,, \quad \forall\, \eta \in \Phi, v \in V_T^{(\mathrm{F})}\,.$$

We claim that $|\Phi|$ bounds from below the number of close-ended type classes for sequences of length n with initial state s_0, and a final state s_n. Let $\eta \in \Phi$, and let $G_{F'}$ be a graph with incidence matrix $F'_{u,v} = \eta_{(u,v)}$ for all $u, v \in V_T^{(\mathrm{F})}$. Since $\eta \in \Phi$, there exists an Eulerian path γ from s_0 to s_n in $G_{F'}$. The path γ defines, via the tagging function ω, a string of length n, $x^n = \omega(\gamma)$. By Lemma 4.14(i), the final state of x^n is s_n, and by Theorem 4.15, there exists an Eulerian path γ' from s_0 to s_n in $G_F(x^n)$, such that $x^n = \omega(\gamma')$. Hence, we have $\omega(\gamma) = \omega(\gamma')$, and by Lemma 4.14(ii), we get $F(x^n) = F'$. Thus, for every $\eta \in \Phi$, there exists a string x^n with pseudo-state transition matrix $F(x^n)$ given by η. Since $F(x^n)$ is a function of $N(x^n)$, then $F(x^n) \neq F(y^n)$ implies $N(x^n) \neq N(y^n)$, and this in turn implies that $\mathcal{T}^*(x^n) \neq \mathcal{T}^*(y^n)$ by Lemma 4.5. Hence, there are at least as many close-ended type classes as assignments of counters in Φ.

Let γ' be a fixed path from s_0 to s_n in $G_T^{(\mathrm{F})}$. For each cyclic connected assignment of counters η' of weight $n - \psi(\gamma')$ for $G_T^{(\mathrm{F})}$, the assignment of counters $\eta = \eta' + \zeta(\gamma')$ belongs to Φ. Hence, by Lemma 4.35(i), we have $|\Phi| \geq C(n - \psi(\gamma'))^{|E_T^{(\mathrm{F})}|-|V_T^{(\mathrm{F})}|}$, which concludes the proof.

□

[7] Recall that $n_{y_e}(y^m)$ is the number of occurrences of the *reverse* of y_e in y^{m-1}.

To prove that $n^{|E_{T_c}|-|V_{T_c}|} = O(\mathcal{N}_T)$ as required for Theorem 4.33 we will show that, for any context tree T, we have $|E_T^{(\mathrm{F})}| - |V_T^{(\mathrm{F})}| = |E_T| - |V_T|$. The claim will then follow from Lemma 4.36. To prove this we will need some auxiliary results.

For $s \in S_T$ we define $A_{(s)}$ as the set of edges in $E_T^{(\mathrm{F})}$ associated to forced pseudo-state sequences of all states s' for which s is the first state, $\nu_1(s')$, of the forced state sequence of s'. Thus,

$$A_{(s)} = \{(\mu_i(s'), \mu_{i+1}(s')) \in E_T^{(\mathrm{F})} : s' \in S_T, 1 \le i < \ell_{s'}, \nu_1(s') = s\}\,. \tag{4.58}$$

We also define $B_{(s)}$ as the set of edges in $E_T^{(\mathrm{F})}$ associated to context-dropping transitions of the form $(u, \rho(u))$, where u and its parent, $\rho(u)$, are entry points for the forced pseudo-state sequences of states s' and s'' such that s is the first state in the forced state sequence of both s' and s'', i.e., $u = \mu_1(s')$, $\rho(u) = \mu_1(s'')$, and $\nu_1(s') = \nu_1(s'') = s$. Formally,

$$B_{(s)} = \{(u, \rho(u)) \in E_T^{(\mathrm{F})} : u \in \bar{\Lambda}(s), \mathrm{tail}(u) \in \mathcal{I}(T)\}\,, \tag{4.59}$$

where we recall that $\bar{\Lambda}(s)$ is the set of proper descendants of s defined in Section 4.1. The condition $\mathrm{tail}(u) \in \mathcal{I}(T)$ guarantees, by (4.6), that there exist states s' and s'' such that $u = \mu_1(s')$ and $\rho(u) = \mu_1(s'')$, and the condition $u \in \bar{\Lambda}(s)$ guarantees that $\nu_1(s') = \nu_1(s'') = s$. Let $A^-_{(s)}$ be the set of edges of $A_{(s)}$ where the order of the source and destination is inverted $A^-_{(s)} = \{(v, u) : (u, v) \in A_{(s)}\}$ and let $V_{(s)}$ be the union of $\{s\}$ with the set of endpoints of the edges in $B_{(s)} \cup A^-_{(s)}$. We denote by S_1 the set of states $s \in S_T$ such that no forced state sequence is imposed to reached s, i.e., $S_1 = \{s \in S_T : \ell_s = 1\}$. We will show that for each state s' in $S_T \setminus S_1$, its forced pseudo-state sequence entry point, $\mu_1(s')$, is an endpoint of an edge in $B_{(s)}$ for some $s \in S_1$. Thus, we can construct a path from s' to $s = \nu_1(s')$ following edges from $A^-_{(s)}$ and $B_{(s)}$. More specifically, we show the following lemma.

4.37. Lemma. *For all $s \in S_T$ the 1-graph $(V_{(s)}, B_{(s)} \cup A^-_{(s)})$ is a tree with sink s. When $s \in S_T \setminus S_1$, the set $B_{(s)} \cup A^-_{(s)}$ is empty, and $V_{(s)} = \{s\}$. Furthermore, $\{V_{(s)} : s \in S_1\}$ is a partition of the set of vertices $V_T^{(F)}$.*

Proof. When $s \notin S_1$, we have $\mathrm{tail}(s) \notin \mathcal{I}(T)$ and, therefore, $\mathrm{tail}(u) \notin \mathcal{I}(T)$ for every $u \in \bar{\Lambda}(s)$. Hence $B_{(s)}$ is empty for $s \notin S_1$. Furthermore, if $\nu_1(s') = s$, then $s \preceq \mu_1(s')$ and, hence, by (4.6), we have $\mathrm{tail}(s) \in \mathcal{I}(T)$. Thus, $A_{(s)}$ is also empty for $s \notin S_1$. This proves that $B_{(s)} \cup A^-_{(s)}$ is empty for s in $S_T \setminus S_1$.

Next, we prove that for any state s, the 1-graph $(V_{(s)}, B_{(s)} \cup A^-_{(s)})$ is a tree with sink s.

We first show that every node in $V_{(s)}$ is the source of at most one edge in $B_{(s)} \cup A^-_{(s)}$. If $(u, \rho(u)) \in B_{(s)}$, then $\mathrm{tail}(u) \in \mathcal{I}(T)$. Thus, we have $\mathrm{tail}(u) \notin U$ and, therefore, $(\mathrm{tail}(u), u)$ is not an edge of $A_{(s)}$. Hence, $(u, \rho(u))$ is the only edge departing from u. If (u, v) is an edge in $A^-_{(s)}$, then we have $u = \mu_{i+1}(s')$ and $v = \mu_i(s')$ for some state s' and some $1 \le i < \ell_{s'}$. Thus, the destination $v = \mathrm{tail}(u)$ is uniquely determined by u and, hence, (u, v) is the only edge departing from u in $A^-_{(s)}$. Furthermore, since $\mathrm{tail}(u) = v$ and $v \in U$, we have $\mathrm{tail}(u) \notin \mathcal{I}(T)$ and, therefore, u is not the source of any edge in $B_{(s)}$.

We now show that from every $u \in V_{(s)}$ there exists a path in $(V_{(s)}, B_{(s)} \cup A^-_{(s)})$ from u to s. The claim is trivially true for $u = s$. Assume now that it is also true for all $u' \in V_{(s)}$ such that $|u'| < |u|$. If $u \in \bar{\Lambda}(s)$ and $\mathrm{tail}(u) \in \mathcal{I}(T)$, then $(u, \rho(u)) \in B_{(s)}$ and, since $|\rho(u)| < |u|$, we have a path to s. In other case, we have that $(u, \mathrm{tail}(u))$ must be an edge of $A^-_{(s)}$ and, since $|\mathrm{tail}(u)| < |u|$, we have a path to s.

The claim that $(V_{(s)}, B_{(s)} \cup A^-_{(s)})$ is a tree with sink s follows from the fact that s has no outgoing edge in $(V_{(s)}, B_{(s)} \cup A^-_{(s)})$. Indeed, s is not the source of any edge in $B_{(s)}$ (because $s \notin \bar{\Lambda}(s)$) and, when $A_{(s)}$ is not empty, we have already shown that s must belong to S_1. Thus, s is not the destination of any edge of the form $(\mu_i(s'), \mu_{i+1}(s'))$.

We next show that $\{V_{(s)} : s \in S_1\}$ is a partition of the set of vertices $V_T^{(\mathrm{F})}$. Notice that $\bigcup_{s \in S_1} V_{(s)}$ covers the set $V_T^{(\mathrm{F})} = U$, since, by the definition of U, every $u \in U$ must be the destination of an edge of $A_{(s)}$ for some s, or it must belong to S_1. If u is an endpoint of an edge of both $A_{(s)}$ and $A_{(t)}$ for some $s, t \in S_1$, we have that $u = \mu_i(s')$ for some i, $1 \le i \le \ell_{s'}$ and $s' \in S_T$ with $\nu_1(s') = s$, and $u = \mu_j(t')$ for some j, $1 \le j \le \ell_{t'}$ and $t' \in S_T$ with $\nu_1(t') = t$. Then, we have $\mu_i(s') = \mu_j(t')$ and, therefore, $\mu_{i-k}(s') = \mu_{j-k}(t')$ for all k, $k < i, k < j$, since for any state w and $1 \le m \le \ell_w$, we have $\mu_{m-k}(w) = \mathrm{tail}^k(\mu_m(w))$, where $\mathrm{tail}^k(\cdot)$ denotes the composition of k applications of $\mathrm{tail}(\cdot)$. Hence, by (4.6), we must have $\mu_1(s') = \mu_1(t')$ and, therefore, $\nu_1(s') = \nu_1(t')$. Thus, we must have $s = t$. If u is an endpoint of an edge of both $B_{(s)}$ and $B_{(t)}$, we have that s and t are both prefixes of u and they must be equal. Finally, if u is an endpoint of an edge of both $A_{(s)}$ and $B_{(t)}$, we have that $u = \mu_i(s')$ for some i, $1 \le i \le \ell_{s'}$ and $s' \in S_T$ with $\nu_1(s') = s$. Also, by the definition of $B_{(t)}$, we know that $\mathrm{tail}(u) \in \mathcal{I}(T)$, and we must have $i = 1$ by (4.6). Thus, we get $u = \mu_1(s')$, and $\nu_1(s') = s$. Therefore, s is a prefix of u, and so is t by definition of $B_{(t)}$. Thus, we must have $s = t$. We conclude that $\{V_{(s)} : s \in S_1\}$ is a partition of $V_T^{(\mathrm{F})}$ as claimed. □

We are now ready to prove the following result, which, together with Lemma 4.36, yields the relation $n^{|E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|} = O(\mathcal{N}_T)$, called for in Theorem 4.33.

4.38. Lemma. *For an arbitrary context tree T, we have $|E_T^{(F)}| - |V_T^{(F)}| = |E_T| - |V_T|$.*

Proof. We partition the set $E_T^{(\mathrm{F})}$ into three subsets, A_1, A_2, A_3, suitable for the application of Lemma 4.37. This will help us connect the quantities $|E_T^{(\mathrm{F})}|$ and $|V_T^{(\mathrm{F})}|$, and compute their difference. Specifically, we define

$$\begin{aligned} A_1 &= \{(\mu_i(s), \mu_{i+1}(s)) \in E_T^{(\mathrm{F})} : s \in S_T, 1 \le i < \ell_s\}, \\ A_2 &= \{(s, \tau(s,b)) \in E_T^{(\mathrm{F})} : s \in S_T, b \in \mathcal{A}, |\tau(s,b)| \le |s|\}, \\ A_3 &= \{(u, \rho(u)) \in E_T^{(\mathrm{F})} : u \in V_T^{(\mathrm{F})}\}. \end{aligned}$$

We observe that the sets A_1, A_2, A_3 indeed partition the set $E_T^{(\mathrm{F})}$. In particular, notice that if $|\tau(s,b)| = |s|+1$, then $(s, \tau(s,b))$ is included in A_1 but not in A_2. Also notice that, as s varies in the set of states, the union of disjoint sets $\bigcup_s B_{(s)} \cup A_{(s)}$ covers the edges in $A_1 \cup A_3$ except for edges of the form $(u, \rho(u))$ with $\mathrm{tail}(u) \notin \mathcal{I}(T)$, which are excluded from $B_{(s)}$ in (4.59). Thus, we define

$$B'_{(s)} = \{(u, \rho(u)) \in E_T^{(\mathrm{F})} : u \in \bar{\Lambda}(s), \mathrm{tail}(u) \in S_T\} \tag{4.60}$$

and we notice that $A_1 \cup A_3$ is partitioned into $\bigcup_s A_{(s)} \cup B_{(s)} \cup B'_{(s)}$. We then have,

$$\begin{aligned} |E_T^{(\mathrm{F})}| &= |A_1| + |A_2| + |A_3| \\ &= |A_2| + \sum_{s \in S_T} |A_{(s)}| + |B_{(s)}| + |B'_{(s)}| \,. \end{aligned}$$

Since $A_{(s)}$ and $B_{(s)}$ are empty sets for $s \notin S_1$ by Lemma 4.37, we have

$$|E_T^{(\mathrm{F})}| = |A_2| + \sum_{s \in S_T} |B'_{(s)}| + \sum_{s \in S_1} |A_{(s)}| + |B_{(s)}| \,.$$

Now, as $(V_{(s)}, B_{(s)} \cup A^-_{(s)})$ is a tree by Lemma 4.37, we have that $|A_{(s)}| + |B_{(s)}| + 1 = |V_{(s)}|$. Thus, we get

$$|E_T^{(\mathrm{F})}| = |A_2| + \sum_{s \in S_T} |B'_{(s)}| + \sum_{s \in S_1} \left(|V_{(s)}| - 1 \right) ,$$

and since $\{V_{(s)} : s \in S_1\}$ is a partition of $V_T^{(\mathrm{F})}$ by Lemma 4.37, we obtain

$$|E_T^{(\mathrm{F})}| = |A_2| + |V_T^{(\mathrm{F})}| - |S_1| + \sum_{s \in S_T} |B'_{(s)}| \,. \tag{4.61}$$

Our next step connects the sets $B'_{(s)}$ with certain sets of edges of $E_T^{(\mathrm{F})}$ of the form $(s, \tau(s,b))$, which we will combine with A_2 and, later on, relate to the state transition support graph G_T. We claim that $|B'_{(s)}| = |A'_{(s)}|$, where

$$A'_{(s)} = \{(s', u) \in E_T^{(\mathrm{F})} : s' \in S_T, u \in \bar{\Lambda}(s), s' = \mathrm{tail}(u)\} \,.$$

The source s' of $(s', u) \in A'_{(s)}$ is uniquely determined by the destination u. Also the destination of $(u, \rho(u)) \in B'_{(s)}$ is uniquely determined by its source u. Hence, it is sufficient to show that

$$\{u : (s', u) \in A'_{(s)}\} = \{u : (u, \rho(u)) \in B'_{(s)}\} \,.$$

If $(u, \rho(u)) \in B'_{(s)}$, then, by the definition of $B'_{(s)}$ in (4.60), we know that $u \in \bar{\Lambda}(s)$ and $\mathrm{tail}(u)$ belongs to S_T. Thus, we have $(s', u) \in A'_{(s)}$ for $s' = \mathrm{tail}(u)$. On the other hand, if $(s', u) \in A'_{(s)}$, then, by the definition of $A'_{(s)}$, we have that $u \in \bar{\Lambda}(s)$ and $\mathrm{tail}(u) = s' \in S_T$. Therefore, we have $(u, \rho(u)) \in B'_{(s)}$ by the definition of $B'_{(s)}$. We conclude that $\sum_{s \in S_T} |B'_{(s)}|$ equals $\sum_{s \in S_T} |A'_{(s)}|$, which in turn is equal to the number of edges in the set

$$\left\{(s, \tau(s,b)) \in E_T^{(\mathrm{F})} : s \in S_T, b \in \mathcal{A}, |\tau(s,b)| = |s| + 1, \tau(s,b) \notin S_T\right\} .$$

Hence, the sum $|A_2| + \sum_{s \in S_T} |B'_{(s)}|$ in (4.61) equals the cardinality of

$$\begin{aligned} A'_2 &= \left\{(s, \tau(s,b)) \in E_T^{(\mathrm{F})} : s \in S_T, b \in \mathcal{A}, |\tau(s,b)| \leq |s| \text{ or } \tau(s,b) \notin S_T\right\} \\ &= \left\{(s, \tau(s,b)) \in E_T^{(\mathrm{F})} : s \in S_T, b \in \mathcal{A}, s' \prec bs \text{ for some } s' \in S_T\right\} . \end{aligned}$$

Thus, (4.61) becomes

$$|E_T^{(\mathrm{F})}| = |A'_2| + |V_T^{(\mathrm{F})}| - |S_1| \,. \tag{4.62}$$

We now relate the size of A'_2, which is a subset of $E_T^{(\mathrm{F})}$, to the size of a certain subset of E_T. This will let us establish the claimed relation, $|E_T^{(\mathrm{F})}| - |V_T^{(\mathrm{F})}| = |E_T| - |V_T|$, by means of (4.62). Specifically, let E_1 be the subset of E_T with destinations in S_1, i.e.,

$$E_1 = \{(s, s') \in E_T : s' \in S_1\}\,.$$

We show next that $|A'_2| = |E_1|$, by showing that the mapping that takes $(s, \tau(s, b)) \in A'_2$ to (s, s'), where $s' \in S_T$ is the unique state that satisfies $s' \prec bs$, defines a one-to-one correspondence between A'_2 and E_1. If $(s, \tau(s, b)) \in A'_2$ and s' is a state with $s' \prec bs$, then $\mathrm{tail}(s') \prec s$ and, by the definition of E_T in (4.22), we have $(s, s') \in E_T$. Moreover, since $\mathrm{tail}(s') \prec s$ we have that $(s, s') \in E_1$. Suppose now that $(s, s') \in E_1$. Since $s' \in S_1$, we have $\mathrm{tail}(s') \in \mathcal{I}(T)$. Thus, by the definition of E_T in (4.22), we know that $\mathrm{tail}(s') \prec s$. Hence, with $b = \mathrm{head}(s')$, we have $s' \prec bs$, and thus $(s, \tau(s, b)) \in A'_2$. Moreover, in order to get $s' \prec bs$, b must be taken equal to $\mathrm{head}(s')$ and, hence, $(s, \tau(s, b))$ is the unique element of A'_2 that is mapped to (s, s'). Then, we have that

$$|A'_2| = |E_1| = \sum_{s' \in S_1} |\{s : (s, s') \in E_T\}|\,. \tag{4.63}$$

When $s' \notin S_1$ and $(s, s') \in E_T$ we have, by the definition of E_T in (4.22), that s is the unique state such that $s \preceq \mathrm{tail}(s')$. Thus, for fixed s', we have $|\{(s, s') \in E_T\}| = 1$ and we rewrite (4.63) as

$$\begin{aligned} |A'_2| &= \sum_{s' \in S_T} |\{s : (s, s') \in E_T\}| - (|S_T| - |S_1|) \\ &= |E_T| + |S_1| - |S_T| \\ &= |E_T| + |S_1| - |V_T|\,. \end{aligned}$$

Substituting in (4.62) we get

$$|E_T^{(\mathrm{F})}| = |V_T^{(\mathrm{F})}| - |S_1| + |E_T| + |S_1| - |V_T|\,, \tag{4.64}$$

thus $|E_T^{(\mathrm{F})}| - |V_T^{(\mathrm{F})}| = |E_T| - |V_T|$ as claimed. □

We have now all the elements to prove Theorem 4.33.

Proof of Theorem 4.33

Since $\mathcal{N}_{T_\mathrm{c}} = O(\mathcal{N}_T)$, and $|E_{T_\mathrm{c}}^{(\mathrm{F})}| - |V_{T_\mathrm{c}}^{(\mathrm{F})}| = |E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|$ by Lemma 4.38, the fact that $n^{|E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|} = O(\mathcal{N}_T)$ follows from Lemma 4.36. We need to show that $\mathcal{N}_T = O(n^{|E_{T_\mathrm{c}}| - |V_{T_\mathrm{c}}|})$. Let ψ be a weight function for the state transition support graph of T_c, G_{T_c}, constantly equal to 1. Clearly G_{T_c} is strongly connected, there are no circuits of weight zero, and, for any $b \in \mathcal{A}$, there is a circuit of weight 1 from the state $bbb\cdots b$ to itself. Thus, G_{T_c} satisfies the assumptions of Lemma 4.35.

For each state s of T_c, let γ_s be a fixed path from s to s_0 in G_{T_c}. Consider a string x^n with final state s_n in T_c. Let η be an assignment of counters for G_{T_c} defined as $\eta_{(u,v)} = N_\mathrm{c}(x^n)_{u,v}$ for all $u, v \in V_{T_\mathrm{c}}$. The assignment of counters $\eta' = \eta + \zeta(\gamma_{s_n})$ is cyclic by construction. Hence,

by Lemma 4.35(ii), η' can be fully described by the values η'_e for a set of $|E_{T_c}| - |V_{T_c}|$ edges. Given s_n, γ_{s_n} is fixed, and therefore this is equivalent to give the values η_e for the same set of edges. This completely describes η', and η can then be recovered from η'. Thus, η can be fully described by giving the final state s_n in T_c, which determines γ_{s_n}, and the values η_e for a set of $|E_{T_c}| - |V_{T_c}|$ edges. Since each value η_e is not greater than n, and there are a constant number of states in T_c, we have $\mathcal{N}_{T_c} = O\left(n^{|E_{T_c}|-|V_{T_c}|}\right)$. The proof is completed by recalling that $\mathcal{N}_T \leq \mathcal{N}_{T_c}$ since T_c is a refinement of T. □

4.5 Type classes with respect to the FSM closure of T

In this section we compare T-classes with T_{suf}-classes, where we recall from Section 2.3.2 that T_{suf} is the FSM closure of T. The difference $|E_{T_c}|-|V_{T_c}|$, appears as a factor in the term of order $\log n$ of $\mathrm{E}_{\langle T,p_T\rangle}\left[\,|\mathcal{T}(X^n)|\,\right]$ in Theorem 4.18, and in the exponent of the number of type classes in Theorem 4.33. In the case of FSM context trees, and in particular in the FSM closure of T, the difference $|E_{T_{\text{suf}}}|-|V_{T_{\text{suf}}}|$ reduces to $|S_{T_{\text{suf}}}|(\alpha-1)$. We next define the (FSM) *over-refinement* of T, which connects $|E_{T_c}|-|V_{T_c}|$ with $|S_{T_{\text{suf}}}|(\alpha-1)$. In Chapter 5, this connection will be exploited to show how an enumerative code for T can be obtained by enumerating the sequences in the T_{suf}-class of the input string, rather than in the T-class, while still being universal with optimal convergence rate with respect to T. This will be an alternative to the classical enumerative code with respect to type classes of the target model, in this case represented by T, that we also explore in Chapter 5.

For a node $t \in \mathcal{I}(T_{\text{suf}})\backslash\mathcal{I}(T_c)$ we define the (FSM) *over-refinement* of t as $\kappa_t = |\{a \in \mathcal{A} : at \notin \mathcal{I}(T_c)\}|$. The name given to κ_t stems from the fact that it counts symbols for which an extension from t was not needed in order to determine a next-state transition in T_c, yet it was added in the process of constructing the FSM closure T_{suf}. The total *over-refinement* of T is now defined as

$$\kappa_T = \sum_{t\in\mathcal{I}(T_{\text{suf}})\backslash\mathcal{I}(T_c)} (\alpha-1)(\kappa_t - 1)\,. \tag{4.65}$$

The following lemma connects κ_T with $|E_{T_c}| - |V_{T_c}|$ and $|E_{T_{\text{suf}}}| - |V_{T_{\text{suf}}}| = |S_{T_{\text{suf}}}|(\alpha-1)$.

4.39. Lemma. *Let $S_{T_{\text{suf}}}$ denote the set of states of T_{suf}. Then,*

$$\kappa_T = -(|E_{T_c}|-|V_{T_c}|) + |S_{T_{\text{suf}}}|(\alpha-1)\,. \tag{4.66}$$

Proof. Let s be a state of T_c and W the subtree of T_{suf} rooted at s, $W = \{w \in \mathcal{A}^* : sw \in T_{\text{suf}}\}$. For $a \in \mathcal{A}$ let $W_a = \{w \in \mathcal{A}^* : asw \in T_c\}$. Since T_{suf} is FSM, clearly $W_a \subseteq W$. The number $A_a(s)$ of edges departing from s with symbol a in the graph $G_{T_c} = (V_{T_c}, E_{T_c})$ equals the number of leaves in the full tree W_a, i.e.,

$$A_a(s) = (\alpha-1)|\mathcal{I}(W_a)| + 1\,.$$

Thus, the total number of edges in E_{T_c} departing from s is

$$A(s) = \sum_{a\in\mathcal{A}} (\alpha-1)|\mathcal{I}(W_a)| + 1 = \alpha + (\alpha-1)\sum_{a\in\mathcal{A}} |\mathcal{I}(W_a)|\,. \tag{4.67}$$

Now, by (4.67), we have,

$$\sum_{t\in\mathcal{I}(W)}(\alpha-1)(\kappa(t)-1)=(\alpha-1)\left(\sum_{t\in\mathcal{I}(W)}\kappa(t)\right)-(\alpha-1)|\mathcal{I}(W)|\,,$$

and by the definition of $\kappa(t)$, letting $\mathbb{1}_{\{t\in\mathcal{I}(W_a)\}}=1$ if $t\in\mathcal{I}(W_a)$, and zero otherwise,

$$\sum_{t\in\mathcal{I}(W)}(\alpha-1)(\kappa(t)-1)=(\alpha-1)\left(\sum_{t\in\mathcal{I}(W)}\sum_{a\in\mathcal{A}}\left(1-\mathbb{1}_{\{t\in\mathcal{I}(W_a)\}}\right)\right)-(\alpha-1)|\mathcal{I}(W)|\,.$$

We separate terms in the summation as, $\sum_{t\in\mathcal{I}(W)}\sum_{a\in\mathcal{A}}1=\alpha|\mathcal{I}(W)|$, and, recalling that $W_a\subseteq W$, $\sum_{t\in\mathcal{I}(W)}\sum_{a\in\mathcal{A}}\mathbb{1}_{\{t\in\mathcal{I}(W_a)\}}=\sum_{a\in\mathcal{A}}|\mathcal{I}(W_a)|$. Thus,

$$\begin{aligned}\sum_{t\in\mathcal{I}(W)}(\alpha-1)(\kappa(t)-1) &= (\alpha-1)\alpha|\mathcal{I}(W)|-(\alpha-1)\sum_{a\in\mathcal{A}}|\mathcal{I}(W_a)|-(\alpha-1)|\mathcal{I}(W)|\\ &= (\alpha-1)^2|\mathcal{I}(W)|-(\alpha-1)\sum_{a\in\mathcal{A}}|\mathcal{I}(W_a)|\,.\end{aligned}$$

Denoting $S_W(s)$ the set of states of T_{suf} that descend from s, i.e., the set of leaves of W, we have

$$\begin{aligned}\sum_{t\in\mathcal{I}(W)}(\alpha-1)(\kappa(t)-1) &= (\alpha-1)^2\frac{|S_W(s)|-1}{\alpha-1}-(\alpha-1)\sum_{a\in\mathcal{A}}|\mathcal{I}(W_a)|\\ &= (\alpha-1)|S_W(s)|-\left(\alpha-1+(\alpha-1)\sum_{a\in\mathcal{A}}|\mathcal{I}(W_a)|\right)\,,\end{aligned}$$

and by (4.67) we get,

$$\sum_{t\in\mathcal{I}(W)}(\alpha-1)(\kappa(t)-1)=(\alpha-1)|S_W(s)|-A(s)+1\,.$$

Hence, the *over-refinement* of T is

$$\begin{aligned}\kappa(T) &= \sum_{t\in\mathcal{I}(T_{\text{suf}})\setminus\mathcal{I}(T_{\text{c}})}(\alpha-1)(\kappa(t)-1)\\ &= \sum_{s\in S_{T_{\text{c}}}}(\alpha-1)|S_W(s)|-A(s)+1\\ &= |S_{T_{\text{suf}}}|(\alpha-1)-(E_{T_{\text{c}}}-V_{T_{\text{c}}})\,.\end{aligned}$$

□

By Theorem 4.33, the exponent in the asymptotic growth of the number of type classes in T and T_{suf} is given by $|E_{T_{\text{c}}}|-|V_{T_{\text{c}}}|$ and $|S_{T_{\text{suf}}}|(\alpha-1)$, respectively. Hence, we get the following corollary to Lemma 4.39, which states that there is, asymptotically, a factor of n^{κ_T} more type classes in T_{suf} than in T.

4.40. Corollary. *Let T be a context tree and let T_{suf} be its FSM closure. Then, we have*

$$\mathcal{N}_{T_{\text{suf}}}=\Theta\left(n^{\kappa_T}\mathcal{N}_T\right)\,.$$

This chapter contains material published in [50].

Chapter 5

Enumerative coding for tree sources

In this chapter we present an *enumerative code* [11] for tree sources, based on the method of types, that can be efficiently computed[1] and is universal with optimal convergence rate. Such enumerative code is comprised of two parts, namely, a preamble that describes the type class to which the input sequence belongs followed by an index that identifies the sequence within its type class. Since all sequences in a type class are equiprobable, a uniform encoding of the index minimizes the expected length of the second part. Hence, the enumeration of type classes presented in Chapter 4 yields an efficient implementation of a uniform, and thus optimal, code for the index of the sequence. We are then left with the problem of efficiently implementing a code for the first part based on an optimal assignment of probabilities to type classes.

As we first observed in Section 1.5, using a uniform code for the first part is optimal for FSM models, but may be suboptimal in general for tree models. For example, consider the context trees T_1 and T_2 of Figure 5.1. It follows from Theorem 4.33 that there are $\Theta(n^5)$ type classes in T_1 and in T_2, even though T_1 has only four states. Furthermore, it can be shown that each type class of T_1 is partitioned into up to a constant number of type classes in T_2. Hence, a uniform coding for both the set of type classes and the set of sequences of each type class, would yield essentially the same code length for both context trees, without taking advantage of the smaller parameter space of T_1. On the other hand, it is readily verified that the vector of five counts, $A = \left(n_s^{(a)}(x^n)\right)_{s \in S_{T_2}}$, for any choice of $a \in \{0, 1\}$, together with the final state of x^n in T_2, suffice to completely describe $\mathcal{T}(T_2, x^n)$ and also $\mathcal{T}(T_1, x^n)$. Observe, however, that a type class with significantly different conditional empirical distributions for states 00 and 01 of T_2 will have small probability under a model based on T_1 for *any* choice of the model parameter, which suggests using a *non-uniform* encoding for describing type classes. More precisely, given the empirical conditional distribution, $\hat{p}$, in state 0 of T_1, and the number of occurrences, n_s, of the pattern 00, we can estimate the component of A corresponding to the number of occurrences, $n_s^{(a)}$, of symbol a in context 00 as $\hat{n}_s^{(a)} = n_s \hat{p}(a)$. If n_s and $\hat{p}$ have already been described to the decoder, we can then encode the difference $n_s^{(a)} - \hat{n}_s^{(a)}$ by assigning high probability to small absolute differences, suggesting that a non-uniform code may allow us to recover the code-length advantage of the smaller context tree T_1. This observation will be generalized to define a collection of non-uniform codes for encoding counts of occurrences of certain patterns within the input sequence. We will then apply these non-uniform codes to optimally encode a set of counts that uniquely determine the type class of

[1] We recall that by "efficient computation" we mean one whose encoding running time is polynomial in the length of the input sequence, and with code construction time that is also polynomial in the dimension of the parameter space of the model.

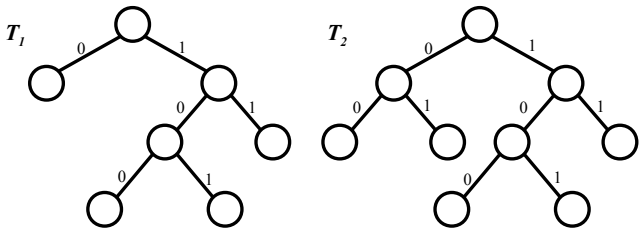

Figure 5.1: *Context tree over* $\mathcal{A} = \{0, 1\}$

a sequence with respect to an arbitrary context tree.

After introducing some notation and the formal setting in Section 5.1, in Section 5.2 we introduce variable length codes for symbol counts, following the above observation. These codes, which are based on Golomb codes [33] and are dubbed *symbol count codes* (*SCCs*), are of both theoretical and practical interest. Indeed, in Section 5.3, we re-derive the asymptotic bound of Theorem 4.18 on the expected size of the type class of a random sequence, which we first obtained by analyzing the exact formula of Theorem 4.15. In this case, we make use of the FSM closure and SCCs, as theoretical tools, to obtain the same bound based on known results for FSMs. In Section 5.4 we generalize the construction of SCCs to codes for pattern counts, termed *string count codes* (*SCCs**), which yield an efficient description of the type class and, combined with the aforementioned bound, lead to a universal enumerative code with optimal convergence rate. The optimality of the convergence rate of this code also yields, via a coding argument, a lower bound on the expected size of the type class of a random sequence, which turns out to differ by $O(1)$ from the upper bound in Theorem 4.18. The SCCs* also yield an alternative construction of enumerative codes, taken with respect to an extension of the original context tree T, and where the increment in the model cost term is compensated exactly by the reduction in the expected size of the class, maintaining an expected code length that is still optimal with respect to T. In particular, such a code can be derived from the FSM closure of T using known techniques for enumerating sequences in FSM type classes. Finally, in Section 5.5 we present two approaches for the twice-universal setting, in which the context tree T is unknown. The first approach is a standard plug-in scheme where the context tree is first estimated, and then the previously derived universal enumerative code is applied as-is, using the estimated tree in lieu of the true one. In the second approach, we take advantage of the observation that for any given context tree, there will be sequences that are "atypical" for the context tree, and will not estimate it regardless of the model parameter. The enumerative code is significantly simplified in the twice-universal setting by excluding such sequences from the coding space for the estimated tree.

5.1 Preliminaries

Except when we switch to the twice-universal setting in Section 5.5, we consider a fixed tree model $\langle T, p_T \rangle$, with p_T unknown. Thus, we keep the same simplified notation of Chapter 4, where sometimes the dependence on T is not made explicit. For the purpose of selecting

states, we assume that x^n is preceded by a *fixed* semi-infinite string $x_{-\infty}^0$. This convention, is consistent with Chapter 4 by selecting $x_{-\infty}^0$ to have $\overline{s_0}$ as a suffix, where s_0 is a state of maximal depth of T. When we move on to the twice-universal setting, however, no context tree is fixed a priori. In this case, $x_{-\infty}^0$ guarantees a well defined (permanent) state sequence $\{s_i\}_{0\le i\le n}$, $s_i = \sigma_T(x_{-\infty}^0 x^i)$, for each candidate tree T for the context tree estimation.

The occurrence counts of contexts, n_u, and symbols in a given context, $n_u^{(a)}$, is adapted accordingly. Namely,

$$n_u^{(a)}(x^n) = |\{\, i \,:\, 0 \le i < n,\ x_{i-k+1}^i = \overline{u},\ x_{i+1} = a \,\}|\,,$$

and we recall that $n_u(x^n) = \sum_{a\in\mathcal{A}} n_u^{(a)}(x^n)$. Again we omit the dependence on x^n when clear from the context. Notice that we also have $n_u = \sum_{a\in\mathcal{A}} n_{ua}$. Furthermore, denoting by $\mathbf{i}(u)$ and $\mathbf{f}(u)$ the indicator functions of the predicates $\overline{u} = x_{-|u|+1}^0$ (x_1 occurs in context $\overline{u}$) and $\overline{u} = x_{n-|u|+1}^n$ ($\overline{u}$ occurs at the end of x^n), we have

$$n_{au} + \mathbf{f}(au) = n_u^{(a)} + \mathbf{i}(au), \quad \text{for every } a \in \mathcal{A}\,.$$

To simplify expressions, we will use a generic constant δ to account for border adjustments due to terms of the form $\mathbf{i}(u)$ and $\mathbf{f}(u)$. In coding situations these terms will be known to the decoder, and in any case border effects will have no bearing on the asymptotic results.

For a context tree T, and a sequence x^n, we denote by $\mathcal{K}(T, x^n)$ the collection of counts $\{\, n_s^{(a)} \,\}_{s\in S_T,\, a\in\mathcal{A}}$. The type class of x^n with respect to T is then

$$\mathcal{T}(T, x^n) = \{\, y^n \in \mathcal{A}^n \,:\, \mathcal{K}(T, y^n) = \mathcal{K}(T, x^n) \,\}\,.$$

A context tree T and a sequence x^n determine a probability assignment, $\hat{\mathrm{P}}_T$, defined by the empirical conditional probabilities $\hat{\mathrm{P}}_T(a|s) = n_s^{(a)}(x^n)/n_s(x^n)$ (as before, we omit the dependence of the distribution $\hat{\mathrm{P}}_T$ on x^n when clear from the context); $\hat{\mathrm{P}}_T(x^n)$ is the *maximum likelihood probability* of x^n under T.

We seek an enumerative code for T comprised of two parts: a description of $\mathcal{K}(T, x^n)$, and an index of x^n within $\mathcal{T}(T, x^n)$. By Theorem 4.18, the expected length of the second part of the enumerative code is bounded by $n\mathcal{H} - \frac{1}{2}(|E_{T_c}| - |V_{T_c}|)\log n + O(1)$. As for the first part of the code, it follows from the proof of the relation $\mathcal{N}_T = O(n^{|E_{T_c}|-|V_{T_c}|})$ in Theorem 4.33, which is based in the (constructive) proof of Part (ii) of Lemma 4.35, that $|E_{T_c}| - |V_{T_c}|$ carefully selected counts of the state transition matrix N_c, suffice to describe $\mathcal{K}(T, x^n)$, modulo a constant number of bits required to describe the completion of an assignment of counters derived from N_c to make it cyclic. Notice that a uniform encoding of these counts requires[2] $(|E_{T_c}|-|V_{T_c}|)\log n + O(1)$ bits, yielding a total expected length for a uniform enumerative code of $n\mathcal{H} + \frac{1}{2}(|E_{T_c}|-|V_{T_c}|)\log n + O(1)$ bits. Except when T is FSM, $|E_{T_c}|-|V_{T_c}|$ is strictly larger than $(\alpha-1)|S_T|$, and a uniform encoding is generally suboptimal, as previously observed. In the special case in which T is FSM, T is also canonical, and $|E_{T_c}| - |V_{T_c}| = (\alpha-1)|S_T|$.

[2] To simplify discussions, we will sometimes ignore fractional parts of code lengths, referring, for example, to $\log n$ bits instead of the more precise "at most $\lceil \log n \rceil$ bits." This loose convention will be immaterial to the main asymptotic results.

Thus, in this case, a uniform encoding of appropriately selected $(\alpha-1)|S_T|$ counts results in a universal enumerative code with optimal convergence rate, with a normalized expected redundancy of $\frac{(\alpha-1)|S_T|\log n}{2n}+O(1/n)$ bits. Indeed, we show next that for $b\in\mathcal{A}$, the collection $\mathcal{K}_b(T,x^n)$ of $(\alpha-1)|S_T|$ counts, $\{n_s^{(a)}\}_{s\in S_T,\,a\in\mathcal{A}\setminus\{b\}}$, suffice to describe $\mathcal{K}(T,x^n)$ if the final state is also encoded. This known resul is re-derived here for completeness, based on the more general result of Lemma 4.35.

5.1. Lemma. *If a context tree T is FSM, then for any $b\in\mathcal{A}$, $\mathcal{K}_b(T,x^n)$ and the final state, s_n, of x^n in T completely determine $\mathcal{K}(T,x^n)$.*

Proof. Consider a constant weight function $\psi(e)=1$ for the state transition support graph $G=G_T$, and let s be the state $s=bb\ldots b$. Let γ_t be a fixed path from t to s_0 in G. Notice that since T is FSM, the sequence of state transitions given by γ_t is always compatible with a fixed string of symbols that takes T from state t to s_0. Let η be the cyclic assignment of counters obtained by letting $\eta_{(u,v)}$ be the number of state transitions from u to v in the state sequence of x^n concatenated with γ_{s_n}. The self loop from s to itself is a circuit of weight one, and the rest of the state transitions for symbol b (a unique transition for each state $s'\neq s$) form a spanning tree with a sink s. Thus, by Lemma 4.35, and recalling that γ_{s_n} is fixed, $\mathcal{K}_b(T,x^n)$ determines η, and therefore it also determines $\mathcal{K}(T,x^n)$. □

5.2 Non-uniform codes for symbol counts

The discussion above motivates the introduction in this section of a class of non-uniform codes, which we call *symbol count codes* (SCCs), for describing symbol counts $n_w^{(a)}$ in certain contexts w. These codes will be used, together with the FSM closure, as a theoretical tool in Section 5.3 for a new derivation of Theorem 4.18, which bounds the expected length of the second part of an enumerative code (i.e., the logarithm of the type class size). With a further generalization of SCCs, we then construct the actual non-uniform code for $\mathcal{K}(T,x^n)$ in Section 5.4. Together, both results will lead to an efficiently computable universal enumerative code with optimal convergence rate for general tree models.

Consider a symbol a and a fixed context w such that $s\prec w$ for some state s of T. Define

$$z_{w,a}=n_w^{(a)}-\frac{n_s^{(a)}}{n_s}n_w\,,\quad Z_{w,a}=|z_{w,a}|\,,\quad\text{and}\quad \mathrm{sg}_{w,a}=\begin{cases}1 & z_{w,a}>0\,,\\ 0 & \text{otherwise}\,.\end{cases}$$

As customary, denote by $\lfloor z\rfloor$ (resp. $\lceil z\rceil$) the largest (resp. smallest) integer satisfying $\lfloor z\rfloor\le z\le\lceil z\rceil$. Given $\mathrm{sg}_{w,a}$, $\lfloor Z_{w,a}\rfloor$, n_s, $n_s^{(a)}$, and n_w, it is possible to reconstruct $n_w^{(a)}$ as

$$n_w^{(a)}=\begin{cases}\lfloor Z_{w,a}\rfloor+\left\lceil\frac{n_s^{(a)}}{n_s}n_w\right\rceil\,, & \mathrm{sg}_{w,a}=1\,,\\ \left\lfloor\frac{n_s^{(a)}}{n_s}n_w\right\rfloor-\lfloor Z_{w,a}\rfloor\,, & \text{otherwise}\,.\end{cases}\tag{5.1}$$

Hence, if n_s, $n_s^{(a)}$, and n_w are known by a decoder, encoding $\mathrm{sg}_{w,a}$ and $\lfloor Z_{w,a}\rfloor$ suffices to describe $n_w^{(a)}$. For $n_w>0$ we will encode $\lfloor Z_{w,a}\rfloor$ using a Golomb code of parameter $\lceil\sqrt{n_w}\rceil$.

Specifically, we use a unary code[3] for the integer division $\lfloor Z_{w,a} \rfloor^\top = \lfloor Z_{w,a} \rfloor / \lceil \sqrt{n_w} \rceil$, using $\lfloor Z_{w,a} \rfloor^\top + 1$ bits, and encode $\lfloor Z_{w,a} \rfloor^\perp = \lfloor Z_{w,a} \rfloor \mod \lceil \sqrt{n_w} \rceil$ uniformly with $\log \lceil \sqrt{n_w} \rceil$ bits. When $n_w = 0$, we have $n_w^{(a)} = 0$ for all a, and encoding $n_w^{(a)}$ is not necessary; we define $\lfloor Z_{w,a} \rfloor^\top = 0$ in this case, to simplify discussions on expectations.

The intuition behind this code, which we denote $C_{w,a}\left(n_w^{(a)}\right)$, is that we can think of $Z_{w,a}$ as the absolute difference between the true value of $n_w^{(a)}$ and the estimate $\frac{n_s^{(a)}}{n_s} n_w$ that the decoder could guess from the known counts n_s, $n_s^{(a)}$ and n_w for a typical sequence of a model based on T. The probability of the estimate $\frac{n_s^{(a)}}{n_s} n_w$ differing from the true value decays exponentially fast, which leads to a constant expectation of $\lfloor Z_{w,a} \rfloor^\top$, as stated in the following lemma.

5.2. Lemma. *Let $\langle T, p_T \rangle$ be a tree source with all conditional probabilities different from zero. Then, the expectation $E_{\langle T, p_T \rangle}\left[\lfloor Z_{w,a} \rfloor^\top \right]$ is upper-bounded by a constant independent of n.*

The proof of Lemma 5.2 is deferred to Appendix G. It follows using a large deviations argument based on [40, Theorem 2].

By Lemma 5.2, the expected length of the unary part of the Golomb code used in SCCs is upper-bounded by a constant. On the other hand, when $n_w > 0$, a uniform encoding of $\lfloor Z_{w,a} \rfloor^\perp = \lfloor Z_{w,a} \rfloor \mod \lceil \sqrt{n_w} \rceil$ takes $\log \lceil \sqrt{n_w} \rceil$ bits. The code length of this uniform part can be upper-bounded by $\log\left(\sqrt{n_w} + 1\right) \le 1 + \frac{1}{2} \log n$.

The foregoing discussion yields the following corollary to Lemma 5.2.

5.3. Corollary. *The expected length of the code $C_{w,a}\left(n_w^{(a)}\right)$ is upper-bounded by $\frac{1}{2} \log n + O(1)$.*

Thus, under appropriate conditions, SCCs can code occurrence counts using, on average, half the length of the naive encoding.

5.3 The expected size of $\mathcal{T}(T, X^n)$ revisited

In this section we study the asymptotic behavior of $\mathrm{E}_{\langle T, p_T \rangle}\left[|\mathcal{T}(T, X^n)| \right]$, thus estimating the expected length of a uniform encoding of the index of x^n within its type class. We re-derive the result of Theorem 4.18 via a coding argument, as opposed to the analysis of the exact formula (4.15) employed in the proof of the theorem in Chapter 4.

By applying the trivial bound $|\mathcal{T}(T, x^n)| \le \prod_{s \in S_T} \frac{n_s!}{\prod_{a \in \mathcal{A}} n_s^{(a)}!}$, Lemma 4.27 yields the following result.

5.4. Lemma. *Let $\langle T, p_T \rangle$ be a tree source with entropy rate $\mathcal{H}$ and all conditional probabilities nonzero. Then,*

$$\frac{1}{n} E_{\langle T, p_T \rangle}\left[\log |\mathcal{T}(T, x^n)| \right] \le \mathcal{H} - |S_T|(\alpha - 1) \frac{\log n}{2n} + O\left(\frac{1}{n}\right). \tag{5.2}$$

[3] A unary code encodes a natural number m with a string of m consecutive zeros followed by a final symbol 1, which marks the end of the code word.

When T is FSM, the bound in (5.2) for $\frac{1}{n}\mathrm{E}_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(T,x^n)|\big]$ is tight, and as mentioned, it leads readily, by means of Lemma 5.1, to the optimality of enumerative coding where type classes are encoded uniformly. As it follows from Theorem 4.18, however, the coefficient $\frac{1}{2}|S_T|(\alpha-1)$ in the negative term of order $\frac{\log n}{n}$ in (5.2) is not the best one can obtain for general tree models, and the larger coefficient $|E_{T_\mathrm{c}}|-|V_{T_\mathrm{c}}|$ is indeed necessary to offset a corresponding length increase in the type class description part.

We first show how an economic description of $\mathcal{K}(T_{\mathrm{suf}},x^n)$, which determines the type class of x^n with respect to the FSM closure of T, can be obtained by means of SCCs from a description of $\mathcal{K}(T,x^n)$. This, together with the bound (5.2) applied to T_{suf}, and a coding argument, will lead to the desired tight bound. The description of $\mathcal{K}(T,X^n)$ itself will be discussed in Section 5.4, and will require additional tools.

By Corollary 4.40, there are, asymptotically, a factor of n^{κ_T} more T_{suf}-classes than T-classes, which suggests that roughly κ_T counts of $\log n$ bits each would suffice to describe $\mathcal{K}(T_{\mathrm{suf}},x^n)$ from $\mathcal{K}(T,x^n)$. The next lemma confirms this intuition, and establishes the fact that the counts can be encoded, on average, with a cost of at most $\frac{1}{2}\log n$ bits each.

5.5. Lemma. *Given $\mathcal{K}(T,x^n)$, the collection $\mathcal{K}(T_{\mathrm{suf}},x^n)$ can be described with SCC encodings of κ_T counts $n_w^{(a)}$ as discussed following (5.1), plus a constant number of bits used to describe $\sigma_{\mathbf{F}}(x^n)$, the final state of x^n in T_{suf}.*

Proof. By Corollary 4.21, $\mathcal{K}(T,x^n)$ determines $\mathcal{K}(T_\mathrm{c},x^n)$, and we can therefore assume that the latter is given. Let $\Delta=\mathcal{I}(T_{\mathrm{suf}})\backslash\mathcal{I}(T_\mathrm{c})$, where we recall that $\mathcal{I}(T)$ denotes the set of internal nodes of T. We will describe $n_{tc}^{(a)}$ for every child tc of $t\in\Delta$ and every $a\in\mathcal{A}$, proceeding in ascending order of length of t. We claim that proceeding this way we are sure that n_{tc} is known (has been described) when describing $n_{tc}^{(a)}$, and, thus, the latter can be encoded using SCCs. Indeed, if $t=bu\in\Delta$, with $b\in\mathcal{A}$ and $u\in T_\mathrm{c}$, then $n_{bu}=n_u^{(b)}+\delta$ is known. Otherwise, if $t=bu$ but $u\notin T_\mathrm{c}$, then u is an internal node of T_{suf} shorter than bu, and thus $n_{buc}=n_{uc}^{(b)}+\delta$ is known for every child buc of bu. Thus, the claimed order of description is satisfied. Consider now a node $tc\in\Delta$, $c\in\mathcal{A}$. For every symbol a such that at is an internal node of T_c, $atc\in T_\mathrm{c}$ and therefore $n_{tc}^{(a)}=n_{atc}+\delta$ is known, and requires no further description. We take now a symbol b such that bt is not an internal node of T_c, and, for each child tc of t with $c\neq b$, we describe κ_t-1 counts $n_{tc}^{(a)}$, $a\neq b$ such that at is not an internal node of T_c. We can then compute $n_{tc}^{(b)}=n_{tc}-\sum_{a\neq b}n_{tc}^{(a)}$ and then $n_{tb}^{(a)}=n_t^{(a)}-\sum_{c\neq b}n_{tc}^{(a)}$ for every $a\in\mathcal{A}$. Overall, we obtain $\mathcal{K}(T_{\mathrm{suf}},x^n)$ from $\mathcal{K}(T_\mathrm{c},x^n)$ and $\sigma_{\mathbf{F}}(x^n)$ by providing $(\alpha-1)(\kappa_t-1)$ counts for each $t\in\Delta$. Thus, recalling the definition of κ_T from (4.65), we describe a total of κ_T counts, each of which can be encoded using SCCs. □

In the following theorem, we apply the results of Lemma 5.5 and Corollary 5.3 in a coding argument, obtaining the same upper bound on the expectation of $|\mathcal{T}(T,X^n)|$ previously presented in Theorem 4.18.

5.6. Theorem. *Let X^n be a random sequence emitted by a tree source $\langle T,p_T\rangle$ with entropy*

rate $\mathcal{H}$ and all conditional probabilities different from zero. Then,

$$\frac{1}{n}E_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(T,X^n)|\big] \le \mathcal{H} - \frac{|E_{T_c}|-|V_{T_c}|}{2n}\log n + O(\frac{1}{n})\,. \tag{5.3}$$

Proof. A sequence in $\mathcal{T}(T,x^n)$ can be encoded by describing the subset $\mathcal{T}(T_{\rm suf},x^n)$ to which the sequence belongs, and then, uniformly, its index within that subset. Hence, by Lemma 5.5, and Corollary 5.3, we have

$$\mathrm{E}_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(T,X^n)|\big] \le \mathrm{E}_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(T_{\rm suf},X^n)|\big] + \frac{1}{2}\kappa_T\log n + O(1)\,, \tag{5.4}$$

the left-hand side of (5.4) being a lower bound on the expected length of any such description, since the sequences in the type class are equiprobable. Normalizing, and applying Lemma 5.4 to $T_{\rm suf}$, we obtain

$$\frac{1}{n}\mathrm{E}_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(T_{\rm suf},X^n)|\big] \le \mathcal{H} - \frac{|S_{T_{\rm suf}}|(\alpha-1)}{2n}\log n + O(\frac{1}{n})\,,$$

which, together with (5.4) and Lemma 4.39, yields

$$\frac{1}{n}\mathrm{E}_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(T,X^n)|\big] \le \mathcal{H} - \frac{|E_{T_c}|-|V_{T_c}|}{2n}\log n + O(\frac{1}{n})\,. \tag{5.5}$$

□

5.4 Encoding the type class

In this section, we present an efficiently computable description of the type class $\mathcal{T}(T,x^n)$ (or, equivalently, the counts $\mathcal{K}(T,x^n)$), which, together with the enumeration of the type class, will yield the sought universal enumerative code with optimal convergence rate for tree sources. We assume, throughout, that the context tree is not trivial, i.e., $|S_T|>1$.

For FSM context trees, $\mathcal{K}_b(T,x^n)$ essentially suffices to describe $\mathcal{K}(T,x^n)$ by Lemma 5.1. In general, however, $\mathcal{K}_b(T,x^n)$ is insufficient, and, as discussed in Section 5.1, $|E_{T_c}|-|V_{T_c}|$ counts of $\log n$ bits each are needed to describe $\mathcal{K}(T,x^n)$ uniformly, which results in a suboptimal normalized expected "redundancy" of $\frac{|E_{T_c}|-|V_{T_c}|}{2n}\log n+O(\frac{1}{n})$ bits over $\mathcal{H}$. We will show that starting from a uniform encoding of $\mathcal{K}_b(T,x^n)$ with $|S_T|(\alpha-1)$ counts of $\log n$ bits each, we can complete the description of $\mathcal{K}(T,x^n)$ by encoding an additional $|E_{T_c}|-|V_{T_c}|-|S_T|(\alpha-1)$ counts requiring on average $\frac{1}{2}\log n+O(1)$ bits of description each. Together with the bound of Theorem 5.6, this reduction in code length for the additional counts will result in an optimal normalized expected redundancy of $\frac{|S_T|(\alpha-1)}{2n}\log n+O(1/n)$ bits over $\mathcal{H}$.

Let h and d denote, respectively, the minimal and maximal depth of leaves in T. For $h\le m\le d$, let $T^{[m]}$ denote the truncation of T to depth m, and let $T_{\mathbf{c}}^{[m]}$ denote the minimal canonical extension of $T^{[m]}$. Notice that, by the definition of a forgetful state (Section 4.3), no state of maximal depth of $T^{[m]}$ is refined in $T_{\mathbf{c}}^{[m]}$, and therefore $T_{\mathbf{c}}^{[m]}$ has the same depth as $T^{[m]}$. We denote by $S_{\mathbf{c}}^{[m]}$ the set of states of $T_{\mathbf{c}}^{[m]}$, and by $\sigma_{\mathbf{c}}^{[m]}(u)$ the state selected by

u in $T_{\mathbf{c}}^{[m]}$. Algorithm **EncodeTypeClass**, shown in Figure 5.2, lists the main steps in the proposed encoding of $\mathcal{K}(T, x^n)$. The algorithm starts by encoding, uniformly, the counts in $\mathcal{K}_b(T, x^n)$ with $\log n$ bits per count, and the final state of x^n in T_{c}, using a constant number of bits. It then iterates to describe, incrementally, each count set $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$ given a previously described set $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$, for $h+1 \leq k < d$. Notice that since $T^{[h]}$ is a full balanced context tree, $T^{[h+1]}$ is FSM, and, hence, $T_{\mathbf{c}}^{[h+1]} = T^{[h+1]}$. By Lemma 5.1, $\mathcal{K}(T^{[h+1]}, x^n)$ is completely determined by $\mathcal{K}_b(T, x^n)$ and the given final state. Thus, a decoder can reconstruct $\mathcal{K}(T_{\mathbf{c}}^{[h+1]}, x^n)$ from the information provided in Step 1 of **EncodeTypeClass**, and can recover $\mathcal{K}(T_{\mathbf{c}}^{[d]}, x^n)$ from the information encoded in the loop of Steps 3–4. Since $T_{\mathbf{c}}^{[d]}$ is a refinement of T, this is sufficient to reconstruct $\mathcal{K}(T, x^n)$.

EncodeTypeClass(T, x^n)

1. Encode $\mathcal{K}_b(T, x^n)$ and the final state of x^n in T_{c}.
2. Set $h = \min\{|s| : s \in S_T\}$ and $d = \max\{|s| : s \in S_T\}$.
3. For $k = h + 1$ to $d - 1$
4. Encode $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$ given $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$.

Figure 5.2: *Encoding of* $\mathcal{K}(T, x^n)$

Clearly, the crucial step in **EncodeTypeClass** is the encoding of the refinement of counters from $T_{\mathbf{c}}^{[k]}$ to $T_{\mathbf{c}}^{[k+1]}$ in Step 4, which we address next. For $u, v \in \mathcal{A}^*$, we denote by $n_{u,v}$ the number of times a transition from context $\overline{u}$ to context $\overline{v}$ occurs in x^n. In particular, when u and v are states of a context tree, $n_{u,v} = N_{u,v}$ denotes the number of times state v is selected immediately after state u. Notice that, for a state t, we have

$$n_t = \sum_{s \in S_T} n_{t,s} = \sum_{s \in S_T} n_{s,t} + \delta \,. \tag{5.6}$$

Our implementation of Step 4 will amount to describing all state transition counts $n_{s,t}$, with $s, t \in S_{\mathbf{c}}^{[k+1]}$. This set of counts is sufficient to determine $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$ as we have $n_s^{(a)} = \sum_{au \in S_{\mathbf{c}}^{[k+1]}} n_{s,au}$. However, not all the counts in the set will be explicitly described, since some will be derivable from $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$ and earlier portions of $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$. The crux of the encoding is to find a minimum subset of transition counts, and the order in which they are described, that suffice to determine *all* of them, and, at the same time, can be described economically.

We will make extensive use of the conditions presented in the following lemma, which will allow for a further simplification in the description of $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$. The proof of the lemma is straightforward, and is omitted here.

5.7. Lemma. *For k such that $h + 1 \leq k \leq d - 1$, let t be a state of $T_{\mathbf{c}}^{[k+1]}$ such that $tail(t) \notin \mathcal{I}(T_{\mathbf{c}}^{[k+1]})$. Then, all transitions into t depart from a unique state $s = \sigma_{\mathbf{c}}^{[k+1]}(\overline{tail(t)})$. Thus, from (5.6), $n_{s,t} = n_t + \delta$, and encoding $n_{s,t}$ is equivalent to encoding n_t.*

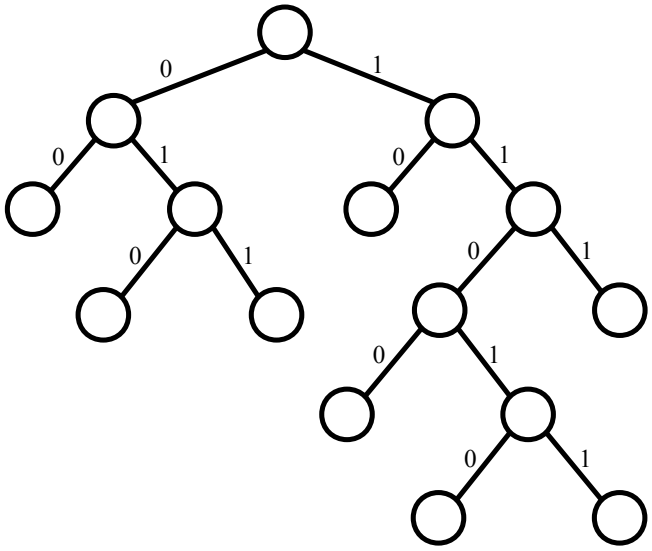

Figure 5.3: *State $t = 1100$ satisfies the conditions for the application of SCCs*, with $s' = 10$*

In our implementation of Step 4 of **`EncodeTypeClass`**, all explicit encodings will be for counts $n_{s,t}$ chosen with t satisfying the conditions of Lemma 5.7, and the following additional condition: there exists a state s' *of* T such that $s' \prec \text{tail}(t)$. In the example of Figure 5.3, the state $t = 1100$ satisfies the required conditions. It turns out that the SCCs of Section 5.2, which are defined for individual symbols, do not suffice to implement these encodings optimally. (They did suffice in Section 5.3, when used to encode symbol counts for nodes that refined the original context tree T; here, however, we need to encode counts for nodes of T.) Therefore, next, we generalize SCCs to *string count codes* (*SCCs**), which, as their name suggests, are defined on string counts rather than counts of individual symbols.

Consider a fixed string $u = u^q$ such that $s' \prec \overline{u^{q-1}}$ for some $s' \in S_T$. We define a code for $n_{\overline{u}}$, and analyze its expected code length. Let $l = \max\{\, j : 1 \le j < q-1,\ \overline{u^j} \in T \,\}$. Since $u_1 \in T$, l is well defined. Also, all proper prefixes u^i of u, $l < i < q$, are sufficiently long for $\overline{u^i}$ to determine a state in T. For $l < i < q$, define the following short-hand notations:

$$s_i = \sigma_T(u^i)\,, \qquad n_i = n_{s_i}\,, \qquad n_i^\alpha = n_{s_i}^{(u_{i+1})}\,, \qquad m_i = n_{\overline{u^i}}\ . \tag{5.7}$$

The occurrence of context u^q in x^n under the above conditions implies the occurrence of the state sequence $\{s_i\}$, $l{+}1 \le i \le q{-}1$, with symbol u_{i+1} occurring in state s_i (except possibly in border situations, which we shall ignore). In the language of Chapter 4, this is part of a forced state sequence. Indeed, in the case $\overline{u} \in S_T$, the definition of l resembles the definition of ℓ_s of Chapter 4, applied to $s = \overline{u}$. Here, however, the sequence $\{s_i\}$, $l{+}1 \le i \le q{-}1$, may be strictly shorter than the forced state sequence of s, when $\overline{u^l}$ is not an internal node but a state of T. For example, in the context tree of Figure 5.3, the state $s = 11010$ has $\ell_s = 4$, with a forced state sequence $10 \to 010 \to 10 \to 11010$. However, for the context $u = \overline{s} = 01011$, we have $l = 3$, since $\overline{u^3} = 010 \in T$.

This forced state sequence knowledge is exploited in the following manner: given m_{l+1}, the number of times context u^{l+1} occurs within x^n, we can estimate $n_{\overline{u}}$ by $m_{l+1}\prod_{i=l+1}^{q-1}\frac{n_i^\alpha}{n_i}$. Define (with a slight abuse of notation previously defined for SCCs),

$$z_u = n_{\overline{u}} - m_{l+1}\prod_{i=l+1}^{q-1}\frac{n_i^\alpha}{n_i}\,, \quad Z_u = |z_u|\,, \quad \text{and} \quad \mathrm{sg}_u = \begin{cases} 1 & z_u > 0\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{5.8}$$

In the encoding of $n_{\overline{u}}$ with SCCs*, denoted $C_u^\star(n_{\overline{u}})$, we encode $\lfloor Z_u \rfloor$ using a Golomb code of parameter $\lceil\sqrt{m_{l+1}}\rceil$, namely, a unary code for the integer quotient $\lfloor Z_u \rfloor^\top = \lfloor Z_u \rfloor / \lceil\sqrt{m_{l+1}}\rceil$, using $\lfloor Z_u \rfloor^\top + 1$ bits, concatenated with a uniform code for $\lfloor Z_u \rfloor^\perp = \lfloor Z_u \rfloor \bmod \lceil\sqrt{m_{l+1}}\rceil$ using $\log\left(\lceil\sqrt{m_{l+1}}\rceil\right)$ bits. When $n_i = 0$ for some $l < i < q$, or when $m_{l+1} = 0$, we also have $n_{\overline{u}} = 0$, and no encoding is necessary; we define $\lfloor Z_u \rfloor^\top = 0$ in this case. The results below are analogues of Lemma 5.2 and Corollary 5.3.

5.8. Lemma. *Let $\langle T, p_T\rangle$ be a tree source with all conditional probabilities different from zero. Then, the expectation $E_{\langle T,p_T\rangle}\left[\,\lfloor Z_u \rfloor^\top\,\right]$ is upper-bounded by a constant independent of n.*

5.9. Corollary. *The expected code length of $C_u^\star(n_{\overline{u}})$ is upper-bounded by $\frac{1}{2}\log n + O(1)$.*

Corollary 5.9 follows straightforwardly from Lemma 5.8. The proof of the latter is given in Appendix G.

Corollary 5.9 shows that SCCs* provide an efficient way to encode certain string occurrence counts. Lemma 5.7 provides a way to recover transition counts $n_{s,t}$ from string counts n_t, and certain subsets of these transition counts are sufficient to reconstruct $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$. We will next show that it is possible to select a minimal set of counts for strings t satisfying Lemma 5.7, and the order in which they are described, to obtain a complete description of $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$ using SCCs*. We distinguish between states of $T_{\mathbf{c}}^{[k+1]}$ that are added to $T_{\mathbf{c}}^{[k]}$ for they belong to $T^{[k+1]}$, and states that arise in $T_{\mathbf{c}}^{[k+1]}$ in order to take $T^{[k+1]}$ to canonical form. Let $U'_{k+1} = (T_{\mathbf{c}}^{[k+1]} \backslash T^{[k+1]}) \backslash T_{\mathbf{c}}^{[k]}$ be the set of nodes of $T_{\mathbf{c}}^{[k+1]}$ that are not in the original context tree $T^{[k+1]}$ and are not in $T_{\mathbf{c}}^{[k]}$. Let U_{k+1} be the set of parent nodes of elements of U'_{k+1}, $U_{k+1} = \{z : za \in U'_{k+1}, a \in \mathcal{A}\}$. The following lemma and corollary will allow us to identify an appropriate set of counts to describe $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$.

5.10. Lemma. *For $h+1 \le k \le d-1$, we have $U_{k+1} \subseteq S_{\mathbf{c}}^{[k]}$.*

5.11. Corollary. *For $h+1 \le k \le d-1$, $T_{\mathbf{c}}^{[k+1]}$ refines states of $T_{\mathbf{c}}^{[k]}$ by at most one level.*

The proof of Lemma 5.10 is deferred to Appendix G. Corollary 5.11 then follows readily.

All the encodings in Step 4 of **`EncodeTypeClass`** will be done through the auxiliary procedure **`P`** shown in Figure 5.4. The procedure relies on Corollary 5.11, as shown in the comments in Figure 5.4. We denote by $S_{\mathbf{c}}^{[m]}(r)$ the set of states of $T_{\mathbf{c}}^{[m]}$ that are children of r, i.e., of the form ra, $a \in \mathcal{A}$. Given a node r and a symbol c, $\mathbf{P}(r, c)$ describes $n_{s,t}$ for every $s \in S_{\mathbf{c}}^{[k+1]}(r)$, and every state $t \in S_{\mathbf{c}}^{[k+1]}$ such that $c = \mathrm{head}(t)$. We assume (and will later verify) that when the procedure is called, these states t satisfy the conditions of Lemma 5.7, so that $n_{s,t} = n_t + \delta$.

Procedure P(r, c)
Assumption: $cr \in \mathcal{I}(T_{\mathbf{c}}^{[k+1]})$, so Lemma 5.7 holds when coding n_t.

1. If r and cr are leaves of $T_{\mathbf{c}}^{[k]}$ but internal nodes of $T_{\mathbf{c}}^{[k+1]}$
   /* From Corollary 5.11, $rd \in S_{\mathbf{c}}^{[k+1]}$ and $crd \in S_{\mathbf{c}}^{[k+1]}$ $\forall d \in \mathcal{A}$. */
2. Use SCCs* to encode $\alpha-1$ counts $n_t = n_{crd}$, $d \in \mathcal{A}_b$.
3. [Reconstruct $n_{crb} = n_{cr} - \sum_{d \neq b} n_{crd}$
   and $n_{s,t} = n_{crd} + \delta$ for all $d \in \mathcal{A}$, with $s = rd, t = crd$].
4. else, for each $s \in S_{\mathbf{c}}^{[k+1]}(r)$
5. If cs is an internal node of $T_{\mathbf{c}}^{[k+1]}$
6. Let W' be the set of states $W' = \{csv \in T_{\mathbf{c}}^{[k+1]} \backslash T_{\mathbf{c}}^{[k]}\}$.
7. Let $W = \{w : wa \in W'\}$ be the parent nodes of W'.
8. For each $csu \in W$ /* $csu \in T_{\mathbf{c}}^{[k]}$ by Corollary 5.11. */
9. Use SCCs* to encode $\alpha-1$ counts $n_t = n_{csud}$, $d \in \mathcal{A}_b$.
10. [Reconstruct $n_{csub} = n_{csu} - \sum_{d \neq b} n_{csud}$
    and $n_{s,t} = n_{csud} + \delta$ for all $d \in \mathcal{A}$, with $t = csud$.]
11. For each state $t = csv \notin W'$ of $T_{\mathbf{c}}^{[k+1]}$
12. [Reconstruct $n_{s,t} = n_{csv} + \delta$, from $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$.]
13. else /* cs is a leaf of $T_{\mathbf{c}}^{[k+1]}$ by the assumptions. */
    /* Either $cs \in T_{\mathbf{c}}^{[k]}$, or $s \in T_{\mathbf{c}}^{[k]}$. Otherwise, by Corollary 5.11, their respective parents cr and r, would belong to $S_{\mathbf{c}}^{[k]}$ and Step 1 would have not branched to 4. */
14. [Reconstruct $n_{s,t} = n_{cs} + \delta$, from $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$, for $t = cs$]

Figure 5.4: *Encoding of state transition counts*

In the procedure, b is a fixed but arbitrary symbol from $\mathcal{A}$, and $\mathcal{A}_b = \mathcal{A} \setminus \{b\}$. Decoding steps are shown in brackets, to verify the losslessness of the code.

Notice the use of SCCs* in Steps 2 and 9. In Step 2, $t = crd$ is a state of $T_{\mathbf{c}}^{[k+1]}$ and, thus, $|cr| < k+1$. Hence, r is a leaf of $T_{\mathbf{c}}^{[k]}$ with $|r| < k$, which implies that there exists a state in T that is a proper prefix of rd for all $d \in \mathcal{A}$, which is a condition for SCCs* to be applicable. In the case of Step 9, it can be shown that the required condition is guaranteed by Step 5, using similar arguments. Furthermore, in the application of SCCs* in Steps 2 and 9, all states s_i in the definition (5.7), have a length smaller than $k+1$. Thus, n_i, n_i^α, and m_{l+1} are known from $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$.

We are now ready to present the full implementation of Step 4 of **EncodeTypeClass**, which is shown as Procedure **RefineTypeClass** in Figure 5.5. Procedure **RefineTypeClass** selects transition counts and an order of description that allows Procedure **P** (and, thus, SCCs*) to be used, and, as we shall prove, such that the total number of counts that are actually encoded is precisely $|E_{T_{\mathbf{c}}}| - |V_{T_{\mathbf{c}}}| - |S_T|(\alpha - 1)$, as needed to achieve optimal expected redundancy.

We define $R_i = \{r \in \mathcal{A}^* : ra \in S_{\mathbf{c}}^{[i]} \text{ for some } a \in \mathcal{A}\}$, namely, the set of parent nodes of states of $T_{\mathbf{c}}^{[i]}$. Procedure **RefineTypeClass** iterates over nodes $r \in R_{k+1}$, and for each $s \in S_{\mathbf{c}}^{[k+1]}(r)$, it describes all potentially nonzero state transition counts $n_{s,t}$ with $t \in S_{\mathbf{c}}^{[k+1]}$.

Procedure RefineTypeClass

1. For each $r \in R_{k+1}$ taken in ascending order of length $|r|$
2. If $r \in U_{k+1}$ /* This implies also $r \in S_{\mathbf{c}}^{[k]}$. */
3. Take $d \in \mathcal{A}$ such that $dr \notin \mathcal{I}(T_{\mathbf{c}}^{[k]})$. /* Such d must exist; otherwise, r would be a forgetful state of $T_{\mathbf{c}}^{[k]}$. */
4. Use $\mathbf{P}(r, c)$ to describe $n_{s,csu}$ for all $s \in S_{\mathbf{c}}^{[k+1]}(r)$, $c \in \mathcal{A}_d$.
5. [Let $n_{s,ds} = n_s^{(d)} = n_s - \sum_{c \neq d} n_s^{(c)}$, for all $s \in S_{\mathbf{c}}^{[k+1]}(r)$.]
6. else, If the children of r belong to $T_{\mathbf{c}}^{[k]}$
7. For each $s \in S_{\mathbf{c}}^{[k+1]}(r)$, and $c \in \mathcal{A}$, such that $cs \notin T_{\mathbf{c}}^{[k+1]}$
8. Let $s' = \sigma_{\mathbf{c}}^{[k+1]}(\overline{cs})$.
9. [Take $n_{s,s'} = n_s^{(c)}$, known from $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$.]
10. For each $s \in S_{\mathbf{c}}^{[k+1]}(r)$, and $c \in \mathcal{A}$, such that $cs \in T_{\mathbf{c}}^{[k+1]}$
11. Use $\mathbf{P}(r, c)$ to describe counts $n_{s,csu}$.
12. else, for each $s \in S_{\mathbf{c}}^{[k+1]}(r)$
13. [For $a \in \mathcal{A}_b$, $s' = \sigma_{\mathbf{c}}^{[k+1]}(\overline{as})$, let $n_{s,s'} = n_s^{(a)}$.]
14. [For $s' = \sigma_{\mathbf{c}}^{[k+1]}(\overline{bs})$, let $n_{s,s'} = n_s - \sum_{a \neq b} n_s^{(a)}$.]

Figure 5.5: *Coding and decoding of $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$ from $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$*

The correctness of the procedure is established in the following lemma. The code length is analyzed later in Lemma 5.13.

5.12. LEMMA. *Assuming that $\mathcal{K}_b(T, x^n)$, $\mathcal{K}(T_{\mathbf{c}}^{[k]}, x^n)$, and the final state of x^n in T_c are known, Procedure* **RefineTypeClass** *correctly encodes $\mathcal{K}(T_{\mathbf{c}}^{[k+1]}, x^n)$.*

Proof. The algorithm iterates over all nodes $r \in R_{k+1}$. Then, in each of the three cases distinguished by the conditions of Steps 2, 6, 12, its computations (possibly involving the use of Procedure **P**) allow the the decoder to recover the counts for all state transitions that depart from every child of r that is a state of $T_{\mathbf{c}}^{[k+1]}$. What we need to show is that required conditions at various points of the computation are satisfied, and, in particular, that the assumptions of **P** are satisfied when the procedure is invoked. The losslessness of **P** itself was established in Figure 5.4 and its discussion.

In Step 3 we ask for a symbol d such that $dr \notin \mathcal{I}(T_{\mathbf{c}}^{[k]})$. The condition $r \in U_{k+1}$ satisfied in Step 2, together with Lemma 5.10, implies that r is a state of $T_{\mathbf{c}}^{[k]}$. If $dr \in \mathcal{I}(T_{\mathbf{c}}^{[k]})$ for all $d \in \mathcal{A}$, then r would be a forgetful state, contradicting the definition of $T_{\mathbf{c}}^{[k]}$. Hence, the symbol d called for in Step 3 must exist.

In Steps 4 and 11, Procedure **P** is used to encode state transition counts departing from a state s with a symbol c, requiring $cs \in T_{\mathbf{c}}^{[k+1]}$, so that the encoded counts are of the form $n_{s,csu}$ for some string u. We claim that the condition $cs \in T_{\mathbf{c}}^{[k+1]}$ is satisfied in Step 4. Indeed, since $r \in U_{k+1}$ (as tested in Step 2), and, as argued above, r is a state of $T_{\mathbf{c}}^{[k]}$, its refinement in $T_{\mathbf{c}}^{[k+1]}$ must have been part of the process of taking $T^{[k+1]}$ to canonical form. Therefore, r is forgetful in $T^{[k+1]}$, and, thus, cr is an internal node of $T^{[k+1]}$, which implies that cr is also

an internal node of $T_{\mathbf{c}}^{[k+1]}$. In Step 11, the condition $cs \in T_{\mathbf{c}}^{[k+1]}$ is also satisfied since it is imposed in Step 10.

In Step 5 we compute $n_{s,ds}$ as $n_s^{(d)}$, implicitly assuming that ds is a state of $T_{\mathbf{c}}^{[k+1]}$. As argued above for cr, dr must be an internal node of $T_{\mathbf{c}}^{[k+1]}$. However dr is not an internal node of $T_{\mathbf{c}}^{[k]}$ by definition in Step 3, thus by Corollary 5.11, dr is a leaf of $T_{\mathbf{c}}^{[k]}$ and ds is a leaf of $T_{\mathbf{c}}^{[k+1]}$.

The computations in Steps 5 and 14 require the knowledge of n_s, the occurrence count of a state $s \in S_{\mathbf{c}}^{[k+1]}$. We claim that when the algorithm takes an element r of R_{k+1} in an iteration of the loop in Step 1, the decoder knows n_s for every $s \in S_{\mathbf{c}}^{[k+1]}(r)$. Consider a state $s \in S_{\mathbf{c}}^{[k+1]}(r)$, and let $v = \text{tail}(s)$. If v is an internal node of $T_{\mathbf{c}}^{[k+1]}$, $v \in T_{\mathbf{c}}^{[k]}$ by Corollary 5.11, and with $a = \text{head}(s)$ we have $n_s = n_v^{(a)} + \delta$, which is known, given the final state in T_{c}. If otherwise v is not an internal node of $T_{\mathbf{c}}^{[k+1]}$, all transitions into s come from a single state s' of $T_{\mathbf{c}}^{[k+1]}$ by Lemma 5.7, and we have $n_s = n_{s',s} + \delta$. In Lemma 5.7, s' is determined by $\overline{v}$ as the unique state $s' \preceq v$. Thus, s' is shorter than s, and, so, $n_{s',s}$ has already been computed in a previous iteration of the loop in Step 1, as the elements of R_{k+1} are taken in ascending order of length. Thus, the claim is proven, showing the validity of the computations in Steps 5 and 14.

In Step 6 we branch on whether the children of r belong to $T_{\mathbf{c}}^{[k]}$ or not. In the latter case, the algorithm skips to Step 12, and for every $s \in S_{\mathbf{c}}^{[k+1]}(r)$ we have that $s \notin T_{\mathbf{c}}^{[k]}$, and $s \notin U'_{k+1}$, since $r \notin U_{k+1}$ in Step 2. Hence, by definition of U'_{k+1}, all states $s \in S_{\mathbf{c}}^{[k+1]}(r)$ belong to $T^{[k+1]} \backslash T_{\mathbf{c}}^{[k]}$, thus $|s| = k+1$ and $\overline{as}$ is sufficiently long to determine a state in $T_{\mathbf{c}}^{[k+1]}$ for every symbol a. This validates the definition of s' in Steps 13 and 14, as well as the use of $\mathcal{K}_b(T, x^n)$ to determine $n_s^{(a)}$ in Step 13. □

We next analyze the expected length of the code defined by **`EncodeTypeClass`** in Figure 5.2. Clearly, we require $|S_T|(\alpha - 1)\log n + O(1)$ bits to describe $\mathcal{K}_b(T, x^n)$, and the final state of x^n in T_{c}, in the first step. We are interested now in the number of counts that are actually encoded by Procedure **`P`** as the algorithm iterates through the loop in Steps 3–4 of **`EncodeTypeClass`**. By carefully following the different cases managed by the algorithm, we will show that the number of counts given to refine the counters from $T_{\mathbf{c}}^{[k]}$ to $T_{\mathbf{c}}^{[k+1]}$ in Step 4 of **`EncodeTypeClass`** equals

$$\left(|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|\right) - \left(|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|\right) - (\alpha - 1)\left(|S_{T^{[k+1]}}| - |S_{T^{[k]}}|\right), \tag{5.9}$$

where we recall that $G_{T_{\mathbf{c}}^{[m]}} = \left(V_{T_{\mathbf{c}}^{[m]}}, E_{T_{\mathbf{c}}^{[m]}}\right)$ is the state transition support graph of $T_{\mathbf{c}}^{[m]}$. Adding over all the iterations in the loop of **`EncodeTypeClass`**, Equation (5.9) gives rise to a telescopic summation, which collapses to yield the following lemma.

5.13. Lemma. *The number of counts encoded by* **`EncodeTypeClass`** *as the algorithm iterates through the loop in Steps 3-4 is* $(|E_{T_c}| - |V_{T_c}|) - (\alpha - 1)|S_T|$

The full proof of Lemma 5.13 is presented in Appendix G. A simplified outline for canonical context trees follows, which nevertheless contains most of the main ideas.

Assume that T is canonical. It is readily verified that then, all context trees $T^{[h+1]} \dots T^{[d]}$ are canonical, and all nodes added to $T_{\mathbf{c}}^{[k]}$ to form $T_{\mathbf{c}}^{[k+1]}$ do in fact belong to T and have depth $k+1$. For a state s of $T_{\mathbf{c}}^{[k]}$, consider the set of edges in $E_{T_{\mathbf{c}}^{[k]}}$ and $E_{T_{\mathbf{c}}^{[k+1]}}$ that depart from s (or the children of s if it is refined in $T_{\mathbf{c}}^{[k+1]}$). Consider also the set of descendants from s in $V_{T_{\mathbf{c}}^{[k+1]}}$, i.e., $\{s\}$ when s is not refined, or $\{sb : b \in \mathcal{A}\}$ otherwise. Suppose first that s remains a state in $T_{\mathbf{c}}^{[k+1]}$. The set of edges in $E_{T_{\mathbf{c}}^{[k]}}$ that depart from s is only altered in $E_{T_{\mathbf{c}}^{[k+1]}}$ if $csu \in S_{\mathbf{c}}^{[k]}$ is refined in $T_{\mathbf{c}}^{[k+1]}$ for some u. In this case, there is an increment of $\alpha - 1$ in the number of edges from s to the children of csu in $E_{T_{\mathbf{c}}^{[k+1]}}$ with respect to the single edge from s to csu in $E_{T_{\mathbf{c}}^{[k]}}$. On the other hand the number of descendants from s in $V_{T_{\mathbf{c}}^{[k+1]}}$ is not altered with respect to $V_{T_{\mathbf{c}}^{[k]}}$ as s is not refined. Thus, we have a contribution of $\alpha - 1$ to the difference $\left(|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|\right) - \left(|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|\right)$, which is exactly the number of counts described by Procedure **P** in Step 9 (we rule out counts given in Step 2 of **P** since, by our assumption of T being canonical, only nodes at depth k are refined going from $T_{\mathbf{c}}^{[k]}$ to $T_{\mathbf{c}}^{[k+1]}$; since r and rc have different lengths, the condition in Step 1 cannot hold).

Suppose now that s is refined with a full complement of children in $T_{\mathbf{c}}^{[k+1]}$. This causes an increment of $\alpha - 1$ in the number of vertices in $V_{T_{\mathbf{c}}^{[k+1]}}$ with respect to $V_{T_{\mathbf{c}}^{[k]}}$. On the other hand, since T is canonical, s must be at level k and therefore there are α edges in $E_{T_{\mathbf{c}}^{[k]}}$ departing from s, and also α edges in $E_{T_{\mathbf{c}}^{[k+1]}}$ departing from each of the α children of s. Thus, we have an increment of $\alpha^2 - \alpha$ in the number of edges in $E_{T_{\mathbf{c}}^{[k+1]}}$ with respect to $E_{T_{\mathbf{c}}^{[k]}}$, and an increment of $\alpha - 1$ in the number of vertices in $V_{T_{\mathbf{c}}^{[k+1]}}$ with respect to $V_{T_{\mathbf{c}}^{[k]}}$, yielding a contribution of $(\alpha-1)^2$ to the difference $\left(|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|\right) - \left(|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|\right)$. Since T is canonical, and s is refined in $T_{\mathbf{c}}^{[k+1]}$, the conditions in Steps 2 and 6 of **RefineTypeClass** do not hold, and all transition counts departing from the children of s are reconstructed in Steps 13 and 14. Thus, we do not need to describe counts in these cases, and since we have a total of $\left(|S_{T^{[k+1]}}| - |S_{T^{[k]}}|\right)/(\alpha - 1)$ states in this situation, the negative term $-(\alpha-1)\left(|S_{T^{[k+1]}}| - |S_{T^{[k]}}|\right)$ in (5.9) arises.

The proof of Lemma 5.13 in Appendix G essentially follows the same idea. When the context tree is not canonical, however, keeping track of the different contributions to $\left(|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|\right) - \left(|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|\right)$ and the number of counts that are encoded becomes more intricate, as forgetful states are refined during the process.

EnumCodeT(T, x^n)

---

1. Encode $\mathcal{K}(T, x^n)$ using **EncodeTypeClass**
2. Encode the index of x^n within $\mathcal{T}(T, x^n)$

Figure 5.6: *Universal enumerative code with optimal convergence rate for tree sources*

All the components of a universal enumerative code with optimal convergence rate for tree sources are now in place. We call the code *EnumCodeT* and the overall scheme is

summarized in Figure 5.6 in the form of Algorithm **EnumCodeT**. The enumeration of the type class for Step 2 was the studied in Chapter 4. By Lemma 5.13 and the discussion preceding Figure 5.2, Algorithm **EncodeTypeClass** for Step 1 gives an expected code length for the description of the type class of

$$\begin{aligned}
&\mathrm{E}_{\langle T,p_T\rangle}\big[\,\mathcal{L}(\mathcal{T}(X^n))\,\big] \leq \\
&\quad\leq\ (\alpha-1)|S_T|\log n + \Big(\,(|E_{T_\mathrm{c}}|-|V_{T_\mathrm{c}}|)-(\alpha-1)|S_T|\,\Big)\frac{1}{2}\log n + O(1) \qquad (5.10)\\
&\quad=\ \frac{1}{2}(\alpha-1)|S_T|\log n + \frac{1}{2}\,(|E_{T_\mathrm{c}}|-|V_{T_\mathrm{c}}|)\log n + O(1)\,, \qquad (5.11)
\end{aligned}$$

where the first term in (5.10) comes from the number of bits used to encode $\mathcal{K}_b(T,x^n)$, and the second term from SCCs*, which, by Lemma 5.13, are used $(|E_{T_\mathrm{c}}|-|V_{T_\mathrm{c}}|)-(\alpha-1)|S_T|$ times taking, by Corollary 5.9, an average of at most $\frac{1}{2}\log n + O(1)$ bits each. Moreover, a straightforward analysis of the computation shows that **EncodeTypeClass** can be executed in time polynomial (at most quadratic) in $|S_T|$. Normalizing, and applying Theorem 5.6 to bound the expected code length of Step 2, we arrive at Theorem 5.14 below, which summarizes the main result of the chapter.

5.14. Theorem. *Let $\langle T,p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$ and all conditional probabilities different from zero. Then, EnumCodeT can be efficiently implemented, and its code length, $\mathcal{L}(X^n)$, for a random sequence X^n emitted by T satisfies*

$$E_{\langle T,p_T\rangle}\big[\,\frac{\mathcal{L}(X^n)}{n}\,\big] = \mathcal{H} + \frac{|S_T|(\alpha-1)\log n}{2n} + O(\,\frac{1}{n}\,)\,.$$

The foregoing results yield the following corollary, which will be useful to derive an alternative enumerative coding strategy.

5.15. Corollary. *The type class of x^n relative to the FSM closure, T_{suf}, of T, can be described using, on average, $\frac{1}{2}(\alpha-1)\left(|S_T|+|S_{T_{\mathrm{suf}}}|\right)\log n + O(1)$ bits.*

Proof. By Lemma 5.5, given a description of $\mathcal{K}(T,x^n)$, we can obtain one of $\mathcal{K}(T_{\mathrm{suf}},x^n)$ with a cost of $\frac{1}{2}\kappa_T\log n + O(1)$ additional bits on average. Combining with the result of Lemma 4.39, and accounting for the cost of the description of $\mathcal{K}(T,x^n)$ from (5.11), yields the claimed result. □

The result of Corollary 5.15 suggests the following alternative enumerative coding strategy: To encode x^n, encode $\mathcal{K}(T_{\mathrm{suf}},x^n)$ as described in the corollary, and then describe the index of x^n in an enumeration of $\mathcal{T}(T_{\mathrm{suf}},x^n)$. Using the bound of Lemma 5.4, applied to T_{suf}, for the expected code length of this index, we obtain an expected total normalized code length of $\mathcal{H} + \frac{1}{2}(\alpha-1)|S_T|\frac{\log n}{n} + O(1/n)$ bits. Notice that although the enumeration is done on $\mathcal{T}(T_{\mathrm{suf}},x^n)$, the expected redundancy is still optimal with respect to T.

As a final remark in this section, we notice that the optimal convergence rate of EnumCodeT to the entropy rate, given by Theorem 5.14, shows that the upper bound of Theorem 4.18 and Theorem 5.6 on the expected size of the type class of a random sequence, is tight, as stated in the following corollary.

5.16. COROLLARY. *Let X^n be a random sequence emitted by a tree source $\langle T, p_T\rangle$ with entropy rate $\mathcal{H}$ and all conditional probabilities different from zero. Then,*

$$\frac{1}{n} E_{\langle T,p_T\rangle}\big[\log|\mathcal{T}(T,X^n)|\big] \geq \mathcal{H} - \frac{|E_{T_c}|-|V_{T_c}|}{2n}\log n - O(\frac{1}{n})\,. \tag{5.12}$$

Proof. By Theorem 7.5 and the Remark 7.1 in [16], we must have

$$\mathrm{E}_{\langle T,p_T\rangle}\big[\frac{\mathcal{L}(X^n)}{n}\big] - \frac{H(X^n)}{n} + O(\frac{1}{n}) \geq \frac{|S_T|(\alpha-1)\log n}{2n}\,, \tag{5.13}$$

where $\mathcal{L}(x^n)$ denotes the code length given by EnumCodeT to x^n.

The code assigned by EnumCodeT to x^n is comprised of two parts, a description of $\mathcal{T}(x^n)$ using **`EncodeTypeClass`** and a uniform description of the index of x^n within $\mathcal{T}(x^n)$. Thus, we have

$$\mathcal{L}(x^n) = \mathcal{L}(\mathcal{T}(x^n)) + \log|\mathcal{T}(x^n)|\,,$$

where $\mathcal{L}(\mathcal{T}(x^n))$ denotes the length of the encoding of $\mathcal{T}(x^n)$ given by **`EncodeTypeClass`**.

Hence, by (5.11), we get, from (5.13),

$$\mathrm{E}_{\langle T,p_T\rangle}\big[\frac{\log|\mathcal{T}(x^n)|}{n}\big] - \frac{H(X^n)}{n} + \frac{|E_{T_c}|-|V_{T_c}|}{2n}\log n + O(\frac{1}{n}) \geq 0\,, \tag{5.14}$$

The claim then follows, since the convergence of $\frac{H(X^n)}{n}$ to the entropy rate, $\mathcal{H}$, is $O(\frac{1}{n})$, which follows from a Perron-Frobenius analysis. □

5.5 Twice-universal Coding

In this section we switch to a twice-universal setting in which the actual context tree T is unknown. Our first approach follows a conceptually simple, standard plug-in strategy in which we estimate $\hat{T}$ and then use EnumCodeT with $\hat{T}$ as if it were the true context tree underlying the model. Later, we will demonstrate an alternative approach in which the implementation of EnumCodeT can be greatly simplified for the twice-universal setting. We consider a class of penalized maximum likelihood context tree estimators. Specifically, given a sequence x^n, we assign to a context tree T a cost $K(T,x^n) = -\log\hat{\mathrm{P}}_T(x^n) + K_T f(n)$ where the *penalization coefficient* K_T is increasing with $|S_T|$, and $f(n)$ is increasing with n. We have

$$K(T,x^n) = -\sum_{s\in S_T, a\in\mathcal{A}} n_s^{(a)} \log\frac{n_s^{(a)}}{n_s} + K_T f(n)\,.$$

The context tree estimate $\hat{T}(x^n)$ for x^n is defined as the tree that minimizes the cost function $K(T,x^n)$ over all possible context trees, that is,

$$\hat{T}(x^n) = \arg\min_T\{K(T,x^n)\}\,. \tag{5.15}$$

Efficient algorithms are known for finding the minimizing context tree $\hat{T}(x^n)$ for cost functions of this kind [17]. In particular the cost function $K(T,x^n)$ obtained by taking $K_T = \frac{\alpha-1}{2}|S_T|$,

and $f(n) = \log n$, arises as the asymptotic cost assigned to T in the minimization procedure of algorithm Context, studied in Chapter3 (Equation (3.1)). For coding applications typically $f(n) = \log n$, although we may select a different function in simulation applications that we investigate in Chapter 6, and for which we will use some of the tools developed in this section.

We present a twice-universal code for tree sources, *Twice-EnumCodeT*, in the form of an algorithm, **Twice-EnumCodeT**, in Figure 5.7. Notice that since $\hat{T}$ is not known in advance, the leading string $x_{-\infty}^{0}$ may turn out to select a state that is not of maximal depth in $\hat{T}$. Thus, Step 3 requires a pre-agreement of an initial state selection strategy between the encoder and the decoder, say, for instance, the smallest lexicographically among maximal depth states of $\hat{T}$. This initial state selection may not agree with $x_{-\infty}^{0}$, in which case the encoder needs to adjust the initial setting of counts $\{n_s^{(a)}\}$.

Twice-EnumCodeT(x^n)

1. Compute the estimate $\hat{T}(x^n)$ of T.
2. Describe $\hat{T}$ to the decoder.
3. Encode x^n using **EnumCodeT** with respect to the context tree $\hat{T}$.

Figure 5.7: *Twice universal enumerative code for tree sources*

Using a natural code [56] for describing the full tree $\hat{T}(x^n)$, Step 2 of **Twice-EnumCodeT** requires one bit per node. To estimate the cost of Step 3, we must analyze the code length of EnumCodeT when applied to $\hat{T}(x^n)$ rather than T. The analysis will rely on upper bounds on the probabilities of over-estimation and under-estimation of T, which are stated in the two lemmas below. Similar bounds are well known for several estimators. For completeness we present proofs for the lemmas, adapted from [84], in Appendix H.

5.17. LEMMA. *Let $\langle T, p_T\rangle$ be a tree model and consider a penalization coefficient of the form $K_T = \beta|S_T|$. Let $O^n \subset \mathcal{A}^n$ be the set of strings for which a state of T is refined by the estimated context tree $\hat{T}$. Then, as soon as $\beta f(n) > \alpha \log n + 2$, we have $P_{\langle T,p_T\rangle}\{X^n \in O^n\} \leq |S_T| n^{\alpha^2+1} 2^{\beta(1-\alpha)f(n)+2\alpha}$.*

5.18. LEMMA. *Let $\langle T, p_T\rangle$ be a minimal tree model with all conditional probabilities different from zero, and consider a penalization coefficient of the form $K_T = \beta|S_T|$, and $f(n) = o(n)$. Let $U^n \subset \mathcal{A}^n$ be the set of sequences whose estimated context tree $\hat{T}$ has a state that is refined by T. Then $P_{\langle T,p_T\rangle}\{X^n \in U^n\} \leq R2^{-nD}$ for positive constants R, D and n sufficiently large.*

From Lemma 5.17 and Lemma 5.18 it follows that we can choose β and $f(n)$ to make the contribution of sequences with estimated context tree $\hat{T} \neq T$ to the expected code length negligible, as long as the code length is upper-bounded by a polynomial in n. We verify this fact next. In **Twice-EnumCodeT** we describe $\mathcal{K}(\hat{T}, x^n)$ by encoding the final state of x^n with respect to $\hat{T}_{\mathbf{c}}$, encoding $\mathcal{K}_b(\hat{T}, x^n)$ with $|S_{\hat{T}}|(\alpha-1)$ counts of $\log n$ bits each, and finally giving an additional set of $|E_{\hat{T}_{\mathbf{c}}}| - |V_{\hat{T}_{\mathbf{c}}}| - |S_{\hat{T}}|(\alpha-1)$ counts described with SCCs*, which take $O(\sqrt{n})$

bits each. Thus, the complete description of $\mathcal{K}(\hat{T}, x^n)$ takes $O\left(\left(|E_{\hat{T}_\mathbf{c}}| - |V_{\hat{T}_\mathbf{c}}|\right)\sqrt{n}\right)$ bits. From the definition of a forgetful state, it is readily verified that $|E_{\hat{T}_\mathbf{c}}| - |V_{\hat{T}_\mathbf{c}}| \leq |E_{\hat{T}}| - |V_{\hat{T}}|$, and from the definition of $|E_{\hat{T}}|$ it is not difficult to see that $|E_{\hat{T}}| = O(|S_{\hat{T}}|)$. Hence, the cost of describing the type class of x^n with respect to $\hat{T}$ is $O(|S_{\hat{T}}|\sqrt{n})$. Since the index of x^n within its class takes no more than n bits, we upper-bound the total code length of Twice-EnumCodeT by $O\left(|S_{\hat{T}}|\sqrt{n} + n\right)$. To obtain the desired bound on the total code length, it remains to bound $|S_{\hat{T}}|$ by a polynomial, which follows from the lemma below. The proof of the lemma uses similar arguments as those used in Chapter 3, from [3], to bound the size of the context tree estimated by the algorithm Context.

5.19. Lemma. *Let $\hat{T}$ be the estimated context tree for x^n with a penalization coefficient of the form $K_T = \beta|S_T|$. We have $|S_{\hat{T}}| = O(n/f(n))$.*

Proof. Since $\hat{T}$ minimizes $K(T, x^n)$, comparing it with the single node context tree (i.e., a zero-order model) we get,

$$-\log \hat{\mathrm{P}}_{\hat{T}}(x^n) + \beta|S_{\hat{T}}|f(n) \leq n\hat{\mathcal{H}}(x^n) + \beta f(n)\,,$$

where $\hat{\mathcal{H}}(x^n) \leq \log\alpha$ is the memoryless empirical entropy of x^n. Since $-\log \hat{\mathrm{P}}_{\hat{T}}(x^n) \geq 0$ we conclude that $|S_{\hat{T}}| = O(n/f(n))$. □

Lemma 5.19 and the preceding discussion yield the desired polynomial bound on the total code length, leading to the following result.

5.20. Theorem. *Let $\langle T, p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$ and with all conditional probabilities different from zero. Taking $f(n)$ of order at least $\log n$, and a penalization coefficient $K_T = \beta|S_T|$ with β sufficiently large, the normalized expected code length of Twice-EnumCodeT is*

$$E_{\langle T,p_T\rangle}\left[\,\frac{\mathcal{L}(X^n)}{n}\,\right] = \mathcal{H} + \frac{|S_T|(\alpha-1)\log n}{2n} + O(1/n)\,.$$

In the rest of the section we present an alternative code EnumCodeT′, whose implementation is a simplification of **EnumCodeT**, applicable when the target context tree is an estimate $\hat{T}$, as in Step 3 of **Twice-EnumCodeT**. Recall that a fundamental tool in EnumCodeT is the use of SCCs* codes, a generalization of SCCs. The latter rely on the quantities $\Delta(w,s,a) = \left|n_w^{(a)} - \frac{n_s^{(a)}}{n_s}n_w\right| n_w^{-\frac{1}{2}}$, for properly defined strings s, w and symbol a, being small with high probability. In other words, sequences with large values $\Delta(w,s,a)$ have small probability under *any* model parameter, and we would expect an estimate $\hat{T}(x^n) \neq T$ for such sequences. The following lemma formalizes these claims.

5.21. Lemma. *Let $\hat{T} \triangleq \hat{T}(x^n)$ be a context tree estimate for x^n, and let s and w be strings such that $s \in S_{\hat{T}}$, $s \prec w$, and $n_s > 0$. Then, for any refinement T' of $\hat{T}$ that contains w, we have*

$$\left|n_w^{(a)} - \frac{n_s^{(a)}}{n_s}n_w\right| \leq \sqrt{2(\ln 2)(K_{T'} - K_{\hat{T}})n_s f(n)}\,.$$

Proof.

Since K_T is increasing in the number of states of T, it is sufficient to consider the case in which T' is the smallest refinement of $\hat{T}(x^n)$ that contains w, i.e., the tree that results from refining $\hat{T}(x^n)$ by adding $w'b$ for all proper prefixes w' of w and all symbols $b \in \mathcal{A}$. Let $W = \{su : su \in S_{T'}\}$. Since $\hat{T}(x^n)$ minimizes the cost function $K(T, x^n)$, we have

$$-\sum_{t \in S_{\hat{T}}, a \in \mathcal{A}} n_t^{(a)} \log \frac{n_t^{(a)}}{n_t} + K_{\hat{T}} f(n) \leq -\sum_{t \in S_{T'}, a \in \mathcal{A}} n_t^{(a)} \log \frac{n_t^{(a)}}{n_t} + K_{T'} f(n)\,. \tag{5.16}$$

Therefore,

$$-\sum_{t \in S_{\hat{T}}, a \in \mathcal{A}} n_t^{(a)} \log \frac{n_t^{(a)}}{n_t} + \sum_{t \in S_{T'}, a \in \mathcal{A}} n_t^{(a)} \log \frac{n_t^{(a)}}{n_t} \leq (K_{T'} - K_{\hat{T}}) f(n)\,, \tag{5.17}$$

which reduces to

$$-\sum_{a \in \mathcal{A}} n_s^{(a)} \log \frac{n_s^{(a)}}{n_s} + \sum_{su \in W, a \in \mathcal{A}} n_{su}^{(a)} \log \frac{n_{su}^{(a)}}{n_{su}} \leq (K_{T'} - K_{\hat{T}}) f(n)\,. \tag{5.18}$$

Since $n_s > 0$, we further obtain

$$-\sum_{a \in \mathcal{A}} \frac{n_s^{(a)}}{n_s} \log \frac{n_s^{(a)}}{n_s} + \sum_{su \in W} \frac{n_{su}}{n_s} \sum_{a \in \mathcal{A}} \frac{n_{su}^{(a)}}{n_{su}} \log \frac{n_{su}^{(a)}}{n_{su}} \leq (K_{T'} - K_{\hat{T}}) \frac{f(n)}{n_s}\,. \tag{5.19}$$

Let $\hat{p}(\cdot|s)$ be the probability mass function over $\mathcal{A}$ given by $\hat{p}(a|s) = \frac{n_s^{(a)}}{n_s}$ and analogously for $su \in W$, $\hat{p}(a|su) = \frac{n_{su}^{(a)}}{n_{su}}$. Consider also a PMF $\hat{p}(\cdot)$ over W given by $\hat{p}(su) = \frac{n_{su}}{n_s}$. Let X, Y be random variables such that Y takes values in W with $Y \sim \hat{p}(\cdot)$, and X takes values in $\mathcal{A}$ with conditional distribution $P(X = a|Y = su) = \hat{p}(a|su)$. Then, the marginal distribution of X is $X \sim \hat{p}(\cdot|s)$, and the joint distribution of X and Y is

$$P(X = a, Y = su) = P(X = a|Y = su)P(Y = su) = \hat{p}(a|su)\hat{p}(su) = \frac{n_{su}^{(a)}}{n_{su}} \frac{n_{su}}{n_s} = \frac{n_{su}^{(a)}}{n_s}\,.$$

From Equation (5.19),

$$I(X;Y) = H(X) - H(X|Y) \leq (K_{T'} - K_{\hat{T}}) \frac{f(n)}{n_s}\,.$$

Let Q be a joint distribution given by the product of the marginal distributions of X, Y, i.e., $Q(X = a, Y = su) = P(X = a)P(Y = su) = \frac{n_s^{(a)}}{n_s} \frac{n_{su}}{n_s}$. Then, by Pinsker's inequality [12, Lemma 12.6.1], we have

$$\frac{1}{2 \ln 2} \|P - Q\|_1^2 \leq D(P||Q) = I(X;Y) \leq (K_{T'} - K_{\hat{T}}) \frac{f(n)}{n_s}\,. \tag{5.20}$$

Therefore,

$$\left(\sum_{a\in\mathcal{A},su\in W}|P(a,su)-Q(a,su)|\right)^2 \le 2(\ln 2)(K_{T'}-K_{\hat{T}})\frac{f(n)}{n_s}\,, \tag{5.21}$$

which takes the form

$$\sum_{a\in\mathcal{A},su\in W}\left|\frac{n_{su}^{(a)}}{n_s}-\frac{n_s^{(a)}}{n_s}\frac{n_{su}}{n_s}\right| \le \sqrt{2(\ln 2)(K_{T'}-K_{\hat{T}})\frac{f(n)}{n_s}}\,. \tag{5.22}$$

In particular, taking only the term corresponding to $su = w$ in the summation on the left hand side of (5.22), we conclude that

$$\left|n_w^{(a)}-\frac{n_s^{(a)}}{n_s}n_w\right| \le \sqrt{2(\ln 2)(K_{T'}-K_{\hat{T}})n_s f(n)}\,. \tag{5.23}$$

□

With a linear penalization coefficient of the form $K_T = \beta|S_T|$, we have $K_{T'} - K_{\hat{T}} = \beta(|S_{T'}| - |S_{\hat{T}}|) \le \beta\alpha|w|$, implying the following corollary.

5.22. Corollary. *Let $\hat{T}(x^n)$ be a context tree estimate for x^n with a penalization coefficient $K_T = \beta|S_T|$, $s \in S_{\hat{T}}$, $s \prec w$, and $n_s > 0$. Then,*

$$|z_{w,a}| = \left|n_w^{(a)}-\frac{n_s^{(a)}}{n_s}n_w\right| \le \sqrt{2(\ln 2)\beta\alpha|w|n_s f(n)}\,.$$

It follows from Corollary 5.22 that, when considering coding with respect to a context tree $\hat{T}$ estimated taking $f(n) = \log n$, and using a linear penalization coefficient $K_T = \beta|S_T|$, it may be advantageous to replace the use of SCCs with a uniform coding of $z_{w,a}$ in the range $\left(-\sqrt{2\beta\alpha|w|n_s\ln n}, \sqrt{2\beta\alpha|w|n_s\ln n}\right)$. The code length obtained would be $\frac{1}{2}\log n_s + o(\log n)$, which is of similar main order as the expected code length of SCCs (cf. Corollary 5.3). Notice, however, that the upper bound here is *pointwise*, and not just in expectation. The same idea can be generalized to SCCs* by means of Lemma 5.23 below, for which we recall the definitions from (5.7).

5.23. Lemma. *Let $\hat{T}$ be a context tree estimate for x^n with a penalization coefficient $K_T = \beta|S_T|$, and let u^q be a string such that an SCC* code is applicable in $\hat{T}$. Then,*

$$|z_u| = \left|n_{\overline{u}} - m_{l+1}\prod_{i=l+1}^{q-1}\frac{n_i^\alpha}{n_i}\right| \le q^{3/2}\sqrt{2(\ln 2)\beta\alpha n f(n)}\,. \tag{5.24}$$

Proof. We prove the result by repeatedly applying Corollary 5.22 as follows. Notice that $m_i = n\frac{(u_i)}{u^{i-1}}$. Then, by Corollary 5.22, we have $\left|m_i - \frac{n_{i-1}^\alpha}{n_{i-1}}m_{i-1}\right| \le \sqrt{2(\ln 2)\beta\alpha|u|nf(n)}$ for all $l < i \le q$ (we extend the definition of m_i for $i = q$ as $m_q = n_{\overline{u^q}}$).

Applying Corollary 5.22 again and grouping terms, we further obtain $\left| m_i - \frac{n_{i-1}^{\alpha}}{n_{i-1}} \frac{n_{i-2}^{\alpha}}{n_{i-2}} m_{i-2} \right| \leq \left(1 + \frac{n_{i-1}^{\alpha}}{n_{i-1}}\right) \sqrt{2(\ln 2)\beta\alpha|u|nf(n)}$. Starting from $m_i = m_q = n_{\overline{u^q}}$, and repeatedly applying the same arguments, we finally bound $|z_u|$ and the claim of the lemma follows readily. □

Using a uniform encoding for the numbers z_u taking $f(n) = \log n$ leads, by (5.24), to an analogue of Corollary 5.9 in the twice-universal setting. As before, we note that the bound here is *pointwise*, as opposed to in expectation as in Corollary 5.9. The results take advantage of the idea suggested in Section 1.5, and the beginning of this chapter, namely, that for some type classes of $\hat{T}$, no sequence in the class will estimate $\hat{T}$. Therefore, these "atypical" classes can be excluded from the coding space. To implement this idea, we define the code EnumCodeT′ exactly as EnumCodeT, but replacing the use of SCCs*, $C_u^\star(n_{\overline{u}})$, by a uniform encoding of z_u in the range $\left(-|u|^{3/2}\sqrt{2\beta\alpha n \ln n}, |u|^{3/2}\sqrt{2\beta\alpha n \ln n}\right)$. In EnumCodeT, SCCs* are applied to strings u that are prefixes of states of the minimal canonical extension of the context tree. Hence, $|u|$ is bounded by the depth of $\hat{T}(x^n)$. For sequences that estimate $\hat{T}(x^n) = T$, $|u|$ is bounded by the depth of T, and the uniform encoding of z_u, by (5.24), takes $\frac{1}{2}\log n + O(\log\log n)$ bits. Thus, when $\hat{T}(x^n) = T$, the upper bounds of the expected code lengths of EnumCodeT′ and EnumCodeT differ by $O(\log\log n)$ bits. As a result, using essentially the same arguments as in Theorem 5.20, we can prove the following theorem for *Twice-EnumCodeT′*, the code obtained by substituting an implementation of EnumCodeT′ for **`EnumCodeT`** in **`Twice-EnumCodeT`**.

5.24. Theorem. *Let $\langle T, p_T\rangle$ be a tree source with entropy rate $\mathcal{H}$ and with all conditional probabilities nonzero. Estimating $\hat{T}(x^n)$ with $f(n) = \log n$ and a penalization coefficient $K_T = \beta|S_T|$, with β sufficiently large, the normalized expected code length of Twice-EnumCodeT′ is*

$$E_{\langle T,p_T\rangle}\left[\frac{\mathcal{L}(X^n)}{n}\right] = \mathcal{H} + \frac{|S_T|(\alpha-1)\log n}{2n} + O\left(\frac{\log\log n}{n}\right).$$

Remark. As with Twice-EnumCodeT, Twice-EnumCodeT′ requires a pre-agreement of an initial state selection strategy between the encoder and the decoder since, in principle, the fixed leading string $x_{-\infty}^0$ may turn out to select a state that is not of maximal depth in $\hat{T}$. In this case, a count $n_w^{(a)}$ may change to $n'^{(a)}_w$ after fixing the new initial conditions. Since $|n'^{(a)}_w - n_w^{(a)}| \leq |w|$, it is not difficult to show that Corollary 5.22 is still valid for $\left| n'^{(a)}_w - \frac{n'^{(a)}_s}{n'_s} n'_w \right|$ with the addition of a correction term of order $O(|w|^2)$ on the right hand side, which does not affect the validity of Theorem 5.24. Alternatively, the encoder may describe $\mathcal{T}(\hat{T}, x^n)$ by giving the original counts $\{n_s^{(a)}\}$ with respect to $x_{-\infty}^0$, together with additional information that lets the decoder compute the modified counts. For example, the encoder may send $\log|\mathcal{I}(\hat{T})| + \log\alpha$ bits to describe the longest internal node of $\hat{T}$, u, such that $\overline{u} \prec x^n$, and the symbol $x_{|u|+1}$. This, again, does not affect the validity of Theorem 5.24. Notice that this alternative can be interpreted as describing the type class $\mathcal{T}'(\hat{T}, x^n)$, defined in Appendix E, which takes into account transient states of $\hat{T}$. Thus, the second part of the encoding may be slightly shortened in this case by describing the index of x^n within the smaller set $\mathcal{T}'(\hat{T}, x^n)$, as in Appendix E.

Chapter 6

Universal tree type classes and simulation of individual sequences

Universal type classes were presented in [67] as a partition of the set of sequences $\mathcal{A}^n$ into equivalence classes that can be seen as analogous to conventional type classes in situations where the model structure (e.g., Markov order) is not known a priori. More precisely, in [67] a universal (LZ) type class is defined as the set of all sequences of a given length that yield the same LZ parsing tree [91]. It is shown in [67] that any two sequences in the same class satisfy the following statistical similarity property.

P1 For any fixed positive integer j, the variational distance between the empirical distributions of j-tuples corresponding to the two sequences is a vanishing function of n.

It is also required in [67] that the universal type class of a sequence x^n be "as large as possible", in a sense to be discussed in the sequel.

In this chapter we present a partition of $\mathcal{A}^n$ that satisfies similar properties, but it exhibits a better convergence rate of the statistics as n grows. Our approach is based on some of the tools developed in Chapter 5. Specifically, in Section 5.5 we observed that if we know the context tree estimate of x^n, $\hat{T} = \hat{T}(x^n)$, then the number of occurrences of certain patterns within x^n can be estimated to order $\sqrt{f(n)}$ given a well characterized set of symbol counts that depend on $\mathcal{T}(\hat{T}, x^n)$ (see Lemma 5.23). This observation, which led to an economic description of $\mathcal{T}(\hat{T}, x^n)$ in a twice-universal enumerative code for tree sources, suggests that other sequences in $\mathcal{T}(\hat{T}, x^n)$ that also estimate $\hat{T}$ would have similar statistics as x^n. Thus, we define the *universal tree type class* of x^n as

$$\mathcal{U}(x^n) = \{y^n \in \mathcal{T}(\hat{T}(x^n), x^n) : \hat{T}(y^n) = \hat{T}(x^n)\},$$

i.e., two sequences belong to the same universal tree type class if and only if they estimate the same context tree, $\hat{T}$, and they belong to the same type class with respect to $\hat{T}$. A similar notion of *universal Markov type class*, denoted $\mathcal{M}(x^n)$, was defined in [47] where a plain Markov order estimator substitutes for the context tree estimator. As we shall see, the convergence of the statistics for an length-r pattern as n grows is similar for universal Markov type classes and universal tree type classes, but the latter exhibit a better behavior with respect to the convergence rate as a function of r.

In Section 6.1 we study the statistical similarity properties of universal tree type classes. We show that Property P1 is satisfied for any two sequences in the same class, and, moreover, we characterize precisely the convergence rate of the statistics. This naturally motivates the application of universal tree type classes to *simulation of individual sequences* in the same

setting as [67]. In this setting the goal is to emit a simulated sequence that is statistically similar to a given training sequence, in a well mathematically defined sense, while, at the same time, keeping as much uncertainty as possible on the output given the input sequence. In Section 6.2 we formally state the problem of simulation of individual sequences and we present a simulation scheme based on universal tree type classes. We compare the performance of this simulation scheme, in terms of entropy of the output given the input, against other simulators with similar statistics preservation properties.

When estimating context trees $\hat{T}$, we focus on penalized maximum-likelihood estimators of the kind shown in (5.15) in Chapter 5, with a linear penalization coefficient $K_T = \beta|S_T|$. Thus,

$$\hat{T}(x^n) = \arg\min_T\{K(T, x^n)\}\,; \quad K(T, x^n) = -\sum_{s\in S_T, a\in\mathcal{A}} n_s^{(a)} \log \frac{n_s^{(a)}}{n_s} + \beta|S_T| f(n)\,, \tag{6.1}$$

where, in the *penalization function* $\beta|S_T|f(n)$, β is a positive constant and $f(n)$ is increasing with n.

6.1 Statistical similarity properties

Theorem 6.1 below establishes the statistical similarity between two sequences in the same universal tree type class. The theorem shows that sequences in the same universal tree type class have similar statistics of any order. This property is analogous to the exact equiprobability of sequences within conventional type classes. We recall from Section 5.1 that n_w denotes the number of times a symbol of x^n occurs in context $\overline{w}$, where we assume that x^n is preceded by a fixed semi-infinite string $x_{-\infty}^0$. Thus, n_w is the number of occurrences of the pattern $\overline{w}$ within x^n, up to a difference bounded by $|w|$ due to border conditions.

6.1. THEOREM. *Let $y^n \in \mathcal{U}(x^n)$ and let w^r be any fixed string. Then,*

$$\left|\frac{n_{\overline{w}}(x^n)}{n} - \frac{n_{\overline{w}}(y^n)}{n}\right| = O\left(r^{3/2}\sqrt{\frac{f(n)}{n}}\right). \tag{6.2}$$

Proof. Let $\hat{T} = \hat{T}(x^n) = \hat{T}(y^n)$. If $\overline{w} \in \hat{T}$, then $n_{\overline{w}}(x^n) = n_{\overline{w}}(y^n)$, and there is nothing to prove. Otherwise let $l = \max\{j : 1 \le j \le r, \overline{w^j} \in \hat{T}\}$, and for $l \le i \le r$ define

$$\begin{aligned}
s_i &= \sigma_{\hat{T}}(w^i),\\
n_i &= n_{s_i}(x^n) = n_{s_i}(y^n),\\
n_i^\alpha &= n_{s_i}^{(w_{i+1})}(x^n) = n_{s_i}^{(w_{i+1})}(y^n), \text{ for } i < r,\\
u_i &= \overline{w^i},\\
m_i &= n_{u_i}(x^n),\\
m_i' &= n_{u_i}(y^n).
\end{aligned}$$

Let T_i be the smallest extension of $\hat{T}$ such that $u_i \in T_i$, and let $M_i = \sqrt{2(\ln 2)(K_{T_i} - K_{\hat{T}})}$. By the linearity of K_T, $K_{T_i} - K_{\hat{T}} = \beta(|S_{T_i}| - |S_{\hat{T}}|) \leq \beta(\alpha - 1)i$. Thus,

$$M_i \leq \sqrt{2(\ln 2)\beta(\alpha - 1)i}\,.$$

Notice that $m_i = n_{u_{i-1}}^{(w_i)}(x^n)$ and, by Lemma 5.21, we have

$$\frac{n_{i-1}^{\alpha}}{n_{i-1}} m_{i-1} - M_{i-1}\sqrt{nf(n)} \leq m_i \leq \frac{n_{i-1}^{\alpha}}{n_{i-1}} m_{i-1} + M_{i-1}\sqrt{nf(n)}\,. \tag{6.3}$$

By recursively applying (6.3) to m_{i-1} in the role of m_i on the right hand side of (6.3) we obtain,

$$m_i \leq \frac{n_{i-1}^{\alpha}}{n_{i-1}} \left(\frac{n_{i-2}^{\alpha}}{n_{i-2}} m_{i-2} + M_{i-2}\sqrt{nf(n)} \right) + M_{i-1}\sqrt{nf(n)}\,,$$

which we further bound as,

$$m_i \leq \frac{n_{i-1}^{\alpha}}{n_{i-1}} \frac{n_{i-2}^{\alpha}}{n_{i-2}} m_{i-2} + (M_{i-1} + M_{i-2})\sqrt{nf(n)}\,.$$

In general, by repeatedly applying the same argument, we conclude that

$$m_r \leq m_{l+1} \prod_{i=l+1}^{r-1} \frac{n_i^{\alpha}}{n_i} + \left(\sum_{j=l+1}^{r-1} M_j \right) \sqrt{nf(n)}\,.$$

With the same derivation applied on the left hand side of (6.3), we get

$$\left| m_r - m_{l+1} \prod_{i=l+1}^{r-1} \frac{n_i^{\alpha}}{n_i} \right| \leq \left(\sum_{j=l+1}^{r-1} M_j \right) \sqrt{nf(n)}\,, \tag{6.4}$$

and, similarly, for y^n we have

$$\left| m'_r - m'_{l+1} \prod_{i=l+1}^{r-1} \frac{n_i^{\alpha}}{n_i} \right| \leq \left(\sum_{j=l+1}^{r-1} M_j \right) \sqrt{nf(n)}\,. \tag{6.5}$$

Now, by the definition of l, we know that $u_l \in \hat{T}$ and thus we have

$$m_{l+1} = \sum_{s \in S_{\hat{T}} : u_l \preceq s} n_s^{(w_{i+1})} = m'_{l+1}\,. \tag{6.6}$$

Hence, from (6.4), (6.5) and (6.6) we get

$$|m_r - m'_r| \leq 2 \left(\sum_{j=l+1}^{r-1} M_j \right) \sqrt{nf(n)}\,,$$

and dividing by n we obtain

$$\left| \frac{n_{\overline{w}}(x^n)}{n} - \frac{n_{\overline{w}}(y^n)}{n} \right| \leq 2 \left(\sum_{j=l+1}^{r-1} M_j \right) \sqrt{\frac{f(n)}{n}}\,. \tag{6.7}$$

The proof is completed by noting that

$$\sum_{j=l+1}^{r-1} M_j \le \sqrt{2(\ln 2)\beta(\alpha-1)} \sum_{j=1}^{r-1} \sqrt{i} = O(r^{3/2}) \,. \tag{6.8}$$

□

The asymptotic rate of convergence of the L_1 distance of the empirical probability for a fixed pattern in (6.2) is similar to that obtained for universal Markov type classes in [47], which in turn improves on the $O(1/\log n)$ rate shown for the LZ-based partition of [67]. The dependence on the pattern length, r, however, is different. While the dependence is linear in r in [67] and $O(r^{3/2})$ in Theorem 6.1, all that we can prove for universal Markov type classes is that the accumulated L_1 distance over the exponentially many length r patterns grows exponentially with r [47], but without a specific nontrivial bound for individual patterns. This can be attributed to the gain in flexibility of context trees, which can grow unbalanced to fit the data faster with different Markov orders for different patterns, as opposed to plain Markov order estimators, which fit the same order to every pattern.

The constant β in the linear penalization coefficient in (6.1) only affects the constant in the upper bound for $\left|\frac{n_{\overline{w}}(x^n)}{n} - \frac{n_{\overline{w}}(y^n)}{n}\right|$ in the proof of the theorem (equations (6.7) and (6.8) above). On the other hand, the function $f(n)$ determines the order of the convergence rate of the statistics and, moreover, it also determines the size of the classes and the number of classes. A slower growth of $f(n)$ favors larger context trees, which yields a larger number of different classes. Diminishing $f(n)$ also yields smaller classes as we can foresee by observing that the statistical similarity condition of Theorem 6.1 becomes tighter when f decreases, thus reducing the universe of sequences that satisfy this condition.

We show in Lemma 6.2 below that all sequences in $\mathcal{T}(\hat{T}(x^n), x^n) \backslash \mathcal{U}(x^n)$ estimate context trees that are refinements of $\hat{T}(x^n)$.

6.2. Lemma. *Assume that ties in (6.1) are consistently broken in favor of either smaller or larger context trees. If $y^n \in \mathcal{T}(\hat{T}(x^n), x^n) \backslash \mathcal{U}(x^n)$, then $\hat{T}(x^n) \subset \hat{T}(y^n)$.*

Proof. Assume without loss of generality that ties are broken in favor of smaller context trees in (6.1). Suppose that s is a state of $\hat{T}(y^n)$ that is refined in $\hat{T}(x^n)$. Let $W = \{sw \in \hat{T}(x^n) : w \in \mathcal{A}^+\}$, and let S_W be the set of leaves descending from s in $\hat{T}(x^n)$. Define the context trees

$$\hat{T}'(y^n) = \hat{T}(y^n) \cup W \,, \qquad \hat{T}'(x^n) = \hat{T}(x^n) \backslash W \,.$$

The context trees $\hat{T}(x^n)$ and $\hat{T}'(x^n)$ differ only in that s is refined by a subtree W in $\hat{T}(x^n)$. Hence, we have

$$-\log \hat{\mathrm{P}}_{\hat{T}(x^n)}(x^n) + \log \hat{\mathrm{P}}_{\hat{T}'(x^n)}(x^n) = -\sum_{su \in S_W} \sum_{a \in \mathcal{A}} n_{su}^{(a)}(x^n) \log \frac{n_{su}^{(a)}(x^n)}{n_{su}(x^n)} + \sum_{a \in \mathcal{A}} n_s^{(a)}(x^n) \log \frac{n_s^{(a)}(x^n)}{n_s(x^n)} \,,$$

or, since $y^n \in \mathcal{T}(\hat{T}(x^n), x^n)$, we can also write

$$-\log \hat{\mathrm{P}}_{\hat{T}(x^n)}(x^n)+\log \hat{\mathrm{P}}_{\hat{T}'(x^n)}(x^n) = -\sum_{su\in S_W}\sum_{a\in\mathcal{A}} n_{su}^{(a)}(y^n)\log\frac{n_{su}^{(a)}(y^n)}{n_{su}(y^n)}+\sum_{a\in\mathcal{A}} n_s^{(a)}(y^n)\log\frac{n_s^{(a)}(y^n)}{n_s(y^n)}\,. \tag{6.9}$$

The context trees $\hat{T}(y^n)$ and $\hat{T}'(y^n)$ also differ only in that s is refined by a subtree W in $\hat{T}'(y^n)$. Hence, we have

$$-\log \hat{\mathrm{P}}_{\hat{T}'(y^n)}(y^n)+\log \hat{\mathrm{P}}_{\hat{T}(y^n)}(y^n) = -\sum_{su\in S_W}\sum_{a\in\mathcal{A}} n_{su}^{(a)}(y^n)\log\frac{n_{su}^{(a)}(y^n)}{n_{su}(y^n)}+\sum_{a\in\mathcal{A}} n_s^{(a)}(y^n)\log\frac{n_s^{(a)}(y^n)}{n_s(y^n)}\,,$$

and, by (6.9), we get

$$-\log \hat{\mathrm{P}}_{\hat{T}'(y^n)}(y^n) + \log \hat{\mathrm{P}}_{\hat{T}(y^n)}(y^n) = -\log \hat{\mathrm{P}}_{\hat{T}(x^n)}(x^n) + \log \hat{\mathrm{P}}_{\hat{T}'(x^n)}(x^n)\,. \tag{6.10}$$

Since $\hat{T}(x^n)$ is the context tree estimate for x^n, by (6.1) we have

$$-\log \hat{\mathrm{P}}_{\hat{T}(x^n)}(x^n) + \log \hat{\mathrm{P}}_{\hat{T}'(x^n)}(x^n) \le \beta(|S_{\hat{T}'(x^n)}| - |S_{\hat{T}(x^n)}|)f(n)\,. \tag{6.11}$$

Moreover, since ties are broken in favor of smaller context trees, the inequality in (6.11) must be strict and we further get

$$-\log \hat{\mathrm{P}}_{\hat{T}(x^n)}(x^n) + \log \hat{\mathrm{P}}_{\hat{T}'(x^n)}(x^n) < \beta(|S_{\hat{T}'(x^n)}| - |S_{\hat{T}(x^n)}|)f(n)\,,$$

or, since $\hat{T}(x^n) \setminus \hat{T}'(x^n) = W = \hat{T}'(y^n) \setminus \hat{T}(y^n)$, we also have

$$-\log \hat{\mathrm{P}}_{\hat{T}(x^n)}(x^n) + \log \hat{\mathrm{P}}_{\hat{T}'(x^n)}(x^n) < \beta(|S_{\hat{T}(y^n)}| - |S_{\hat{T}'(y^n)}|)f(n)\,.$$

Thus, by (6.10), we get

$$-\log \hat{\mathrm{P}}_{\hat{T}'(y^n)}(y^n) + \log \hat{\mathrm{P}}_{\hat{T}(y^n)}(y^n) < \beta(|S_{\hat{T}'(y^n)}| - |S_{\hat{T}(y^n)}|)f(n)\,,$$

which contradicts $\hat{T}(y^n)$ being the context tree estimate for y^n. □

Lemma 6.2 confirms the intuition that diminishing $f(n)$ yields smaller classes; consider two penalization functions given by $K_T f(n)$ and $K_T f'(n)$, with $f > f'$, and suppose that x^n estimates the same context tree, $\hat{T}$, with both estimators. Let $\mathcal{U}(x^n)$ and $\mathcal{U}'(x^n)$ denote the classes of x^n calculated with respect to the estimator induced by f and f', respectively. Since a smaller penalization function favors larger context trees, some sequence of $\mathcal{U}(x^n)$ may estimate extensions of $\hat{T}$ with the smaller penalization function f'. Then, we must have $\mathcal{U}(x^n) \supset \mathcal{U}'(x^n)$ since, by Lemma 6.2, there is no sequence in $\mathcal{T}(\hat{T}, x^n)$ that estimates a context tree smaller than $\hat{T}$, which could otherwise belong to $\mathcal{U}'(x^n)\backslash\mathcal{U}(x^n)$.

6.2 Simulation of individual sequences

The partition of $\mathcal{A}^n$ that we have presented has applications in the problem of simulation of individual sequences [67]. A *simulator*, or a *simulation scheme* for individual sequences is defined in [67] as a function, $\mathcal{S}$, that maps sequences x^n, for all n, to random sequences $Y^n \in \mathcal{A}^n$. Thus, a simulation scheme takes an input, x^n, and emits a random *simulated sequence*, Y^n, which we would like to be statistically similar to x^n. Formally, the simulator $\mathcal{S}$ is said to be *faithful* [67], if for each positive integer j, for each $x^n \in \mathcal{A}^n$, and for every $\delta, \epsilon > 0$, we have

$$\text{Prob}\{d_j(x^n, Y^n) < \epsilon\} > 1 - \delta\,, \quad \text{for sufficiently large } n\,, \tag{6.12}$$

where $d_j(x^n, Y^n)$ denotes the variational distance (see, e.g., [12, Chapter 11]) between the empirical distributions of j-tuples corresponding to x^n and Y^n. Observe that, as noticed in [54] (in a stochastic setting), and also in [67], a faithful simulator can be obtained trivially by taking y^n equal to x^n. However, we will be interested in simulation schemes that satisfy the following additional condition.

P2 Given that (6.12) is satisfied, there is as much uncertainty as possible in the choice of y^n.

Property P2 is desirable, so as to make the simulated sequence look as "original" as possible. In [67] this is addressed by maximizing the entropy of the output Y^n, given the input x^n conditioned on the fact that (6.12) is satisfied.

Since P1, and thus (6.12), is satisfied by any pair of sequences in a universal tree type class by Theorem 6.1, a faithful simulator can be obtained by picking y^n at random from $\mathcal{U}(x^n)$. Since we are also interested in maximizing the uncertainty in the choice of y^n, we maximize the entropy of the output random sequence given x^n by drawing y^n with a uniform distribution on $\mathcal{U}(x^n)$. We summarize this simulation algorithm, called **`TreeSim`**, in Figure 6.1.

`TreeSim`

1. `Based on a training sequence` x^n`, estimate a context tree` $\hat{T} = \hat{T}(x^n)$
2. `Draw uniformly at random from` $\mathcal{U}(x^n)$`, i.e., from the subset of sequences of` $\mathcal{T}(\hat{T}, x^n)$ `for which the estimated context tree is also` $\hat{T}$

Figure 6.1: *Simulation algorithm based on universal tree type classes*

We next compare the performance of **`TreeSim`** against other simulation schemes. Let a *weakly faithful* simulator be one that satisfies the following property: for any fixed k, the k-th order empirical entropy rate of the training sequence x^n and the output sequence y^n satisfy $\hat{\mathcal{H}}_k(y^n) < \hat{\mathcal{H}}_k(x^n) + \gamma_k(n)$, where each $\gamma_k(n) = o(1)$ (not necessarily uniformly in k). Notice that this criterion does not impose any statistical similarity between the original sequence and the simulated sequence. Also notice that any simulator that satisfies P1, like **`TreeSim`** by Theorem 6.1, is a weakly faithful simulator. Let $M_n(x^n)$ be a partition of the sequence space

$\mathcal{A}^n$ into N_n classes that yield a weakly faithful simulator and such that $(\log N_n)/n = o(1)$. The following theorem establishes the asymptotic optimality of the simulator induced by the partitions $\{M_n\}$. As in the analogous result in [67], the theorem relies strongly on the sample converse to the Source Coding Theorem [4, Theorem 3.1][36], from which it also inherits an "exception set".

6.3. THEOREM. *For any stationary ergodic measure, μ, and any weakly faithful simulator,*

$$\limsup_{n\to\infty} \frac{H(Y^n|x^n) - \log|M_n(x^n)|}{n} \le 0$$

for all x, except possibly in a set of vanishing μ-volume.

Proof. Consider the probability assignment $Q(x^n) = (N_n|M_n(x^n)|)^{-1}$. By the sample converse to the Source Coding Theorem [4, Theorem 3.1], $\liminf \frac{-\log Q(X^n)}{n} \ge \mathcal{H}$ *a.s.* where $\mathcal{H}$ is the entropy rate of the given measure μ. Since $\frac{\log N_n}{n} = o(1)$, we then have

$$\liminf_{n\to\infty} \frac{\log|M_n(x^n)|}{n} \ge \mathcal{H}$$

in a set of μ-volume one.

Let $N_{n,k}$ denote the number of Markov type classes of a fixed order k, and let $\mathcal{T}_{n,k}$ denote a Markov type class of order k. For every sequence y^n, the k-th order empirical probability of y^n is $\hat{\mathrm{P}}_{n,k}(y^n) = 2^{-n\hat{\mathcal{H}}_k(y^n)}$. Thus, we have

$$\sum_{y^n} 2^{-n\hat{\mathcal{H}}_k(y^n)} = \sum_{\mathcal{T}_{n,k}} \sum_{y^n\in\mathcal{T}_{n,k}} \hat{\mathrm{P}}_{n,k}(y^n)\,. \tag{6.13}$$

Since $\hat{\mathrm{P}}_{n,k}(y^n)$ is constant within each type class, we define $\hat{\mathrm{P}}_{n,k}(\mathcal{T}_{n,k}) = |\mathcal{T}_{n,k}|\hat{\mathrm{P}}_{n,k}(y^n)$, where y^n is any sequence in $\mathcal{T}_{n,k}$. We then get, from (6.13),

$$\sum_{y^n} 2^{-n\hat{\mathcal{H}}_k(y^n)} = \sum_{\mathcal{T}_{n,k}} \hat{\mathrm{P}}_{n,k}(\mathcal{T}_{n,k}) \le N_{n,k}\,. \tag{6.14}$$

Now, for a given sequence x^n, denote by $S_n(x^n)$ the set of sequences that have positive probability of being emitted by a weakly faithful simulator S, on input x^n, and let $N_{n,S}(x^n) = |S_n(x^n)|$. Then, we have

$$N_{n,k} \ge \sum_{y^n\in S_n(x^n)} 2^{-n\hat{\mathcal{H}}_k(y^n)} \ge N_{n,S}(x^n)2^{-n(\hat{\mathcal{H}}_k(x^n)+\gamma_k(n))}\,, \tag{6.15}$$

where the first inequality follows from (6.14) and the second inequality follows from the fact that S is weakly faithful. Therefore, since $H(Y^n|x^n)$ has a maximum equal to $\log N_{n,S}$ achieved by the uniform distribution over $S_n(x^n)$, we get

$$\frac{1}{n}H(Y^n|x^n) \le \frac{1}{n}\log N_{n,S}(x^n) \le \hat{\mathcal{H}}_k(x^n) + \gamma_k(n) + \frac{\log N_{n,k}}{n}\,,$$

where the last inequality follows from (6.15). Since $N_{n,k}$ grows polynomially with n and $\gamma_k(n) = o(1)$ for S weakly faithful, this implies that, for any fixed k,

$$\limsup \left(\frac{1}{n} H(Y^n|x^n) - \hat{\mathcal{H}}_k(x^n) \right) \leq 0 \,.$$

The theorem then follows from the fact that $\lim_k \lim_n \hat{\mathcal{H}}_k(x^n) = \mathcal{H}$ *a.s.* [1]. □

Theorem 6.3 states that in order for a partition-based weakly faithful simulator to achieve an asymptotic optimal output entropy it is sufficient to prevent the number of equivalence classes, N_n, from growing too fast with n. Both the original LZ-based universal type classes of [67] and the universal Markov type classes of [47] satisfy this condition. We next show that, with an appropriate choice of $f(n)$, universal tree type classes also satisfy it and, thus, **`TreeSim`** exhibits an asymptotic optimal output entropy when compared to all weakly faithful simulators (including all faithful simulators).

6.4. Lemma. *The number N_n of universal tree type classes over $\mathcal{A}^n$ satisfy $(\log N_n)/n = o(1)$, as long as $\frac{\log n}{f(n)} = o(1)$.*

Proof. By using a natural code [56, 88], a full tree can be described with as many bits as the number of nodes of the tree, which for a context tree with k states (leaves) is $\frac{\alpha k - 1}{\alpha - 1} \leq 2k$. Thus, the number T_k of different context trees with k states is upper-bounded by 2^{2k}. Let w_n be the maximum number of states of an estimate $\hat{T}(x^n)$ among all sequences $x^n \in \mathcal{A}^n$. For each context tree with k states there are at most $n^{\alpha k}$ type classes, since we have α symbol occurrence counts in each of the k states. Hence, we have

$$N_n \leq \sum_{k=1}^{w_n} 2^{2k} n^{\alpha k} \leq w_n 2^{2w_n} n^{\alpha w_n} \,.$$

Thus, we get

$$\frac{\log N_n}{n} \leq \frac{\log w_n}{n} + 2\frac{w_n}{n} + \frac{\alpha w_n \log n}{n} \,,$$

which tends to zero as long as $\frac{\log n}{f(n)} = o(1)$, since $w_n = O(n/f(n))$ by Lemma 5.19.

□

A comment is in order with respect to the implementation of this simulation algorithm. While selecting a sequence at random from an LZ-based universal type class can be implemented in a rather efficient manner [67], an enumeration of $\mathcal{U}(x^n)$ that would efficiently solve Step 2 of **`TreeSim`** seems a challenging problem. We may, however, circumvent the problem by selecting a candidate string y^n uniformly at random from $\mathcal{T}(\hat{T}(x^n), x^n)$ as described in Chapter 4, checking whether $\hat{T}(y^n) = \hat{T}(x^n)$, and repeating until a sequence from $\mathcal{U}(x^n)$ is drawn. This strategy succeeds in a constant expected number of steps (with high probability, only one step) under any finite order Markov measure for x^n, since a minimal context tree representation of the Markov measure is recovered with probability one for a suitable context tree estimator [17]. In the case of universal Markov type classes, it can be shown [47]

that, with an appropriate Markov order estimator, $\hat{k}(x^n)$, all but a negligible fraction of the sequences in the Markov type class of order $\hat{k}(x^n)$ of x^n estimate the same Markov order as x^n. For universal tree type classes, the problem remains open.

Chapter 7

Conclusions and directions for further research

We studied information-theoretic properties of tree models, and algorithms related to them. We first noticed that the statistics needed for the estimation of a context tree (e.g. in Context data compression algorithm, or in the applications studied in chapters 5 and 6), can be efficiently collected by constructing a compact suffix tree of the input sequence, which is not full in general. This observation led to the GCTM extension, which we took advantage of as an algorithmic tool. The GCTM formalism, however, also offers appealing parameter economization features when compared to classical tree models. The development of practical applications that exploit this potential reduction in the dimension of the model parameter still presents significant computational complexity challenges. Specifically, the problem of finding efficient algorithms for estimating an optimal GCT for a given sequence, remains an open one, and a promising direction for further research.

The apparent algorithmic drawback of context trees stemming from the lack of a next-state function, was overcome by means of the FSM closure formalism, which we discussed in Chapter 2. We characterized the FSM closure of a GCT theoretically, and also investigated efficient algorithms for constructing it. We exploited the construction in the implementation of **`SPContextFSM`**, the first algorithm for linear-time encoding/decoding of the semi-predictive version of Context, a twice-universal code in the class of tree models. In this application we were able to combine tree models and FSM models enjoying from both the potential for parameter space dimension reduction of tree sources but also from the computationally complexity advantages of FSMs. Beyond providing an efficient mechanism for resolving context transitions, the FSM closure turns out to be a valuable theoretical tool, providing a nexus between tree models and FSM models, which are, in general, easier to analyze than tree models. For example, in Chapter 5 we were able to derive a bound on the expected logarithm of the size of a context tree type class, based on a known analogous result for FSMs.

The general scheme that we used in the definition of **`SPContextFSM`** can be followed to implement a wide range of two-pass encoders obtained by varying the sequential coding stage and the model optimization function. For instance, one can replace the Krichevskii-Trofimov estimator, which, in practice, may not be the best choice for large alphabets (see, e.g., [78]), and adjust the pruning phase to optimize the resulting cost function. One may even try to heuristically estimate a GCT during the pruning phase, which, although possibly not the best in its class, could still improve the performance achieved by the best full context tree. In all these cases, the construction of the FSM closure of the selected model would provide, as in **`SPContextFSM`**, an efficient state transition mechanism. These ideas could be taken as the basis for future experimental research.

Another direction for future work may be related to efficiency improvements on the model

optimization step that takes place at the encoder side of **SPContextFSM**. Even though the construction of the compact suffix tree of a sequence requires linear time and memory usage, the needs for memory may become prohibitive in practice for very large sequences. Besides the obvious alternatives of splitting the input into chunks or limiting the model size, both of which invalidate the universality claim for the original algorithm, one can explore alternative suffix tree constructions methods that facilitate an efficient memory administration, e.g., [77]. In practice, for large sequences, these algorithms may outperform the classical linear-time suffix tree construction techniques surveyed in [32], even though their time complexity is not necessarily linear. Again, in any case, the FSM closure provides an efficient state transition mechanism for the sequential encoding phase. We point out that the decoder side requires, in general, less resources than the encoder, as the search for the optimal model is performed only by the latter. This observation makes the semi-predictive approach especially attractive in situations where the computational requirements are asymmetric and a low complexity decoder is needed.

In Chapter 4, we solved the hitherto open problem of the characterization of type classes of tree models. We studied both the size of the type class of a given string and the number of type classes for sequences of length n induced by a given context tree. We derived an exact formula for the size of the type class of a string, which generalizes Whittle's formula for FSMs, and we analyzed this formula to characterize the asymptotic behavior of the expected size of the type class of a random sequence emitted by a tree source. As in the re-derivation of Whittle's formula in [34], we reduced the problem to counting Eulerian unlabeled paths in a graph. For tree models however, the derivation is far more involved, due to the lack of a next-state function. Moreover, it is interesting to note that the asymptotic behavior of the size of the type class as $\exp(n\hat{\mathcal{H}}(x^n))$, which is valid for common models such as FSM, Markov, and of course memoryless, does not extend to tree models. It is not clear whether a similar elementary asymptotic expression for individual sequences exists for tree models, and we leave it as an open question. Although we focused our study of type classes on full tree models, the application of similar tools to other hierarchical aggregation of Markov models, such as GCTs, seems plausible for future research.

By establishing a one-to-one correspondence between strings in a type class, and Eulerian unlabeled paths in a graph, we automatically obtained an efficient enumeration algorithm for the strings of a type class, which in turn yields applications in data compression and simulation. These applications however, are not immediate. In the case of data compression, a classical enumerative source code in which both type classes and sequences within each type class are encoded uniformly turns out to be suboptimal for tree models, in opposition to FSM models. Instead, we developed in Chapter 5 a non uniform encoding of type classes that yields a universal enumerative code with optimal convergence rate in the class of tree sources. The developed tools show an interesting trade-off between the cost of encoding the type class with respect to a richer model, and the gain in code length that stems from the reduction of the size of the type classes. By carefully encoding type classes, we saw that we could refine the model structure used to partition the universe of sequences, while preserving optimal code length with respect to the original model.

In the case of simulation, the enumeration algorithm presented in Chapter 4 allows for the application of the scheme in [54], where the output sequence is selected uniformly at random from the type class of a training sequence x^n. Under the strong law preservation properties required in [54], this simulation algorithm is optimal, in the sense of minimizing the mutual information between the training data and the output sequence. We also defined and investigated the universal tree type class of a sequence, which parallels conventional type classes, in the same spirit as [67]. The partition of $\mathcal{A}^n$ into universal tree type classes results in a simulation scheme for individual sequences analogous to the one presented in [67] and also the one in [47]. This scheme is optimal in a well defined mathematical sense, and, compared to universal Markov type classes [47], it exhibits an improvement with respect to the convergence rate of the simulated sequence statistics as a function of the order. In [47], it is shown that the same algorithm based on universal Markov type classes for the simulation of individual sequences can be regarded as a *twice-universal simulator* in the class of Markov sources, in the sense that it achieves asymptotically the same optimal performance (in the sense of [54]) as a simulator that knows the true order of the source. An analogue result for tree sources remains open and provides another direction for future investigation.

Appendix A

Minimality conditions for generalized context tree models

Clearly, a model is minimal if and only if it remains so after normalization. Thus, to characterize minimality, we can assume without loss of generality that T is normal. We say that a pseudo-leaf $v \in T$ is a *pseudo-child* of $u \in T$ if $v \in \text{CHLD}_T(u)$, or v is a leaf of the form $v'b$, where $b \in \mathcal{A}$, $v' \in \text{CHLD}_T(u)$, and $\alpha = 2$. By the discussion on normalization in Section 2.2, the case in which $v \notin \text{CHLD}_T(u)$ corresponds to the only case in which a node can be eliminated from a normal GCT in an unnormalization step making its parent a pseudo-leaf. If v or v' are atomic children of u, then v is called an atomic pseudo-child of u. Let $L_T(u)$ denote the set of atomic pseudo-children of u, and let

$$\nu_T(u) = \max\{\,1,\ |\{\,a \in \mathcal{A} \,|\, ua \notin T\,\}|\,\}\,.$$

The following theorem characterizes minimal normal GCTMs.

A.1. THEOREM. *A normal GCTM $\langle T, p\rangle$ with $\mathcal{A}^* = \mathcal{A}_P^*$ is minimal if and only if every node $u \in T$ satisfies the following conditions:*

(i) *Any subset of nodes in $L_T(u)$ that share a common CPMF is of size $\nu_T(u)$ or less.*

(ii) *If u is a permanent state, and v a pseudo-child of u, then $p(\cdot\,|\,u) \neq p(\cdot\,|\,v)$.*

In addition, for $\alpha = 3$, if $\langle T, p\rangle$ is minimal and $\langle T', p'\rangle$ generates the same process with T' normal, then T' is an extension of T.

It can be shown that if T is not normal, the conditions of Theorem A.1 hold on T_N if and only if they hold on T; therefore, the step of normalizing T can be avoided. The theorem implies that the minimal normal GCTM is unique for $\alpha = 3$. However, it might not be so for $\alpha \neq 3$. For example, let $\alpha = 4$, denote $\mathcal{A} = \{a_i\}_{i=1}^4$, and consider the normal GCTs $T = \{\lambda, a_1, a_2\}$ and $T' = \{\lambda, a_3, a_4\}$. Clearly, if $p(\cdot\,|\,a_1) = p(\cdot\,|\,a_2) = p'(\cdot\,|\,\lambda)$ (as functions), $p'(\cdot\,|\,a_3) = p'(\cdot\,|\,a_4) = p(\cdot\,|\,\lambda) \neq p'(\cdot\,|\,\lambda)$, and $p(\cdot\,|\,\lambda\$) = p'(\cdot\,|\,\lambda\$)$, $\langle T, p\rangle$ and $\langle T', p'\rangle$ generate the same process and are both minimal. This example can be generalized to any $\alpha > 4$ in an obvious manner (but not to $\alpha < 4$, since T and T' must be normal). For $\alpha = 2$, let $\mathcal{A} = \{0, 1\}$, $T = \{\lambda, 01, 10\}$, $T' = \{\lambda, 00, 11\}$, $p(\cdot\,|\,01) = p(\cdot\,|\,10) = p'(\cdot\,|\,\lambda)$, and $p'(\cdot\,|\,00) = p'(\cdot\,|\,11) = p(\cdot\,|\,\lambda) \neq p'(\cdot\,|\,\lambda)$. Again, assuming that the two models use identical CPMFs for the transient states $\lambda\$$, $0\$$, and $1\$$, both $\langle T, p\rangle$ and $\langle T', p'\rangle$ are minimal GCT models that generate the same process.

Proof. First, we show the necessity of the conditions. If u does not satisfy condition (i) we can modify the model, without affecting the process, as follows. If $u \in S_T$, add new

pseudo-leaves ua for all $a \in \mathcal{A}$ such that $ua \notin T$, and associate them with the CPMF of u. By Lemma 2.1, u ceases to be a permanent state. By our assumption, there are $\nu' > \nu_T(u)$ atomic pseudo-children of u that share the same CPMF. These nodes can be eliminated so that the strings they accepted are now accepted by u, which also inherits their common CPMF (for $\alpha = 2$, the elimination of an atomic pseudo-child $v \notin \textsc{chld}_T(u)$ also causes the elimination of $\textsc{par}_T(v)$, which is not a permanent state as T is normal). Overall, the number of permanent states in T decreases by at least $\nu' - \nu_T(u) > 0$. If $u \in S_T$ and does not satisfy condition (ii), then it has a pseudo-child v with the same CPMF, which can be eliminated without affecting the process, decreasing the number of permanent states of T (again, for $\alpha = 2$, the elimination of v may also imply the elimination of $\textsc{par}_T(v)$). In both cases, not satisfying the condition implies that T is not minimal. Notice that, since the CPMFs of the transient states are assumed to satisfy the constraint (2.3), the above state eliminations are not impeded by the transient CPMF $p(\cdot\,|\,u\$)$.

Next, we prove the sufficiency of the conditions. Consider two normal GCTMs $\langle T, p\rangle$ and $\langle T', p'\rangle$ that generate the same process P, with $\mathcal{A}^* = \mathcal{A}_P^*$, and assume that $\langle T, p\rangle$ satisfies conditions (i) and (ii). We first state two general properties that will be used in the proof.

(P1) If $u \in \mathcal{A}^*$ is such that $u \notin \textsc{word}(T')$, then $V_T(u) \in S_T$, $V_{T'}(u) \in S_{T'}$, and $p(\cdot\,|\,V_T(u)) = p'(\cdot\,|\,V_{T'}(u))$ (as functions). Moreover, if $u \in T$, then u is a leaf of T.

(P2) If $u \in T \setminus T'$, then either u is a pseudo-leaf of T, or $\alpha = 2$ and there exists $b \in \mathcal{A}$ such that ub is a leaf of T. In addition, denoting by v the claimed pseudo-child of $\textsc{par}_T(u)$ (either u or ub), we have $p(\cdot\,|\,v) = p'(\cdot\,|\,V_{T'}(u))$ (as functions).

For $u \notin T$, **(P1)** is an obvious consequence of the existence of $b \in \mathcal{A}$ such that $ub \notin \textsc{word}(T)$, of Lemma 2.1, and of $P(\cdot|z\overline{u})$ being independent of $z \in \mathcal{A}^*$ (as the two processes are identical and all strings have nonzero probability); conditions (i) and (ii) on $\langle T, p\rangle$ are not required. The case $u \in T$ and the second part of **(P1)** follow from the fact that any nontrivial subtree of T must contain permanent states with at least two different CPMFs, for otherwise the subtree would contain a node all of whose children are leaves and that violates either condition (i) or condition (ii).

To prove **(P2)**, observe first that since $u \notin T'$ the set $A_u = \{a \in \mathcal{A} : ua \notin \textsc{word}(T')\}$ contains at least $\alpha - 1$ symbols. Thus, since T is normal, if $u \in S_T$ then there exists a symbol $a' \in A_u$ such that $ua' \notin T$, and therefore, by **(P1)**, $p(\cdot\,|\,u) = p'(\cdot\,|\,V_{T'}(u))$. Now, consider the strings $ub \in \textsc{word}(T)$, $b \in A_u$, and let $uby \in \textsc{chld}_T(u)$, $y \in \mathcal{A}^*$. By **(P1)**, uby must be a leaf of T. If u is not a pseudo-leaf of T, then there exists at least one such leaf uby and, moreover, uc must be a leaf of T for *every* $c \in A_u$, for otherwise $u \in S_T$ and it would have the same CPMF $p'(\cdot\,|\,V_{T'}(u))$ as uby, violating condition (ii). This case can only occur for $\alpha = 2$, for otherwise the number of leaves sharing the same CPMF would be $\alpha - 1 > 1 = \nu_T(u)$, violating condition (i). The proof of **(P2)** is complete.

Now, to prove the sufficiency of conditions (i) and (ii), we will show that if a GCTM $\langle T', p'\rangle$ generates the same process as $\langle T, p\rangle$ and it also satisfies the conditions, then T and T' are identical up to transformations of $\langle T', p'\rangle$ that do not affect neither $|S_{T'}|$ nor the generated process. Without loss of generality, we can assume that T' is also normal. Clearly,

it suffices to prove that if $u \in T \cap T'$, then after such transformations, u is unaffected and $\text{CHLD}_T(u) = \text{CHLD}_{T'}(u)$.

The claim is obvious for $u \notin S_T \cup S_{T'}$, as u has a full complement of atomic children in both GCTs. Next, we show that if $u \in S_T$, then $u \in S_{T'}$ or, equivalently, that $u \in S_T \setminus S_{T'}$ implies $u \notin T'$. If the claim did not hold, we would have $uc \in T'$ for all $c \in \mathcal{A}$. Since T is normal and $u \in S_T$, there exist $a, a' \in \mathcal{A}$, $a \neq a'$, such that $ua, ua' \notin T$. Thus, by **(P2)**, T' has pseudo-children in the directions of a and a' sharing the same CPMF. Since $\nu_{T'}(u) = 1$, T' does not satisfy condition (i), a contradiction.

Consequently, it suffices to consider the case $u \in S_T \cap S_{T'}$. Assume first $\alpha > 2$. If $ua \in T \setminus T'$ for some $a \in \mathcal{A}$, then, by **(P2)**, ua is a pseudo-leaf of T and $p'(\cdot \,|\, u) = p(\cdot \,|\, ua)$. Therefore, if $p(\cdot \,|\, u) = p'(\cdot \,|\, u)$, the sets of atomic children of u for T and T' coincide for otherwise condition (ii) is violated. If $p(\cdot \,|\, u) \neq p'(\cdot \,|\, u)$, we must have $ua \in T \cup T'$ for all $a \in \mathcal{A}$, for otherwise there exists $b \in \mathcal{A}$ such that $uab \notin \text{WORD}(T')$ and $uab \notin T$, and a contradiction to **(P1)** follows. Moreover, in order for both T and T' to satisfy condition (i), the sets $\{ua \in T \setminus T'\}$ and $\{ua \in T' \setminus T\}$ must have the same size. Therefore, by deleting from T' all nodes in the latter set (which are pseudo-leaves) and adding the nodes in the former set, the size of $S_{T'}$ remains unchanged, while the process $\langle T', p' \rangle$ is preserved by replacing the distribution $p'(\cdot \,|\, u)$ with $p(\cdot \,|\, u)$ and associating $p'(\cdot \,|\, u)$ with the added pseudo-leaves (this operation does not affect the transient states). After this transformation, again, the sets of atomic children of T and T' coincide, and $p(\cdot \,|\, u) = p'(\cdot \,|\, u)$.

For $\alpha = 2$, the two sets of atomic children are empty by normality. We show that we can also assume $p(\cdot \,|\, u) = p'(\cdot \,|\, u)$. For all $c \in \mathcal{A}$, there exists $b \in \mathcal{A}$ such that $ucb \notin \text{WORD}(T')$. If $p(\cdot \,|\, u) \neq p'(\cdot \,|\, u)$, then ucb is a leaf of T, for otherwise a contradiction to **(P1)** would follow. We can assume, without loss of generality, that $c = b$. Thus, letting $\mathcal{A} = \{a, a'\}$, T has leaves uaa and $ua'a'$, with CPMF $p'(\cdot \,|\, u)$. Similarly, T' has leaves uaa' and $ua'a$, with CPMF $p(\cdot \,|\, u)$. Therefore, we can replace the subtree of T' rooted at u with the corresponding subtree of T (replacing also the associated CPMFs), without affecting the size of $S_{T'}$ or the generated process.

To complete the sufficiency proof, it remains to show that, for any α, if $uay \in \text{CHLD}_T(u)$ and $uaz \in \text{CHLD}_{T'}(u)$ for some $a \in \mathcal{A}$, $y, z \in \mathcal{A}^+$, and $p(\cdot \,|\, u) = p'(\cdot \,|\, u)$, then $y = z$. Suppose it is not. Then, either $V_{T'}(uay) = u$ or $V_T(uaz) = u$. Assume, without loss of generality, that the former holds. Then, by **(P2)**, u has a pseudo-child v with $p(\cdot \,|\, v) = p'(\cdot \,|\, u) = p(\cdot \,|\, u)$, violating condition (ii).

Finally, assume that $\langle T', p' \rangle$ generates the same process as $\langle T, p \rangle$, with T' normal, $\langle T, p \rangle$ minimal, and $\alpha = 3$. We prove that $T \subseteq T'$. Since $T \subseteq T_N$, we can assume, without loss of generality, that T is normal. Suppose, to the contrary, that $w \in T \setminus T'$. Clearly, there exists $v \preceq w$ such that $v \in T \setminus T'$ and $\text{PAR}_T(v) \preceq V_{T'}(v) \stackrel{\triangle}{=} u$. Let $v = uax$, with $a \in \mathcal{A}$ and $x \in \mathcal{A}^*$. By **(P2)**, v is a pseudo-leaf of T and $p(\cdot \,|\, v) = p'(\cdot \,|\, u)$. Since T' is normal, there exists $b \in \mathcal{A} \setminus \{a\}$ such that $ub \notin T'$. Moreover, for any $a' \in \{a, b\}$ we must have $ua' \in T$, for otherwise $V_T(u) \in S_T$ and $ua'c \notin T \cup \text{WORD}(T')$ for some $c \in \mathcal{A}$, implying, by **(P1)**, $p(\cdot \,|\, V_T(u)) = p'(\cdot \,|\, u) = p(\cdot \,|\, v)$, violating condition (ii) for T. Thus, $u \in T$ and $v = ua$. Again by **(P2)**, ub is a pseudo-leaf of T with $p'(\cdot \,|\, u) = p(\cdot \,|\, ub)$. It follows that $\{ua, ub\}$ is a subset

of pseudo-leaves in $L_T(u)$ that share the same CPMF, violating condition (i) for T since $\nu_T(u) = 1$ as $\alpha = 3$. □

Appendix B

Proof of Theorem 2.9

We claim that the total number of comparisons made during computations of $C_T(x)$ is upper-bounded by $2N_E + N'$. By the discussion preceding the theorem, this fact establishes the claimed upper bound, as the other operations take constant time per node of T_{suf}.

Let $T_0 = T$, and, for $i > 0$, let T_i be a snapshot of T' after the i-th computation of $C_T(x)$ in Step 2 of **`Verify`**, and the corresponding call to **`Insert`** in Step 4, if such a call was made. Let C_i be the total number of comparisons made in computations of $C_T(\cdot)$ up to that point, and C the total number after completion of the algorithm, when $T' = T_{\text{suf}}$. Denote by T_i^* the set of nodes of T_i excluding the root, by T_i^- the subset of those nodes that have not been visited at the time of the snapshot, and by $T_{L,i}^-$ the subset of nodes in T_i^- that are leaves of T_i. We construct two sequences of functions $f_i : T_i^* \to \mathbb{Z}$ and $g_i : T_i^* \to \mathbb{Z}$, $i \geq 0$, such that

$$f_0(uv) = 2|v|, \quad uv \in T_0^*, \;\; u = \text{PAR}_{T_0^*}(uv), \;\; u \xrightarrow{v} uv,$$

$g_0 \equiv 0$, and, for $i > 0$, and each node $uav \in T_i^*$, with $u = \text{PAR}_{T_i^*}(uav)$,

$$f_i(uav) = \begin{cases} 2|av| & uav \in T_{L,i}^- \text{ or } \texttt{Traversed[}u, a\texttt{]=}\mathit{false}, \\ 0 & \text{otherwise}, \end{cases}$$

and

$$g_i(uav) = \begin{cases} |av| & uav \in T_i^- \setminus T_{L,i}^- \text{ and } \texttt{Traversed[}u, a\texttt{]=}\mathit{true}, \\ 0 & \text{otherwise}. \end{cases}$$

Notice that the condition for $g_i(uav) \neq 0$ can only hold when uav was just created as a result of an edge split in Step 5 of **`Insert`**. We prove, by induction on i, that the following condition holds for all $i \geq 0$ for which the relevant quantities are defined:

$$C_i \leq |T_i^*| + 2N_E - \sum_{t \in T_i^*} f_i(t) - \sum_{t \in T_i^*} g_i(t). \tag{B.1}$$

Condition (B.1) implies that at the end of the execution of the algorithm, after all nodes have been visited, we have $C \leq N' + 2N_E$, as claimed. The inequality is clearly satisfied for $i = 0$. Assume now it is satisfied for all $i \leq n - 1$. We determine f_n and g_n after the next execution of Step 2 and any necessary insertions, assuming **`Verify`** was called with argument $w = cx$, and $C_T(x) = \langle r, u, v \rangle$. Since cx is being visited, we have

$$f_n(cx) = g_n(cx) = 0. \tag{B.2}$$

If $u \neq \lambda$, let $u = au'$, $a \in \mathcal{A}$. In this case, **Insert** was called, and an internal node was created by splitting an edge $r \xrightarrow{uy} ruy$ into $r \xrightarrow{u} ru \xrightarrow{y} ruy$. Hence, we have

$$f_n(ru) = \begin{cases} 2|u| & \texttt{Traversed}[r, a] = \mathit{false}, \\ 0 & \text{otherwise}, \end{cases} \tag{B.3}$$

$$f_n(ruy) = \begin{cases} 2|y| & \texttt{Traversed}[r, a] = \mathit{false}, \\ 0 & \text{otherwise}, \end{cases} \tag{B.4}$$

and

$$g_n(ru) = \begin{cases} |u| & \texttt{Traversed}[r, a] = \mathit{true}, \\ 0 & \text{otherwise}. \end{cases} \tag{B.5}$$

For node ruy, notice that if $\texttt{Traversed}[r, a] = \mathit{true}$, ruy had been visited either immediately after creation from Step 7 or 9, or after setting $\texttt{Traversed}(r, a) = \mathit{true}$ from Step 16. Therefore, we have

$$g_n(ruy) = 0. \tag{B.6}$$

Notice that from (B.3), (B.4), and the fact that edges resulting from a split inherit the "traversed" status of the original edge, it follows that

$$f_n(ru) + f_n(ruy) - f_{n-1}(ruy) = 0. \tag{B.7}$$

If $v \neq \lambda$, a new leaf node ruv was created, and has not yet been visited, i.e. $ruv \in T^-_{L,i}$. Thus, we have

$$f_n(ruv) = 2|v|, \tag{B.8}$$

and

$$g_n(ruv) = 0. \tag{B.9}$$

All other values of f_n and g_n are unchanged from their values at $i = n - 1$. We prove that the condition in (B.1) holds for $i = n$. Assume first that the invocation of **Verify** at which snapshot n was taken was made recursively from Step 7. It follows from the discussion preceding the theorem that in this case, $C_{T'}$ was computed in *fast* mode, since we know that x was a word of T_{n-1}. This also implies that $v = \lambda$, and ru was the only node possibly created (if any). Let $\hat{c}\hat{x}$ be the argument of the call to **Verify** that caused the recursive call in Step 7 with argument cx, and let $\langle \hat{r}, \hat{u}, \hat{v} \rangle = C_{T'}(\hat{x})$, i.e., $cx = \hat{r}\hat{u}$. The search in fast mode for $x = \text{tail}(\hat{r}\hat{u})$ in T_n begins at $\text{tail}(\hat{r})$ and requires as many comparisons as nodes in the path from $\text{tail}(\hat{r})$ to the node immediately before exhausting the $|\hat{u}|$ remaining symbols. Thus, the number of comparisons needed in the fast computation of $C_{T'}(x)$ is upper bounded by $|\hat{u}| - |u| + 1$ if $u \neq \lambda$, or $|\hat{u}|$ otherwise. Thus, we can write

$$C_n - C_{n-1} \leq |\hat{u}| - |u| + |T^*_n| - |T^*_{n-1}|. \tag{B.10}$$

Also, we have $f_{n-1}(cx) = 0$, since we call **Verify** from Step 7 only for nodes that are not leaves, and whose incoming edge has **Traversed** $= \mathit{true}$. Therefore, combining with (B.2), and (B.7), we get

$$\sum_{t \in T^*_n} f_n(t) - \sum_{t \in T^*_{n-1}} f_{n-1}(t) = 0. \tag{B.11}$$

As for the functions g, we have $g_n(cx) = 0$ by (B.2), and, since coming from Step 7, cx is a newly created internal node that did not exist at $i = n - 1$,

$$g_{n-1}(cx) = |\hat{u}|. \tag{B.12}$$

By (B.5),

$$g_n(ru) \leq |u|. \tag{B.13}$$

Also,

$$g_n(ruy) = 0, \tag{B.14}$$

and

$$g_{n-1}(ruy) = 0 \text{ whenever } ruy \neq cx. \tag{B.15}$$

The last two equations follow from the fact that if the incoming edge of ruy has **`Traversed`**$=$ *true*, the node had already been visited at $i = n - 1$. The only exception is the coincidental case where $ruy = cx$, the node being visited at $i = n$, in which case the node had not been visited at $i = n - 1$ and the value from (B.12) takes precedence. Other values of g_n remain unchanged from g_{n-1}, and, hence, it follows from (B.12)–(B.15) that

$$-\sum_{t \in T_n^*} g_n(t) + \sum_{t \in T_{n-1}^*} g_{n-1}(t) \geq |\hat{u}| - |u|. \tag{B.16}$$

Now, from (B.10),(B.11),(B.16), and the induction hypothesis we obtain (B.1) for $i = n$, as desired.

Assume now that the invocation of **`Verify`** at which snapshot n was taken was made recursively from Step 9. As before, we let $\hat{c}\hat{x}$ be the argument of the call to **`Verify`** that caused the recursive call in Step 9 with argument cx, and let $\langle \hat{r}, \hat{u}, \hat{v} \rangle = C_{T'}(\hat{x})$. We have $\hat{u} = \lambda$, $\hat{v} \neq \lambda$, and $cx = \hat{r}\hat{v}$. The search for $x = \text{tail}(\hat{r}\hat{v})$ in T_n begins at $\text{tail}(\hat{r})$ and requires as many symbol to symbol comparisons as needed to either "fall off" the tree, or reach $x = \text{tail}(\hat{r}\hat{v})$. Thus, the number of comparisons made in computing $C_{T'}(x)$ is $|\hat{v}| - |v| + 1$ when $v \neq \lambda$, or $|\hat{v}|$ otherwise. Hence,

$$C_n - C_{n-1} \leq |\hat{v}| - |v| + |T_n^*| - |T_{n-1}^*|. \tag{B.17}$$

Since $cx = \hat{r}\hat{v}$ is a leaf added to T_{n-1} just before calling **`Verify`** with argument cx, we have $f_{n-1}(cx) = 2|\hat{v}|$. Therefore, combining with (B.2), (B.7), and (B.8), we get

$$-\sum_{t \in T_n^*} f_n(t) + \sum_{t \in T_{n-1}^*} f_{n-1}(t) = 2|\hat{v}| - 2|v|. \tag{B.18}$$

Also, $g_n(cx) = 0$, and $g_{n-1}(cx) = 0$ for cx is a leaf. By (B.5),

$$g_n(ru) \leq |u|, \tag{B.19}$$

and by the same arguments as in the previous case,

$$g_n(ruy) = 0, \tag{B.20}$$

and

$$g_{n-1}(ruy) = 0. \tag{B.21}$$

Since ruv is a leaf in case $v \neq \lambda$, $g_n(ruv) = 0$, and, thus,

$$-\sum_{t\in T_n^*} g_n(t) + \sum_{t\in T_{n-1}^*} g_{n-1}(t) \geq -|u|. \tag{B.22}$$

From (B.18) and (B.22) we obtain

$$-\sum_{t\in T_n^*} f_n(t) + \sum_{t\in T_{n-1}^*} f_{n-1}(t) - \sum_{t\in T_n^*} g_n(t) + \sum_{t\in T_{n-1}^*} g_{n-1}(t) \geq 2|\hat{v}| - 2|v| - |u| \geq |\hat{v}| - |v|, \tag{B.23}$$

where the rightmost inequality follows from $|\hat{v}| \geq |u| + |v|$, which in turn follows from $ruv = x = \mathrm{tail}(\hat{r})\hat{v}$ and $\mathrm{tail}(\hat{r}) \preceq r$. Finally, combining with (B.17), and applying the induction hypothesis, we obtain (B.1) for $i = n$ also in this case.

It remains to consider the case where the invocation of **`Verify`** at which snapshot n was taken was made from Step 16. Let $\hat{w} = \hat{c}\hat{x}$ be the argument of the call to **`Verify`** that caused the recursive call in Step 16 with argument $cx = \hat{w}az$. The search for $x = \mathrm{tail}(\hat{w}az)$ in T_n begins at $\mathrm{tail}(\hat{w})$ and requires as many symbol to symbol comparisons as needed to either "fall off" the tree, or reach $x = \mathrm{tail}(\hat{w}az)$. Thus, the number of comparisons made in computing $C_{T'}(x)$ is $|az| - |v| + 1$ when $v \neq \lambda$, or $|az|$ otherwise. Hence,

$$C_n - C_{n-1} \leq |az| - |v| + |T_n^*| - |T_{n-1}^*|. \tag{B.24}$$

Since at snapshot $n-1$, **`Traversed[`**$\hat{w}, a$**`]`** $=$ *false*, we have $f_{n-1}(cx) = 2|az|$. Therefore, combining with (B.2), (B.7), and (B.8), we get

$$-\sum_{t\in T_n^*} f_n(t) + \sum_{t\in T_{n-1}^*} f_{n-1}(t) = 2|az| - 2|v|. \tag{B.25}$$

Also, $g_n(cx) = 0$ by (B.2), and $g_{n-1}(cx) = 0$ for **`Traversed[`**$\hat{w}, a$**`]`** $=$ *false*. By (B.5),

$$g_n(ru) \leq |u|, \tag{B.26}$$

and by the same arguments as in the previous cases,

$$g_n(ruy) = 0, \tag{B.27}$$

and

$$g_{n-1}(ruy) = 0. \tag{B.28}$$

Since ruv is a leaf in case $v \neq \lambda$, $g_n(ruv) = 0$, and, thus,

$$-\sum_{t\in T_n^*} g_n(t) + \sum_{t\in T_{n-1}^*} g_{n-1}(t) \geq -|u|. \tag{B.29}$$

From (B.25) and (B.29) we obtain

$$-\sum_{t\in T_n^*} f_n(t) + \sum_{t\in T_{n-1}^*} f_{n-1}(t) - \sum_{t\in T_n^*} g_n(t) + \sum_{t\in T_{n-1}^*} g_{n-1}(t) \geq 2|az| - 2|v| - |u| \geq |az| - |v|, \tag{B.30}$$

where the rightmost inequality follows from $|az| \geq |u| + |v|$, which in turn follows from $ruv = x = \mathrm{tail}(\hat{w})az$ and $\mathrm{tail}(\hat{w}) \preceq r$. Finally, combining with (B.24), and applying the induction hypothesis, we obtain the desired result. □

Appendix C

Linear-time decoding

In this appendix, we describe in full detail the two alternative decoding algorithms outlined in Chapter 3. Both algorithms yield linear-time decoding.

C.1 Decoding using incremental FSM closure construction

We now present a decoding algorithm that does not require additional complex data structures (as **Jump[**u**]** in Chapter 3). The implementation of the semi-predictive approach to Context algorithm derived from this decoding algorithm is denoted **SPContextFSMi**. The decoder starts by constructing $\hat{T}'(x^n)$ and $\hat{T}'_F(x^n)$ as before. However, $\hat{T}'_F(x^n)$ is not used statically as in the algorithm in Chapter 3. Instead, new permanent states of $T_F(x^n) \setminus \hat{T}'_F(x^n)$ are added to $\hat{T}'_F(x^n)$ as they are found while symbols of x^n are decoded. Formally, we recursively define the sequence of FSM GCTs $\tilde{T}'_{F_i}(x^n)$, for $0 \le i \le n$ by $\tilde{T}'_{F_0}(x^n) = \hat{T}'_F(x^n)$; $\tilde{T}'_{F_i}(x^n) = \tilde{T}'_{F_{i-1}}(x^n) \cup \{s_i\}$ for $0 < i \le n$ adding any necessary bifurcation and all nodes necessary to satisfy the *suffix property*, namely, every suffix of a node of $\tilde{T}'_{F_i}(x^n)$ is a node of $\tilde{T}'_{F_i}(x^n)$.

Let $\tilde{s}_0 \stackrel{\triangle}{=} \lambda$; $\tilde{s}_i \stackrel{\triangle}{=} \sigma_{\tilde{T}'_{F_{i-1}}(x^n)}(x^i)$ for $0 < i \le n$, and $\tilde{z}_i$ such that $\tilde{z}_0 = \lambda$; $\tilde{s}_i\tilde{z}_i$ is the longest prefix of $\overline{x^i}$ that is a word of $\tilde{T}'_{F_{i-1}}(x^n)$ for $0 < i \le n$. Also define $\tilde{b}_i \stackrel{\triangle}{=} x_{i-|\tilde{s}_i\tilde{z}_i|}$ in case $|\tilde{s}_i\tilde{z}_i| < i$ and $\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|} \in T_F(x^n)$, or $\tilde{b}_i = \lambda$ otherwise. The idea now is to use $\tilde{s}_i$, $\tilde{z}_i$, and $\tilde{b}_i$ to determine s_i for every i, $0 \le i \le n$. The connection is given by the following lemma for which we remove the \$ symbol from transient states.

C.1. Lemma. *For every i, $0 \le i < n$, we have $s_i = \tilde{s}_i\tilde{z}_i\tilde{b}_i$.*

Proof. If $\tilde{b}_i = \lambda$, either $\tilde{s}_i\tilde{z}_i$ is a leaf of $T_F(x^n)$ or $\tilde{s}_i\tilde{z}_i = \overline{x^i}$ and in any case $\tilde{s}_i\tilde{z}_i = s_i$. If $\tilde{b}_i \ne \lambda$, $\tilde{s}_i\tilde{z}_i \prec s_i$ by the definition of $\tilde{b}_i$, and since $s_i = \hat{s}_i z_i b_i$ by Lemma 3.3, we have $\tilde{s}_i\tilde{z}_i \prec \hat{s}_i z_i b_i$. Since $\tilde{T}'_{F_{i-1}}(x^n)$ is an extension of $\hat{T}'_F(x^n)$, also $\hat{s}_i z_i \preceq \tilde{s}_i\tilde{z}_i$. Hence, we have $\hat{s}_i z_i \preceq \tilde{s}_i\tilde{z}_i \prec \hat{s}_i z_i b_i$, and therefore $\tilde{s}_i\tilde{z}_i = \hat{s}_i z_i$. Thus, $\tilde{s}_i\tilde{z}_i\tilde{b}_i = \hat{s}_i z_i b_i$ and the claim follows from Lemma 3.3. □

We now describe the decoding stage conceptually, and postpone implementation details for a later discussion. After building $\tilde{T}'_{F_0}(x^n)$ and setting $s_0 = \lambda$, for each i, $0 < i \le n$ the algorithm decodes x_i using the statistics pointed by **Origin[**s_{i-1}**]** in $\tilde{T}'_{F_{i-1}}(x^n)$, determines s_i, and constructs constructs $\tilde{T}'_{F_i}(x^n)$ from $\tilde{T}'_{F_{i-1}}(x^n)$ by adding s_i and the necessary suffixes.

In order for the decoder to use the same statistics as the encoder, the algorithm must guarantee that **Origin[**uv**]**=**Origin[**uw**]** for all nodes uv and uw where u is a leaf of $T(x^n)$. To this end, for any leaf u of $T(x^n)$ and $uv \in \tilde{T}'_{F_i}(x^n)$, **Origin[**$uv$**]** will point to the node uw of $\tilde{T}'_{F_i}(x^n)$, where w is the shortest string such that $uw \in \tilde{T}'_{F_j}(x^n)$, and j is the minimum

Evolve $(\tilde{T}'_{F_{i-1}}(x^n), \tilde{s}_i, \mathtt{head}(\tilde{z}_i), |\tilde{z}_i|, x_{i-|\tilde{s}_i\tilde{z}_i|})$

1. Start from $\tilde{T}'_{F_i}(x^n) = \tilde{T}'_{F_{i-1}}(x^n)$
2. Set $s_i = \tilde{s}_i$, and $\tilde{b}_i = \lambda$
3. If $\tilde{z}_i \neq \lambda$
4. Add $\tilde{s}_i\tilde{z}_i$ to $\tilde{T}'_{F_i}(x^n)$, and set $s_i = \tilde{s}_i\tilde{z}_i$
5. Set Transitions$[\tilde{s}_i\tilde{z}_i]$=Transitions$[\tilde{s}_i]$
6. Verify*$(\tilde{s}_i\tilde{z}_i)$
7. If $|\tilde{s}_i\tilde{z}_i| < i$ and $\tilde{s}_i\tilde{z}_i$ is an internal node of $T_F(x^n)$
8. Set $\tilde{b}_i = x_{i-|\tilde{s}_i\tilde{z}_i|}$
9. Add $\tilde{s}_i\tilde{z}_i\tilde{b}_i$ to $\tilde{T}'_{F_i}(x^n)$, and set $s_i = \tilde{s}_i\tilde{z}_i\tilde{b}_i$
10. Set Transitions$[\tilde{s}_i\tilde{z}_i\tilde{b}_i]$=Transitions$[\tilde{s}_i\tilde{z}_i]$
11. Verify*$(\tilde{s}_i\tilde{z}_i\tilde{b}_i)$

Figure C.1: *Computation of* $\tilde{T}'_{F_i}(x^n)$

index in the range $0 \leq j \leq i$ such that $u \in \text{WORD}(\tilde{T}'_{F_j}(x^n))$. During the creation of $\tilde{T}'_{F_i}(x^n)$, $i > 0$, we make use of the routine **Verify***, equal to **Verify** from Chapter 2, except for the variations (to **Verify** itself, and to the auxiliary routine **Insert**) that we describe next.

For a node uv created as a leaf child of u (steps 2 and 9 of **Insert** in Figure 2.6), **Origin**$[uv]$ is set to uv if u is an internal node of $T(x^n)$, or to **Origin**$[u]$ otherwise. For a node uv created splitting an edge $u \xrightarrow{vx} uvx$ into $u \xrightarrow{v} uv \xrightarrow{x} uvx$ (Step 5 of **Insert**), **Origin**$[uv]$ is set to uv if uv is an internal node of $T(x^n)$ or to **Origin**$[uvx]$ otherwise. These modifications implement the desired behavior for the pointers **Origin**$[u]$, which guarantee a correct determination of the decoding statistics.

When a new node u is inserted in $\tilde{T}'_{F_i}(x^n)$, it is also necessary to define new outgoing FSM transitions for u, and update those FSM transitions of $\tilde{T}'_{F_{i-1}}(x^n)$ that must now lead to u. To achieve this, **Verify***(w) sets $f(x', c) = w$ in **Transitions**$[x']$, where $c = \text{head}(w)$, not only for node $x' = \text{tail}(w)$ (in Step 11 of **Verify**), but also to all its descendants $x'y$ that shared the same next-state for symbol c in $\tilde{T}'_{F_{i-1}}(x^n)$, i.e., $f(x'y, c) = f(x', c)$. In addition, if node x' is created as a new child of z, it sets $f(x', c') = f(z, c')$ in **Transitions**$[x']$ for all $c' \in \mathcal{A}$, $c' \neq c$.

Using **Verify***, and assuming that $\tilde{s}_i$ and $|\tilde{z}_i|$ have been determined, the routine **Evolve** of Figure C.1 evolves a tree from $\tilde{T}'_{F_{i-1}}(x^n)$ to $\tilde{T}'_{F_i}(x^n)$, and determines the symbol $\tilde{b}_i$, and the state s_i. The construction of $\tilde{T}'_{F_i}(x^n)$ starts from $\tilde{T}'_{F_{i-1}}(x^n)$. If $\tilde{z}_i \neq \lambda$, a node $\tilde{s}_i\tilde{z}_i$ is added to $\tilde{T}'_{F_i}(x^n)$. As $\tilde{s}_i\tilde{z}_i$ lies within a composite edge departing from $\tilde{s}_i$ in the direction of $\text{head}(\tilde{z}_i)$, the arguments $\text{head}(\tilde{z}_i)$, and $|\tilde{z}_i|$ suffice to determine the insertion point. Notice that by Lemma C.1 and the definition of $\tilde{z}_i$, either $\tilde{s}_i\tilde{z}_i = s_i$, or $\tilde{s}_i\tilde{z}_i$ is a bifurcation needed to insert $\tilde{s}_i\tilde{z}_i\tilde{b}_i = s_i$. Since $\tilde{T}'_{F_{i-1}}(x^n)$ has the suffix property, $a\tilde{s}_iv \notin \tilde{T}'_{F_{i-1}}(x^n)$ for any $a \in \mathcal{A}$, $v \preceq \tilde{z}_i$, with $v \in \mathcal{A}^+$, for otherwise $\tilde{s}_iv$ would be the state selected by x^i in $\tilde{T}'_{F_{i-1}}(x^n)$ in place of $\tilde{s}_i$. Thus, the set of FSM transitions outgoing from $\tilde{s}_i\tilde{z}_i$, is initially copied from **Transitions**$[\tilde{s}_i]$. Notice that this initial setting may be later modified by the invocation to **Verify***$(\tilde{s}_i\tilde{z}_i)$, which possibly adds suffixes of $\tilde{s}_i\tilde{z}_i$ defining their outgoing FSM transitions, and possibly

updating previously existing transitions. Now, since $T_F(x^n)$ is a full tree, $\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|} \in T_F(x^n)$ for $|\tilde{s}_i\tilde{z}_i| < i$ if and only if $\tilde{s}_i\tilde{z}_i$ is an internal node of $T_F(x^n)$. Thus, $\tilde{b}_i \neq \lambda$ if and only if the condition of Step 7 holds true. In case $\tilde{b}_i \neq \lambda$, a node $\tilde{s}_i\tilde{z}_i\tilde{b}_i$ is created, which, by Lemma C.1, is equal to s_i. Again, since $\tilde{T}'_{F_{i-1}}(x^n)$ has the suffix property, $a\tilde{s}_i v \notin \tilde{T}'_{F_{i-1}}(x^n)$ for any $a \in \mathcal{A}$, $v \preceq \tilde{z}_i\tilde{b}_i$, with $v \in \mathcal{A}^+$, for otherwise $\tilde{s}_i v$ would be the state selected by x^i in $\tilde{T}'_{F_{i-1}}(x^n)$ in place of $\tilde{s}_i$. Thus, the set of FSM transitions outgoing from $\tilde{s}_i\tilde{z}_i\tilde{b}_i$, is initially copied from **`Transitions[`**$\tilde{s}_i\tilde{z}_i$**`]`**. Finally, an invocation of **`Verify*(`**$\tilde{s}_i\tilde{z}_i\tilde{b}_i$**`)`** adds suffixes of $\tilde{s}_i\tilde{z}_i\tilde{b}_i$ defining their outgoing FSM transitions, and possibly updating previously existing transitions.

We now discuss some complexity issues before we show that the whole decoding stage can be implemented in linear time. First, we point out that determining whether a node u of $\tilde{T}'_{F_i}(x^n)$ is an internal node of $T_F(x^n)$, as required by Step 7 of **`Evolve`**, can be implemented efficiently with the help of an extra boolean flag **`Internal[`**u**`]`** associated to every node u of $\tilde{T}'_{F_i}(x^n)$ with $0 \leq i \leq n$. Start by assigning **`Internal[`**u**`]=true`** to all nodes $u \in \tilde{T}'_{F_0}(x^n)$, which, by Lemma 3.2, are internal nodes of $T_F(x^n)$. Then, since $\hat{T}'_F(x^n) = \tilde{T}'_{F_0}(x^n) \subseteq \tilde{T}'_{F_i}(x^n) \subseteq T_F(x^n)$, by Lemma 3.2 (ii), every node w added as a leaf to build $\tilde{T}'_{F_i}(x^n)$ for $0 \leq i < n$ is not internal in $T_F(x^n)$, and we set **`Internal[`**w**`]=false`**. On the other hand, for those nodes $w = uv$ that are created splitting an edge $u \xrightarrow{vy} uvy$ into $u \xrightarrow{v} uv \xrightarrow{y} uvy$, since $uvy \in T_F(x^n)$ we set **`Internal[`**w**`]=true`**. A similar but simpler technique can be use for determining whether a node is an internal node of $T(x^n)$ using the fact that all such nodes are words of $\hat{T}'(x^n)$. This allows an efficient setting of the pointers **`Origin[`**u**`]`** by **`Verify*`**.

In order to achieve linear-time decoding it is also necessary an efficient implementation of the transition propagation in **`Verify*`**. This can be solved by choosing a representation for **`Transitions[`**u**`]`** that groups all transitions that lead to the same destination (from all states) in one single place. That is, all nodes $x'y$ with transition $f(x'y, c) = cx'$ have a pointer, **`Transition[`**$x'y, c$**`]`**, to a cell that in turn points to cx'. This second level indirection can be established in a single traversal of $\hat{T}'_F(x^n)$, upon its original construction. Clearly, the initializations in steps 5 and 10 of **`Evolve`** can be performed in constant time. Now, suppose that a node $w = cx'$ is added as a child of cr as part of the construction of $\tilde{T}'_{F_i}(x^n)$ from $\tilde{T}'_{F_{i-1}}(x^n)$, and **`Verify*(`**$w$**`)`** calls for the insertion of x'. Setting $f(x'y, c) = cx'$ for all descendants $x'y$ of x' that had **`Transition[`**$x'y, c$**`]=`**cr in $\tilde{T}'_{F_{i-1}}(x^n)$, as required by **`Verify*`**, can be done by changing the value of the cell that points to cr, and creating a new cell for all nodes u in the path $r \preceq u \prec x'$. We will show that the update of **`Transition[`**u, c**`]`** for these nodes does not affect the overall linear complexity.

Notice that once s_i has been determined, and possibly added to $\tilde{T}'_{F_i}(x^n)$, clearly $\sigma_{\tilde{T}'_{F_i}(x^n)}(x^i) = s_i$. Thus, given s_i, $\tilde{s}_{i+1}$ can be computed in constant time following the next-state transition **`Transition[`**s_i, x_{i+1}**`]`** in $\tilde{T}'_{F_i}(x^n)$. Starting from $s_0 = \lambda$, the algorithm successively computes $\tilde{s}_i$ from s_{i-1}, and then s_i from $\tilde{s}_i$, using **`Evolve`**, to proceed to the next input symbol. The length of $\tilde{z}_i$, an input to **`Evolve`**, can be efficiently determined in much the same way as the algorithm in Chapter 3. We make use of Lemma C.2 below, for which we define $\tilde{u}_{i+1}$ as $\tilde{u}_{i+1} = \lambda$ if $|\tilde{s}_{i+1}| > |\tilde{s}_i| + 1$, [1] or the string satisfying $x_{i+1}\tilde{s}_i = \tilde{s}_{i+1}\tilde{u}_{i+1}$ otherwise.

[1] Since $\tilde{s}_{i+1}$ and $\tilde{s}_i$ are determined with respect to different GCTs, this condition can indeed hold.

C.2. Lemma. *If* $|\tilde{s}_{i+1}| > |\tilde{s}_i|+1$ *or* $\tilde{s}_{i+1}\tilde{u}_{i+1}head(\tilde{z}_i) \in \text{WORD}(\tilde{T}'_{F_i}(x^n))$*, then either* $\tilde{z}_{i+1} = \lambda$*, or* $\tilde{s}_{i+1}\tilde{z}_{i+1} = x_{i+1}\tilde{s}_i\tilde{z}_i$*.*

Proof. First we show that $\tilde{s}_{i+1}\tilde{z}_{i+1} \preceq x_{i+1}\tilde{s}_i\tilde{z}_i$. Otherwise, since both $\tilde{s}_{i+1}\tilde{z}_{i+1}$, and $x_{i+1}\tilde{s}_i\tilde{z}_i$, are prefixes of $\overline{x^{i+1}}$, $x_{i+1}\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|} \preceq \tilde{s}_{i+1}\tilde{z}_{i+1}$. Thus, $x_{i+1}\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|}$ is a word of $\tilde{T}'_{F_i}(x^n)$, and therefore $x_{i+1}\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|}y \in \tilde{T}'_{F_i}(x^n)$ for some y. But since nodes in $\tilde{T}'_{F_i}(x^n) \setminus \tilde{T}'_{F_{i-1}}(x^n)$ can only be substrings of $\tilde{s}_i\tilde{z}_i\tilde{b}_i$, and $|x_{i+1}\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|}y| > |\tilde{s}_i\tilde{z}_i\tilde{b}_i|$, $x_{i+1}\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|}y$ is also a node of $\tilde{T}'_{F_{i-1}}(x^n)$. By the suffix property $\tilde{s}_i\tilde{z}_i x_{i-|\tilde{s}_i\tilde{z}_i|}y \in \tilde{T}'_{F_{i-1}}(x^n)$, contradicting the definition of $\tilde{z}_i$. We conclude that $\tilde{s}_{i+1}\tilde{z}_{i+1} \preceq x_{i+1}\tilde{s}_i\tilde{z}_i$ as claimed. Assume now that $\tilde{z}_i = \lambda$. In this case, $\tilde{s}_{i+1}\tilde{z}_{i+1} \preceq x_{i+1}\tilde{s}_i$, and therefore we rule out the condition $|\tilde{s}_{i+1}| > |\tilde{s}_i|+1$. Thus, by the assumptions of the lemma, we must have $\tilde{s}_{i+1}\tilde{u}_{i+1} \in \text{WORD}(\tilde{T}'_{F_i}(x^n))$. Hence, by the definition of $\tilde{z}_{i+1}$, $\tilde{s}_{i+1}\tilde{u}_{i+1} \preceq \tilde{s}_{i+1}\tilde{z}_{i+1}$, and since $\tilde{s}_{i+1}\tilde{u}_{i+1} = x_{i+1}\tilde{s}_i$, $x_{i+1}\tilde{s}_i \preceq \tilde{s}_{i+1}\tilde{z}_{i+1}$. But also $\tilde{s}_{i+1}\tilde{z}_{i+1} \preceq x_{i+1}\tilde{s}_i$, thus, $\tilde{s}_{i+1}\tilde{z}_{i+1} = x_{i+1}\tilde{s}_i$. If $\tilde{z}_i \neq \lambda$, by the assumptions, and the prefix relation $\tilde{s}_{i+1}\tilde{z}_{i+1} \preceq x_{i+1}\tilde{s}_i\tilde{z}_i$ that we have shown, we can write $\tilde{s}_{i+1}\tilde{z}_{i+1} = x_{i+1}\tilde{s}_i z'_i$ with $\tilde{z}_i = z'_i z''_i$ and $z'_i \neq \lambda$. Suppose that $z''_i \neq \lambda$. Let w be the shortest string such that $x_{i+1}\tilde{s}_i z'_i w \in \tilde{T}'_{F_i}(x^n)$. Such string exists since $x_{i+1}\tilde{s}_i z'_i = \tilde{s}_{i+1}\tilde{z}_{i+1} \in \text{WORD}\left(\tilde{T}'_{F_i}(x^n)\right)$. If $w = \lambda$, $\tilde{s}_{i+1}$ is $x_{i+1}\tilde{s}_i z'_i$. Thus, $\tilde{z}_{i+1} = \lambda$, and the proof is complete. Suppose now that $w \neq \lambda$. We claim that there exists a string $u \in \mathcal{A}^+$ such that $\tilde{s}_i z'_i u \in \text{WORD}(\tilde{T}'_{F_{i-1}}(x^n))$ and $\text{head}(u) \neq \text{head}(z''_i)$. Notice that $\text{head}(w) \neq \text{head}(z''_i)$, for otherwise $\tilde{z}_{i+1}$ would be longer. If $x_{i+1}\tilde{s}_i z'_i w$ is not a substring of $\tilde{s}_i\tilde{z}_i\tilde{b}_i$, it also belongs to $\tilde{T}'_{F_{i-1}}(x^n)$ and by the suffix property so does $\tilde{s}_i z'_i w$. In this case the string $u = w$ satisfies the claim. If $x_{i+1}\tilde{s}_i z'_i w$ is a substring of $\tilde{s}_i\tilde{z}_i\tilde{b}_i$, then for some string v and $w'b = w$, $vx_{i+1}\tilde{s}_i z'_i w' \in \text{WORD}(\tilde{T}'_{F_{i-1}}(x^n))$. Therefore, there exists a string w'' such that $vx_{i+1}\tilde{s}_i z'_i w'w'' \in \tilde{T}'_{F_{i-1}}(x^n)$ and by suffix property $x_{i+1}\tilde{s}_i z'_i w'w'' \in \tilde{T}'_{F_{i-1}}(x^n)$. Again by the suffix property, $\tilde{s}_i z'_i w'w'' \in \tilde{T}'_{F_{i-1}}(x^n)$, which implies that $w'w'' \neq \lambda$ for otherwise $\tilde{s}_i z'_i$ would be the state at time i . Since $\tilde{T}'_{F_i}(x^n)$ is an extension of $\tilde{T}'_{F_{i-1}}(x^n)$, $x_{i+1}\tilde{s}_i z'_i w'w''$ does also belong to $\tilde{T}'_{F_i}(x^n)$ implying that $\text{head}(w'w'') \neq \text{head}(z''_i)$. Thus, $u = w'w''$ is a string satisfying the claim. We conclude that $\tilde{s}_i z'_i u \in \text{WORD}(\tilde{T}'_{F_{i-1}}(x^n))$, $u \neq \lambda$ and $\text{head}(u) \neq \text{head}(z''_i)$. Therefore, $\tilde{s}_i z'_i$ is a bifurcation and the state at time i would be $\tilde{s}_i z'_i$. The contradiction arises from supposing $z''_i \neq \lambda$, thus $z''_i = \lambda$, which concludes the proof. □

The following lemma will allow us to bound the cost of invocations to **`Verify*`**.

C.3. Lemma. $tail(\tilde{s}_i\tilde{z}_i) \in \tilde{T}'_{F_{i-1}}(x^n)$.

Proof. The claim follows from the suffix property if $\tilde{z}_i = \lambda$. If it is not, by definition $\tilde{s}_i\tilde{z}_i \in \text{WORD}(\tilde{T}'_{F_{i-1}}(x^n))$ and there exists $w \in \mathcal{A}^+$ such that $\tilde{s}_i\tilde{z}_i w \in \tilde{T}'_{F_{i-1}}(x^n)$. Consequently, by the suffix property $\text{tail}(\tilde{s}_i\tilde{z}_i w) \in \tilde{T}'_{F_{i-1}}(x^n)$. By Lemma C.1, $s_i = \tilde{s}_i\tilde{z}_i\tilde{b}_i$ implying that $\tilde{s}_i\tilde{z}_i\tilde{b}_i \in T_F(x^n)$ and $\text{tail}(\tilde{s}_i\tilde{z}_i\tilde{b}_i) \in T_F(x^n)$. $s_{i-1} = \text{tail}(\tilde{s}_i\tilde{z}_i\tilde{b}_i)y$ for some y, and by definition of $\tilde{T}'_{F_{i-1}}(x^n)$, $\text{tail}(\tilde{s}_i\tilde{z}_i\tilde{b}_i)y \in \tilde{T}'_{F_{i-1}}(x^n)$. By definition of $\tilde{z}_i$, $\text{head}(w) \neq \text{head}(\tilde{b}_i y)$. Thus, $\text{tail}(\tilde{s}_i\tilde{z}_i)$ is a bifurcation between $\text{tail}(\tilde{s}_i\tilde{z}_i\tilde{b}_i)y$ and $\text{tail}(\tilde{s}_i\tilde{z}_i w)$. □

By Lemma C.3, the invocation to **`Verify*`** in Step 6 of **`Evolve`** does not lead to recursive calls. Also notice that if $u \in \tilde{T}'_{F_i}(x^n) \setminus \tilde{T}'_{F_{i-1}}(x^n)$, then either $u = \tilde{s}_i\tilde{z}_i$, or u is a suffix of $\tilde{s}_i\tilde{z}_i\tilde{b}_i$,

in which case $u = u'\tilde{b}_i$ is an atomic child of $u' \in \tilde{T}'_{F_{i-1}}(x^n)$.

As we have observed previously, since the states $\tilde{s}_i$, and $\tilde{s}_{i+1}$ are determined with respect to possibly different GCTs, it is plausible to find situations in which $|\tilde{s}_{i+1}| > |\tilde{s}_i| + 1$. The following lemma shows, however, that $\tilde{s}_{i+1}$ may gain at most one extra symbol of context beyond $|\tilde{s}_i|+1$. This will allow us to bound the total cost of determining s_i over all $0 \le i \le n$.

C.4. Lemma. $|\tilde{s}_{i+1}| \le |\tilde{s}_i| + 2$.

Proof. Suppose $|\tilde{s}_{i+1}| > |\tilde{s}_i|+1$. Clearly, $\mathrm{tail}(\tilde{s}_{i+1}) \notin \tilde{T}'_{F_{i-1}}(x^n)$, for otherwise $|\tilde{s}_{i+1}| \le |\tilde{s}_i|+1$. Then, by the suffix property $\tilde{s}_{i+1} \in \tilde{T}'_{F_i}(x^n) \setminus \tilde{T}'_{F_{i-1}}(x^n)$, and by Lemma C.3, $\tilde{s}_{i+1}$ is a suffix of $\tilde{s}_i\tilde{z}_i\tilde{b}_i$, and $\tilde{s}_{i+1} = u'\tilde{b}_i$ is an atomic child of $u' \in \tilde{T}'_{F_{i-1}}(x^n)$. Since, $\mathrm{tail}(u') \in \tilde{T}'_{F_{i-1}}(x^n)$ by the suffix property, $\mathrm{tail}(u') \preceq \tilde{s}_i$, and therefore $|\tilde{s}_i| \ge |u'| - 1 = |\tilde{s}_{i+1}| - 2$. □

Finally we bound the overall complexity of the algorithm.

C.5. Theorem. *`SPContextFSMi` encodes and decodes any sequence x^n in time $O(n)$.*

Proof. We only need to show that the decoder runs in linear time. Due to Lemma C.2, determining $|\tilde{z}_i|$ requires at most $|\tilde{u}_i| + 1$ comparisons, thus determining the insertion point for s_i for all $0 < i \le n$ requires $n + \sum_{i=1}^{n} |\tilde{u}_i|$. By definition of $\tilde{u}_i$ and Lemma C.4, $|\tilde{u}_i| \le |\tilde{s}_{i-1}| - |\tilde{s}_i| + 2$ and therefore $\sum_{i=1}^{n} |\tilde{u}_i|$ is $O(n)$.

We next show that the cost of the invocation `Verify*(`$\tilde{s}_i\tilde{z}_i$) is also proportional to $|\tilde{u}_i|+1$. First notice that by Lemma C.3 $\mathrm{tail}(\tilde{s}_i\tilde{z}_i) \in \tilde{T}'_{F_{i-1}}(x^n)$, thus this invocation does not lead to recursive calls. Moreover, we claim that searching for $\mathrm{tail}(\tilde{s}_i\tilde{z}_i)$ from $\mathrm{tail}(\tilde{s}_i)$ can be done in a node by node traversal and takes at most $|\tilde{u}_i| + 1$ comparisons. To show the claim, we point out that by Lemma C.2, when $|\tilde{z}_i| > |\tilde{u}_i| + 1$, $\mathrm{tail}(\tilde{s}_i\tilde{z}_i) = \tilde{s}_{i-1}\tilde{z}_{i-1}$, and there are no nodes between $\tilde{s}_{i-1}$ and $\tilde{s}_{i-1}\tilde{z}_{i-1}$ except possibly for suffixes of $\tilde{s}_{i-1}\tilde{z}_{i-1}\tilde{b}_{i-1}$ created in previous steps. But by Lemma C.3, these suffixes are created as atomic children of preexisting nodes, thus the number of these suffixes in the path of interest is at most one. Using exactly the same argument, we show that the update of FSM transitions does also take at most $|\tilde{u}_i| + 1$ nodes visited.

Finally, by Lemma C.3, `Verify*(`$\tilde{s}_i\tilde{z}_i\tilde{b}_i$) only creates atomic children of preexisting nodes. Thus, it takes a number of operations proportional to the number of nodes created, and the claim then follows from the fact that all these nodes belong to $T'_F(x^n)$ and $|T'_F(x^n)| = O(n)$. □

C.2 Decoding using incremental suffix tree construction

A linear-time decoder can also be implemented by extending $T(x^n)$ on the fly with the suffix tree of the string decoded so far. A sequential suffix tree building algorithm is presented in [23], and we describe a variation that works extending $T(x^n)$. This approach, which we

denote **SPContextSfx**, does not use the FSM closure and typically requires more storage space than **SPContextFSM** and **SPContextFSMi**.

The algorithm starts from a tree $T_0 = T(x^n)$ and successively constructs T_i inserting the reverse prefix $\overline{x^i}$ in T_{i-1}. Common statistics can be accessed keeping in each inserted node a pointer to the nearest ancestor that belongs to $T(x^n)$. The crucial step of this algorithm from a complexity point of view is the search for suf'_i, the longest prefix of $\overline{x^i}$ that is a word of T_{i-1}, i.e. the insertion point for each new reverse prefix. For $0 < i \leq n$ we define suf_i as the longest prefix of $\overline{x^i}$ that is a substring of $\overline{x^{i-1}}$. It is not difficult to see that suf_i is a prefix of $x_i suf_{i-1}$. Thus, we have,

$$suf_i \preceq x_i suf_{i-1}, \tag{C.1}$$

and also,

$$suf_i \preceq suf'_i. \tag{C.2}$$

A key instrument to achieve linear time complexity is the use of *short-cut* links, which for each node node u of T_i, and each symbol a, points to node auv where v is the shortest string such that $auv \in T_i$, or it is undefined if such a node does not exist. The algorithm starts by initializing short-cut links for all nodes in T_0 and updates the structure as new nodes are inserted. The algorithm starts the construction of T_i from node suf'_{i-1} (we define $suf'_0 \triangleq \lambda$), going into an *upwards traversal*. Namely, it traverses the tree upwards until it finds the first ancestor of suf'_{i-1} that has a short-cut link defined for symbol x_i, or it reaches the root. Let v_i be the node found in the upwards traversal, and w_i the node pointed to by its short-cut link, or $w_i = \lambda$ if the root was reached and it has no short-cut defined for x_i. We will exploit the connection between the nodes w_i, v_i and the string suf_i established by the following proposition.

C.6. Proposition. *If the node w_i found in the upwards traversal at step i is not a node of $T(x^n)$, then $suf_i = x_i v_i$.*

Proof. Since by the assumptions $w_i \in T_{i-1} \setminus T(x^n)$, $|w_i| \leq i-1$, and therefore $|x_i v_i| < i$. If $w_i = x_i v_i$, and there is no edge in the direction of $x_{i-|x_i v_i|}$ from the node $x_i v_i$, then $suf_i = x_i v_i$ and the proof is complete. If this is not the case, we define $w'_i = w_i$ if $w_i \neq x_i v_i$, and otherwise w'_i is the shortest string that is a node of T_{i-1}, and $x_i v_i x_{i-|x_i v_i|} \preceq w'_i$. By definition, $v_i \preceq suf'_{i-1}$, and therefore $x_i v_i \preceq \overline{x^i}$. Since $w_i \notin T(x^n)$, $x_i v_i$ is a substring of $\overline{x^{i-1}}$, and therefore $x_i v_i \preceq suf_i$. Since also $x_i v_i \prec w'_i$, and by the definition of w'_i there are no bifurcation nodes u, $x_i v_i \prec u \prec w'_i$, we must have either $suf_i \prec w'_i$, or $w'_i \preceq suf_i$. Since w'_i is a node of $T_{i-1} \setminus T(x^n)$, there must exist $1 \leq j, k < i$, such that $\overline{x^j} = w'_i z$, $\overline{x^k} = w'_i z'$, and either $\mathrm{head}(z) \neq \mathrm{head}(z')$, or $\overline{x^j} = \overline{x^k} = w'_i$. Then, the reverse prefixes $\overline{x^{j-1}}$, and $\overline{x^{k-1}}$ imply the existence of the node $\mathrm{tail}(w'_i)$ in T_{i-1}, with a short-cut pointing to w'_i for symbol x_i. Thus, in case $w'_i \preceq suf_i$, $\mathrm{tail}(w'_i) \preceq suf_{i-1}$ by (C.1), and furthermore $\mathrm{tail}(w'_i) \preceq suf'_{i-1}$ by (C.2). However, the upwards traversal stopped at v_i although $x_i v_i \prec w'_i$, which implies that $v_i \prec \mathrm{tail}(w'_i)$. We arrived to a contradiction from supposing that $w'_i \preceq suf_i$, and we must then have $suf_i \prec w'_i$. Since suf_i is a substring of $\overline{x^{i-1}}$, $suf_i \preceq \overline{x^j}$ for some $j < i$. Thus,

$suf_i y \in T_{i-1}$ for some y. Now, since $x_i v_i \preceq suf_i \prec w'_i$, and there are no bifurcation nodes u, $x_i v_i \prec u \prec w'_i$, we must have $suf_i = x_i v_i$ (i.e., $w_i \preceq suf_i y$ if $w_i = w'_i$, or $x_i v_i$ does not have in fact an edge in the direction of $x_{i-|x_i v_i|}$ if $w_i = x_i v_i$). □

Now, if the short-cut link found is of the form $w_i = x_i v_i y'_i$ for some $y'_i \in \mathcal{A}^+$, then w_i does not belong to the full tree $T(x^n)$, and by Proposition C.6, $suf'_i = suf_i = x_i v_i$ lies within the composite edge that links w_i with its parent. If $w_i = x_i v_i$, then $suf'_i = w_i y''_i$ for some string y''_i that may not be empty when $suf'_i \in T(x^n)$. In this case the algorithm goes into a *downwards traversal* descending from w_i to the leaf of $T(x^n)$, suf'_i, following past symbols that have already been decoded (a transient state is reached if $\overline{x^{i-1}}$ is exhausted). Once suf'_i is determined, a new leaf representing $\overline{x^i}$ is added and all nodes in the unsuccessful portion of the upwards traversal are given short-cut links pointing to it for symbol x_i. If suf'_i was within a composite edge, short-cut links of nodes $u \preceq v_i$ that previously pointed to $w_i = x_i v_i y'_i$ are updated to point to $x_i v_i$.

Some implementation considerations are needed to achieve linear-time decoding. During a downwards traversals, an auxiliary data structure associated to node v_i, **jump[**v_i**]**, is constructed. The structure **jump[**v_i**]** maps a symbol $a = x_i$ and an index j to the j-th node traversed downwards. Notice that if the preceding upwards traversal is not empty, there are no nodes in the sub-path from $v_i = \text{tail}(w_i)$ to $\text{tail}(w_i y''_i)$ of the upwards traversal, for otherwise this node would have a short-cut link for symbol x_i. Thus, **jump[**v_i**]** is a mapping between substrings y of a composite edge departing from v_i into nodes $x_i v_i y$. This structure can be updated in constant time whenever a composite edge is split.

The goal of **jump[**v_i**]** is to avoid revisiting nodes of $T(x^n)$ during downwards traversals. Consider the downwards traversal at time i, and let y be the string such that $v_i y = suf'_{i-1}$. We recall that in case a downwards traversal occurs, we must have $w_i = x_i v_i$. Let $z = z_0 z_1 z_2$, with $z_0, z_1, z_2 \in \mathcal{A}^*$, such that $suf'_i = x_i v_i z$, and z_1 has been previously downwards traversed. This means that a reverse prefix $x_i v_i z_0 z_1 z'$ was inserted at some time $j < i$, and $v_i z_0 z_1 z'$ was also inserted at time $j - 1$. $v_i z_0 z_1 z'$ can not be a prefix of $v_i y$ since it has a short-cut link for symbol x_i contradicting the definition of v_i. If $v_i y$ is not a prefix of $v_i z_0 z_1 z'$, then there must be a bifurcation node that would also contain a short-cut link. We conclude that $v_i y$ is a prefix of $v_i z_0 z_1 z'$, and **jump(**v_i**)** can be used to avoid traversing $z_0 z_1$.

In the following theorem we study the complexity of **SPContextSfx**.

C.7. Theorem. *SPContextSfx encodes and decodes any sequence x^n in time $O(n)$.*

Proof. We only need to show that the decoder runs in linear time. To find the complexity of the decoder we must find the sum over all i of of the number of nodes visited in upwards traversals, the number of nodes in the path w_i to $w_i y''_i$ (downwards traversal), and the number of nodes whose short-cut links are updated when suf'_i splits a composite edge. By the preceding discussion, each node of $T(x^n)$ is visited at most once in a downwards traversal, and, as argued in Chapter 3, $|T(x^n)| = O(n)$. As for upwards traversals, all nodes in the unsuccessful portion of a traversal looking for a short-cut pointer for symbol a, are then given a short-cut pointer for that symbol. Thus, every node is visited at most α times. Since, the

whole (compact) suffix tree has $O(n)$ nodes, and $T(x^n)$ has $O(n)$ nodes, the overall cost of upwards traversals is $O(n)$.

When suf'_i splits a composite edge, a node $suf'_i = suf_i = x_i v_i$ is created with no short-cut link. Thus, if $i < n$, the next upwards traversal incudes at least the parent of the new node, $r = \text{PAR}_{T_i}(suf_i)$, and therefore $v_{i+1} \preceq r$. Now, either $x_{i+1}v_{i+1} = suf_{i+1}$ if $x_{i+1}v_{i+1} \notin T(x^n)$, or $suf_{i+1} \preceq x_{i+1}v_{i+1}$ otherwise. Thus, in any case, $|suf_{i+1}| \leq |v_{i+1}| + 1 \leq |r| + 1$. Hence, the cost of updating short-cut links of nodes $u \preceq v_i$ that pointed to $w_i = x_i v_i y'_i$, to make them point to to suf_i, is bounded by $|suf_i| - |r| \leq |suf_i| - |suf_{i+1}| + 1$. Notice that although this kind of short-cut link update occurs only when suf'_i splits a composite edge, $|suf_i| - |suf_{i+1}| + 1$ is non negative for all i by (C.1). We then bound the total cost of these updates for $1 \leq i < n$, by the telescopic summation $\sum_{i=1}^{n-1} |suf_i| - |suf_{i+1}| + 1 = O(n)$. For $i = n$, the cost is trivially bound by n. □

Appendix D

Proofs for Chapter 4

D.1 Proof of Lemma 4.13

For every s_h in the state sequence of x^n, we have $s_h \preceq \overline{\overline{s_0}x^h}$, and by (4.5) we get

$$\mu_i(s_h) \preceq \overline{(\overline{s_0}x)^{|s_0|+h-\ell_{s_h}+i}} \text{ for } 0 \le h \le n, 1 \le i \le \ell_{s_h}\,. \tag{D.1}$$

If also j is an integer such that $h > j$ and $j > h - \ell_{s_h}$, the index $i = \ell_{s_h} + j - h$ satisfies $0 < i < \ell_{s_h}$ as stated in Part (i). We then get from (D.1),

$$\begin{aligned} \mu_i(s_h) \preceq \overline{(\overline{s_0}x)^{|s_0|+j}}, \quad & \text{for } 0 \le h \le n,\, h > j,\, j > h - \ell_{s_h}, \\ & \text{where } i = \ell_{s_h} + j - h\,. \end{aligned} \tag{D.2}$$

We claim that if $h > 0$, j must be positive. Otherwise, $j' = 0$ also satisfies $h > j'$ and $j' > h - \ell_{s_h}$, and from (D.2) we get $\mu_{i'}(s_h) \preceq s_0$ with $i' = \ell_{s_h} - h$. Then, since no pseudo-state is a proper prefix of a state, we get $\mu_{i'}(s_h) = s_0$. However, we have $|\mu_{i'}(s_h)| < |s_h|$ since $i' < \ell_{s_h}$, which is a contradiction. We conclude that

$$h - \ell_{s_h} \ge 0 \quad \text{for } 0 < h \le n\,. \tag{D.3}$$

If $j > 0$, which is guaranteed when $h > 0$, (D.2) reduces to $\mu_i(s_h) \preceq \overline{x^j}s_0$. Hence, we have

$$\begin{aligned} s_j = \nu_i(s_h) \quad & \text{for } 0 < h \le n,\, h > j,\, j > h - \ell_{s_h}, \\ & \text{where } i = \ell_{s_h} + j - h\,. \end{aligned} \tag{D.4}$$

To conclude the proof of Part (i), it remains to show that $|s_j| < |s_h|$, and $\ell_{s_h} - \ell_{s_j} \ge h - j$. The former follows from the fact that $i < \ell_{s_h}$ and the definition of $\nu_i(s_h)$. The latter follows by Lemma 4.7(i), from which we get $i \ge \ell_{s_j}$. Thus, we have $\ell_{s_h} + j - h \ge \ell_{s_j}$, or, $\ell_{s_h} - \ell_{s_j} \ge h - j$ as claimed.

For Part (ii) it is sufficient to show that $1 \le j_\Delta \le \ell_{s_{\vec{j}}}$, since in this case, (D.1) with j_Δ in the role of i, and $\vec{j}$ in the role of h yields

$$\mu_{j_\Delta}(s_{\vec{j}}) \preceq \overline{(\overline{s_0}x)^{|s_0|+\vec{j}-\ell_{s_{\vec{j}}}+j_\Delta}}\,, \tag{D.5}$$

or, since $\vec{j} - \ell_{s_{\vec{j}}} + j_\Delta = j$ by definition,

$$\mu_{j_\Delta}(s_{\vec{j}}) \preceq \overline{x^j}s_0\,. \tag{D.6}$$

Let $0 \le j \le n$. Since $\vec{j} \ge j$ by definition, it is clear that $j_\Delta = j - \vec{j} + \ell_{s_{\vec{j}}} \le \ell_{s_{\vec{j}}}$. Let $m = \max\{m : m \ge j, j > m - \ell_{s_m}\}$. We have $m \le \vec{j}$, for otherwise $m > \vec{j}, m - \ell_{s_m} < j \le \vec{j}$

and $\vec{\jmath}$ would not belong to J. Suppose $m < \vec{\jmath}$. Then, by definition of $\vec{\jmath}$, we have $m \notin J$, and therefore there exists $m' > m$ such that $m > m' - \ell_{s_{m'}}$. Since m' is positive, (D.4) with m' in the role of h, and m in the role of j, yields

$$s_m = \nu_{i'}(s_{m'}) \quad \text{where } i' = \ell_{s_{m'}} + m - m' \,. \tag{D.7}$$

By Lemma 4.7(i), with s_m in the role of s' and $s_{m'}$ in the role of s, we obtain $i' \geq \ell_{s_m}$. Thus, since $m' - \ell_{s_{m'}} = m - i'$ by (D.7), we get $m' - \ell_{s_{m'}} \leq m - \ell_{s_m} < j$. We then have $m' - \ell_{s_{m'}} < j$ but $m' > m$, which is a contradiction since m is the maximum index satisfying $m - \ell_{s_m} < j$. We conclude that $m = \vec{\jmath}$. Thus $j > \vec{\jmath} - \ell_{s_{\vec{\jmath}}}$, and therefore $j_\Delta = j - \vec{\jmath} + \ell_{s_{\vec{\jmath}}} \geq 1$. This concludes the proof of Part (ii).

By definition of J, we know that $n \in J$, and by (D.3), we have $0 \in J$. Thus, $t_0 = 0$ and $t_r = n$ as claimed in the first part of (iii). To prove the remaining of Part (iii), and Part (iv), we apply Part (ii) taking in particular $j = t_i + 1$ for $0 \leq i < r$. In this case we have $\vec{\jmath} = t_{i+1}$. Then, we have $j_\Delta = t_i + 1 - t_{i+1} + \ell_{s_{t_{i+1}}}$, which is positive by Part (ii), and therefore $t_i \geq t_{i+1} - \ell_{s_{t_{i+1}}}$. But also $t_i \leq t_{i+1} - \ell_{s_{t_{i+1}}}$, for otherwise t_i would not belong to J. Hence, we have $t_i = t_{i+1} - \ell_{s_{t_{i+1}}}$ as claimed in (iii). Now, replacing $t_i = t_{i+1} - \ell_{s_{t_{i+1}}}$ in $j_\Delta = t_i + 1 - t_{i+1} + \ell_{s_{t_{i+1}}}$, we get $j_\Delta = 1$. Thus, we get $\mu_1(s_{t_{i+1}}) \preceq \overline{x^{t_i+1}} s_0$ by Part (ii). Since also we have $x_{t_i+1} s_{t_i} \preceq \overline{x^{t_i+1}} s_0$ and, by (4.6), $x_{t_i+1} s_{t_i} \not\preceq \mu_1(s_{t_{i+1}})$, we conclude that $\mu_1(s_{t_{i+1}}) \prec x_{t_i+1} s_{t_i}$, which proves Part (iv).

For Part (v) we write $N^{(\mu)}$ as

$$N^{(\mu)} \;=\; \sum_{i=1}^{n} \mathbf{1}_{s_{i-1}, s_i} \,.$$

Splitting the summation in intervals between indexes in J, and recalling that $t_0 = 0$ and $t_r = n$, we get

$$\begin{aligned} N^{(\mu)} &= \sum_{i=1}^{r} \sum_{j=t_{i-1}+1}^{t_i} \mathbf{1}_{s_{j-1}, s_j} \\ &= \sum_{i=1}^{r} \left(\mathbf{1}_{s_{t_{i-1}}, s_{t_{i-1}+1}} + \sum_{j=t_{i-1}+2}^{t_i} \mathbf{1}_{s_{j-1}, s_j} \right) . \end{aligned}$$

Since $t_{i-1} = t_i - \ell_{s_{t_i}}$ by Part (iii), this becomes

$$N^{(\mu)} = \sum_{i=1}^{r} \left(\mathbf{1}_{s_{t_{i-1}}, s_{t_{i-1}+1}} + \sum_{j=t_i - \ell_{s_{t_i}}+2}^{t_i} \mathbf{1}_{s_{j-1}, s_j} \right) .$$

For each j in the inner summation of the last equation, we have $\vec{\jmath} = t_i$. Thus, by Part (ii), we have $s_j = \nu_{j_\Delta}(s_{t_i})$, and similarly $s_{j-1} = \nu_{(j-1)_\Delta}(s_{t_i})$. We then get

$$N^{(\mu)} = \sum_{i=1}^{r} \left(\mathbf{1}_{s_{t_{i-1}}, s_{t_{i-1}+1}} + \sum_{j=t_i - \ell_{s_{t_i}}+2}^{t_i} \mathbf{1}_{\nu_{(j-1)_\Delta}(s_{t_i}), \nu_{j_\Delta}(s_{t_i})} \right) .$$

From the definition of j_Δ, we have $(j-1)_\Delta = j - 1 - t_i + \ell_{s_{t_i}}$, and $j_\Delta = j - t_i + \ell_{s_{t_i}}$. Thus, with a change of index variable, the last equation becomes

$$N^{(\mu)} = \sum_{i=1}^{r} \left(\mathbf{1}_{s_{t_{i-1}}, s_{t_{i-1}+1}} + \sum_{j=2}^{\ell_{s_{t_i}}} \mathbf{1}_{\nu_{j-1}(s_{t_i}), \nu_j(s_{t_i})} \right) . \tag{D.8}$$

By the definition of $\tilde{\Delta}^-$ we get

$$N^{(\mu)} = \sum_{i=0}^{r-1} \left(\mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^-(s_{t_{i+1}}) \right) , \tag{D.9}$$

as claimed in Part (v).

For Part (vi) we proceed in a similar way. We start from the definition of K and write

$$K = N^{(\mu)} + \sum_{i=1}^{n} \Theta(s_i) .$$

Splitting the summation in intervals between indexes in J, and recalling that $t_0 = 0$ and $t_r = n$, we get

$$K = N^{(\mu)} + \sum_{i=1}^{r} \sum_{j=t_{i-1}+1}^{t_i} \Theta(s_j) .$$

For each j in the inner summation of the last equation, we have $\vec{j} = t_i$. Thus, by Part (ii), we get $s_j = \nu_{j_\Delta}(s_{t_i})$. Since $j_\Delta = j - t_i + \ell_{s_{t_i}}$, we obtain

$$K = N^{(\mu)} + \sum_{i=1}^{r} \sum_{j=t_{i-1}+1}^{t_i} \Theta(\nu_{j-t_i+\ell_{s_{t_i}}}(s_{t_i})) . \tag{D.10}$$

By Part (iii), we have $t_{i-1} = t_i - \ell_{s_{t_i}}$, which yields

$$K = N^{(\mu)} + \sum_{i=1}^{r} \sum_{j=t_i-\ell_{s_{t_i}}+1}^{t_i} \Theta(\nu_{j-t_i+\ell_{s_{t_i}}}(s_{t_i})) . \tag{D.11}$$

Now, changing the index variable,

$$K = N^{(\mu)} + \sum_{i=1}^{r} \sum_{j=1}^{\ell_{s_{t_i}}} \Theta(\nu_j(s_{t_i})) \tag{D.12}$$

$$= N^{(\mu)} + \sum_{i=1}^{r} \left(\Theta(s_{t_i}) + \sum_{j=1}^{\ell_{s_{t_i}}-1} \Theta(\nu_j(s_{t_i})) \right) . \tag{D.13}$$

We recall the definition of $\Theta(s) = \tilde{\Theta}(s) - \sum_{t=1}^{\ell_s - 1} \Theta(\nu_t(s))$, from which we get

$$K = N^{(\mu)} + \sum_{i=1}^{r} \tilde{\Theta}(s_{t_i}) . \tag{D.14}$$

Now, by Part (v), we get

$$K = \sum_{i=0}^{r-1} \left(\mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^-(s_{t_{i+1}}) \right) + \sum_{i=0}^{r-1} \tilde{\Theta}(s_{t_{i+1}}) \,,$$

and by the definition $\tilde{\Theta}(s) = \tilde{\Delta}^+(s) - \tilde{\Delta}^-(s)$, this yields

$$K = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^+(s_{t_{i+1}}) \,, \tag{D.15}$$

as claimed in Part (vi).

If $u, v \in S_T$, and $\tilde{\Delta}^+_{u,v}(s_{t_{i+1}}) > 0$, we must have $u = \text{tail}(v)$, and therefore $\tau(u, \text{head}(v)) = v$, which yields $d_{u,v} = 0$. Hence, applying the definition of D to (D.15), we get

$$D = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, s_{t_i+1}} + d(s_{t_i}, s_{t_i+1}) + \tilde{\Delta}^+(s_{t_{i+1}}) \,.$$

Also, since $t_i \in J$, we have $x_{t_i+1} s_{t_i} \notin T$. Thus, we have $\mathbf{1}_{s_{t_i}, s_{t_i+1}} + d(s_{t_i}, s_{t_i+1}) = \mathbf{1}_{s_{t_i}, \tau(s_{t_i}, x_{t_i+1})}$, and we get Part (vii),

$$D = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, \tau_i} + \tilde{\Delta}^+(s_{t_{i+1}}) \,. \tag{D.16}$$

We prove Part (viii) next. From the last equation, we obtain

$$D_{w*} = \sum_{i=0}^{r-1} \delta_{s_{t_i}, w} + \tilde{\Delta}^+_{w*}(s_{t_{i+1}}) \,.$$

Notice that, by the definition of $\tilde{\Delta}^+(s)$, we have $\tilde{\Delta}^+_{w*}(s) + \delta_{w,s} = \tilde{\Delta}^+_{*w}(s) + \delta_{w,\mu_1(s)}$. Thus, we get

$$D_{w*} = \sum_{i=0}^{r-1} \delta_{s_{t_i}, w} + \tilde{\Delta}^+_{*w}(s_{t_{i+1}}) + \delta_{w,\mu_1(s_{t_{i+1}})} - \delta_{w, s_{t_{i+1}}} \,.$$

The terms $\delta_{s_{t_i}, w}$, and $-\delta_{w, s_{t_{i+1}}}$ cancel each other along the summation, except for $\delta_{s_{t_0}, w}$ and $-\delta_{w, s_{t_r}}$. Recalling that $t_0 = 0$ and $t_r = n$ by Part (iii), we get

$$D_{w*} = \delta_{s_0, w} - \delta_{s_n, w} + \sum_{i=0}^{r-1} \tilde{\Delta}^+_{*w}(s_{t_{i+1}}) + \delta_{w,\mu_1(s_{t_{i+1}})} \,. \tag{D.17}$$

From (D.16) we also have

$$D_{*w} = \sum_{i=0}^{r-1} \delta_{w, \tau_i} + \tilde{\Delta}^+_{*w}(s_{t_{i+1}}) \,. \tag{D.18}$$

Thus, subtracting (D.17) from (D.18) we get,

$$D_{*w} - D_{w*} = \delta_{s_n, w} - \delta_{s_0, w} + \sum_{i=0}^{r-1} \delta_{w, \tau_i} - \delta_{w,\mu_1(s_{t_{i+1}})} \,. \tag{D.19}$$

Consider now a pseudo-state $u \in U \backslash S_T$, and let $v = \rho(u)$. We recall, from the definition (4.12) of B, that $B_{u,v} = \sum_{w \in \Lambda(u)} D_{*w} - D_{w*}$. Thus, since $\delta_{s_0,w} = \delta_{s_n,w} = 0$ for all $w \in \Lambda(u)$, summing (D.19) in $w \in \Lambda(u)$ we get $B_{u,v} = \sum_{w \in \Lambda(u)} \sum_{i=0}^{r-1} \delta_{w,\tau_i} - \delta_{w,\mu_1(s_{t_{i+1}})}$ as claimed in (viii).

For any $u \in U$ we claim that

$$\sum_{i=0}^{r-1} \sum_{w \in \Lambda(u)} \delta_{w,\tau_i} - \delta_{w,\mu_1(s_{t_{i+1}})} \geq 0\,, \text{ with equality when } u \in S_T. \tag{D.20}$$

For fixed i, δ_{w,τ_i} is non zero for at most one $w \in \Lambda(u)$, and also $\delta_{w,\mu_1(s_{t_{i+1}})}$ is non zero for at most one value of w. As $\tau_i \in \Lambda(\mu_1(s_{t_{i+1}}))$ by Part (iv), if $\delta_{w,\mu_1(s_{t_{i+1}})} = 1$ for w, there exists $w' \in \Lambda(w) \subset \Lambda(u)$ such that $\delta_{w',\tau_i} = 1$. This shows the non-negativity of (D.20). On the other hand, if $\delta_{w,\tau_i} = 1$ for some $w \in \Lambda(u)$, w belongs simultaneously to $\Lambda(u)$ and $\Lambda(\mu_1(s_{t_{i+1}}))$ by Part (iv). When $u \in S_T$, no $v \in U$ is a prefix of u, and we must have $\Lambda(\mu_1(s_{t_{i+1}})) \subset \Lambda(u)$. Hence, there exists $w' \in \Lambda(u)$ such that $\delta_{w',\mu_1(s_{t_{i+1}})} = 1$, which shows that (D.20) is zero when $u \in S_T$. We conclude from Part (viii) and (D.20) that for $u \in U \backslash S_T$ and $v = \rho(u)$, we have $B_{u,v} \geq 0$. From (D.16) we see that also D is non-negative and, therefore, $F_{i,j} \geq 0$ for all i, j. Also, for $u \in U \backslash S_T$, we have

$$\begin{aligned} B_{u,v} &= \sum_{w \in \Lambda(u)} D_{*w} - D_{w*} \\ &= D_{*u} - D_{u*} + \sum_{w \in \bar{\Lambda}(u)} D_{*w} - D_{w*} \\ &= D_{*u} - D_{u*} + \sum_{w : \rho(w) = u} B_{w,u}\,. \end{aligned} \tag{D.21}$$

Thus, since $B_{u,v} = B_{u*}$ by the definition of B, we get $F_{*u} = F_{u*}$. Similarly, for $u \in S_T$ we get summing (D.19) for $w \in \Lambda(u)$, and applying (D.20),

$$\sum_{w \in \Lambda(u)} D_{*w} - D_{w*} = \delta_{s_n,u} - \delta_{s_0,u}\,.$$

Thus,

$$\begin{aligned} \delta_{s_n,u} - \delta_{s_0,u} &= D_{*u} - D_{u*} + \sum_{w \in \bar{\Lambda}(u)} D_{*w} - D_{w*} \\ &= D_{*u} - D_{u*} + \sum_{v : \rho(v) = u} B_{v,u}\,, \end{aligned}$$

and since $B_{u*} = 0$ for $u \in S_T$, we get

$$\delta_{s_n,u} - \delta_{s_0,u} = F_{*u} - F_{u*}\,.$$

This concludes the proof of Part (ix).

Notice that, by the definitions of $\tilde{\Delta}^+$ and $\tilde{\Delta}^-$, we have $\sum_{i,j} \tilde{\Delta}^+_{i,j}(s) = \sum_{i,j} \tilde{\Delta}^-_{i,j}(s)$ for all states s. Hence, the fact that $\sum_{i,j} D_{i,j} = \sum_{i,j} K_{i,j} = n$, claimed in Part (x), follows from

parts (v), (vi), and (vii). Now, for $u \in S_T$, we have $B_{u*} = 0$ and, therefore, $F_{u*} = D_{u*} \leq n$. Also, from (D.21), for $u \in U \backslash S_T$, we have

$$B_{u,\rho(u)} + D_{u*} = D_{*u} + \sum_{w \in \bar{\Lambda}(u)} (D_{*w} - D_{w*}) .$$

Since D is non-negative, we have $B_{u,\rho(u)} + D_{u*} \leq D_{*u} + \sum_{w \in \bar{\Lambda}(u)} D_{*w}$. Hence, since $F_{u*} = B_{u,\rho(u)} + D_{u*}$, we get $F_{u*} \leq n$. To bound F_{*u} we notice that, by Part (ix), we have $F_{*u} \leq F_{u*}$ except when $u = s_n \neq s_0$, in which case $F_{*u} = F_{u*} + 1$. For $u = s_n$, we have $F_{u*} = D_{u*} \leq n$ since $B_{u*} = 0$ for $u \in S_T$. By Part (vii), recalling that $t_0 = 0$, we see that $D_{s_0 *} > 0$. Therefore, we have $D_{u*} < n$ for $u \neq s_0$. Thus, we get $F_{*u} \leq n$ as claimed.

As for Part (xi), let $v \prec u$ and $F_{u,v} > 0$. From (D.16) it is clear that $D_{u,v} = 0$. Hence, we have $F_{u,v} = B_{u,v}$ and $v = \rho(u)$ by the definition (4.12) of B. Thus, we have $B_{u,v} > 0$, which by Part (viii) implies that the summation $\sum_{i=0}^{r-1} \sum_{w \in \Lambda(u)} \delta_{w,\tau_i} - \delta_{w,\mu_1(s_{t_{i+1}})}$ is positive. Hence, there exists an index i, $0 \leq i < r$, such that $\tau_i = w \in \Lambda(u)$. Thus, we must have $u \preceq x_{t_i+1} s_{t_i}$, and therefore $\mathrm{tail}(u) \preceq s_{t_i}$.

D.2 Proof of Lemma 4.29

For each $t \in S_T$, we let b_t denote an arbitrary but fixed symbol, such that $b_t t$ is not an internal node of T. Since T is canonical, such symbol does exist for all states t. Consider a state $s \in S_T$, and a symbol $b \in \mathcal{A}$, such that bs is not an internal node of T. We claim that there exists a string w, such that for every $z, z' \in \mathcal{A}^*$, we have for $x^n = z\overline{s}bwz'$ and $j = |z\overline{s}|$ that $s_j = \mu_{j_\Delta}(s_{\vec{j}})$. Take w of length $k = \mathsf{depth}(T)$, defined as $w_i = b_{s'_i}$ where

$$s'_i = \begin{cases} \sigma_T(\overline{s}b) & i = 1 , \\ \sigma_T(\overline{s'_{i-i} b_{s'_{i-i}}}) & 1 < i \leq k . \end{cases} \tag{D.22}$$

When $b's'$ is not an internal node of T, $\overline{s'}b'$ is sufficiently long to select a state in T, thus (D.22) is well defined. Notice that, with $x^n = z\overline{s}bwz'$ and $j = |z\overline{s}|$, we have that $s_{j+i} = s'_i$ for $1 \leq i \leq k$. By (D.22) and the definition of b_s, we have $|s_{j+i}| \leq |s_{j+i-1}| + 1$ for $1 \leq i \leq k$. Hence, we get $|s_m| \leq |s_j| + m - j$ for $j \leq m \leq j + k$. Since k is the depth of T, we know that $\ell_{s'} \leq k$ for all $s' \in S_T$. Therefore, recalling that $\vec{j} = j + \ell_{s_{\vec{j}}} - j_\Delta$, we get $\vec{j} < j + k$. Hence, taking $m = \vec{j}$ we get $|s_{\vec{j}}| \leq |s_j| + \vec{j} - j$. Thus, we have

$$\begin{aligned} |s_j| &\geq |s_{\vec{j}}| + j - \vec{j} \\ &= |s_{\vec{j}}| - (\ell_{s_{\vec{j}}} - j_\Delta) \\ &= |\mu_{j_\Delta}(s_{\vec{j}})| , \end{aligned}$$

where the last equality follows from (4.5). By Lemma 4.13(ii), we have $s_j = \nu_{j_\Delta}(s_{\vec{j}})$, which is a prefix of $\mu_{j_\Delta}(s_{\vec{j}})$ and, therefore, we get $s_j = \mu_{j_\Delta}(s_{\vec{j}})$ as claimed. The string w we have constructed is a function of s and b, and we denote it $\varphi(s, b)$.

Consider now an edge of the form $e = (\mu_i(s), \mu_{i+1}(s))$ where s is any state of T. Since T is canonical, let $b \in \mathcal{A}$ be a symbol such that $bs \notin \mathcal{I}(T)$. Let $w = \varphi(s, b)$ and let $y' = \overline{s}bw$. Every

time y' occurs in x^n, i.e., $x^n = z\overline{s}bwz'$ for some $z, z' \in \mathcal{A}^*$, the state s_j with $j = |z\overline{s}|$ satisfies $s_j = s = \mu_{j_\Delta}(s_{\vec{j}})$. By Lemma 4.7(iii), we have $\mu_i(s) = \mu_i(s_{\vec{j}})$ and $\mu_{i+1}(s) = \mu_{i+1}(s_{\vec{j}})$. Thus, the term with $t_{i+1} = \vec{j}$ in equation (D.23) below (from Lemma 4.13(vii)) makes a positive contribution to $D_{u,v}$ with $u = \mu_i(s), v = \mu_{i+1}(s)$. Moreover, if x^n can also be decomposed as $x^n = \tilde{z}\overline{s}bw\tilde{z}'$ for some $\tilde{z}, \tilde{z}' \in \mathcal{A}^*$, and $m = |\tilde{z}\overline{s}|$, then $s_m = s = \mu_{m_\Delta}(s_{\vec{m}})$. Thus, either $\vec{m} \neq \vec{j}$, or $m = j$ for also $s_j = s = \mu_{j_\Delta}(s_{\vec{j}})$. Thus, every different occurrence of y' in x^n provokes a new increment to $D_{u,v}$. Hence, taking $y = \overline{y'}$, we have $F_{u,v}(x^n) \geq n_y(x^n)$.

$$D = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i},\tau_i} + \tilde{\Delta}^+(s_{t_{i+1}})\,. \tag{D.23}$$

Consider now a state $u = s \in S_T$ and a symbol b, such that $v = \tau(s, b)$ is well defined, i.e., bs is not an internal node of T. Let $y' = \overline{s}bw$ where $w = \varphi(s, b)$. Every time y' occurs in x^n, i.e. $x^n = z\overline{s}bwz'$ for some $z, z' \in \mathcal{A}^*$, we have for $j = |z\overline{s}|$ that $s = s_j = \mu_{j_\Delta}(s_{\vec{j}})$. If $\vec{j} = j$, the term $\mathbf{1}_{s_{t_i},\tau_i}$ with $t_i = \vec{j}$ in (D.23) makes a positive contribution to $D_{u,v}$. Otherwise, we have $(j \overset{\rightarrow}{+} 1) = \vec{j}$ and therefore $(j+1)_\Delta = j_\Delta + 1$. Thus, by Lemma 4.13(ii), we get $\mu_{j_\Delta+1}(s_{\vec{j}}) \preceq bs\overline{z}$. Since $\mu_{j_\Delta}(s_{\vec{j}}) = s$, we must have $\mu_{j_\Delta+1}(s_{\vec{j}}) = bs$ and consequently $\tau(s, b) = \mu_{j_\Delta+1}(s_{\vec{j}})$. Hence, the term $\tilde{\Delta}^+(s_{t_{i+1}})$ with $t_{i+1} = \vec{j}$ adds to $D_{u,v}$. Moreover, if x^n can also be decomposed as $x^n = \tilde{z}\overline{s}bw\tilde{z}'$ for some $\tilde{z}, \tilde{z}' \in \mathcal{A}^*$, and $m = |\tilde{z}\overline{s}|$, then $s_m = s = \mu_{m_\Delta}(s_{\vec{m}})$. Thus, either $\vec{m} \neq \vec{j}$, or $m = j$ for also $s_j = s = \mu_{j_\Delta}(s_{\vec{j}})$. Hence, the term $\mathbf{1}_{s_{t_i},\tau_i}$ in case $\vec{j} = j$ is counted only once, and also the term $\tilde{\Delta}^+(s_{t_{i+1}})$ in case $\vec{j} \neq j$ is counted only once. Thus, every occurrence of y' in x^n makes a positive contribution to $D_{u,v}$. Hence, taking $y = \overline{y'}$, we have $F_{u,v}(x^n) \geq n_y(x^n)$.

Let $v = \rho(u)$ with $\mathrm{tail}(u) \in T$, $s \in S_T$ such that $\mathrm{tail}(u) \preceq s$, and $s' \in S_T$ such that $s' \preceq v$. Let also b be a symbol such that $bs' \notin \mathcal{I}(T)$, let $w = \varphi(s', b)$, and $c = \mathrm{head}(s')$. Let $y' = \overline{s}cbw$. Every time y' occurs in x^n, i.e. $x^n = z\overline{s}cbwz'$ for some $z, z' \in \mathcal{A}^*$, we have for $j = |z\overline{s}| + 1$ that $s' = s_j = \mu_{j_\Delta}(s_{\vec{j}})$, and $s = s_{j-1}$. Since $\mathrm{tail}(u) \in T$ and $s' \prec u$, we know that $\mathrm{tail}(s') \in \mathcal{I}(T)$. Thus, we have $\mathrm{tail}(\mu_{j_\Delta}(s_{\vec{j}})) \in \mathcal{I}(T)$ and, therefore, $j - \vec{j} + \ell_{s_{\vec{j}}} = j_\Delta = 1$ by (4.6). Hence, we have $j - 1 = \vec{j} - \ell_{s_{\vec{j}}}$, which belongs to J by Lemma 4.13(iii). We recall Equation (D.24) below from Lemma 4.13(viii).

$$B_{u,\rho(u)} = \sum_{i=0}^{r-1} \sum_{w \in \Lambda(u)} \delta_{w,\tau_i} - \delta_{w,\mu_1(s_{t_{i+1}})}\,. \tag{D.24}$$

Since $\mathrm{tail}(u) \preceq s$, $c = \mathrm{head}(s')$, and $s' \preceq v \prec u$, then $\tau(s, c) \in \Lambda(u)$. Therefore, when $t_i = j - 1$, we know that δ_{w,τ_i} is positive for some $w \in \Lambda(u)$. On the other hand, since $\mu_1(s_{\vec{j}}) = s'$ and $s' \prec u$, we have $\delta_{w,\mu_1(s_{t_{i+1}})} = 0$ for all $w \in \Lambda(u)$ when $t_{i+1} = \vec{j}$. We conclude that the term corresponding to index i such that $t_i = j - 1$ of (D.24) adds to $B_{u,v}$ and, therefore, taking $y = \overline{y'}$ we have $F_{u,v} \geq n_y(x^n)$.

D.3 Proof of Lemma 4.22

With a slight abuse of notation, we extend the dimension of the matrices K,D,B, F to $|U \cup U'| \times |U \cup U'|$ by inserting all-zero rows and all-zero columns for pseudo-states in $U' \backslash U$.

We also extend K',D',B', F' for T' analogously. Notice that this extension does not alter the value of the multinomial factors of (4.15). Let $\{s_i\}$ and $\{s'_i\}$ be the state sequences defined by x^n in T and T' respectively. The length of a forced state sequence of a state s, defined in (4.4), is denoted ℓ_s when computed with respect to T and ℓ'_s when computed with respect to T'. Similarly, a matrix $\tilde{\Delta}^+(s)$ is denoted $\tilde{\Delta}'^+(s)$, a pseudo-state $\mu_i(s)$ is denoted $\mu'_i(s)$, and a forced state $\nu_i(s)$ is denoted $\nu'_i(s)$ when computed with respect to T'. The forced state sequence parcing of x^n with respect to T' is denoted $J'(x^n)$. The function τ' and the matrix function d' are also T' versions of τ and d. Both d and d' are extended to dimension $|U \cup U'| \times |U \cup U'|$ by inserting all-zero rows and all-zero columns.

In order to compare matrices F' and F, we start by studying the difference $\Delta K_n \triangleq K'(x^n) - K(x^n)$. Defining $k'_n = K'(x^n) - K'(x^{n-1})$, and $k_n = K(x^n) - K(x^{n-1})$, we can write ΔK_n for $n > 0$ as

$$\Delta K_n = \Delta K_{n-1} + \Delta k_n; \;\; \Delta k_n = k'_n - k_n \,. \tag{D.25}$$

Notice that while determining ΔK_n involves comparing matrices computed with respect to different context trees, namely, T and T', each term k'_n and k_n depends exclusively on a fixed context tree, which simplifies our analysis. Based on (D.25), we will show that

$$\Delta K_n = \begin{cases} \Delta K_n^{(\ell_w=1)}, & \text{if } \ell_w = 1\,, \\ \Delta K_n^{(\ell_w>1)}, & \text{if } \ell_w > 1\,, \end{cases} \tag{D.26}$$

where

$$\Delta K_n^{(\ell_w=1)} = \sum_{i=1}^{n} \left[\delta_{s_{i-1},w} \left(\mathbf{1}_{s'_{i-1},s'_i} - \mathbf{1}_{s_{i-1},s_i} \right) + \delta_{s_i,w}(1 - \delta_{s_{i-1},w}) \left(\mathbf{1}_{s'_{i-1},s'_i} - \mathbf{1}_{s_{i-1},s_i} \right) \right] , \tag{D.27}$$

and,

$$\Delta K_n^{(\ell_w>1)} = \sum_{i=1}^{n} \Bigg[\overbrace{\delta_{s_{i-1},w} \left(\mathbf{1}_{s'_{i-1},s'_i} - \mathbf{1}_{s_{i-1},s_i} \right)}^{A} \tag{D.28}$$

$$+ \sum_{b \in \mathcal{A}} \delta_{s'_i,wb} \left(\underbrace{(1 - \delta_{\ell_{s_i},\ell'_{s'_i}}) d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})}_{B} + \sum_{j=2}^{\ell_w} \underbrace{\mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b} - \mathbf{1}_{\mu_{j-1}(w),\mu_j(w)}}_{C} \right) \Bigg] .$$

When $\ell_w = 1$, (D.27) simply replaces transitions incoming to, or outgoing from w, by a transition incoming to, or outgoing from one of its descendants in T'. When $\ell_w > 1$, w is accessed always through its forced pseudo-state sequence. In this case, (D.28) "raises" the sequence of pseudo-states $\mu_1(w) \cdots \mu_{\ell_w}(w) = w$, replacing it by a sequence from $\mu_1(w)b$ to $\mu_{\ell_w}(w)b = wb$. A term labeled B of (D.28), as we shall see, has the effect of redirecting a transition of the form $s \to \nu_1(w)$, by one of the form $s \to \mu_1(w)b$. This new transition becomes part of the forced pseudo-state sequence of s'_i, which is in this case longer than that of s_i.

In the sequel, we extensively make use of the following claim.

1. Claim. *The forced pseudo-state sequences of the states of T and T' satisfy:*

- *For all $b \in \mathcal{A}$, $\ell'_{wb} \geq \ell_w$.*
- *If $s \in S_T \cap S_{T'}$, either $s = aw$ with $a \in \mathcal{A}$, or $\ell_s = \ell'_s$.*
- *If $s \in S_T$ has the form $b^{|s|}$ with $b \in \mathcal{A}$, then $\ell_s = 1$.*

Proof. Let i be an integer in the range $1 \leq i \leq \ell_w$. By the definition of ℓ_w in (4.4), there exists a state $s' \in S_T$, such that $s' \preceq (w)_i^{|w|}$. If $i > 1$, then $s' \neq w$ belongs also to $S_{T'}$ and $s' \prec (wb)_i^{|wb|}$. If $i = 1$, wb itself is a prefix of $(wb)_i^{|wb|}$. Hence, by the definition (4.4) we must have $\ell'_{wb} \geq \ell_w$. Consider now a state $s \in S_T \cap S_{T'}$ that is not of the form $s = aw$, and let i be an integer in the range $1 \leq i \leq \ell_s$. By the definition of ℓ_s in (4.4), there exists a state $s' \in S_T$, such that $s' \preceq (s)_i^{|s|}$. If $i = 1$, s itself is a state of $S_{T'}$ which is a prefix of $(s)_i^{|s|}$. If $i > 1$, $(s)_i^{|s|} \neq w$ for s is not of the form $s = aw$, and $|w| \geq \mathsf{depth}(T) - 1$. Hence, s' is also a state of T', and we conclude that $\ell'_s \geq \ell_s$. On the other hand, since T' is a refinement of T, if a state $s' \in S_{T'}$ is a prefix of $(s)_j^{|s|}$ for some j, then there exists also a state $s'' \in S_T$ such that $s'' \preceq s' \prec (s)_j^{|s|}$. Thus, also $\ell_s \geq \ell'_s$ and we must have $\ell'_s = \ell_s$. Finally, if s has the form $b^{|s|}$ with $b \in \mathcal{A}$, $\mathrm{tail}(s) \prec s$ is an internal node of T, and therefore $\ell_s = 1$ by (4.6). □

By Lemma 4.13(vi), which we recall in (D.29) below, determining k'_n and k_n amounts to determining the effect on $J(x^{n-1})$ and $J'(x^{n-1})$ of appending the symbol x_n to x^{n-1}.

$$K = \sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, s_{t_i+1}} + \tilde{\Delta}^+(s_{t_{i+1}}) \,. \tag{D.29}$$

From the definition of forced sequence parsing, it is clear that $J(x^n) \backslash J(x^{n-1}) = \{n\}$, i.e., except for the last index n, the concatenation of a new symbol to x^{n-1} does not add new indexes to J. On the contrary, some indexes of $J(x^{n-1})$ may cease to belong to the forced sequence parsing. Specifically,

$$J_n^{(-)} \triangleq J(x^{n-1}) \backslash J(x^n) = \{t_i \in J(x^{n-1}) : t_i > n - \ell_{s_n}\} \,. \tag{D.30}$$

Similarly, denoting $J'(x^n) = \{t'_i\}$ the forced sequence parsing of x^n with respect to T', we have

$$J'^{(-)}_n \triangleq J'(x^{n-1}) \backslash J'(x^n) = \{t'_i \in J'(x^{n-1}) : t'_i > n - \ell'_{s'_n}\} \,. \tag{D.31}$$

It then follows from (D.29) that

$$k_n = -A + \mathbf{1}_{s_{n-1}, s_n} \delta_{\ell_{s_n}, 1} - B + \tilde{\Delta}^+(s_n) \,, \tag{D.32}$$

where

$$A = \sum_{t_i \in J_n^{(-)}, t_i < n-1} \mathbf{1}_{s_{t_i}, s_{t_i+1}} \,, \tag{D.33}$$

and

$$B = \sum_{t_i \in J_n^{(-)}, t_i > 0} \tilde{\Delta}^+(s_{t_i}) \,, \tag{D.34}$$

and also,

$$k'_n = -A' + \mathbf{1}_{s'_{n-1},s'_n}\delta_{\ell'_{s'_n},1} - B' + \tilde{\Delta}'^{+}(s'_n)\,, \tag{D.35}$$

where

$$A' = \sum_{t'_i \in J'^{(-)}_n, t'_i < n-1} \mathbf{1}_{s'_{t'_i},s'_{t'_i+1}}\,, \tag{D.36}$$

and

$$B' = \sum_{t'_i \in J'^{(-)}_n, t'_i > 0} \tilde{\Delta}'^{+}(s'_{t'_i})\,. \tag{D.37}$$

We distinguish three cases to study (D.32) and (D.35), which lead straightforwardly to (D.27) and (D.28). Namely, for $n > 0$,

(i) For $s_n = w$, the state selected in T' has the form $s'_n = wb$ with $b \in \mathcal{A}$, $b = x_{n-|w|}$. We claim that for $\ell_w = 1$, $\Delta k_n = \mathbf{1}_{s'_{n-1},s'_n} - \mathbf{1}_{s_{n-1},s_n}$ and for $\ell_w > 1$,

$$\Delta k_n = (1 - \delta_{\ell_{s_n},\ell'_{s'_n}})d'(s'_h, s'_{h+1}) + \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b} - \mathbf{1}_{\mu_{j-1}(w),\mu_j(w)}\,, \tag{D.38}$$

where $h = n - \ell_{s_n}$.

(ii) For $s_{n-1} = w$, we claim that

$$\Delta k_n = \mathbf{1}_{s'_{n-1},s'_n} - \mathbf{1}_{s_{n-1},s_{n-1}}\,. \tag{D.39}$$

(iii) For $s_n \neq w$, $s_{n-1} \neq w$, we claim that $k'_n = k_n$.

The following claim will help on the proof of (i), but also in the rest of the proof of this Lemma.

2. Claim. *If $\ell'_{wb} > \ell_w$ for $b \in \mathcal{A}$, then, with $m = \ell'_{wb} - \ell_w$, we have $\mu'_m(wb) = tail(\mu_1(w))b \in S_{T'} \cap S_T$.*

Proof. By Claim 1, the forced state sequence of wb in T' is not shorter than that of w in T. Thus, for $0 \leq i < \ell_w$ we have $\mu'_{\ell'_{wb}-i}(wb) = \mu_{\ell_w - i}(w)b$. In particular for $i = \ell_w - 1$, $\mu'_{\ell'_{wb}-\ell_w+1}(wb) = \mu_1(w)b$. Thus, $\mu'_{m+1}(wb) = \mu_1(w)b$. Now, if $\ell'_{wb} > \ell_w$, so that $m > 0$, we know that $\mu'_m(wb)$, which equals $\text{tail}(\mu_1(w))b$, is not an internal node of T', and therefore it is not an internal node of T either. However, from (4.6), we also know that $\text{tail}(\mu_1(w))$ is an internal node of T. Thus, $\mu'_m(wb)$ must be a state of T. Moreover, since $\ell'_{wb} > 1$ we know that w is not of the form b^m, and therefore $\text{tail}(\mu_1(w))b \neq w$, and $\mu'_m(wb)$ is a state of both T and T'. Claim 2 is proved. □

We now consider (i), and we start by comparing the state sequences $\{s_j\}$ and $\{s'_j\}$ in the range $n - \ell_{s_n} < j < n$.

(i)-1. Claim. *For $n - \ell_{s_n} < j < n$, we claim that $s'_j = s_j$, $\ell'_{s'_j} = \ell_{s_j}$, and $\tilde{\Delta}'^{+}(s'_j) = \tilde{\Delta}^{+}(s_j)$.*

Proof. By Lemma 4.13(i) with $h = n$, we get that $|s_j| < |s_n|$. Thus $s_j \neq w$ and we must have $s_j = s'_j$. Moreover, since $|s_j| < |w|$, by Claim 1 the forced state sequence of s_j with respect to T and T' must coincide. Thus, $\ell'_{s'_j} = \ell_{s_j}$ and $\tilde{\Delta}'^{+}(s'_j) = \tilde{\Delta}^{+}(s_j)$. □

We next relate the forced state sequence lengths $\ell'_{s'_n}$ and ℓ_{s_n} to the sets $J'^{(-)}_n$ and $J^{(-)}_n$. Recall that by Claim 1, $\ell'_{s'_n} \geq \ell_{s_n}$.

(i)-2. Claim. *We claim that $J'^{(-)}_n \supseteq J^{(-)}_n$ with equality if and only if $\ell'_{s'_n} = \ell_{s_n}$. Moreover, if $m = \ell'_{s'_n} - \ell_{s_n} > 0$, there is a unique index $h = n - \ell_{s_n}$ in $J'^{(-)}_n \backslash J^{(-)}_n$. The state s'_h satisfies $s'_h = s_h = \mu'_m(s'_n) = tail(\mu_1(s_n))b$, and the state s_{h+1} is equal to $\nu_1(s_n)$.*

Proof. By Claim (i)-1, an index $t_i > n - \ell_{s_n}$ belongs to $J(x^{n-1})$ if and only if it also belongs to $J'(x^{n-1})$. Hence, recalling that $\ell'_{s'_n} \geq \ell_{s_n}$, we get the inclusion relation $J'^{(-)}_n \supseteq J^{(-)}_n$ directly from the definitions of $J'^{(-)}_n$ and $J^{(-)}_n$ in (D.30) and (D.31). Furthermore, if $t'_i \in J'^{(-)}_n \backslash J^{(-)}_n$, we must have $t'_i \leq n - \ell_{s_n}$. If $\ell'_{s'_n} = \ell_{s_n}$, this immediately gives $J'^{(-)}_n = J^{(-)}_n$. If $\ell'_{s'_n} > \ell_{s_n}$, consider the index $h = n - \ell_{s_n}$. By Lemma 4.13(iii), $h \in J(x^n)$ and a fortiori $h \in J(x^{n-1})$. Hence, by Claim (i)-1, also $h \in J'(x^{n-1})$ and therefore $h \in J'^{(-)}_n \backslash J^{(-)}_n$. Since $h > n - \ell'_{s'_n}$, considering the forced sequence parsing of x^n with respect to T', we have $\vec{h}' = n$ and $h'_\Delta = h - n + \ell'_{s'_n} = -\ell_{s_n} + \ell'_{s'_n} = m$. By Lemma 4.13(ii), $s'_h = \nu'_m(s'_n)$, and by Claim 2, $s'_h = s_h = \mu'_m(s'_n) = \text{tail}(\mu_1(s_n))b$. Also since $h+1 > n - \ell_{s_n}$, considering the forced sequence parsing of x^n with respect to T, we have $(\vec{h+1}) = n$ and $(h+1)_\Delta = h + 1 - n + \ell_{s_n} = 1$. Hence, by Lemma 4.13(ii), $s_{h+1} = \nu_1(s_n)$. It remains to show that h is the unique index in $J'^{(-)}_n \backslash J^{(-)}_n$. Recall that if $t'_i \in J'^{(-)}_n \backslash J^{(-)}_n$, we must have $t'_i \leq n - \ell_{s_n} = h$. Now, if $h > j > n - \ell'_{s'_n}$, then $j > n - m - \ell_{s_n}$ for $m = \ell'_{s'_n} - \ell_{s_n}$. Since by Lemma 4.7(iii), $\ell'_{s'_h} = m$, we have $j > n - \ell'_{s'_h} - \ell_{s_n} = h - \ell'_{s'_h}$. Thus, $j \notin J'(x^{n-1})$ and we conclude that h is the unique index in $J'^{(-)}_n \backslash J^{(-)}_n$. Claim (i)-2 is proved. □

When $\ell'_{s'_n} = \ell_{s_n}$, the indexes t_i and t'_i in the summations of (D.33) and (D.36), and the summations of (D.34) and (D.37), coincide by Claim (i)-2. Moreover, by Claim (i)-1 these summations include exactly the same terms, i.e., $A = A'$, $B = B'$, and we get from (D.35) and (D.32)

$$\Delta k_n = \mathbf{1}_{s'_{n-1},s'_n}\delta_{\ell'_{s'_n},1} - \mathbf{1}_{s_{n-1},s_n}\delta_{\ell_{s_n},1} + \tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}^{+}(s_n)\,, \tag{D.40}$$

or equivalently,

$$\Delta k_n = \begin{cases} \mathbf{1}_{s'_{n-1},s'_n} - \mathbf{1}_{s_{n-1},s_n} & ,\ell_w = 1\,, \\ \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b} - \mathbf{1}_{\mu_{j-1}(w),\mu_j(w)} & ,\ell_w > 1\,, \end{cases} \tag{D.41}$$

which is in agreement with (i).

When $\ell'_{s'_n} > \ell_{s_n}$, we conclude from Claim (i)-2 that $A - A' = -\mathbf{1}_{s'_h,s'_{h+1}}(1 - \delta_{h,n-1})$, and $B - B' = -\tilde{\Delta}'^{+}(s'_h)$ where $h = n - \ell_w$ is the unique index in $J'^{(-)}_n \backslash J^{(-)}_n$, and the factor

$(1-\delta_{h,n-1})$ comes from the condition $t_i < n-1$ in the summation of (D.36). Also, since $\ell'_{s'_n} > \ell_{s_n} \geq 1$, we have $\delta_{\ell'_{s'_n},1} = 0$. We then get from (D.35) and (D.32)

$$\Delta k_n = -\mathbf{1}_{s'_h,s'_{h+1}}(1-\delta_{h,n-1}) - \tilde{\Delta}'^{+}(s'_h) - \mathbf{1}_{s_{n-1},s_n}\delta_{\ell_{s_n},1} + \tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}^{+}(s_n)\,. \tag{D.42}$$

If $\ell_w = 1$, $\tilde{\Delta}^{+}(s_n)$ is null. Also $h = n-1$, and the first term of (D.42) is zero. Recalling that by Claim (i)-2, $s'_h = \mu'_m(s'_n)$ with $m = \ell'_{s'_n} - 1$, we have $\tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}'^{+}(s'_h) = \mathbf{1}_{s'_{n-1},s'_n}$. Thus, Equation (D.42) reduces to

$$\Delta k_n = \mathbf{1}_{s'_{n-1},s'_n} - \mathbf{1}_{s_{n-1},s_n}\,, \tag{D.43}$$

as claimed in (i).

If $\ell_w > 1$, the difference $\tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}'^{+}(s'_h)$ is

$$\tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}'^{+}(s'_h) = \sum_{j=m+1}^{\ell'_{wb}} \mathbf{1}_{\mu'_{j-1}(wb),\mu'_j(wb)}\,, \tag{D.44}$$

or, separating the first term and applying the definition of τ to s'_h and $a = x_{n-\ell_w+1}$,

$$\tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}'^{+}(s'_h) = \mathbf{1}_{s'_h,\tau(s'_h,a)} + \sum_{j=m+2}^{\ell'_{wb}} \mathbf{1}_{\mu'_{j-1}(wb),\mu'_j(wb)}\,, \tag{D.45}$$

which changing the summation index, and using that $m = \ell'_{s'_n} - \ell_{s_n}$, becomes

$$\tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}'^{+}(s'_h) = \mathbf{1}_{s'_h,\tau(s'_h,a)} + \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu'_{j-1+m}(wb),\mu'_{j+m}(wb)}\,. \tag{D.46}$$

Finally, we recall that $\mu'_m(wb) = \mathrm{tail}(\mu_1(w))b$. Thus, $\mu'_{m+1}(wb) = \mu_1(w)b$, and in general $\mu'_{m+j}(wb) = \mu_j(w)b$ for $1 \leq j \leq \ell_w$. Hence,

$$\tilde{\Delta}'^{+}(s'_n) - \tilde{\Delta}'^{+}(s'_h) = \mathbf{1}_{s'_h,\tau(s'_h,a)} + \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b}\,. \tag{D.47}$$

Substituting in (D.42), and recalling that for $\ell_w > 1$ we have $\delta_{h,n-1} = 0$,

$$\Delta k_n = -\mathbf{1}_{s'_h,s'_{h+1}} + \mathbf{1}_{s'_h,\tau(s'_h,a)} + \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b} - \mathbf{1}_{\mu_{j-1}(w),\mu_j(w)}\,. \tag{D.48}$$

Since $d'(s'_h, s'_{h+1}) = \mathbf{1}_{s'_h,\tau(s'_h,a)} - \mathbf{1}_{s'_h,s'_{h+1}}$, we get

$$\Delta k_n = d'(s'_h, s'_{h+1}) + \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b} - \mathbf{1}_{\mu_{j-1}(w),\mu_j(w)}\,, \tag{D.49}$$

which concludes the proof of (i).

We now consider (ii) where $s_{n-1} = w$, and therefore s'_{n-1} has the form $s'_{n-1} = wb$ with $b \in \mathcal{A}$, $b = x_{n-1-|w|}$. Suppose first that $|s_n| \leq |w|$, which implies that $\ell_{s_n} = 1$. Since $|s_n| \leq |w|$, also $|s'_n| \leq |wb|$, and we have $\ell'_{s'_n} = \ell_{s_n} = 1$. Consequently both ${J'}_n^{(-)}$, and $J_n^{(-)}$ are empty. Hence, the terms A,B,A',B', $\tilde{\Delta'}^{+}(s'_n)$, and $\tilde{\Delta}^{+}(s_n)$ in (D.35) and (D.32) are null and we get

$$\Delta k_n = \mathbf{1}_{s'_{n-1},s'_n} - \mathbf{1}_{s_{n-1},s_n} \,. \tag{D.50}$$

If on the other hand, $|s_n| > |w|$, since $|w| \geq \mathsf{depth}(T) - 1$, s_n must have the form $s_n = aw$ with $a = x_n$. Also since $s_n \neq w$, the state selected in T' coincides with s_n, i.e., $s'_n = s_n = aw$. Hence, $\mathrm{tail}(s'_n) = w$ is an internal node of T', and we have $\ell'_{s'_n} = 1$. Thus, $\tilde{\Delta'}^{+}(s'_n) = 0$ and the set ${J'}_n^{(-)}$ is empty, which causes the terms A' and B' to be null in (D.35). Hence,

$$k'_n = \mathbf{1}_{s'_{n-1},s'_n} \,. \tag{D.51}$$

In T however, $\ell_{s_n} > 1$ for we have $\mathrm{tail}(s_n) = w$. Thus, $s_{n-1} = \mu_{\ell_{s_n}-1}(s_n)$, and by Lemma 4.7(iii), we get $\ell_{s_{n-1}} = \ell_{s_n} - 1$. Now, if t_i belongs to $J(x^{n-1})$, either $t_i = n-1$, or, by definition of J, $t_i \leq n-1-\ell_{s_{n-1}} = n - \ell_{s_n}$. Thus, since the latter condition excludes t_i from $J_n^{(-)}$ by (D.30), $J_n^{(-)} = \{n-1\}$. This, yields $A = 0$ in (D.32). As for B, notice that $n-1$ must be strictly positive for the initial state s_0 is of maximal depth in T, but $|s_{n-1}| < |s_n|$. Thus, from (D.34), we get $B = \tilde{\Delta}^{+}(s_{n-1})$. Replacing in (D.32),

$$k_n = -\tilde{\Delta}^{+}(s_{n-1}) + \tilde{\Delta}^{+}(s_n) \,, \tag{D.52}$$

which, since $s_{n-1} = \mu_{\ell_{s_n}-1}(s_n)$, reduces to $\mathbf{1}_{s_{n-1},s_n}$. This, together with (D.51), gives the same expression as in (D.50) for Δk_n, and concludes the proof of the claim in (ii).

We now consider (iii), where $s_{n-1} \neq w$ and $s_n \neq w$. Since $s_{n-1} \neq w$, s_n is not of the form aw with $a \in \mathcal{A}$. Thus, by Claim 1, $\ell'_{s'_n} = \ell_{s_n}$. Furthermore, by Lemma 4.13(i) with n in the role of h, we have $|s_j| < |s_n|$ for all j in the range $n - \ell_{s_n} < j < n$. Thus, $s_j = s'_j$ and by Claim 1 $\ell'_{s'_j} = \ell_{s_j}$, and therefore and $\tilde{\Delta'}^{+}(s'_j) = \tilde{\Delta}^{+}(s_j)$. Then, for all $n - \ell_{s_n} < j < n$, we have by the definition of forced sequence parsing that $j \in J(x^{n-1})$ if and only if $j \in J'(x^{n-1})$. Hence, the sets ${J'}_n^{(-)}$ and $J_n^{(-)}$ are equal yielding exactly the same terms in (D.36) and (D.33), and in (D.37) and (D.34). From (D.35) and (D.32), recalling that $\ell'_{s'_n} = \ell_{s_n}$ and therefore $\tilde{\Delta'}^{+}(s'_n) = \tilde{\Delta}^{+}(s_n)$, we get $k'_n = k_n$. This concludes the proof of (iii).

To derive (D.27) and (D.28) from (i), (ii), and (iii), we notice that $\Delta K_0 = 0$, and therefore $\Delta K_n = \sum_{i=1}^{n} \Delta K_i - \Delta K_{i-1}$. Thus, by (D.25), $\Delta K_n = \sum_{i=1}^{n} \Delta k_i$, from which (D.27) and (D.28) follow, by (i), (ii), and (iii).

We now study $\Delta D_n \triangleq D'(x^n) - D(x^n)$, based on ΔK_n. From the definition of the matrix D, we get

$$\Delta D_n = \Delta K_n + \sum_{u,v \in S_{T'}} \left(K'_n\right)_{u,v} d'(u,v) - \sum_{u,v \in S_T} (K_n)_{u,v}\, d(u,v) \,. \tag{D.53}$$

Since $|w| \geq \mathsf{depth}(T) - 1$, no pseudo-state of T is a proper descendant of w, and for the same reason, no pseudo-state of T' is a proper descendant of wb for any $b \in \mathcal{A}$. Thus, we can

discard all terms with $v = w$ in the first summation, as we know that $d(u, v) = 0$ in these cases. Similarly, we can discard all terms with $v = wb$, $b \in \mathcal{A}$, in the second summation, getting

$$\Delta D_n = \Delta K_n + \sum_{u,v \in S_{T'}, w \not\preceq v} \left(K'_n\right)_{u,v} d'(u, v) - \sum_{u,v \in S_T, v \neq w} (K_n)_{u,v} \, d(u, v) \,. \tag{D.54}$$

We can further simplify (D.54) by grouping in a single summation the terms where $d'(u, v) = d(u, v)$. By Claim 1, if $s \in S_T \cap S_{T'}$, then either $s = aw$ with $a \in \mathcal{A}$, in which case $\ell_s > \ell'_s$, or $\ell_s = \ell'_s$. Thus, the set of pseudo-states that are new in T' with respect to T come from the new states wb with $b \in \mathcal{A}$,

$$U' \backslash U \subset \bigcup_{b \in \mathcal{A}, 1 \leq i \leq \ell'_{wb}} \mu'_i(wb) \,. \tag{D.55}$$

Now, if $\ell'_{wb} > \ell_w$ for $b \in \mathcal{A}$, then, with $m = \ell'_{wb} - \ell_w$, we have $\mu'_m(wb) \in S_{T'} \cap S_T$ by Claim 2. Thus, by Lemma 4.7(iii), with $s' = \mu'_m(wb)$, $\mu'_i(wb) = \mu'_i(s')$ for all $1 \leq i \leq m$. Furthermore, since $m < \ell'_{wb}$, $|s'| < |wb|$, and therefore s' is not of the form aw. Hence, by Claim 1, $\ell_{s'} = \ell'_{s'}$. Thus, $\mu'_i(wb) = \mu'_i(s') = \mu_i(s') \in U$ for all $1 \leq i \leq m$, and (D.55) becomes,

$$U' \backslash U \subset \bigcup_{b \in \mathcal{A}, \ell'_{wb} - \ell_w < i \leq \ell'_{wb}} \mu'_i(wb) \,. \tag{D.56}$$

Notice that (D.56) is still valid when $\ell'_{wb} = \ell_w$. Changing the index variable,

$$U' \backslash U \subset \bigcup_{b \in \mathcal{A}, 1 \leq i \leq \ell_w} \mu'_{\ell'_{wb} - \ell_w + i}(wb) \,, \tag{D.57}$$

or,

$$U' \backslash U \subset \bigcup_{b \in \mathcal{A}, 0 \leq i < \ell_w} \mu'_{\ell'_{wb} - i}(wb) \,. \tag{D.58}$$

Since $\mu'_{\ell'_{wb} - i}(wb) = \mu_{\ell_w - i}(w)b$, we get,

$$U' \backslash U \subset \bigcup_{b \in \mathcal{A}, 1 \leq i \leq \ell_w} \mu_i(w)b \,. \tag{D.59}$$

We now consider the pseudo-states of U that are no longer pseudo-states in T'. This includes the pseudo-states $\mu_1(w) \cdots \mu_{\ell_w}(w)$, which may not belong to U' as $w \notin S_{T'}$, and also pseudo-states $\mu_i(s)$ of states $s \in S_{T'} \cap S_T$ with $\ell_s > \ell'_s$. By Claim 1, only states of the form $s = aw$ have longer forced pseudo-state sequence in T than in T'. In this case, we have $w = \mu_{\ell_s - 1}(s)$ and by Lemma 4.7(iii), $\mu_i(s) = \mu_i(w)$ for all $1 \leq i \leq \ell_w$. Hence, since $\mu_{\ell_w + 1}(s) = s \in U'$,

$$U \backslash U' \subset \bigcup_{1 \leq i \leq \ell_w} \mu_i(w) \,. \tag{D.60}$$

We next characterize the cases in which τ and τ' differ. We define the integer m as the unique index $1 \leq m < \ell_w - 1$ such that $\mu_m(w) \in S_T$, and $\mu_{m+1}(w) \notin U'$ if such index exists, and $m = 0$ otherwise. Notice that, if $t = \mu_j(w) \in S_T$ with $1 \leq j < \ell_w$, then $\ell_t = j$ and $\mu_i(w) = \mu_i(t)$ for all $i < j$ by Lemma 4.7 Part (iii). Since $|\mu_j(w)| < |w|$, t is also a state of

T' and its forced pseudo-state sequences coincide in both context trees by Claim 1. Hence $\mu_i(w) = \mu'_i(t) \in U'$ for all $i < j$, and therefore there can exist at most one index $1 \le j < \ell_w$ such that $\mu_j(w) \in S_T$, and $\mu_{j+1}(w) \notin U'$.

3. Claim. *Let $s \in S_T$, and $s' \in S_{T'}$, such that $s \preceq s'$, and let also a be a symbol of $\mathcal{A}$ such that as is not an internal node of T, and as' is not an internal node of T'. Then, if $\tau'(s', a) \neq \tau(s, a)$, one of the following holds:*

- *$\tau(s, a) = \mu_1(w)$, and $\tau'(s', a) = \mu_1(w)b$ where $b = (as')_{|\mu_1(w)|+1}$. In this case, $\mu_1(w)b \notin U$.*
- *$m > 0$, $s = s' = \mu_m(w)$, and $\tau(s, a) = \mu_{m+1}(w) = as$.*

Proof. Let $u = \tau(s, a)$, and $u' = \tau'(s', a)$, with $u \neq u'$. We first show that $u' \preceq as$. Otherwise, as $u' \preceq as'$ by the definition of τ, s and s' must be different, and therefore $s = w$, and $s' = wc$ with $c \in \mathcal{A}$. Thus, $u' \preceq awc$ but $u' \not\preceq aw$, and we must have $u' = awc$, which is a contradiction for $|w| \ge \mathsf{depth}(T) - 1$. We conclude that $u' \preceq as$. Hence, both u and u' are prefixes of as, and therefore they are also prefixes of as'. Since $u \neq u'$, either $u \prec u'$ in which case we know by the definition of τ that $u' \in U' \backslash U$, or $u' \prec u$ in which case $u \in U \backslash U'$. If $u \prec u'$, $u' \in U' \backslash U$, and u' has the form $u' = \mu_i(w)c$ by (D.59). In this case, $\mu_i(w)$ is the longest prefix of u' in U, and therefore $u = \mu_i(w)$. Since $u' \preceq as$, $\mathrm{tail}(u') \preceq s$. Thus, $\mathrm{tail}(\mu_i(w)c) \preceq s$, and we must have $i = 1$ by (4.6). The symbol c must be equal to $b = (as')_{|\mu_1(w)|+1}$ for $u' = \mu_1(w)c \preceq as'$. If on the other hand $u' \prec u$, $u \in U \backslash U'$, and u has the form $u = \mu_i(w)$ by (D.60). Since $\mu_i(w)c \in U'$ for all $c \in \mathcal{A}$, but $\tau'(s', a) \prec \mu_i(w)$, then $\mu_i(w)c \not\preceq as'$ for all $c \in \mathcal{A}$. Hence, since $\mu_i(w) \preceq as \preceq as'$, we must have $\mu_i(w) = as$, and therefore $s = \mu_{i-1}(w)$. Furthermore, as $|\mu_{i-1}(w)| < |w|$, $s = s'$. Since as', which is equal to $\mu_i(w)$, is not an internal node of T' by the assumptions, $i < \ell_w$. Thus, we have an index $j = i - 1$, such that $1 \le j < \ell_w - 1$, $\mu_j(w) \in S_T$, and $\mu_{j+1}(w) \notin U'$, and therefore $m = j$. The claim is proved. □

We define the set $Z_m \subset S_T \times S_T$ as empty if $m = 0$, and $\{(\mu_m(w), \nu_{m+1}(w))\}$ otherwise. We also define $Z_1 = \{(u, \nu_1(w)) \in S_T \times S_T : u \neq w, \mathrm{tail}(\mu_1(w)) \prec u\}$, and take $Z = Z_m \cup Z_1$. Notice that, if $\tau'(u, a) \neq \tau(u, a)$ for some $u \in S_T \cap S_{T'}$ and $a \in \mathcal{A}$ such that au is not an internal node of T, nor of T', then there exists a unique state $v \in S_T \cap S_{T'}$, such that $v \preceq au$, and by Claim 3 $(u, v) \in Z$. On the other hand, if $u \in S_T \cap S_{T'}$ and $a \in \mathcal{A}$ are such that au is not an internal node of T, but au is an internal node of T', we must have $au = w$. Notice that in this case, the pairs u, w and u, wb with $b \in \mathcal{A}$ are excluded from the summations in (D.54). We then define $W = \{(u, v) \in S_T \times S_T : u \neq w, v \neq w, (u, v) \notin Z\}$, and rewrite (D.54) as (D.61) below,

$$\begin{aligned} \Delta D_n \;=\; & \Delta K_n + \sum_{(u,v)\in W} (\Delta K_n)_{u,v}\, d(u, v) \\ & + (1 - \delta_{\ell_w,1}) \sum_{(u,v)\in Z} \Big(\big(K'_n\big)_{u,v}\, d'(u, v) - (K_n)_{u,v}\, d(u, v) \Big) \\ & + \sum_{b\in\mathcal{A}, v\in S_{T'}} \big(K'_n\big)_{wb,v}\, d'(wb, v) - \sum_{v\in S_T} (K_n)_{w,v}\, d(w, v)\,, \end{aligned} \tag{D.61}$$

where the factor $(1-\delta_{\ell_w,1})$ excludes the case $\nu_1(w)=w$, assuring that $v\neq w$ in $(u,v)\in Z$.

We show next that the first summation in (D.61) is null. From (D.27) and (D.28), it is clear that all nonzero elements $(\Delta K_n)_{u,v}$ with $u,v\in S_T$ and $u\neq w, v\neq w$, come from terms of the form $d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})$ or $\mathbf{1}_{\mu_{j-1}(w),\mu_j(w)}$. In the latter case, if $\mu_{j-1}(w)$, and $\mu_j(w)$ are both states of T, then $d(\mu_{j-1}(w),\mu_j(w))=0$ by definition. On the other hand, when $s'_i=wb$ and $\ell'_{s'_i}>\ell_{s_i}$, so that a term $d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})$ arises in (D.28), we have by Claim (i)-2 that $\mathrm{tail}(\mu_1(w))\prec s_{i-\ell_{s_i}}=s'_{i-\ell_{s_i}}$ and $s_{i-\ell_{s_i}+1}=\nu_1(w)$. Thus, the pair $(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})$ does not belong to W, for either $s'_{i-\ell_{s_i}+1}=s_{i-\ell_{s_i}+1}=\nu_1(w)$ in which case the pair belongs to Z_1, or $s'_{i-\ell_{s_i}+1}\neq s_{i-\ell_{s_i}+1}$, in which case $s'_{i-\ell_{s_i}+1}\notin S_T$. We conclude that

$$\sum_{(u,v)\in W}(\Delta K_n)_{u,v}\,d(u,v)=0\,. \tag{D.62}$$

We now analyze the summation in $(u,v)\in Z$ of (D.61). Namely,

$$\Delta_1=\sum_{(u,v)\in Z}\big(K'_n\big)_{u,v}\,d'(u,v)-\sum_{(u,v)\in Z}(K_n)_{u,v}\,d(u,v)\,, \tag{D.63}$$

where we are assuming $\ell_w>1$. We can also write (D.63) as

$$\Delta_1=\sum_{(u,v)\in Z}(K_n)_{u,v}\big(d'(u,v)-d(u,v)\big)+\sum_{(u,v)\in Z}(\Delta K_n)_{u,v}\,d'(u,v)\,. \tag{D.64}$$

Consider a pair of states $(u,v)\in Z$, such that $(\Delta K_n)_{u,v}\neq 0$. If $(u,v)\in Z_m$, $\mu_{m+1}(w)\notin U'$, and since $\mu_{m+1}(w)\neq w$ for $m+1<\ell_w$ by the definition of m, $\mu_{m+1}(w)\notin S_T$. Thus, $|\nu_{m+1}(w)|\leq|\mu_m(w)|$, and therefore entry $(\Delta K_n)_{u,v}$ is not affected by terms labeled C of (D.28). Similarly, if $(u,v)\in Z_1$, $\mathrm{tail}(\mu_1(w))\prec u$ and therefore $|v|=|\nu_1(w)|\leq|u|$. Thus, entry $(\Delta K_n)_{u,v}$ is not affected by terms labeled C of (D.28) either in this case. Hence, since terms labeled A do only affect entries $(\Delta K_n)_{u',v'}$ with $u'\preceq w$, we see that $(\Delta K_n)_{u,v}$ comes from the negative part, $-\mathbf{1}_{s'_{i-\ell_{s_i}},s'_{i-\ell_{s_i}+1}}$, of terms of (D.28) of the form $d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})$ with $s'_i=wb$ and $\ell'_{s'_i}>\ell_{s_i}$. Thus, for $(u,v)\in Z$,

$$(\Delta K_n)_{u,v}=-\sum_{i=1}^{n}\sum_{b\in\mathcal{A}}\delta_{s'_i,wb}(1-\delta_{\ell_{s_i},\ell'_{s'_i}})\delta_{u,s'_{i-\ell_{s_i}}}\delta_{v,s'_{i-\ell_{s_i}+1}}\,. \tag{D.65}$$

Hence, the last summation of (D.64) is,

$$-\sum_{(u,v)\in Z}\sum_{i=1}^{n}\sum_{b\in\mathcal{A}}\delta_{s'_i,wb}(1-\delta_{\ell_{s_i},\ell'_{s'_i}})\delta_{u,s'_{i-\ell_{s_i}}}\delta_{v,s'_{i-\ell_{s_i}+1}}d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})\,,$$

or,

$$-\sum_{i=1}^{n}\sum_{b\in\mathcal{A}}\delta_{s'_i,wb}(1-\delta_{\ell_{s_i},\ell'_{s'_i}})d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})\sum_{(u,v)\in Z}\delta_{u,s'_{i-\ell_{s_i}}}\delta_{v,s'_{i-\ell_{s_i}+1}}\,. \tag{D.66}$$

Now, by Claim (i)-2, when $s'_i=wb$ and $\ell'_{s'_i}>\ell_{s_i}$, we have that $\mathrm{tail}(\mu_1(w))b=s_{i-\ell_{s_i}}=s'_{i-\ell_{s_i}}$, and $s_{i-\ell_{s_i}+1}=\nu_1(w)$. If $s_{i-\ell_{s_i}+1}\neq s'_{i-\ell_{s_i}+1}$, then $s_{i-\ell_{s_i}+1}=w$. Thus, $\nu_1(w)=w$ and

therefore $\ell_w = 1$. Hence, in this case we have $s'_{i-1} = \mathrm{tail}(w)b = \mathrm{tail}(s'_i)$, and therefore $d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})$ is null. If on the other hand $s'_{i-\ell_{s_i}+1} = s_{i-\ell_{s_i}+1} = \nu_1(w)$, then $(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})$ belongs to Z_1, since also $\mathrm{tail}(\mu_1(w)) \prec \mathrm{tail}(\mu_1(w))b = s'_{i-\ell_{s_i}}$. We conclude that for every i such that $s'_i = wb$ and $\ell'_{s'_i} > \ell_{s_i}$, either $\sum_{(u,v)\in Z} \delta_{u,s'_{i-\ell_{s_i}}} \delta_{v,s'_{i-\ell_{s_i}+1}} = 1$, or $d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1}) = 0$. Thus, (D.66) gives for the last summation of (D.64),

$$-\sum_{i=1}^{n}\sum_{b\in\mathcal{A}} \delta_{s'_i,wb}(1-\delta_{\ell_{s_i},\ell'_{s'_i}}) d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1})\,. \tag{D.67}$$

As for the first summation of (D.64), we can write it as

$$\sum_{(u,v)\in Z} (K_n)_{u,v} \left(\mathbf{1}_{u,\tau'(u,\mathsf{head}(v))} - \mathbf{1}_{u,\tau(u,\mathsf{head}(v))}\right)\,. \tag{D.68}$$

Substituting (D.67) and (D.68) in (D.64), and this together with (D.62) back in (D.61), we get

$$\begin{aligned}\Delta D_n \;=\; \Delta K_n + (1-\delta_{\ell_w,1}) \Bigg[&\sum_{(u,v)\in Z} (K_n)_{u,v} \left(\mathbf{1}_{u,\tau'(u,\mathsf{head}(v))} - \mathbf{1}_{u,\tau(u,\mathsf{head}(v))}\right) \\ &- \sum_{i=1}^{n}\sum_{b\in\mathcal{A}} \delta_{s'_i,wb}(1-\delta_{\ell_{s_i},\ell'_{s'_i}}) d'(s'_{i-\ell_{s_i}}, s'_{i-\ell_{s_i}+1}) \Bigg] \\ &+ \sum_{b\in\mathcal{A},v\in S_{T'}} \left(K'_n\right)_{wb,v} d'(wb,v) - \sum_{v\in S_T} (K_n)_{w,v}\, d(w,v)\,.\end{aligned} \tag{D.69}$$

When $\ell_w = 1$, (D.69) with (D.27) yield

$$\begin{aligned}\Delta D_n^{(\ell_w=1)} \;=\; &\sum_{i=1}^{n} \delta_{s_{i-1},w} \left(\mathbf{1}_{s'_{i-1},s'_i} - \mathbf{1}_{s_{i-1},s_i}\right) \\ &+ \sum_{i=1}^{n} \delta_{s_i,w}(1-\delta_{s_{i-1},w}) \left(\mathbf{1}_{s'_{i-1},s'_i} - \mathbf{1}_{s_{i-1},s_i}\right) \\ &+ \sum_{b\in\mathcal{A},v\in S_{T'}} \left(K'_n\right)_{wb,v} d'(wb,v) - \sum_{v\in S_T} (K_n)_{w,v}\, d(w,v)\,,\end{aligned} \tag{D.70}$$

and when $\ell_w > 1$, (D.69) with (D.28) yield

$$\begin{aligned}\Delta D_n^{(\ell_w>1)} \;=\; &\sum_{i=1}^{n} \delta_{s_{i-1},w} \left(\mathbf{1}_{s'_{i-1},s'_i} - \mathbf{1}_{s_{i-1},s_i}\right) \\ &+ \sum_{i=1}^{n}\sum_{b\in\mathcal{A}} \delta_{s'_i,wb} \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b} - \mathbf{1}_{\mu_{j-1}(w),\mu_j(w)} \\ &+ \sum_{(u,v)\in Z} (K_n)_{u,v} \left(\mathbf{1}_{u,\tau'(u,\mathsf{head}(v))} - \mathbf{1}_{u,\tau(u,\mathsf{head}(v))}\right) \\ &+ \sum_{b\in\mathcal{A},v\in S_{T'}} \left(K'_n\right)_{wb,v} d'(wb,v) - \sum_{v\in S_T} (K_n)_{w,v}\, d(w,v)\,.\end{aligned} \tag{D.71}$$

We now show that the last two summations of (D.70) have the effect of "redirecting" the destination of transitions (s'_{i-1}, s'_i) and (s_{i-1}, s_i) of the first summation to $\tau'(wb, x_i)$ and $\tau(w, x_i)$ respectively. Similarly, the last two summations of (D.71) have the effect of "redirecting" the destination of transitions (s'_{i-1}, s'_i) and (s_{i-1}, s_i) of the first summation to $\tau'(wb, x_i)$ and $\tau(w, x_i)$ respectively. We make use of the following claim.

4. Claim. *Let u be a state of an arbitrary context tree $\tilde{T}$ such that $|u| \geq \mathsf{depth}(\tilde{T}) - 1$. We claim that $\tilde{K}_{u*} = \tilde{N}_{u*}$ and for every state v of $\tilde{T}$, $\tilde{K}_{u,v} = \tilde{N}_{u,v}$, where we are using the notation $\tilde{K}$ and $\tilde{N}$ to emphasize that the matrices are calculated with respect to $\tilde{T}$. If also $\tilde{\ell}_u > 1$, then $\tilde{F}_{*u} = \tilde{F}_{\mathsf{tail}(u),u} = \tilde{N}_{*u}$.*

Proof. Since $|u| \geq \mathsf{depth}(\tilde{T}) - 1$, u can only belong to the forced state sequence of a state of the form $s = au$ with $a \in \mathcal{A}$. In this case we have $\mu_{\ell_s - 1}(s) = \nu_{\ell_s - 1}(s) = u$. Thus, for all states t, v of $\tilde{T}$, we have $\tilde{\Delta}^-(t)_{u,v} = \tilde{\Delta}^+(t)_{u,v}$, which are equal to one if t is of the form au, and zero otherwise. The first part of the claim then follows from parts (v) and (vi) of Lemma 4.13, which yield $\tilde{N}^{(\mu)} - \tilde{K} = \sum_{i=0}^{r-1} \tilde{\Delta}^-(s_{t_{i+1}}) - \tilde{\Delta}^+(s_{t_{i+1}})$. If $\tilde{\ell}_u > 1$, $\tilde{\Delta}^+(u)_{*u} = \tilde{\Delta}^+(u)_{\mathsf{tail}(u),u} = 1$. Also $\tilde{\Delta}^+(t)_{*u} = 0$ for all states t that are not of the form au, and $\tilde{\Delta}^+(t)_{*u} = \tilde{\Delta}^+(t)_{\mathsf{tail}(u),u} = 1$ if $t = au$. Now, since $|u| \geq \mathsf{depth}(\tilde{T}) - 1$, for all j such that $1 \leq j \leq n$ and $\tilde{s}_j = u$, either $j \in \tilde{J}$, or $\tilde{s}_{j+1} = x_{j+1}u$, in which case $j + 1 \in \tilde{J}$ and $j \notin \tilde{J}$. Hence,

$$\left(\sum_{i=0}^{r-1} \tilde{\Delta}^+(s_{t_{i+1}})\right)_{\mathsf{tail}(u),u} = \left(\sum_{i=0}^{r-1} \tilde{\Delta}^+(s_{t_{i+1}})\right)_{*u} = \tilde{N}_{*u}\,. \tag{D.72}$$

Since $|u| \geq \mathsf{depth}(\tilde{T}) - 1$, no pseudo-state of $\tilde{T}$ is a descendant of u, and therefore $\tilde{B}_{*u} = 0$. The claim is proved if we show that $\tilde{D}_{*u} = \tilde{D}_{\mathsf{tail}(u),u} = \tilde{N}_{*u}$. Thus, by Lemma 4.13(vii), and (D.72), we only need to show that $\left(\sum_{i=0}^{r-1} \mathbf{1}_{s_{t_i}, \tau_i}\right)_{*u} = 0$. Now, if $\tau_i = u$, then $\tilde{s}_{t_i+1} = u$ for both τ_i and $\tilde{s}_{t_i+1}$ are prefixes of $\overline{x^{t_i+1}}$. But since $\tilde{\ell}_u > 1$, this implies that $t_i \notin \tilde{J}$, which is a contradiction. □

By Claim 4, we have

$$\sum_{v \in S_T} (K_n)_{w,v}\, d(w, v) = \sum_{i=1}^{n} \delta_{s_{i-1},w} d(s_{i-1}, s_i)\,, \tag{D.73}$$

and also

$$\sum_{b \in \mathcal{A}, v \in S_{T'}} \left(K'_n\right)_{wb,v} d'(wb, v) = \sum_{i=1}^{n} \delta_{s_{i-1},w} d'(s'_{i-1}, s'_i)\,. \tag{D.74}$$

Thus, (D.70) becomes

$$\begin{aligned} \Delta D_n^{(\ell_w = 1)} \;=\; & \sum_{i=1}^{n} \delta_{s_{i-1},w} \left(\mathbf{1}_{s'_{i-1}, \tau'(s'_{i-1}, x_i)} - \mathbf{1}_{s_{i-1}, \tau(s_{i-1}, x_i)}\right) \\ & + \sum_{i=1}^{n} \delta_{s_i,w}(1 - \delta_{s_{i-1},w}) \left(\mathbf{1}_{s'_{i-1}, s'_i} - \mathbf{1}_{s_{i-1}, s_i}\right)\,, \end{aligned} \tag{D.75}$$

and (D.71) becomes

$$\begin{aligned}\Delta D_n^{(\ell_w>1)} &= \sum_{i=1}^{n} \delta_{s_{i-1},w}\left(\mathbf{1}_{s'_{i-1},\tau'(s'_{i-1},x_i)} - \mathbf{1}_{s_{i-1},\tau(s_{i-1},x_i)}\right) \\ &\quad + \sum_{i=1}^{n}\sum_{b\in\mathcal{A}} \delta_{s'_i,wb} \sum_{j=2}^{\ell_w} \mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b} - \mathbf{1}_{\mu_{j-1}(w),\mu_j(w)} \\ &\quad + \sum_{(u,v)\in Z} (K_n)_{u,v}\left(\mathbf{1}_{u,\tau'(u,\mathsf{head}(v))} - \mathbf{1}_{u,\tau(u,\mathsf{head}(v))}\right) . \end{aligned} \tag{D.76}$$

We now derive (4.27) by analyzing (D.75). Since $\ell_w = 1$, by (D.60) and (D.59), all pseudo-states u in $U \cup U'$, such that $w \not\preceq u$, belong simultaneously to U and U'. Notice that the first summation affects exclusively transitions departing from w in D and, from $\{wb : b \in \mathcal{A}\}$ in D'. When $(1 - \delta_{s_{i-1},w}) = 1$, $s'_{i-1} = s_{i-1}$, and therefore each term of the second summation adds and subtracts a transition from the same state. Thus, $D_{u*} = D'_{u*}$ for all u such that $w \not\preceq u$. On the other hand, the second summation affects exclusively transitions arriving at w in D and, at $\{wb : b \in \mathcal{A}\}$ in D'. Since $\ell_w = 1$, by Claim 3 we see that $\tau'(s'_{i-1}, x_i)$ and $\tau(s_{i-1}, x_i)$ coincide except when $\tau(s_{i-1}, x_i) = w$. Thus, the first summation does not change the total number of incoming transition to pseudo-states that do not belong to $\{w\} \cup \{wb : b \in \mathcal{A}\}$. Hence, also $D_{*u} = D'_{*u}$ for all u such that $w \not\preceq u$. As a consequence, since no pseudo-states descend from w in T nor in T' for $|w| \geq \mathsf{depth}(T) - 1$, we have by the definition of B that $B_{u,\rho(u)} = B'_{u,\rho'(u)}$ for all u in $U\backslash S_T$, or equivalently $U'\backslash S_{T'}$. Now, consider a pseudo-state u such that $w \not\preceq u$. Only the second summation may affect transitions departing from u. Recall that when $(1 - \delta_{s_{i-1},w}) = 1$, $s'_{i-1} = s_{i-1}$. Thus, outgoing transitions from u are only affected by terms with index i such that $s_i = w$, $(1 - \delta_{s_{i-1},w}) = 1$, and $u = s'_{i-1} = s_{i-1}$. Each of these terms subtracts $\mathbf{1}_{u,w}$, and adds $\mathbf{1}_{u,wb}$, where $b = (\overline{s_0 x})_{i-|w|}$. Furthermore, since $\ell_w = 1$, $\mathrm{tail}(w)$ is an internal node of T, and therefore $\mathrm{tail}(w) \prec u \preceq \overline{x^{i-1}} s_0$. Thus, the symbol b is uniquely determined by u as $b = b_u = u_{|w|}$. Hence, each term that affects outgoing transitions from u subtracts $\mathbf{1}_{u,w}$, and adds $\mathbf{1}_{u,wb_u}$. Therefore, $\left(\Delta D_n^{(\ell_w=1)}\right)_{u,wb_u} = -\left(\Delta D_n^{(\ell_w=1)}\right)_{u,w}$. Now, since $w \notin U'$, $D'_{u,w} = 0$, and therefore $D_{u,w} = -\left(\Delta D_n^{(\ell_w=1)}\right)_{u,w}$. Analogously, since $wb_u \notin U$, $D_{u,wb_u} = 0$, and therefore $D'_{u,wb_u} = \left(\Delta D_n^{(\ell_w=1)}\right)_{u,wb_u}$. Hence, we have $D'_{u,w} = D_{u,wb_u} = 0$, $D_{u,w} = D'_{u,wb_u}$, and $D_{u,v} = D'_{u,v}$ for all v such that $v \notin \{w, wb_u\}$. This, together with the fact that $B_{u,\rho(u)} = B'_{u,\rho'(u)}$ when u belongs to $U\backslash S_T$, or equivalently $U'\backslash S_{T'}$, gives

$$\frac{F'_{u*}! \,/ \prod_v F'_{u,v}!}{F_{u*}! \,/ \prod_v F_{u,v}!} = 1\,, \text{ for all } u \text{ such that } w \not\preceq u\,.$$

We now consider outgoing transitions from pseudo-states u in $\{w\} \cup \{wb : b \in \mathcal{A}\}$, which are affected exclusively by the first summation. Notice that $B_{u*} = B'_{u*} = 0$ for all such u. Since $wb \notin U$ for any $b \in \mathcal{A}$, $F_{wb,u} = 0$ for all $u \in U \cup U'$, and therefore $F'_{wb,u} = \left(\Delta D_n^{(\ell_w=1)}\right)_{wb,u}$. By (D.75), this is equal to $\sum_{i=1}^{n} \delta_{s'_{i-1},wb}\delta_{u,\tau'(wb,x_i)}$. Thus, with $a = \mathrm{head}(u)$, $F'_{wb,u} = n'^{(a)}_{wb}$ if $\tau'(wb, a) = u$, and $F'_{wb,u} = 0$ otherwise (notice that $\tau'(wb, a)$ is well defined for all symbols a,

as $|wb| = \mathsf{depth}(T')$). Hence, since by the definition of τ, $\tau'(wb, a) = \tau'(wb, c)$ if and only if $a = c$, we conclude that

$$\frac{F'_{wb*}!}{\prod_v F'_{wb,v}!} = \frac{N'_{wb*}!}{\prod_{a\in\mathcal{A}} n'^{(a)}_{wb}!}\,.$$

Similarly, since $w \notin U'$, $F'_{w,u} = 0$ for all $u \in U \cup U'$, and therefore $F_{w,u} = -\left(\Delta D_n^{(\ell_w=1)}\right)_{w,u}$. By (D.75), this is equal to $-\sum_{i=1}^{n} \delta_{s_{i-1},w}\delta_{u,\tau(w,x_i)}$. Thus, with $a = \mathrm{head}(u)$, $F_{w,u} = n_w^{(a)}$ if $\tau(w, a) = u$, and $F_{w,u} = 0$ otherwise (notice that $\tau(w, a)$ is well defined for all symbols a, as $|aw| \geq \mathsf{depth}(T)$). Hence,

$$\frac{F_{w*}!}{\prod_v F_{w,v}!} = \frac{N_{w*}!}{\prod_{a\in\mathcal{A}} n_w^{(a)}!}\,,$$

which concludes the proof of (4.27).

We now study (D.76) to derive (4.28). Looking at the three summations of (D.76), we see that $D_{u,v}$ and $D'_{u,v}$ may differ only for $u \in \{w\} \cup \{wb : b \in \mathcal{A}\}$, for $u \in S_1 \triangleq \{u : (u, v) \in Z\}$, and for $u \in U_w \cup U'_w$, where $U_w = \{\mu_i(w) : 1 \leq i < \ell_w\}$, and $U'_w = \{\mu_i(w)b : 1 \leq i < \ell_w, b \in \mathcal{A}\}$. The set S_1 has, by definition, null intersection with $\{w\} \cup \{wb : b \in \mathcal{A}\}$. We claim that also $\{w\} \cup \{wb : b \in \mathcal{A}\}$ has empty intersection with $U_w \cup U'_w$.

5. Claim. *The set $\{w\} \cup \{wb : b \in \mathcal{A}\}$ has empty intersection with $U_w \cup U'_w$.*

Proof. Let u be an element of $U_w \cup U'_w$. By definition, $|u| \leq w$ with equality only if $u = \mu_{\ell_w-1}(w)b$ for some $b \in \mathcal{A}$. Hence, if the intersection is not null, we must have $w = \mu_{\ell_w-1}(w)b$. But in this case w is of the form $w = b^{|w|}$, and consequently $\ell_w = 1$. We conclude that the sets do not intersect, as claimed. □

As a consequence of the last claim, only the first summation of (D.76) affects outgoing transitions from $u \in \{w\}\cup\{wb : b \in \mathcal{A}\}$, and we observe that it gives rise to the factor $\Pi^{(\ell_w=1)}$ exactly as in the case $\ell_w = 1$. In the sequel we study outgoing transitions from pseudo-states $u \notin \{w\} \cup \{wb : b \in \mathcal{A}\}$, which are not affected by the first summation of (D.76).

We claim that

$$\frac{F'_{u*}!\,/\prod_v F'_{u,v}!}{F_{u*}!\,/\prod_v F_{u,v}!} = 1 \text{ for every } u \in S_1\backslash U_w\,. \tag{D.77}$$

As $u \in S_1$, u is a state of T and therefore $B_{u*} = B'_{u*} = 0$, i.e., we only need to take care of the matrix entries in D and D'. For the same reason, u is not of the form $\mu_i(w)b$. Since also $u \notin U_w$, the difference between $F'_{u,v}$ and $F_{u,v}$ for any $v \in U \cup U'$ is determined by the last summation of (D.76). Recalling that $u \neq \mu_m(w)$ for $u \notin U_w$, it then follows that $F'_{u,v} = F_{u,v}$ for all v, except possibly for $v \in \{\tau(u, a), \tau'(u, a)\}$, where $a = \mathrm{head}(\nu_1(w))$, and $\tau(u, a) \neq \tau'(u, a)$. By Claim 3, $\tau(u, a) = \mu_1(w)$, and $\tau'(u, a) = \mu_1(w)b_u$ where $b_u \in \mathcal{A}$ depends solely on u. Since $\mu_1(w)b_u \notin U$ by Claim 3, $F_{u,\mu_1(w)b_u} = 0$. Thus, $F'_{u,\mu_1(w)b_u} = \left(\Delta D_n^{(\ell_w>1)}\right)_{u,\mu_1(w)b_u} = K_{u,\nu_1(w)}$. Also since $\mu_1(w) \prec \tau'(u, a)$, $F'_{u,\mu_1(w)} = 0$ by Lemma 4.13(vii). Thus, $F_{u,\mu_1(w)} = -\left(\Delta D_n^{(\ell_w>1)}\right)_{u,\mu_1(w)} = K_{u,\nu_1(w)}$. We conclude that $F'_{u,\mu_1(w)b_u} = F_{u,\mu_1(w)}$, and $F_{u,\mu_1(w)b_u} = F'_{u,\mu_1(w)}$, from which (D.77) follows.

We are left with the multinomial factors in the pseudo-states of U_w and U'_w, i.e., with the quotient $\Pi_{T'}/\Pi_T$ where

$$\Pi_T = \prod_{u \in U_w \cup U'_w} \frac{F_{u*}!}{\prod_v F_{u,v}!} \; ; \Pi_{T'} = \prod_{u \in U_w \cup U'_w} \frac{F'_{u*}!}{\prod_v F'_{u,v}!} \, .$$

For convenience we split Π_T and $\Pi_{T'}$ in two factors as

$$\Pi_T = \Pi_{T,U} \times \Pi_{T,U'} \; ; \Pi_{T'} = \Pi_{T',U} \times \Pi_{T',U'} \, ,$$

where

$$\Pi_{T,U'} = \prod_{u \in U'_w} \frac{F_{u*}!}{\prod_v F_{u,v}!} ; \; \Pi_{T,U} = \prod_{u \in U_w \setminus U'_w} \frac{F_{u*}!}{\prod_v F_{u,v}!} \, ,$$

and

$$\Pi_{T',U'} = \prod_{u \in U'_w} \frac{F'_{u*}!}{\prod_v F'_{u,v}!} ; \; \Pi_{T',U} = \prod_{u \in U_w \setminus U'_w} \frac{F'_{u*}!}{\prod_v F'_{u,v}!} \, .$$

We can write $\Pi_{T,U'}$ as

$$\Pi_{T,U'} = \prod_{b \in \mathcal{A}} \prod_{i=1}^{\ell_w - 1} \frac{F_{\mu_i(w)b*}!}{\prod_v F_{\mu_i(w)b,v}!} \, , \tag{D.78}$$

and also for $\Pi_{T',U'}$

$$\Pi_{T',U'} = \prod_{b \in \mathcal{A}} \prod_{i=1}^{\ell_w - 1} \frac{F'_{\mu_i(w)b*}!}{\prod_v F'_{\mu_i(w)b,v}!} \, . \tag{D.79}$$

For $\Pi_{T,U}$ and $\Pi_{T',U}$ however, some indexes in the range from $i = 1$ to $\ell_w - 1$ may be excluded from the product as they may correspond to elements $\mu_i(w)$ that belong to the intersection $U_w \cap U'_w$. We now characterize the elements in $U_w \cap U'_w$, for which we define r as the largest index, in the range from 1 to $\ell_w - 1$, such that $\mu_r(w)$ has the form $c^{|\mu_r(w)|}$ with $c \in \mathcal{A}$, or $r = 1$ if such index does not exist.

6. Claim. *Let i be an integer such that $1 \leq i < \ell_w$. Then, $\mu_i(w) \in U_w \cap U'_w$ if and only if $i \leq r$ and $i > 1$.*

Proof. If $\mu_i(w) \in U_w \cap U'_w$, there exist $b \in \mathcal{A}$ and $1 \leq j < \ell_w$ such that $\mu_i(w) = \mu_j(w)b$. Since $\mu_i(w)$ and $\mu_j(w)b$ are equally long, we must have $i > 1$ and $j = i - 1$, i.e., $\mu_i(w) = \mu_{i-1}(w)b$. Since by definition also $\mu_i(w) = d\mu_{i-1}(w)$ for some $d \in \mathcal{A}$, $\mu_i(w)$ must have the form $b^{|\mu_i(w)|}$. Hence, $i \leq r$ as claimed. On the other hand, if $1 < i \leq r$, then $\mu_i(w) = c^{|\mu_i(w)|}$ by the definition of r. In this case, $\mu_{i-1}(w) = \text{tail}(\mu_i(w)) = c^{|\mu_i(w)|-1}$. Thus, $\mu_i(w) = \mu_{i-1}(w)c$, which belongs to $U_w \cap U'_w$. □

Indexes m and r are related as follows.

7. Claim. *If $m > 0$ and $r > 1$, then $m > r$.*

Proof. If $r > 1$, then $\mu_{r-1}(w) = c^{|\mu_{r-1}(w)|}$ and $\mu_r(w) = c^{|\mu_r(w)|}$. Thus, $\mu_{r-1}(w) \prec \mu_r(w)$ and therefore $\mu_r(w) \notin S_T$. Hence $m \neq r$ by the definition of m. If $1 \leq i < r$, then $\mu_{i+1}(w) \in U'$ by Claim 6, and therefore $i \neq m$ by the definition of m. We conclude that if $m > 0$ and $r > 1$, then $m > r$. □

By Claim 6 we can now express $\Pi_{T,U}$ and $\Pi_{T',U}$ as

$$\Pi_{T,U} = \frac{F_{\mu_1(w)*}!}{\prod_v F_{\mu_1(w),v}!} \prod_{i=r+1}^{\ell_w - 1} \frac{F_{\mu_i(w)*}!}{\prod_v F_{\mu_i(w),v}!}\,, \tag{D.80}$$

and

$$\Pi_{T',U} = \frac{F'_{\mu_1(w)*}!}{\prod_v F'_{\mu_1(w),v}!} \prod_{i=r+1}^{\ell_w - 1} \frac{F'_{\mu_i(w)*}!}{\prod_v F'_{\mu_i(w),v}!}\,. \tag{D.81}$$

Consider the subsets $U_w^+ = \{\mu_i(w) \in U_w : i > 1\}$ and $U'^+_w = \{\mu_i(w)b \in U'_w : i > 1, b \in \mathcal{A}\}$.

8. Claim. *For every* $u \in U_w^+ \cup U'^+_w$, $F_{\mathsf{tail}(u),u} = F_{*u} = F_{u*} + \delta_{s_n,u}$ *and* $F'_{\mathsf{tail}(u),u} = F'_{*u} = F'_{u*} + \delta_{s_n,u}$.

Proof. By Lemma 4.13(xi), $B'_{*u} = B_{*u} = 0$ for every $u \in U_w^+ \cup U'^+_w$, and therefore $F_{*u} = D_{*u}$ and $F'_{*u} = D'_{*u}$. From Lemma 4.13(vii), we observe that for a positive entry $D_{v,u}$ with $u \in U_w^+ \cup U'^+_w$, v must be of the form $v = \mathsf{tail}(u)$. Hence, $F_{*u} = D_{\mathsf{tail}(u),u} = F_{\mathsf{tail}(u),u}$. Also by Lemma 4.13(ix), $F_{u*} + \delta_{s_n,u} = F_{*u}$, where we have ruled out the term $\delta_{s_0,u}$ by the condition of s_0 being of maximal length. Similarly, $F'_{u*} + \delta_{s'_n,u} = F'_{*u} = F'_{\mathsf{tail}(u),u}$, and since $u \notin \{w\} \cup \{wb : b \in \mathcal{A}\}$ by Claim 5, $\delta_{s'_n,u} = \delta_{s_n,u}$. □

By Claim 8 we then have

$$\Pi_{T,U'} = \prod_{b \in \mathcal{A}} \frac{F_{\mu_1(w)b*}!}{F_{\mu_{\ell_w-1}(w)b,wb}!} \left(\prod_{i=1}^{\ell_w - 1} \prod_{v \neq \mu_{i+1}(w)b} F_{\mu_i(w)b,v}! \right)^{-1}, \tag{D.82}$$

and, when $r < \ell_w - 1$, $\Pi_{T,U} = \Pi^{(r)}_{T,U}$ where,

$$\begin{aligned} \Pi^{(r)}_{T,U} \;=\; & \frac{F_{\mu_1(w)*}!}{\prod_v F_{\mu_1(w),v}!} \frac{F_{\mu_{r+1}(w)*}!}{F_{\mu_{\ell_w-1}(w),w}!} \left(\prod_{i=r+1}^{\ell_w - 1} \prod_{v \neq \mu_{i+1}(w)} F_{\mu_i(w),v}! \right)^{-1} \\ & \times \left(\prod_{i=r+2}^{\ell_w - 1} \max\left\{1, F_{*\mu_i(w)} \delta_{s_n,\mu_i(w)}\right\} \right)^{-1}. \end{aligned} \tag{D.83}$$

Notice that the last factor of (D.83) accounts for the term $\delta_{s_n,u}$ in Claim 8, which makes F_{u*} to differ from $F_{\mathsf{tail}(u),u} = F_{*u}$ when $s_n = u$. A similar factor is not necessary in (D.82) as clearly $\mu_i(w)b$ is not a state of T (nor of T' since $i < \ell_w$). In the case $r = \ell_w - 1$, we simply get from (D.80), $\Pi_{T,U} = \Pi^{(R)}_{T,U}$ where,

$$\Pi^{(R)}_{T,U} = \frac{F_{\mu_1(w)*}!}{\prod_v F_{\mu_1(w),v}!}\,. \tag{D.84}$$

Equations (D.82), (D.83), and (D.84) are valid for $\Pi_{T',U}$ and $\Pi_{T',U'}$ replacing F by F'.

We now proceed to analyze the different kinds of factors in (D.83) and (D.82). The following claims are instrumental to this aim.

9. Claim. *Let $u = \mu_i(w)b$ with $b \in \mathcal{A}$ and $1 \le i < \ell_w$. Then, for $v = \mu_{i+1}(w)b$,*

$$D'_{u,v} - D_{u,v} = \begin{cases} N'_{*wb} - N_{*w}, & \text{if } i+1 < r, b = \mathit{head}(u), \\ N'_{*wb}, & \text{otherwise.} \end{cases} \tag{D.85}$$

Also for $v \neq \mu_{i+1}(w)b$,

$$D'_{u,v} - D_{u,v} = \begin{cases} -N_{*w}, & \text{if } i+1 = r, b = \mathit{head}(u), v = \mu_{i+2}(w), \\ 0\,, & \text{otherwise.} \end{cases} \tag{D.86}$$

Proof. Since u is not a state of T, $u \notin S_1$, and therefore $D'_{u,v} - D_{u,v}$ is determined exclusively by the second summation of (D.76). If $i+1 < r$ and $b = \mathrm{head}(u)$, we know by Claim 6 that $\mu_{i+2}(w) = b^{|\mu_{i+2}(w)|}$, and therefore $\mu_{i+1}(w) = \mu_i(w)b$, and $\mu_{i+2}(w) = \mu_{i+1}(w)b$. Furthermore, since w is not of the form $b^{|w|}$ for $\ell_w > 1$, $i+2 < \ell_w$. Hence, the term $\mathbf{1}_{\mu_i(w)b,\mu_{i+1}(w)b}$, which is added N'_{*wb} times, and the term $-\mathbf{1}_{\mu_{i+1}(w),\mu_{i+2}(w)}$, which is added N_{*w} times, affect both the same entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,\mu_{i+1}(w)b}$. This gives the first line of (D.85) where $v = \mu_{i+1}(w)b$. Since these are the only two terms that affect $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ in this case, we get $D'_{u,v} - D_{u,v} = 0$ if $v \neq \mu_{i+1}(w)b$. Thus, to prove the second line of (D.86), it remains to show that when $v \neq \mu_{i+1}(w)b$, $D'_{u,v} - D_{u,v} = 0$ if $i+1 > r$, or $b = \mathrm{head}(u)$, or $v \neq \mu_{i+2}(w)$. Indeed, since $v \neq \mu_{i+1}(w)b$ entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ can only be affected by a term $-\mathbf{1}_{\mu_{i+1}(w),\mu_{i+2}(w)}$ with $\mu_{i+1}(w) = u$ and $\mu_{i+2}(w) = v$. The condition $\mu_{i+1}(w) = u$ implies $i+1 \le r$ and $b = \mathrm{head}(u)$ by Claim 6, which completes the proof of the second line of (D.86). Moreover, if $i+1 = r$ and $b = \mathrm{head}(u)$, so that $\mu_{i+1}(w) = u$ by Claim 6, and also $\mu_{i+2}(w) = v \neq \mu_{i+1}(w)b$, then the only term of (D.76) that affects $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ is $-\mathbf{1}_{\mu_{i+1}(w),\mu_{i+2}(w)}$, which is added N_{*w} times as $i+2 = r+1 \le \ell_w$. This proves the first line of (D.86). Finally, if $i+1 \ge r$, or $b \neq \mathrm{head}(u)$, then we see that for $v = \mu_{i+1}(w)b$, entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ is only affected by the term $\mathbf{1}_{\mu_i(w)b,\mu_{i+1}(w)b}$. Indeed, if the entry were affected by a term $-\mathbf{1}_{\mu_{i+1}(w),\mu_{i+2}(w)}$, then $\mu_{i+2}(w) = \mu_{i+1}(w)b$ and therefore $i+2 \le r$, and $b = \mathrm{head}(u)$ by Claim 6. Since the term $\mathbf{1}_{\mu_i(w)b,\mu_{i+1}(w)b}$ is added N'_{*wb} times, the first line of (D.85) is proved. □

10. Claim. *Let $u = \mu_i(w)$ with $1 \le i < \ell_w$ and $i \neq m$. Then, for $v = \mu_{i+1}(w)$ and $b = \mathit{head}(u)$,*

$$D'_{u,v} - D_{u,v} = \begin{cases} N'_{*wb} - N_{*w}, & \text{if } 1 < i < r, \\ -N_{*w}, & \text{otherwise.} \end{cases} \tag{D.87}$$

Also for $v \neq \mu_{i+1}(w)$ *and* $b = \mathit{head}(u)$,

$$D'_{u,v} - D_{u,v} = \begin{cases} N'_{*wb}, & \text{if } 1 < i = r, v = ub, \\ K_{u,\nu_1(w)}, & \text{if } u \in S_T, \tau(u, \mathit{head}(v)) = \mu_1(w), \tau'(u, \mathit{head}(v)) = \mu_1(w)c = v, \\ -K_{u,\nu_1(w)}, & \text{if } u \in S_T, \tau(u, \mathit{head}(v)) = \mu_1(w) = v, \tau'(u, \mathit{head}(v)) = \mu_1(w)c, \\ 0\,, & \text{otherwise.} \end{cases} \tag{D.88}$$

Proof. We first prove (D.87), i.e., we consider $v = \mu_{i+1}(w)$. If $1 < i \leq r$, then $\mu_i(w) = \mu_{i-1}(w)b$ with $b = \mathrm{head}(u)$ by Claim 6. Hence, $u \notin S_T$ and therefore, only the second summation of (D.76) may affect entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$. If $i < r$, then also $\mu_{i+1}(w) = \mu_i(w)b$. In this case, the term $\mathbf{1}_{\mu_{i-1}(w)b,\mu_i(w)b}$, which is added N'_{*wb} times, and the term $-\mathbf{1}_{\mu_i(w),\mu_{i+1}(w)}$, which is added N_{*w} times, affect the same entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,\mu_{i+1}(w)}$. This proves the first line of (D.87). If $i \geq r$ and $i > 1$, then $\mu_{i+1}(w) \neq \mu_i(w)b$ for all $b \in \mathcal{A}$. Thus, the terms $\mathbf{1}_{\mu_{i-1}(w)b,\mu_i(w)b}$ with $b \in \mathcal{A}$ do not affect $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ for $v = \mu_{i+1}(w)$. Hence, in this case, the second summation of (D.76) makes a negative contribution $-N_{*w}$ to $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ through the term $-\mathbf{1}_{\mu_i(w),\mu_{i+1}(w)}$. The same applies when $i = 1$, as in this case no term of the form $\mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b}$ affects the entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$. We next show that the third summation of (D.76) does not affect $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$, which concludes the proof of the second line of (D.87). Suppose that $(u, v') \in Z$, where v' is the unique state of T such that $v' \preceq v$, and $\tau(u, \mathrm{head}(v)) \neq \tau'(u, \mathrm{head}(v))$. Since $i \neq m$, by Claim 3 we have that $\tau(u, \mathrm{head}(v)) = \mu_1(w)$, and $\tau'(u, \mathrm{head}(v)) = \mu_1(w)b$. Thus, in order to affect entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ with $v = \mu_{i+1}(w)$, we must have $v = \mu_1(w)b$, but $\mu_1(w)b \notin U$ by Claim 3. This concludes the proof of (D.87). We now consider (D.88), where $v \neq \mu_{i+1}(w)$. If $i = r > 1$, then $\mu_i(w) = \mu_{i-1}(w)b$ with $b = \mathrm{head}(u)$. Thus, $u \notin S_T$, and therefore, only the second summation of (D.76) may affect entry $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$. Since $v \neq \mu_{i+1}(w)$, only the term $\mathbf{1}_{\mu_{i-1}(w)b,\mu_i(w)b}$, which is added N'_{*wb} times, affects $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ for $v = ub$. This proves the first line of (D.88). Notice that when $v \neq ub$ we obtain $\left(\Delta D^{(\ell_w>1)}\right)_{u,v} = 0$ in agreement with the last line of (D.88), which is selected since $u \notin S_T$ as required by the second and third line. If $1 < i < r$, then $\mu_i(w)b = \mu_{i+1}(w) \neq v$, and therefore the term $\mathbf{1}_{\mu_{i-1}(w)b,\mu_i(w)b}$ does not affect $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$. Similarly, if $i = 1$, no term of the form $\mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b}$ affects $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$. Thus, in any case, the value of $\left(\Delta D^{(\ell_w>1)}\right)_{u,v}$ is determined exclusively by the third summation of (D.76). The remaining lines of (D.88) then follow by Claim 3 recalling that $i \neq m$. □

The analogous of Claim 10 for the special case $u = \mu_m(w)$ is Claim 11 below. When $m > 0$ we denote the symbol $\mathrm{head}(\mu_{m+1}(w))$ by a_m. Notice that, since $m < \ell_w - 1$ by the definition of m, $\mu_{m+1}(w) \neq w$, and therefore $\mu_{m+1}(w)$ is not an internal node of T'. Thus, $\tau'(u, a_m)$ is well defined for $u = \mu_m(w)$.

11. Claim. *Suppose $m > 0$ and let $u = \mu_m(w)$. Then,*

$$\begin{aligned} D_{u,\mu_{m+1}(w)} &= K_{u,\nu_{m+1}(w)} + N_{*w}\,, \\ D'_{u,\mu_{m+1}(w)} &= 0\,, \\ D'_{u,\tau'(u,a_m)} &= K_{u,\nu_{m+1}(w)}\,, \\ D_{u,\tau'(u,a_m)} &= 0\,. \end{aligned}$$

Also for all $v \notin \{\mu_{m+1}(w), \tau'(u,a_m)\}$,

$$D'_{u,v} - D_{u,v} = \begin{cases} K_{u,\nu_1(w)}, & \text{if } \tau(u,\mathit{head}(v)) = \mu_1(w), \tau'(u,\mathit{head}(v)) = \mu_1(w)c = v, \\ -K_{u,\nu_1(w)}, & \text{if } \tau(u,\mathit{head}(v)) = \mu_1(w) = v, \tau'(u,\mathit{head}(v)) = \mu_1(w)c, \\ 0\,, & \text{otherwise.} \end{cases} \tag{D.89}$$

Proof. Since $\mu_{m+1}(w) \notin U'$, $D'_{u,\mu_{m+1}(w)} = 0$. Hence, $D_{u,\mu_{m+1}(w)} = -\left(\Delta D_n^{(\ell_w>1)}\right)_{u,\mu_{m+1}(w)}$. Also $\tau'(u,a_m) \prec \tau(u,a_m) = \mu_{m+1}(w)$ for $\mu_{m+1}(w) \notin U'$, and we see from Lemma 4.13(vii) that $D_{u,\tau'(u,a_m)} = 0$. As a consequence, $D'_{u,\tau'(u,a_m)} = \left(\Delta D_n^{(\ell_w>1)}\right)_{u,\tau'(u,a_m)}$. The last summation of (D.76) adds $K_{u,\nu_{m+1}(w)}$ times the terms $\mathbf{1}_{u,\tau'(u,a_m)} - \mathbf{1}_{u,\mu_{m+1}(w)}$. As the second summation of (D.76) does only include terms of the form $\mathbf{1}_{\mathsf{tail}(v),v}$, and $\tau'(u,a_m) \prec \mu_{m+1}(w)$, entry $(u,\tau'(u,a_m))$ of $\Delta D_n^{(\ell_w>1)}$ is determined exclusively by the last summation. We then conclude that $D'_{u,\tau'(u,a_m)} = K_{u,\nu_{m+1}(w)}$. As for entry $(u,\mu_{m+1}(w))$, recall that by Claim 7, if $r > 1$ then $m > r$, and therefore $\mu_m(w)$ is not of the form $\mu_{m-1}(w)b$. Thus, no term of the form $\mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b}$ of the second summation of (D.76) affects $\left(\Delta D^{(\ell_w>1)}\right)_{u,\mu_{m+1}(w)}$. On the other hand, the second summation of (D.76) does subtract N_{*w} times the term $\mathbf{1}_{u,\mu_{m+1}(w)}$, which together with the last summation yield $D_{u,\mu_{m+1}(w)} = K_{u,\nu_{m+1}(w)} + N_{*w}$. We finally observe that for $v \notin \{\mu_{m+1}(w), \tau'(u,a_m)\}$, the term $-\mathbf{1}_{\mu_m(w),\mu_{m+1}(w)}$ of the second summation of (D.76) does not affect $\left(\Delta D_n^{(\ell_w>1)}\right)_{u,v}$. A term of the form $\mathbf{1}_{\mu_{j-1}(w)b,\mu_j(w)b}$ does not affect $\left(\Delta D_n^{(\ell_w>1)}\right)_{u,v}$ either, for otherwise, $\mu_{m-1}(w)b = \mu_m(w)$ and therefore, $1 < m \le r$, contradicting Claim 7. Thus, only the third summation of (D.76) may affect $\left(\Delta D_n^{(\ell_w>1)}\right)_{u,v}$, and since $v \notin \{\mu_{m+1}(w), \tau'(u,a_m)\}$, (D.89) follows from Claim 3. □

We now consider the quotient $\Pi_{T',U'}/\Pi_{T,U'}$. By Claim 9, we see that the factors $F_{\mu_i(w)b,v}$, and $F'_{\mu_i(w)b,v}$ of (D.82) cancel each other, except possibly, when they correspond to B and B' entries, or when the first line of (D.86) applies. In the latter case, $i+1 = r$,$b = \mathrm{head}(u)$, and $v = \mu_{i+2}(w)$. Thus, $\mu_i(w)b = \mu_{i+1}(w) = \mu_r(w)$, and $v = \mu_{r+1}(w)$. We then get,

$$\frac{\Pi_{T',U'}}{\Pi_{T,U'}} = \left(\prod_{b\in\mathcal{A}} \frac{F'_{\mu_1(w)b*}!F_{\mu_{\ell_w-1}(w)b,wb}!}{F'_{\mu_{\ell_w-1}(w)b,wb}!F_{\mu_1(w)b*}!} \prod_{i=1}^{\ell_w-1} \frac{B_{\mu_i(w)b,\rho(\mu_i(w)b)}!}{B'_{\mu_i(w)b,\rho'(\mu_i(w)b)}!}\right) \frac{((1-\delta_{r,1})D_{\mu_r(w),\mu_{r+1}(w)})!}{((1-\delta_{r,1})D'_{\mu_r(w),\mu_{r+1}(w)})!}\,. \tag{D.90}$$

A similar factor cancelation occurs in $\Pi_{T',U}/\Pi_{T,U}$ when we look at (D.83). By Claim 10, we see that the factors $F_{\mu_i(w),v}$, and $F'_{\mu_i(w),v}$, with $r < i < \ell_w$, $i \neq m$, and $v \neq \mu_{i+1}(w)$, cancel

each other, except possibly, when they correspond to B and B' entries, or, when the second or third line of (D.88) applies. In the latter case, observe that if $u \in S_T$, $\tau(u, \mathrm{head}(v)) = \mu_1(w)$, and $\tau'(u, \mathrm{head}(v)) = \mu_1(w)c$, then $\mu_1(w)c$ does not belong to U by Claim 3. Hence, if $v = \mu_1(w)c$, $D_{u,v} = 0$ and therefore $D'_{u,v} = K_{u,\nu_1(w)}$ by Claim 10. Also, if $v = \mu_1(w)$, since $\mu_1(w) \prec \tau'(u, \mathrm{head}(v))$, we see from Lemma 4.13(vii) that $D'_{u,v} = 0$, and therefore $D_{u,v} = K_{u,\nu_1(w)}$ by Claim 10. Hence, when the second or third line of (D.88) applies, the factors $F_{\mu_i(w),v}$, and $F'_{\mu_i(w),v'}$ also cancel each other for $v = \mu_1(w)$ and $v' = \mu_1(w)c$. Similarly, if $i = m$, by Claim 11 the factors $F_{\mu_i(w),v}$, and $F'_{\mu_i(w),v}$ cancel each other for $v \notin \{\mu_{m+1}(w), \tau'(u, a_m)\}$, except possibly, when the first or second line of (D.89) applies. In this case, the factors $F_{\mu_i(w),v}$, and $F'_{\mu_i(w),v'}$ cancel each other for $v = \mu_1(w)$ and $v' = \mu_1(w)c$. Thus, when $r < \ell_w - 1$ we get from (D.83),

$$\frac{\Pi^{(r)}_{T',U}}{\Pi^{(r)}_{T,U}} = \frac{F'_{\mu_1(w)*}! \prod_v F_{\mu_1(w),v}!}{F_{\mu_1(w)*} \prod_v F'_{\mu_1(w),v}!} \frac{F'_{\mu_{r+1}(w)*}! F_{\mu_{\ell_w-1}(w),w}!}{F_{\mu_{r+1}(w)*}! F'_{\mu_{\ell_w-1}(w),w}!} \prod_{i=r+1}^{\ell_w-1} \frac{B_{\mu_i(w),\rho(\mu_i(w))}!}{B'_{\mu_i(w),\rho'(\mu_i(w))}!} \times \frac{\tilde{\Pi}}{\big((1-\delta_{m,0})(1-\delta_{m,1})K_{\mu_m(w),\nu_{m+1}(w)}\big)!}, \tag{D.91}$$

where,

$$\tilde{\Pi} = \frac{\prod_{i=r+2}^{\ell_w-1} \max\left\{1, F_{*\mu_i(w)} \delta_{s_n,\mu_i(w)}\right\}}{\prod_{i=r+2}^{\ell_w-1} \max\left\{1, F'_{*\mu_i(w)} \delta_{s_n,\mu_i(w)}\right\}}. \tag{D.92}$$

Notice that if $m > 1$, then $m > r$ by Claim 7. Thus, the factor $K_{\mu_m(w),\nu_{m+1}(w)}!$ arises in the denominator of (D.91), since for $v = \tau'(u, a_m) \neq \mu_{m+1}(w)$, we have $D'_{u,\tau'(u,a_m)} = K_{u,\nu_{m+1}(w)}$, but $D_{u,\tau'(u,a_m)} = 0$ by Claim 11. The same factor cancelation arguments used to derive (D.91) apply also to the quotient $\prod_v F_{\mu_1(w),v}! \Big/ \prod_v F'_{\mu_1(w),v}!$. In this case, if $m \neq 1$ we see by Claim 10 that only the factors with $v = \mu_2(w)$, or those corresponding to B and B' entries, survive. If $m = 1$, by Claim 11 we see that, besides B and B' entries, and the factor with $v = \mu_2(w) = \mu_{m+1}(w)$, a factor $K_{\mu_m(w),\nu_{m+1}(w)}!$ arises in the denominator. This comes from the fact that for $v = \tau'(u, a_m) \neq \mu_{m+1}(w)$, we have $D'_{u,\tau'(u,a_m)} = K_{u,\nu_{m+1}(w)}$, but $D_{u,\tau'(u,a_m)} = 0$. Thus,

$$\frac{\prod_v F_{\mu_1(w),v}!}{\prod_v F'_{\mu_1(w),v}!} = \frac{F_{\mu_1(w),\mu_2(w)}! B_{\mu_1(w),\rho(\mu_1(w))}!}{F'_{\mu_1(w),\mu_2(w)}! B'_{\mu_1(w),\rho'(\mu_1(w))}!} \times \frac{1}{\big(\delta_{m,1} K_{\mu_m(w),\nu_{m+1}(w)}\big)!}. \tag{D.93}$$

Substituting (D.93) in (D.91), we get

$$\frac{\Pi^{(r)}_{T',U}}{\Pi^{(r)}_{T,U}} = \frac{F'_{\mu_1(w)*}! F_{\mu_1(w),\mu_2(w)}! B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*} F'_{\mu_1(w),\mu_2(w)}! B'_{\mu_1(w),\rho'(\mu_1(w))}!} \frac{F'_{\mu_{r+1}(w)*}! F_{\mu_{\ell_w-1}(w),w}!}{F_{\mu_{r+1}(w)*}! F'_{\mu_{\ell_w-1}(w),w}!} \prod_{i=r+1}^{\ell_w-1} \frac{B_{\mu_i(w),\rho(\mu_i(w))}!}{B'_{\mu_i(w),\rho'(\mu_i(w))}!} \times \frac{\tilde{\Pi}}{\big((1-\delta_{m,0})K_{\mu_m(w),\nu_{m+1}(w)}\big)!}. \tag{D.94}$$

Also from (D.93) and (D.84), we get for $r = \ell_w - 1$,

$$\frac{\Pi^{(R)}_{T',U}}{\Pi^{(R)}_{T,U}} = \frac{F'_{\mu_1(w)*}! F_{\mu_1(w),\mu_2(w)}! B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*} F'_{\mu_1(w),\mu_2(w)}! B'_{\mu_1(w),\rho'(\mu_1(w))}!} \times \frac{1}{\big(\delta_{m,1} K_{\mu_m(w),\nu_{m+1}(w)}\big)!}. \tag{D.95}$$

If $r > 1$, then by Claim 7, either $m = 0$, or $m > r$. But since $m < \ell_w - 1$ by definition, we must have $m = 0$ if $r = \ell_w - 1$. On the other hand, if $r = 1$, and $r = \ell_w - 1$, we have $m < \ell_w - 1 = 1$ and therefore $m = 0$. Hence, when $r = \ell_w - 1$, $\delta_{m,1}$ is always zero, as so is $(1 - \delta_{m,0})$. Since also $\tilde{\Pi} = 1$ when $r = \ell_w - 1$, we get combining (D.94) and (D.95),

$$\frac{\Pi_{T',U}}{\Pi_{T,U}} = \frac{F'_{\mu_1(w)*}!F_{\mu_1(w),\mu_2(w)}!B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}!F'_{\mu_1(w),\mu_2(w)}!B'_{\mu_1(w),\rho'(\mu_1(w))}!} \times \frac{\Pi^{(r)} \times \tilde{\Pi}}{\big((1-\delta_{m,0})K_{\mu_m(w),\nu_{m+1}(w)}\big)!}\,, \tag{D.96}$$

where $\Pi^{(r)} = 1$ if $r = \ell_w - 1$, and otherwise,

$$\Pi^{(r)} = \frac{F'_{\mu_{r+1}(w)*}!F_{\mu_{\ell_w-1}(w),w}!}{F'_{\mu_{\ell_w-1}(w),w}!F_{\mu_{r+1}(w)*}!} \prod_{i=r+1}^{\ell_w-1} \frac{B_{\mu_i(w),\rho(\mu_i(w))}!}{B'_{\mu_i(w),\rho'(\mu_i(w))}!}\,. \tag{D.97}$$

We will further simplify (D.96) and (D.90) by means of the claims that we show next.

12. Claim. *Let $u = \mu_i(w)$ with $1 < i < \ell_w$. Then for all $v \in \bar{\Lambda}(u)$, $B_{v,\rho(v)} = 0$, and for all $v \in \bar{\Lambda}'(u)$, $B'_{v,\rho'(v)} = 0$. We then have for all strings u' such that $u \prec u'$,*

$$\sum_{v\in\Lambda(u')} D_{*v} - D_{v*} = \sum_{v\in\Lambda'(u')} D'_{*v} - D'_{v*} = 0\,. \tag{D.98}$$

Proof. Since $i > 1$, $\mathrm{tail}(u) = \mu_{i-1}(w)$ is a pseudo-state of T, and therefore $\mathrm{tail}(uz) \notin T$ for all $z \in \mathcal{A}^+$. Hence, by Lemma 4.13(xi), we get $B_{v,\rho(v)} = 0$ for all $v \in \bar{\Lambda}(u)$. Since $\mathrm{tail}(u) = \mu_{i-1}(w) \neq w$, $\mathrm{tail}(u)$ is not an internal node of T', and therefore $\mathrm{tail}(uz) \notin T'$ for all $z \in \mathcal{A}^+$. Thus, also $B'_{v,\rho'(v)} = 0$ for all $v \in \bar{\Lambda}'(u)$. Equation (D.98) then follows by the definition of matrix B. □

Notice that an immediate consequence of Claim 12 is that $B_{u,\rho(u)} = 0$ and $B'_{u,\rho'(u)} = 0$ for all $u \in U'^{+}_{w}$. Hence, (D.90) reduces to

$$\frac{\Pi_{T',U'}}{\Pi_{T,U'}} = \left(\prod_{b\in\mathcal{A}} \frac{F'_{\mu_1(w)b*}!F_{\mu_{\ell_w-1}(w)b,wb}!}{F'_{\mu_{\ell_w-1}(w)b,wb}!F_{\mu_1(w)b*}!} \frac{B_{\mu_1(w)b,\rho(\mu_1(w)b)}!}{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)}!}\right) \frac{((1-\delta_{r,1})D_{\mu_r(w),\mu_{r+1}(w)})!}{((1-\delta_{r,1})D'_{\mu_r(w),\mu_{r+1}(w)})!}\,. \tag{D.99}$$

Also notice that $F_{\mu_{\ell_w-1}(w)b,wb} = 0$ for $wb \notin U$ for all $b \in \mathcal{A}$. Moreover, since wb has maximal depth in T', $F'_{\mu_{\ell_w-1}(w)b,wb} = N'_{*wb}$ by Claim 4. Hence, (D.99) becomes

$$\frac{\Pi_{T',U'}}{\Pi_{T,U'}} = \left(\prod_{b\in\mathcal{A}} \frac{F'_{\mu_1(w)b*}!}{N'_{*wb}!F_{\mu_1(w)b*}!} \frac{B_{\mu_1(w)b,\rho(\mu_1(w)b)}!}{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)}!}\right) \frac{((1-\delta_{r,1})D_{\mu_r(w),\mu_{r+1}(w)})!}{((1-\delta_{r,1})D'_{\mu_r(w),\mu_{r+1}(w)})!}\,. \tag{D.100}$$

We next analyze the factors $B_{\mu_1(w)b,\rho(\mu_1(w)b)}$ and $B'_{\mu_1(w)b,\rho'(\mu_1(w)b)}$ of the last equation. The following claims are instrumental to this aim.

13. Claim. *Let $u' = \mu_1(w)bz$ with $b \in \mathcal{A}$ and $z \in \mathcal{A}^+$. Then, for $m = 0$, or $u' \notin \{\mu_{m+1}(w), \tau'(\mu_m(w), a_m)\}$,*

$$D'_{*u'} - D'_{u'*} = D_{*u'} - D_{u'*}\,, \tag{D.101}$$

for $m > 0$ *and* $u' = \mu_{m+1}(w)$,

$$D'_{*u'} - D'_{u'*} = D_{*u'} - D_{u'*} - K_{\mu_m(w),\nu_{m+1}(w)}\,, \tag{D.102}$$

and for $m > 0$ *and* $u' = \tau'(\mu_m(w), a_m)$,

$$D'_{*u'} - D'_{u'*} = D_{*u'} - D_{u'*} + K_{\mu_m(w),\nu_{m+1}(w)}\,. \tag{D.103}$$

Proof. By Claim 5 $\mu_1(w)b \neq w$. Hence, $u' \notin \{w\} \cup \{wc : c \in \mathcal{A}\}$ and therefore the first summation of (D.76) does not affect outgoing transitions from u'. The last summation of (D.76) does not affect the total amount of outgoing transitions from any pseudo-state. On the other hand, by Claim 3 and recalling that $u' \notin \{\mu_1(w)c : c \in \mathcal{A}\} \cup \{\mu_1(w)\}$ for $z \in \mathcal{A}^+$, the first and last summation of (D.76) do not affect incoming transitions to u' if $m = 0$ or $u' \notin \{\mu_{m+1}(w), \tau'(\mu_m(w), a_m)\}$. Since the second summation of (D.76) preserves the flow balance in u', for $u' \notin \{w\} \cup \{wc : c \in \mathcal{A}\}$, and $u' \notin \{\mu_1(w)c : c \in \mathcal{A}\} \cup \{\mu_1(w)\}$, we conclude that for $m = 0$, or $u' \notin \{\mu_{m+1}(w), \tau'(\mu_m(w), a_m)\}$,

$$D'_{*u'} - D'_{u'*} = D_{*u'} - D_{u'*}\,. \tag{D.104}$$

For $u' = \mu_{m+1}(w)$ with $m > 0$, Claim 11 gives $D'_{\mu_m(w),u'} - D_{\mu_m(w),u'} = -K_{\mu_m(w),\nu_{m+1}(w)} - N_{*w}$. Since $\mu_{m+1}(w) \notin U'$ by the definition of m, $D'_{*u'} = 0$. Also since $\mu_m(w) = \text{tail}(u')$, we observe from Lemma 4.13(vii) that all input transitions to u' in D come from $\mu_m(w)$. Thus, $D'_{*u'} - D_{*u'} = -K_{\mu_m(w),\nu_{m+1}(w)} - N_{*w}$. For output transitions from $u' = \mu_{m+1}(w)$ we apply Claim 10 with $i = m+1$, which by the definition of m satisfies $i < \ell_w$. Since $\mu_{m+1}(w) = u' = \mu_1(w)bz$ with $z \in \mathcal{A}^+$, $m > 1$ and therefore $m > r$ by Claim 7. Thus, the first line of (D.87), and the first line of (D.88) in Claim 10 do not apply. Hence, $D'_{u'*} - D_{u'*} = -N_{*w}$, and we conclude that for $u' = \mu_{m+1}(w)$,

$$D'_{*u'} - D'_{u'*} = D_{*u'} - D_{u'*} - K_{\mu_m(w),\nu_{m+1}(w)}\,. \tag{D.105}$$

Suppose now that $u' = \tau'(\mu_m(w), a_m)$ with $m > 0$. Since $u' \prec \mu_{m+1}(w)$, and $\mu_m(w)$ is a state of T, then $\text{tail}(u')$ is an internal node of T. As a consequence, for all $j > 1$, u' is not of the form $\mu_j(w)$ nor of the form $\mu_j(w)c$ by (4.6). But, since $u' = \mu_1(w)bz$ with $z \in \mathcal{A}^+$, u' is not of the form $\mu_1(w)$ nor of the form $\mu_1(w)c$ either. Hence, the second summation of (D.76) does not affect incoming transitions to u', nor outgoing transitions from u'. Also since $u' \notin \{\mu_1(w)c : c \in \mathcal{A}\} \cup \{\mu_1(w)\}$ for $z \in \mathcal{A}^+$, by Claim 3 the first and last summation of (D.76) do not affect the total number of incoming transitions to u', except for the term $(K_n)_{\mu_m(w),\nu_{m+1}(w)}\,(\mathbf{1}_{\mu_m(w),\tau'(\mu_m(w),a_m)} - \mathbf{1}_{\mu_m(w),\mu_{m+1}(w)})$. Thus, $D'_{*u'} - D_{*u'} = K_{\mu_m(w),\nu_{m+1}(w)}$. Regarding output transitions from u', we recall that $u' \notin \{w\} \cup \{wc : c \in \mathcal{A}\}$ and therefore the first summation of (D.76) does not affect outgoing transitions from u'. The third summation does not affect them either, since $\mu_1(w)b \prec u'$ and therefore $u' \notin S_T$. Hence, $D'_{u'*} - D_{u'*} = 0$, and we conclude

$$D'_{*u'} - D'_{u'*} = D_{*u'} - D_{u'*} + K_{\mu_m(w),\nu_{m+1}(w)}\,. \tag{D.106}$$

□

14. Claim. *Let $u = \mu_1(w)b$ with $b \in \mathcal{A}$. If $u \notin U$, $F'_{u*} = B'_{u,\rho'(u)} + N'_{*wb}$. If on the other hand $u \in U$, $B'_{u,\rho'(u)} - B_{u,\rho(u)} = -N'_{*wb}$, and $F'_{u*} - F_{u*} = -\delta_{u,\mu_2(w)} N_{*w}$.*

Proof. If $u \notin U$, wb is the only state of T' that includes $\mu_1(w)b$ in its forced pseudo-state sequence. Since wb has maximal depth in T', $j \in J$ for every j such that $0 \le j \le n$ and $s'_j = wb$. Then, by Lemma 4.13(vii), $D'_{u*} = D'_{\mu_1(w)b,\mu_2(w)b} = N'_{*wb}$. As a consequence, $F'_{u*} = B'_{u,\rho'(u)} + N'_{*wb}$. Suppose now that $u \in U$. By Claim 9 with $i = 1$,

$$\left(D_n^{(\ell_w>1)}\right)_{u*} = \begin{cases} N'_{*wb} - N_{*w} & \text{if } i+1 \le r, b = \text{head}(u); \\ N'_{*wb} & \text{otherwise}, \end{cases} \tag{D.107}$$

or, equivalently,

$$\left(D_n^{(\ell_w>1)}\right)_{u*} = N'_{*wb} - \delta_{u,\mu_2(w)} N_{*w}\,. \tag{D.108}$$

Suppose first that $m = 0$ or $\mu_{m+1}(w) \notin \Lambda(u)$. In this case, recalling that $u \in U$, we know by Claim 3 that only the second summation of (D.76) may affect incoming transitions to u. This in turn can only occur if u coincides with $\mu_2(w)$. Hence,

$$\left(D_n^{(\ell_w>1)}\right)_{*u} = -\delta_{u,\mu_2(w)} N_{*w}\,. \tag{D.109}$$

Since $\mu_{m+1}(w) \notin \Lambda(u)$, by Claim 13 and the definition of B we have

$$\begin{aligned} B'_{u,\rho'(u)} - B_{u,\rho(u)} &= \left(D'_{*u} - D'_{u*}\right) - \left(D_{*u} - D_{u*}\right) \\ &= \left(D'_{*u} - D_{*u}\right) - \left(D'_{u*} - D_{u*}\right)\,. \end{aligned} \tag{D.110}$$

Substituting (D.108) and (D.109) in (D.110) we get,

$$B'_{u,\rho'(u)} - B_{u,\rho(u)} = -N'_{*wb}\,,$$

which combined with (D.108) yields

$$F'_{u*} - F_{u*} = -\delta_{u,\mu_2(w)} N_{*w}\,.$$

We now consider the case in which $m > 0$ and $\mu_{m+1}(w) \in \Lambda(u)$. In fact, since $\mu_{m+1}(w) \notin U'$ by the definition of m, and $u = \mu_1(w)b \in U'$, then $u \neq \mu_{m+1}(w)$, and we must have $\mu_{m+1}(w) \in \bar{\Lambda}(u)$. We start by analyzing incoming transitions to u in $D' - D$ by means of (D.76). Since $\mu_1(w)b \in U$, by Claim 3 we see that the first summation of (D.76) does not affect any incoming transitions to u, and the third summation may affect $\left(D_n^{(\ell_w>1)}\right)_{*u}$ only through the term $(K_n)_{\mu_m(w),\nu_{m+1}(w)} \left(\mathbf{1}_{\mu_m(w),\tau'(\mu_m(w),a_m)} - \mathbf{1}_{\mu_m(w),\mu_{m+1}(w)}\right)$. Since we have argued that $u \neq \mu_{m+1}(w)$, this term may affect $\left(D_n^{(\ell_w>1)}\right)_{*u}$ only if $\tau'(\mu_m(w), a_m) = \mu_1(w)b$. The second summation may affect incoming transitions to u only if $\mu_1(w)b = \mu_2(w)$. In this case, (D.76) includes N_{*w} times the term $-\mathbf{1}_{\mu_1(w),\mu_1(w)b}$. We then conclude that

$$\left(D_n^{(\ell_w>1)}\right)_{*u} = \delta_{u,\tau'(\mu_m(w),a_m)} K_{\mu_m(w),\nu_{m+1}(w)} - \delta_{u,\mu_2(w)} N_{*w}\,. \tag{D.111}$$

We now study the difference $B'_{u,\rho'(u)} - B_{u,\rho(u)}$. We start by analyzing $B'_{*u} - B_{*u}$ for which we have,

$$B'_{*u} - B_{*u} = \sum_{v':\rho'(v')=u} B'_{v',u} - \sum_{v:\rho(v)=u} B_{v,u}\,. \tag{D.112}$$

We recall that $B'_{v',u} = \sum_{t\in\Lambda'(v')} D'_{*t} - D'_{t*}$, and $B_{v,u} = \sum_{t\in\Lambda(v)} D_{*t} - D_{t*}$. Hence, since $D'_{*t} = D'_{t*} = 0$ for $t \in U\backslash U'$, and $D_{*t} = D_{t*} = 0$ for $t \in U'\backslash U$, we can equivalently write

$$B'_{*u} - B_{*u} = \sum_{t\in\bar\Lambda'(u)\cup\bar\Lambda(u)} (D'_{*t} - D'_{t*}) - (D_{*t} - D_{t*})\,. \tag{D.113}$$

Each $t \in \bar\Lambda'(u) \cup \bar\Lambda(u)$, has the form $\mu_1(w)bz$ with $z \in \mathcal{A}^+$, and therefore we can apply Claim 13. By the definition of τ, $\tau'(\mu_m(w), a_m)$ is the longest prefix of $\mu_{m+1}(w)$ that belongs to U'. Thus, since $\mu_{m+1}(w) \in \bar\Lambda(u)$, either $\tau'(\mu_m(w), a_m) = u$, or there exists $t \in \bar\Lambda'(u)$ such that $\tau'(\mu_m(w), a_m) = t$. Hence, Claim 13 applied to the terms in (D.113) gives,

$$B'_{*u} - B_{*u} = -\delta_{u,\tau'(\mu_m(w),a_m)} K_{\mu_m(w),\nu_{m+1}(w)}\,, \tag{D.114}$$

where $K_{\mu_m(w),\nu_{m+1}(w)}$ comes from (D.102) with $t = \mu_{m+1}(w)$, which is canceled by (D.103) if and only if $\tau'(\mu_m(w), a_m) \neq u$. Summing (D.114) to (D.111) we get,

$$F'_{*u} - F_{*u} = -\delta_{u,\mu_2(w)} N_{*w}\,. \tag{D.115}$$

Since $u = \mu_1(w)b$ is not a state of T, nor of T', we must have $F'_{*u} - F_{*u} = F'_{u*} - F_{u*}$ by Lemma 4.13(ix). Thus,

$$F'_{u*} - F_{u*} = -\delta_{u,\mu_2(w)} N_{*w}\,. \tag{D.116}$$

Since $F'_{u*} - F_{u*} = D'_{u*} - D_{u*} + B'_{u,\rho'(u)} - B_{u,\rho(u)}$, subtracting (D.108) from (D.116) we get,

$$B'_{u,\rho'(u)} - B_{u,\rho(u)} = -N'_{*wb}\,.$$

□

We now apply Claim 14 to the factors of (D.100). For a symbol $b \in \mathcal{A}$ such that $\mu_1(w)b \notin U$, $F_{\mu_1(w)b*} = 0$, $B_{\mu_1(w)b,\rho(\mu_1(w)b)} = 0$, and, by Claim 14, $F'_{\mu_1(w)b*} = B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb}$. Hence, the factor

$$\frac{F'_{\mu_1(w)b*}!}{N'_{*wb}!F_{\mu_1(w)b*}!}\frac{B_{\mu_1(w)b,\rho(\mu_1(w)b)}!}{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)}!}\,, \tag{D.117}$$

reduces to

$$\begin{pmatrix} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb} \\ N'_{*wb} \end{pmatrix}.$$

On the other hand, for a symbol $b \in \mathcal{A}$ such that $\mu_1(w)b \in U$, by Claim 14, $B_{\mu_1(w)b,\rho(\mu_1(w)b)} = B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb}$, and $F'_{\mu_1(w)b*} - F_{\mu_1(w)b*} = -\delta_{\mu_1(w)b,\mu_2(w)} N_{*w}$. Hence, (D.117) becomes

$$\begin{pmatrix} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb} \\ N'_{*wb} \end{pmatrix} \times \frac{(\delta_{\mu_1(w)b,\mu_2(w)} F'_{\mu_1(w)b*})!}{(\delta_{\mu_1(w)b,\mu_2(w)} F_{\mu_1(w)b*})!}\,. \tag{D.118}$$

Thus, since $\mu_1(w)b = \mu_2(w)$ if and only if $r > 1$ and $b = \mathrm{head}(\mu_2(w))$, we get replacing in (D.100),

$$\frac{\Pi_{T',U'}}{\Pi_{T,U'}} = \left[\prod_{b\in\mathcal{A}} \binom{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb}}{N'_{*wb}}\right] \times \frac{((1-\delta_{r,1})F'_{\mu_2(w)*})!}{((1-\delta_{r,1})F_{\mu_2(w)*})!} \times \frac{((1-\delta_{r,1})D_{\mu_r(w),\mu_{r+1}(w)})!}{((1-\delta_{r,1})D'_{\mu_r(w),\mu_{r+1}(w)})!}\,. \tag{D.119}$$

We now turn to (D.96). We start by simplifying (D.97) by means of the following claim.

15. Claim. *Let $u = \mu_i(w)$ with $r < i < \ell_w$. For $i \neq m+1$, $B_{u,\rho(u)} = B'_{u,\rho'(u)}$, and for $i = m+1$, $B_{u,\rho(u)} = K_{\mu_m(w),\nu_1(w)}$ and $B'_{u,\rho'(u)} = 0$.*

Proof. Notice that since $i > r$, we know by Claim 6 that $\mu_i(w) \notin U'_w$. The second summation of (D.76) preserves the flow balance in u, i.e., the incoming transition term $\mathbf{1}_{\mu_{i-1}(w),u}$ is subtracted as many times as the outgoing term $\mathbf{1}_{u,\mu_{i+1}(w)}$. The last summation of (D.76) preserves the total count of outgoing transitions from a pseudo-state. By Claim 3, both the first and the last summation of (D.76) only affect incoming transitions to $\mu_1(w)$, $\mu_{m+1}(w)$, $\tau'(\mu_m(w), a_m)$, and $\mu_1(w)b$ for $b \in \mathcal{A}$. Now, since $i > r \geq 1$, we know that $u \neq \mu_1(w)$, and by Claim 6, $u \neq \mu_1(w)b$ for all $b \in \mathcal{A}$. Also, as $\tau'(\mu_m(w), a_m) \prec \mu_{m+1}(w)$ when $m > 0$, and $\mu_m(w)$ is a state of T, then $\mathrm{tail}(\tau'(\mu_m(w), a_m))$ is an internal node of T. Thus, by (4.6), $\tau'(\mu_m(w), a_m) \neq u$, for $u = \mu_i(w)$ with $i > 1$. Hence, the unique term of (D.76) that affects incoming transitions to u is $(K_n)_{\mu_m(w),\nu_{m+1}(w)} (\mathbf{1}_{\mu_m(w),\tau'(\mu_m(w),a_m)} - \mathbf{1}_{\mu_m(w),\mu_{m+1}(w)})$ when $u = \mu_{m+1}(w)$. We conclude that

$$D_{*u} - D_{u*} = D'_{*u} - D'_{u*} \text{ if } i \neq m+1\,, \tag{D.120}$$

and,

$$D_{*u} - D_{u*} = D'_{*u} - D'_{u*} + K_{\mu_m(w),\nu_{m+1}(w)} \text{ if } i = m+1\,. \tag{D.121}$$

Suppose that $i \neq m+1$. If $u \in U'$, the definition of B and Claim 12 yield $B_{u,\rho(u)} = B'_{u,\rho'(u)}$. If $u \notin U'$, $B'_{u,\rho'(u)} = 0$ by definition. Also $D'_{*u} = D'_{u*} = 0$, and (D.120) reduces to $D_{*u} - D_{u*} = 0$. This implies by the definition of B and Claim 12 that $B_{u,\rho(u)}$ is also zero. Thus, in any case $B_{u,\rho(u)} = B'_{u,\rho'(u)}$ for $i \neq m+1$. We now consider the case in which $i = m+1$. Since $u \notin U'$ by the definition of m, $B'_{u,\rho'(u)} = 0$ and also $D'_{*u} = D'_{u*} = 0$. Thus, (D.121) gives $D_{*u} - D_{u*} = K_{\mu_m(w),\nu_{m+1}(w)}$, from which we get $B_{u,\rho(u)} = K_{\mu_m(w),\nu_{m+1}(w)}$ by the definition of B and Claim 12. Claim 15 is proved. □

Now, by Claim 15, the factors $B_{\mu_i(w),\rho(\mu_i(w))}!$ and $B'_{\mu_i(w),\rho'(\mu_i(w))}!$ of (D.97) cancel each other, except, possibly, for $i = m+1$ where $B_{\mu_i(w),\rho(\mu_i(w))} = K_{\mu_m(w),\nu_{m+1}(w)}$, and $B'_{\mu_i(w),\rho'(\mu_i(w))} = 0$. Now, if $r > 1$, by Claim 7, either $m = 0$ or $m > r$. In the latter case, (D.97) includes a factor with $i = m+1$, since also $m+1 \leq \ell_w - 1$ by the definition of m. If $r = 1$, either $m = 0$, or $m \geq r$ and therefore (D.97) includes, also in this case, a factor with $i = m+1$. Hence, (D.97) becomes,

$$\Pi^{(r)} = \frac{F'_{\mu_{r+1}(w)*}!F_{\mu_{\ell_w-1}(w),w}!}{F'_{\mu_{\ell_w-1}(w),w}!F_{\mu_{r+1}(w)*}!} \times \big((1-\delta_{m,0})K_{\mu_m(w),\nu_{m+1}(w)}\big)!\,. \tag{D.122}$$

Notice that, since $w \notin U'$, $F'_{\mu_{\ell_w-1}(w),w} = 0$. Also since $|w| \geq \mathsf{depth}(T) - 1$, $F_{\mu_{\ell_w-1}(w),w} = N_{*w}$ by Claim 4. Hence, (D.122) becomes,

$$\Pi^{(r)} = \frac{F'_{\mu_{r+1}(w)*}!N_{*w}!}{F_{\mu_{r+1}(w)*}!} \times \big((1-\delta_{m,0})K_{\mu_m(w),\nu_{m+1}(w)}\big)! \,. \tag{D.123}$$

Then, for $r < \ell_w - 1$, we get from (D.96)

$$\frac{\Pi_{T',U}}{\Pi_{T,U}} = \frac{F'_{\mu_1(w)*}!F_{\mu_1(w),\mu_2(w)}!B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}!F'_{\mu_1(w),\mu_2(w)}!B'_{\mu_1(w),\rho'(\mu_1(w))}!} \frac{F'_{\mu_{r+1}(w)*}!N_{*w}!}{F_{\mu_{r+1}(w)*}!} \times \tilde{\Pi} \tag{D.124}$$

By Claim 8, $F_{\mu_1(w),\mu_2(w)} = F_{\mu_2(w)*} + \delta_{s_n,\mu_2(w)} = F_{*\mu_2(w)}$, and $F'_{\mu_1(w),\mu_2(w)} = F'_{\mu_2(w)*} + \delta_{s_n,\mu_2(w)} = F'_{*\mu_2(w)}$. Thus, we can substitute $F_{\mu_2(w)*}$ for $F_{\mu_1(w),\mu_2(w)}$, and $F'_{\mu_2(w)*}$ for $F'_{\mu_1(w),\mu_2(w)}$ in (D.124), multiplying by an additional factor $\frac{\max\{1,F_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}}{\max\{1,F'_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}}$.

$$\frac{\Pi_{T',U}}{\Pi_{T,U}} = \frac{F'_{\mu_1(w)*}!F_{\mu_2(w)*}!B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}!F'_{\mu_2(w)*}!B'_{\mu_1(w),\rho'(\mu_1(w))}!} \frac{F'_{\mu_{r+1}(w)*}!N_{*w}!}{F_{\mu_{r+1}(w)*}!} \times \tilde{\Pi} \times \frac{\max\{1,F_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}}{\max\{1,F'_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}} \,. \tag{D.125}$$

Now, if $r = 1$ the factors $F_{\mu_2(w)*}!$ and $F_{\mu_{r+1}(w)*}!$ cancel each other, and the same occurs with $F'_{\mu_2(w)*}!$ and $F'_{\mu_{r+1}(w)*}!$. Thus, multiplying (D.125) by (D.119) we obtain

$$\begin{aligned}\frac{\Pi_{T',U}}{\Pi_{T,U}} \times \frac{\Pi_{T',U'}}{\Pi_{T,U'}} \;=\; & \left[\prod_{b\in\mathcal{A}} \binom{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb}}{N'_{*wb}}\right] \frac{F'_{\mu_1(w)*}!B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}!B'_{\mu_1(w),\rho'(\mu_1(w))}!} \times N_{*w}! \times \tilde{\Pi} \\ & \times \frac{\max\{1,F_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}}{\max\{1,F'_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}}.\end{aligned} \tag{D.126}$$

If on the other hand $1 < r < \ell_w - 1$, we know by Claim 8 that $F_{\mu_{r+1}(w)*} + \delta_{s_n,\mu_{r+1}(w)} = F_{\mu_r(w),\mu_{r+1}(w)} = F_{*\mu_{r+1}(w)}$, and $F'_{\mu_{r+1}(w)*} + \delta_{s_n,\mu_{r+1}(w)} = F'_{\mu_r(w),\mu_{r+1}(w)} = F'_{*\mu_{r+1}(w)}$. Thus, we can substitute $F_{\mu_r(w),\mu_{r+1}(w)}$ for $F_{\mu_{r+1}(w)*}$, and $F'_{\mu_r(w),\mu_{r+1}(w)}$ for $F'_{\mu_{r+1}(w)*}$ in (D.125), multiplying by an additional factor $\frac{\max\{1,F_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}\}}{\max\{1,F'_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}\}}$, obtaining

$$\begin{aligned}\frac{\Pi_{T',U}}{\Pi_{T,U}} \;=\; & \frac{F'_{\mu_1(w)*}!F_{\mu_2(w)*}!B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}!F'_{\mu_2(w)*}B'_{\mu_1(w),\rho'(\mu_1(w))}!} \frac{F'_{\mu_r(w),\mu_{r+1}(w)}!N_{*w}!}{F_{\mu_r(w),\mu_{r+1}(w)}!} \times \tilde{\Pi} \\ & \times \frac{\max\{1,F_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}}{\max\{1,F'_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}} \frac{\max\{1,F_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}\}}{\max\{1,F'_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}\}} \,.\end{aligned} \tag{D.127}$$

When we multiply (D.127) by (D.119), the factor $F_{\mu_2(w)*}! \big/ F'_{\mu_2(w)*}!$ of (D.127) cancels with the inverse factor of (D.119). Also the factor $F'_{\mu_r(w),\mu_{r+1}(w)}! \big/ F_{\mu_r(w),\mu_{r+1}(w)}!$ of (D.127) cancels with $D_{\mu_r(w),\mu_{r+1}(w)}! \big/ D'_{\mu_r(w),\mu_{r+1}(w)}!$ of (D.119). Thus, for $1 < r < \ell_w - 1$ we obtain,

$$\begin{aligned}\frac{\Pi_{T',U}}{\Pi_{T,U}} \times \frac{\Pi_{T',U'}}{\Pi_{T,U'}} \;=\; & \left[\prod_{b\in\mathcal{A}} \binom{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb}}{N'_{*wb}}\right] \frac{F'_{\mu_1(w)*}!B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}!B'_{\mu_1(w),\rho'(\mu_1(w))}!} \times N_{*w}! \times \tilde{\Pi} \\ & \times \frac{\max\{1,F_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}}{\max\{1,F'_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}\}} \frac{\max\{1,F_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}\}}{\max\{1,F'_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}\}} \,.\end{aligned} \tag{D.128}$$

We combine (D.126), which is valid for $r = 1$, with (D.128) to obtain a single equation valid for $r < \ell_w - 1$,

$$\frac{\Pi_{T',U}}{\Pi_{T,U}} \times \frac{\Pi_{T',U'}}{\Pi_{T,U'}} = \left[\prod_{b\in\mathcal{A}} \binom{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} + N'_{*wb}}{N'_{*wb}}\right] \frac{F'_{\mu_1(w)*}! B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}! B'_{\mu_1(w),\rho'(\mu_1(w))}!} \times N_{*w}! \times \tilde{\Pi}' , \tag{D.129}$$

where, noting that $(1 - \delta_{\ell_w,2}) = 1$ for $r < \ell_w - 1$,

$$\tilde{\Pi}' = \tilde{\Pi} \times \frac{\max\{1, F_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}(1-\delta_{\ell_w,2})\}}{\max\{1, F'_{*\mu_2(w)}\delta_{s_n,\mu_2(w)}(1-\delta_{\ell_w,2})\}} \frac{\max\{1, F_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}(1-\delta_{r,1})\}}{\max\{1, F'_{*\mu_{r+1}(w)}\delta_{s_n,\mu_{r+1}(w)}(1-\delta_{r,1})\}} . \tag{D.130}$$

Although the factor $(1 - \delta_{\ell_w,2})$ in (D.130) is never null for $r < \ell_w - 1$, it makes this equation also applicable in the case $r = \ell_w - 1$ that we analyze next.

If $r = \ell_w - 1$, we recall from the discussion preceding (D.96) that $m = 0$. Thus, (D.96) reduces to

$$\frac{\Pi_{T',U}}{\Pi_{T,U}} = \frac{F'_{\mu_1(w)*}! F_{\mu_1(w),\mu_2(w)}! B_{\mu_1(w),\rho(\mu_1(w))}!}{F_{\mu_1(w)*}! F'_{\mu_1(w),\mu_2(w)}! B'_{\mu_1(w),\rho'(\mu_1(w))}!} . \tag{D.131}$$

Since $r = \ell_w - 1$, $\mu_{r+1}(w) = w$. Hence, if also $r > 1$, the factor $D'_{\mu_r(w),\mu_{r+1}(w)}$ of (D.119) is null, and the factor $D_{\mu_r(w),\mu_{r+1}(w)}$ equals N_{*w} by Claim 4. Also by Claim 8, $F_{\mu_1(w),\mu_2(w)} = F_{\mu_2(w)*} + \delta_{s_n,\mu_2(w)} = F_{*\mu_2(w)}$ and $F'_{\mu_1(w),\mu_2(w)} = F'_{\mu_2(w)*} + \delta_{s_n,\mu_2(w)} = F'_{*\mu_2(w)}$. Thus, (D.131) together with (D.119) yields the same Equation (D.129) that we have derived for $r < \ell_w - 1$. If in particular $r = 1$, i.e., $\ell_w = 2$, then $\mu_2(w) = w$ and therefore $F_{\mu_1(w),\mu_2(w)} = N_{*w}$ by Claim 4, and $F'_{\mu_1(w),\mu_2(w)} = 0$. Thus, (D.131) together with (D.119) yield (D.129) once again. The following claim will allow us to further simplify (D.129).

16. Claim. *If $\mu_1(w) \notin U'$, $F_{\mu_1(w)*} = B_{\mu_1(w),\rho(\mu_1(w))} + N_{*w}$ and $\sum_{b\in\mathcal{A}} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} = B_{\mu_1(w),\rho(\mu_1(w))} + \delta_{s_n,\mu_1(w)}$. If $\mu_1(w) \in U'$, $B'_{\mu_1(w),\rho'(\mu_1(w))} = B_{\mu_1(w),\rho(\mu_1(w))}$ and $F'_{\mu_1(w)*} - F_{\mu_1(w)*} = -N_{*w}$.*

Proof. We first study the difference between $\sum_{b\in\mathcal{A}}(D'_{*\mu_1(w)b} - D_{*\mu_1(w)b})$ and $D'_{*\mu_1(w)} - D_{*\mu_1(w)}$. By Claim 3, the first and last summation of (D.76) subtract as many incoming transitions to $\mu_1(w)$, as they add to the pseudo-states in the set $\{\mu_1(w)b : b \in \mathcal{A}\}$. The only exception is, when $m > 0$, the term $(K_n)_{\mu_m(w),\nu_{m+1}(w)} (\mathbf{1}_{\mu_m(w),\tau'(\mu_m(w),a_m)} - \mathbf{1}_{\mu_m(w),\mu_{m+1}(w)})$, which adds incoming transitions to $\tau'(\mu_m(w), a_m)$, which may be of the form $\mu_1(w)b$, but subtracts incoming transitions from $\mu_{m+1}(w) \neq \mu_1(w)$. The second summation of (D.76) does not affect incoming transitions to $\mu_1(w)$, although it may subtract incoming transitions to $\mu_1(w)b$ in the case $\mu_1(w)b = \mu_2(w)$. Since these are the only terms involving incoming transitions to pseudo-states in $\{\mu_1(w)b : b \in \mathcal{A}\} \cup \{\mu_1(w)\}$, we conclude that

$$\begin{aligned} \sum_{b\in\mathcal{A}} D'_{*\mu_1(w)b} - D_{*\mu_1(w)b} \;=\; & -\left(D'_{*\mu_1(w)} - D_{*\mu_1(w)}\right) \\ & + (1-\delta_{m,0}) \sum_{b\in\mathcal{A}} \delta_{\mu_1(w)b,\tau'(\mu_m(w),a_m)} K_{\mu_m(w),\nu_{m+1}(w)} \\ & - \sum_{b\in\mathcal{A}} \delta_{\mu_2(w),\mu_1(w)b} N_{*w} . \end{aligned} \tag{D.132}$$

We now study the total amount of outgoing transitions from pseudo-states in $\{\mu_1(w)b : b \in \mathcal{A}\} \cup \{\mu_1(w)\}$. The last summation of (D.76) preserves the number of outgoing transitions from any pseudo-state, and the first summation does not involve pseudo-states in $U_w \cup U'_w$ by Claim 5. The second summation subtracts N_{*w} times the term $\mathbf{1}_{\mu_1(w),\mu_2(w)}$ and adds the same number of terms of the form $\mathbf{1}_{\mu_1(w)b,\mu_2(w)b}$ with $b \in \mathcal{A}$. Additionally, if $\mu_1(w)b = \mu_2(w)$ for some $b \in \mathcal{A}$, $\mu_2(w)$ must be of the form b^k, which differs from w as $\ell_w > 1$. Hence, $\ell_w > 2$ and the second summation also subtracts N_{*w} times the term $\mathbf{1}_{\mu_1(w)b,\mu_3(w)}$. As a consequence,

$$\sum_{b\in\mathcal{A}} D'_{\mu_1(w)b*} - D_{\mu_1(w)b*} = \left(1 - \sum_{b\in\mathcal{A}} \delta_{\mu_2(w),\mu_1(w)b}\right) N_{*w}\,, \tag{D.133}$$

and,

$$D'_{\mu_1(w)*} - D_{\mu_1(w)*} = -N_{*w}\,. \tag{D.134}$$

Subtracting (D.133) from (D.132) we get

$$\begin{aligned}
&\sum_{b\in\mathcal{A}} \left(D'_{*\mu_1(w)b} - D_{*\mu_1(w)b}\right) - \sum_{b\in\mathcal{A}} \left(D'_{\mu_1(w)b*} - D_{\mu_1(w)b*}\right) \\
&\quad = -\left(D'_{*\mu_1(w)} - D_{*\mu_1(w)}\right) - N_{*w} + (1-\delta_{m,0}) \sum_{b\in\mathcal{A}} \delta_{\mu_1(w)b,\tau'(\mu_m(w),a_m)} K_{\mu_m(w),\nu_{m+1}(w)}\,.
\end{aligned}$$

Combining the last equation with (D.134), and reordering terms, we get

$$\begin{aligned}
\sum_{b\in\mathcal{A}} \left(D'_{*\mu_1(w)b} - D'_{\mu_1(w)b*}\right) - \sum_{b\in\mathcal{A}} \left(D_{*\mu_1(w)b} - D_{\mu_1(w)b*}\right) &= \\
&= \left(D'_{\mu_1(w)*} - D_{\mu_1(w)*}\right) - \left(D'_{*\mu_1(w)} - D_{*\mu_1(w)}\right) \\
&\quad + (1-\delta_{m,0}) \sum_{b\in\mathcal{A}} \delta_{\mu_1(w)b,\tau'(\mu_m(w),a_m)} K_{\mu_m(w),\nu_{m+1}(w)}\,.
\end{aligned} \tag{D.135}$$

Now, by Claim 13, we have,

$$\begin{aligned}
&\sum_{b\in\mathcal{A}} \sum_{u\in\bar{\Lambda}(\mu_1(w)b)} \left(D'_{*u} - D'_{u*}\right) - \sum_{b\in\mathcal{A}} \sum_{u\in\bar{\Lambda}(\mu_1(w)b)} \left(D_{*u} - D_{u*}\right) \\
&\quad = \quad -(1-\delta_{m,0}) \sum_{b\in\mathcal{A}} \delta_{\mu_1(w)b,\tau'(\mu_m(w),a_m)} K_{\mu_m(w),\nu_{m+1}(w)}\,,
\end{aligned}$$

where the negative term $-K_{\mu_m(w),\nu_{m+1}(w)}$ arises when $\tau'(\mu_m(w), a_m) = \mu_1(w)b$ for some $b \in \mathcal{A}$, so that (D.102) applies to $\mu_{m+1}(w) \in \bar{\Lambda}(\mu_1(w)b)$, and it is not canceled by (D.103). Summing this equation to (D.135) we get,

$$\begin{aligned}
&\sum_{b\in\mathcal{A}} \sum_{u\in\Lambda(\mu_1(w)b)} \left(D'_{*u} - D'_{u*}\right) - \sum_{b\in\mathcal{A}} \sum_{u\in\Lambda(\mu_1(w)b)} \left(D_{*u} - D_{u*}\right) \\
&\quad = \quad \left(D'_{\mu_1(w)*} - D_{\mu_1(w)*}\right) - \left(D'_{*\mu_1(w)} - D_{*\mu_1(w)}\right)\,,
\end{aligned}$$

or equivalently,

$$\sum_{b\in\mathcal{A}} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} - \sum_{v:\rho(v)=\mu_1(w)} B_{v,\mu_1(w)} = \left(D'_{\mu_1(w)*} - D_{\mu_1(w)*}\right) - \left(D'_{*\mu_1(w)} - D_{*\mu_1(w)}\right) . \tag{D.136}$$

Now, if $\mu_1(w) \in U'$, $\rho'(\mu_1(w)b) = \mu_1(w)$ for all $b \in \mathcal{A}$. Thus, the left hand side of (D.136) is simply $B'_{*\mu_1(w)} - B_{*\mu_1(w)}$. In this case, by Lemma 4.13(ix), we get from (D.136)

$$B'_{\mu_1(w),\rho'(\mu_1(w))} = B_{\mu_1(w),\rho(\mu_1(w))} \quad \text{if } \mu_1(w) \in U' , \tag{D.137}$$

where we have used in the application Lemma 4.13 that $\delta_{s_n,\mu_1(w)} = \delta_{s'_n,\mu_1(w)}$ for $\mu_1(w) \neq w$, and $\delta_{s_0,\mu_1(w)} = \delta_{s'_0,\mu_1(w)} = 0$ for $\mu_1(w)$ is not of maximal depth. This combined with (D.134) gives

$$F'_{\mu_1(w)*} - F_{\mu_1(w)*} = -N_{*w} . \tag{D.138}$$

If on the contrary $\mu_1(w) \notin U'$, $D'_{\mu_1(w)*} = 0$ and (D.134) gives $D_{\mu_1(w)*} = N_{*w}$. Hence, $F_{\mu_1(w)*} = B_{\mu_1(w),\rho(\mu_1(w))} + N_{*w}$. Also $D'_{*\mu_1(w)} = 0$ and (D.136) reduces to

$$\sum_{b\in\mathcal{A}} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} - B_{*\mu_1(w)} = D_{*\mu_1(w)} - D_{\mu_1(w)*} , \tag{D.139}$$

which, by Lemma 4.13(ix), gives

$$\sum_{b\in\mathcal{A}} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} = B_{\mu_1(w),\rho(\mu_1(w))} + \delta_{s_n,\mu_1(w)} . \tag{D.140}$$

The proof of Claim 16 is completed. □

We now apply Claim 16 to Equation (D.129). If $\mu_1(w) \in U'$, the factors $B'_{\mu_1(w),\rho'(\mu_1(w))}$ and $B_{\mu_1(w),\rho(\mu_1(w))}$ of (D.129) cancel each other. Since also $F_{\mu_1(w)*} = F'_{\mu_1(w)*} + N_{*w}$, (D.129) reduces to,

$$\frac{\Pi_{T',U}}{\Pi_{T,U}} \times \frac{\Pi_{T',U'}}{\Pi_{T,U'}} = \frac{\prod_{b\in\mathcal{A}} \binom{B'_{\mu_1(w)b,\rho'(\mu_1(w)b)}+N'_{*wb}}{N'_{*wb}}}{\binom{F_{\mu_1(w)*}}{N_{*w}}} \times \tilde{\Pi}' . \tag{D.141}$$

If on the contrary $\mu_1(w) \notin U'$, $F'_{\mu_1(w)*} = 0$ and $B'_{\mu_1(w),\rho'(\mu_1(w))} = 0$. Since also $F_{\mu_1(w)*} = B_{\mu_1(w),\rho(\mu_1(w))} + N_{*w}$, (D.129) reduces to (D.141) again.

To complete the proof of the lemma it remains to show (4.31). Claim 16 already states that, for $\mu_1(w) \notin U'$, $\sum_{b\in\mathcal{A}} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} = B_{\mu_1(w),\rho(\mu_1(w))} + \delta_{s_n,\mu_1(w)}$, and $B_{\mu_1(w),\rho(\mu_1(w))} = F_{\mu_1(w)*} - N_{*w}$. If otherwise $\mu_1(w) \in U'$, by (4.6) $\mathrm{tail}(\mu_1(w))$ is an internal node of T, and a fortiori an internal node of T'. Hence, since $\mu_1(w)$ has a full complement of children in U', i.e., $\mu_1(w)b \in U'$ for all $b \in \mathcal{A}$, $\tau'(s,a) \neq \mu_1(w)$ for all states s of T' and all symbols a. Hence, by Lemma 4.13(vii), $D'_{*\mu_1(w)} = 0$. As a consequence, $\sum_{b\in\mathcal{A}} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} = B'_{*\mu_1(w)} = F'_{*\mu_1(w)}$. This in turn is equal to $F'_{\mu_1(w)*} + \delta_{s_n,\mu_1(w)}$ by Lemma 4.13(ix). Since by Claim 16, $F'_{\mu_1(w)*} - F_{\mu_1(w)*} = -N_{*w}$, we conclude that $\sum_{b\in\mathcal{A}} B'_{\mu_1(w)b,\rho'(\mu_1(w)b)} = F_{\mu_1(w)*} - N_{*w} + \delta_{s_n,\mu_1(w)}$. □

Appendix E

Type classes under general initial conditions

In this appendix we extend the results of Chapter 4 to the more general setting in which the initial conditions are determined by transient states of the context tree, rather than a fixed initial permanent state s_0. In this case, a sequence x^n determines a state sequence $s'_0, s'_1 \cdots s'_n$, where each s'_i may be a transient state of the form $u\$ \in S_T^{\$}$, with $u \in \mathcal{I}(T)$. The probability assignment $\mathrm{P}_{\langle T,p_T\rangle}(\cdot)$ of (4.1) is now replaced by

$$\mathrm{P}_{\langle T,p_T\rangle}(\lambda) = 1; \quad \mathrm{P}_{\langle T,p_T\rangle}(x^n) = \prod_{i=1}^{n} p_T(x_i|s'_{i-1}), \quad n \geq 1\,, \tag{E.1}$$

which depends on the conditional probability mass functions of all the states of T, S_T^{A}, including permanent and transient states. Notice that this setting is more general in that it lets the model allocate arbitrary probabilities for the first symbols. As a consequence, the defining condition of a type class, namely, containing equiprobable sequences under any model parameter, becomes more constraining. Thus, we can foresee that, in relation to the original setting, type classes shrink, and there is a slight increment in the number of type classes.

We will study how this modification of the initial conditions affect the results we have derived in Chapter 4, either asymptotically or in the exact formula of Theorem 4.15. To avoid confusion with the notation introduced in Chapter 4 we define n'_s as the number of occurrences of the state $s \in S_T^{\mathrm{A}}$ in the sequence $s'_0, s'_1 \cdots s'_{n-1}$, and $n'^{(a)}_s$ as the number of occurrences of symbol a in state $s \in S_T^{\mathrm{A}}$. Notice that this definition agrees with our previous definition of n_s and $n_s^{(a)}$ in Chapter 3, if we eliminate the sign \$ from transient states. Clearly the type class of x^n is now

$$\mathcal{T}'(x^n) = \left\{ y^n \in \mathcal{A}^n : n'^{(a)}_s(y^n) \;=\; n'^{(a)}_s(x^n)\ \forall s \in S_T^{\mathrm{A}}, a \in \mathcal{A} \right\}. \tag{E.2}$$

We define $i_0(x^n)$, or simply i_0 when x^n is clear from the context, as the last index in the state sequence of x^n where a transient state shows up, i.e., $i_0 = \max\{i : \overline{x^i} \in \mathcal{I}(T)\}$. The following lemma characterizes $\mathcal{T}'(x^n)$.

E.1. Lemma. *$y^n \in \mathcal{T}'(x^n)$ if and only if $y^n \in \mathcal{T}(x^n)$, $x^{i_0(x^n)+1} \preceq y^n$, and $i_0(y^n) = i_0(x^n)$.*

Proof. Notice that for fixed s_0 of maximal depth, the counts $\{n_s^{(a)}(y^n)\}_{s\in S_T, a\in\mathcal{A}}$ together with the prefix $y^{i_0(y^n)+1}$ determines the counts $\{n'^{(a)}_s(y^n)\}_{s\in S_T^{\mathrm{A}}, a\in\mathcal{A}}$. Thus, the converse part follows immediately from this observation. Now, let $s = s'_{i_0(x^n)}(x^n)$, which, by definition of $i_0(x^n)$, is equal to $\overline{x^{i_0(x^n)}}\$$. If $y^n \in \mathcal{T}'(x^n)$, then $n'^{(a)}_s(y^n) = n'^{(a)}_s(x^n) = \delta_{a,x_{i_0(x^n)+1}}$, and therefore $x^{i_0(x^n)+1} \preceq y^n$. Moreover, all states $s'_i(y^n)$ with $i > i_0(x^n)$ are permanent states,

for otherwise $n'_{s'_i(y^n)} = 1$ in y^n, but $n'_{s'_i(y^n)} = 0$ in x^n. Thus, we have $i_0(y^n) = i_0(x^n)$. Since the counts $\{n'^{(a)}_s(y^n)\}_{s\in S^{\mathrm{A}}_T, a\in\mathcal{A}}$ together with the prefix $y^{i_0(y^n)+1}$ determines the counts $\{n^{(a)}_s(y^n)\}_{s\in S_T, a\in\mathcal{A}}$, we must also have $y^n \in \mathcal{T}(x^n)$. □

Observe that Lemma E.1 suffices to extend the asymptotic results of Chapter 4. Since $\mathcal{T}'(x^n) \subset \mathcal{T}(x^n)$, the upper bound of Theorem 4.18 is of course valid for the expected size of $\mathcal{T}'(X^n)$. Moreover, since each type class $\mathcal{T}(x^n)$ is partitioned into up to a constant number of subclasses $\mathcal{T}'$ for $S^{\$}_T$ is finite, the asymptotic result of Theorem 4.33 remains valid. The following lemma will help us in deriving an exact formula for the size of $\mathcal{T}'(x^n)$, and an enumeration algorithm for it. As we did in Chapter 4, we will establish a one-to-one mapping between unlabeled paths in a graph, and the set of strings in

$$\mathcal{T}'^*(x^n) \left\{ y^n \in \mathcal{T}'(x^n) : s_n(x^n) = s_n(y^n) \right\} .$$

By the same arguments used in Chapter 4, an enumeration algorithm, and an exact formula for the size of $\mathcal{T}'^*(x^n)$, yield straightforwardly their counterpart for $\mathcal{T}'(x^n)$.

E.2. Lemma. *Let J be the forced sequence parsing of x^n. We have $i_0 \in J$.*

Proof. Let $i = i_0$, and $h = \vec{i_0}$. By (4.5) we decompose s_h as

$$s_h = (s_h)_1^{\ell_{s_h}-j} \mu_j(s_h), \quad 1 \le j \le \ell_{s_h} .$$

We take in particular $j = i_\Delta$, which by Lemma 4.13(ii) satisfies $1 \le i_\Delta \le \ell_{s_h}$, and since $\ell_{s_h} - j = h - i$ by the definition of i_Δ, we get

$$s_h = (s_h)_1^{h-i} \mu_{i_\Delta}(s_h) . \tag{E.3}$$

Since $s_h \preceq \overline{x^h} s_0$, we have $(s_h)_1^{h-i} = \overline{x^h_{i+1}}$, and sice $\overline{x^i} \prec s_i$ by the definition of i_0, we know that $\overline{x^h} \prec (s_h)_1^{h-i} s_i$. Furthermore, since $s_i = \nu_{i_\Delta}(s_h)$ by Lemma 4.13(ii), we get

$$\overline{x^h} \prec (s_h)_1^{h-i} \mu_{i_\Delta}(s_h) .$$

Thus, by (E.3), we know that $\overline{x^h} \prec s_h$, and we must have $h = i$ by the definition of i_0. □

Since $i_0(x^n) \in J(x^n)$, the unlabeled path $\xi_{\$} = s_0\varsigma_1 \cdots \varsigma_j$, where $t_j = i_0$, is by Theorem 4.15 a prefix of the unique Eulerian unlabeled path ξ from s_0 to s_n in G_F such that $\omega(\xi) = x^n$. By Lemma 4.13(ii) and the definition of ς_i, it follows that $\omega(\xi_{\$}) = x^{i_0}$. Furthermore, by Lemma 4.14, there is a unique unlabeled path $\xi_{\$}$ from s_0 to s_{i_0} such that $\omega(\xi_{\$}) = x^{i_0}$. Hence, by Lemma E.1, and noting that $s_{i_0}(y^n) = s_{i_0}(x^n)$ for all $y^n \in \mathcal{T}'^*(x^n)$ since s_0 is fixed, we conclude that $\xi_{\$}$ is a prefix of any Eulerian unlabeled path γ from s_0 to s_n in G_F such that $\omega(\gamma) \in \mathcal{T}'^*(x^n)$. Furthermore, since $x^{i_0(x^n)+1}$ is a prefix of all strings $y^n \in \mathcal{T}'^*(x^n)$, $\xi_{\$}$ must necessarily be followed by the pseudo-state $\tau(s_{i_0}, x_{i_0+1})$ in any such Eulerian unlabeled path.

We define $u_0 = \arg\max\{|u| : u \in U, u \preceq \overline{x^{i_0(x^n)+1}}\}$, and β' as the unlabeled path formed by context-dropping transitions from $\tau(s_{i_0}, x_{i_0+1})$ to u_0, i.e., β' is empty if $\tau(s_{i_0}, x_{i_0+1}) = u_0$,

and otherwise $\beta' = v_0 v_1 \cdots v_m$ where $v_0 = \tau(s_{i_0}, x_{i_0+1})$, $v_m = u_0$, and $v_j = \rho(v_{j-1})$ for $j = 1 \cdots m$. We also denote by $G'_F(x^n)$ a graph obtained from $G_F(x^n)$ by deleting the set of edges of any path that is a representative of the unlabeled path $\xi_0 = \xi_{\$}\tau(s_{i_0}, x_{i_0+1})\beta'$, i.e., the incidence matrix of $G'_F(x^n)$ is given by $F'_{u,v} = F_{u,v} - |\{i : 1 \le i \le l, v_{i-1} = u, v_i = v\}|$, where $\xi_0 = v_0 \cdots v_l$.

Observe that ξ_0 and F' are determined by the counts $\{n'^{(a)}_s(x^n)\}_{s \in S^{\mathrm{A}}_T, a \in \mathcal{A}}$, which characterize $\mathcal{T}'(x^n)$. Hence, only the type class, and not the specific string x^n, is sufficient to obtain F'. Indeed, the longest transient state $s = u\$$ with a positive count $n'^{(a)}_s = 1$ determines $x^{i_0(x^n)+1} = \overline{u}a$. This, together with the fixed state s_0, determine the initial portion of the state sequence $s_0 \cdots s_{i_0}$, which allows for the computation of the whole state transition matrix N, and hence of F. Since $x^{i_0(x^n)+1}$ is known, we can compute the forced sequence parsing of x^{i_0}, from which we get $\xi_{\$}$, and then complete it with the concatenation of $\tau(s_{i_0}, x_{i_0+1})$ and context-dropping transitions until we obtain ξ_0. The following theorem parallels Theorem 4.15 in the context of general initial conditions. We define the normalized $|U| \times |U|$ matrix $\hat{F}'$ as $\hat{F}'_{i,j} = F'_{i,j}/F'_{i*}$ if $F'_{i*} > 0$ and $\hat{F}'_{i,j} = 0$ otherwise.

E.3. Theorem. *Let x^n be a sequence in $\mathcal{A}^n$.*

(**i**) *The unlabeled path ξ', obtained from $\xi = s_0\varsigma_1 \cdots \varsigma_r$ by deleting the prefix ξ_0, is an Eulerian unlabeled path from u_0 to s_n in G'_F such that $x^{i_0(x^n)+1}\omega(\xi') = x^n$.*

(**ii**) *The function ω defines a one-to-one correspondence between the set of Eulerian unlabeled paths from u_0 to s_n in G'_F and the sequences in $\mathcal{T}'^*(x^n)$.*

(**iii**) *Let M denote the cofactor of entry (s_n, u_0) in $I - \hat{F}'$. Then, $M \le 1$ and*

$$|\mathcal{T}'^*(x^n)| = M \frac{\prod_i F'_{i*}!}{\prod_{i,j} F'_{i,j}!}\,. \tag{E.4}$$

Proof. We will show that for an Eulerian unlabeled path γ from s_0 to s_n in G_F, $\omega(\gamma) \in \mathcal{T}'^*(x^n)$ if and only if ξ_0 is a prefix of γ, which proves parts (i) and (ii). The proof of Part (iii) follows exactly as Part (iii) of Theorem 4.15.

Suppose first that the string $y^n = \omega(\gamma)$ belongs to $\mathcal{T}'^*(x^n)$. By Theorem 4.15(i), $\gamma = s_0\varsigma_1 \cdots \varsigma_r$, where the unlabeled paths ς_i are defined with respect to the forced sequence parsing of y^n, $J(y^n) = \{t_i\}_{i=0}^r$. Moreover, by the discussion preceding the theorem, we know that $\xi_{\$}\tau(s_{i_0}, x_{i_0+1})$ is a prefix of γ, where $\xi_{\$} = s_0\varsigma_1 \cdots \varsigma_j$ and $t_j \in J(y^n)$ is $t_j = i_0(y^n) = i_0(x^n)$. By the definition of ς_i before Theorem 4.15, and the definition of β' above, to prove that ξ_0 is a prefix of γ we must show that $\mu_1(s_{t_{j+1}}(y^n)) \preceq u_0$. Suppose on the contrary that $\mu_1(s_{t_{j+1}}) \not\preceq u_0$, where $s_{t_{j+1}}$ hereafter stands for $s_{t_{j+1}}(y^n)$. Since $u_0 \preceq \overline{x^{i_0(x^n)+1}}$, and $\mu_1(s_{t_{j+1}}) \preceq \overline{y^{i_0(y^n)+1}}s_0 = \overline{x^{i_0(x^n)+1}}s_0$, but $\mu_1(s_{t_{j+1}}) \not\preceq u_0$, we must have $u_0 \prec \mu_1(s_{t_{j+1}})$. Hence, by the definition of u_0 we know that $\mu_1(s_{t_{j+1}}) \not\preceq \overline{x^{i_0(x^n)+1}}$, and therefore

$$\overline{x^{i_0(x^n)+1}} \prec \mu_1(s_{t_{j+1}})\,. \tag{E.5}$$

Now, by (4.5) we decompose $s_{t_{j+1}}$ as

$$s_{t_{j+1}} = (s_{t_{j+1}})_1^{\ell_{s_{t_{j+1}}} - i}\mu_i(s_{t_{j+1}}), \quad 1 \le i \le \ell_{s_{t_{j+1}}}\,. \tag{E.6}$$

We take $i = 1$ in (E.6) and, recalling that $\ell_{s_{t_{j+1}}} = t_{j+1} - t_j$ by Lemma 4.13(iii), we get

$$s_{t_{j+1}} = (s_{t_{j+1}})_1^{t_{j+1}-t_j-1}\mu_1(s_{t_{j+1}})\,. \tag{E.7}$$

Notice that $s_{t_{j+1}} \preceq \overline{y^{t_{j+1}}}$ and therefore $(s_{t_{j+1}})_1^{t_{j+1}-t_j-1} = y_{t_j+2}^{t_{j+1}}$. Hence, by (E.5) and (E.7), we get $\overline{y_{t_j+2}^{t_{j+1}}}\,\overline{x^{i_0(x^n)+1}} \prec s_{t_{j+1}}$, and since $\overline{x^{i_0(x^n)+1}} = \overline{y^{i_0(y^n)+1}}$, we obtain $\overline{y^{t_{j+1}}} \prec s_{t_{j+1}}$. We arrive to a contradiction by Lemma E.1 since in this case we have $i_0(y^n) \geq t_{j+1} > t_j = i_0(x^n)$. We conclude that $\mu_1(s_{t_{j+1}}(y^n)) \preceq u_0$ and therefore ξ_0 is a prefix of γ.

Suppose now that γ is an Eulerian unlabeled path from s_0 to s_n in G_F such that ξ_0 is a prefix of γ. We will prove that $y^n = \omega(\gamma)$ belongs to $\mathcal{T}'^*(x^n)$. Clearly $x^{i_0(x^n)+1} \preceq y^n$ for $\omega(\xi_0) = x^{i_0(x^n)+1}$. Since also $y^n \in \mathcal{T}^*(x^n)$ by Theorem 4.15, by Lemma E.1 we only need to show that $i_0(y^n) = i_0(x^n)$. Let $\xi_0 = v_0 \cdots v_l$, and $\gamma = v_0 \cdots v_l v_{l+1} \cdots v_m$. Each pair of consecutive vertexes in γ satisfies $|v_{i+1}| \leq |v_i| + 1$, and the equality may hold only if $|\omega(v_0 \cdots v_{i+1})| > |\omega(v_0 \cdots v_i)|$. Hence, since $v_l = u_0 \preceq \overline{x^{i_0(x^n)+1}} = \overline{\omega(v_0 \cdots v_l)}$, we have $v_i \preceq \overline{\omega(v_0 \cdots v_i)}$ for all $i \geq l$ by Part (i) of Lemma 4.14. Hence, for all $i \geq l$, the string $\omega(v_0 \cdots v_i)$ selects the unique permanent state of T that is a prefix of v_i. Thus, all prefixes of y^n of length grater than $i_0(x^n)$ select permanent states, and therefore we must have $i_0(y^n) = i_0(x^n)$. □

Appendix F

Eulerian unlabeled paths enumeration algorithm

Consider a graph $G = (V, E)$ and fix a complete order on the set V. Without loss of generality, let $V = \{v_1 \dots v_k\}$, and $v_i < v_j$ whenever $i < j$. We extend the order to length-m unlabeled paths lexicographically, i.e., for $\gamma = u_0, u_1 \cdots u_m$ and $\gamma' = u'_0, u'_1 \cdots u'_m$, we have $\gamma < \gamma'$ if and only if for some $i \leq m$, $u_i < u'_i$, and $u_j = u'_j$ for all $j < i$. Let $\Gamma_{u,v}(G)$ be the set of Eulerian unlabeled path from u to v in G, and $\mathbf{N}_{u,v}(G)$ the cardinality of $\Gamma_{u,v}(G)$. For $\gamma \in \Gamma_{u,v}(G)$, we denote $\mathrm{Idx}(G, \gamma)$ the number of Eulerian unlabeled paths in $\Gamma_{u,v}(G)$ that appear before γ lexicographically, i.e.,

$$\mathrm{Idx}(G, \gamma) = \left|\left\{\gamma' \in \Gamma_{u,v}(G) : \gamma' < \gamma\right\}\right| \; ; \quad \gamma \in \Gamma_{u,v}(G)\,.$$

For an edge $e \in E$, we denote $G - e$ the graph $G' = (V, E\backslash\{e\})$. Furthermore, for $u, u' \in V$ such that $(u, u') \in E$, denote by $G - (u, u')$ a graph $G' = G - e$, where e is any edge with source u, and destination u'. Letting $\gamma = u_0, u_1 \cdots u_m$, and denoting $\mathrm{tail}(\gamma) = u_1 \cdots u_m$, it is readily verified by induction that the following recurrence holds.

$$\mathrm{Idx}(G, \gamma) = \left(\sum_{u' \in V : u' < u_1, (u_0, u') \in E} \mathbf{N}_{u', u_m}(G - (u_0, u')) \right) + \mathrm{Idx}(G - (u_0, u_1), \mathrm{tail}(\gamma))\,. \quad \text{(F.1)}$$

Given a way of computing $\mathbf{N}_{u,v}(G)$, implementing an algorithm that computes $\mathrm{Idx}(G, \gamma)$ using (F.1) is straightforward, either recursively or iteratively. We next demonstrate that such an algorithm can be implemented with polynomial complexity in the number of edges of G.

F.1. Proposition. $\mathrm{Idx}(G, \gamma)$ *can be computed in a polynomial order number of bit operations with respect to the number of edges of* G*, i.e., regarding* $|V|$ *as constant.*

Proof. In each step of the recurrence in (F.1), the number of edges of the graph is decremented by one. Thus, computing $\mathrm{Idx}(G, \gamma)$ amounts to computing $\mathbf{N}_{u,v}(G)$ up to $|V| \times |E|$ times. By [22], using the same arguments as for the derivation of the formula in Theorem 4.15, we have

$$\mathbf{N}_{u,v}(G) = M \frac{\prod_i \tilde{G}_{i*}!}{\prod_{i,j} \tilde{G}_{i,j}!}\,,$$

where $\tilde{G}$ is the incidence matrix of G, and M denotes the cofactor of entry (v, u) in $I - \hat{G}$, with $\hat{G}_{i,j} = \tilde{G}_{i,j}/\tilde{G}_{i*}$ if $\tilde{G}_{i*} > 0$ and $\hat{\tilde{G}}_{i,j} = 0$ otherwise. Multiplying by $\tilde{G}_{u*}$ the u-th row of $\hat{G}$ for every $u \in V$, we get

$$\mathbf{N}_{u,v}(G) = \left(\prod_{i \neq v} \tilde{G}_{i*} \right)^{-1} M' \frac{\prod_i \tilde{G}_{i*}!}{\prod_{i,j} \tilde{G}_{i,j}!}\,, \quad \text{(F.2)}$$

where M' is the cofactor of entry (v, u) in L, with $L_{u,v} = \delta_{u,v}\tilde{G}_{u*} - \tilde{G}_{u,v}$. The matrix L is integer, and M' can be computed with a number of operations that depends only on $|V|$ with operands of $|V| \log |E|$ bits, which take of course a polynomial number of bit operations with respect to $|E|$.

As for the multinomial coefficients in (F.2), we can compute $\tilde{G}_{i*}!/\prod_{i,j} \tilde{G}_{i,j}!$ as the product of $|V|-1$ binomial coefficient. Each binomial coefficient $\binom{n}{r}$ can be computed, by means of the Pascal triangle (see e.g. [25]), with $r(n-r)$ additions. Thus, working with $|E|$ bits numeric representations, the computation of the multinomial coefficients in (F.2) take $O(|E|^3)$ bit operations.

Finally, again working with $|E|$ bits numeric representations, dividing by $\prod_{i \neq v} \tilde{G}_{i*}$ takes $O(|E|^2)$ bit operations, even with naive implementations of integer multiplications an divisions. Hence, the overall complexity of computing $\mathbf{N}_{u,v}(G)$ is polynomial, and so is the complexity of computing $\mathrm{Idx}(G, \gamma)$. □

The inverse problem of obtaining γ given $\mathrm{Idx}(G, \gamma)$ can also be derived from (F.1) straightforwardly, and yields the same complexity as Proposition F.1 by the same arguments.

In the application of Proposition F.1 to the enumeration of $\mathcal{T}(x^n)$ in Chapter 4, the graph G_F has a set of vertices U that depends only on T. As for the number of edges, $|E_\mu|$, clearly $|E_\mu| = O(n)$ by Lemma 4.13(x). Thus, by Proposition F.1 and Theorem 4.15, $\mathcal{T}(x^n)$ can be enumerated in polynomial time in n.

Appendix G

Proofs for Chapter 5

G.1 Proof of Lemma 5.2

For the proof of Lemma 5.2 we make use of the following theorem from [40], which we present below in a simplified form, which is nevertheless suitable for our setting.

G.1. THEOREM. *[40, Theorem 2]Suppose $\{X_n\}$ is an ergodic finite-state chain with a set of states S and a stationary distribution π. Let $F : S \to \mathbb{R}$ be any bounded function and $\bar{F} = \sup_s |F(s)|$. Then for any $\epsilon > 0$ we have,*

$$\log P\left\{\sum_{i=0}^{n-1}\left(F(X_i) - E_\pi\left[\,F\,\right]\right) \geq n\epsilon\right\} \leq -\frac{n-1}{2}\left(\frac{\epsilon}{d\bar{F}} - \frac{3}{n-1}\right)^2 \tag{G.1}$$

as long as $n \geq 1 + 3d\bar{F}/\epsilon$, where d is a positive constant and $E_\pi\left[\,F\,\right]$ is the expectation of $F(X_i)$ under π.

We prove next the following Lemma G.2, which includes Lemma 5.2 in Part (ii). Part (i) is used in the proof of Lemma 5.8 in Section G.2.

G.2. LEMMA. *Let $\langle T, p_T\rangle$ be a tree source such that all conditional probabilities are positive. Then,*

(i) *For every $k \geq 1$, and $n > N_0$ independent of k,*

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{\frac{Z_{w,a}}{\sqrt{n_w}} \geq k \,\middle|\, n_w > 0\right\} \leq \frac{4}{\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}} \exp\left\{-\frac{n-1}{2}\left(r\frac{k}{2\sqrt{n}} - \frac{3}{n-1}\right)^2\right\} \tag{G.2}$$

where r is a positive constant independent of k.

(ii) *The expectation $E_{\langle T,p_T\rangle}\left[\,\lfloor Z_{w,a}\rfloor^\top\,\right]$ with respect to the tree source $\langle T, p_T\rangle$ is bounded by a constant independent of n.*

Proof. The expectation of $\lfloor Z_{w,a}\rfloor^\top$ is

$$\mathrm{E}_{\langle T,p_T\rangle}\left[\,\lfloor Z_{w,a}\rfloor^\top\,\right] = \sum_{k\geq 1}\mathrm{P}_{\langle T,p_T\rangle}\left\{\lfloor Z_{w,a}\rfloor^\top \geq k\right\}. \tag{G.3}$$

Since $\lfloor Z_{w,a}\rfloor^\top = 0$ when $n_w = 0$ by definition, for $k \geq 1$, we have

$$\begin{aligned}\mathrm{P}_{\langle T,p_T\rangle}\left\{\lfloor Z_{w,a}\rfloor^\top \geq k\right\} &= \mathrm{P}_{\langle T,p_T\rangle}\left\{\lfloor Z_{w,a}\rfloor^\top \geq k, n_w > 0\right\}\\ &= \mathrm{P}_{\langle T,p_T\rangle}\left\{\lfloor Z_{w,a}\rfloor^\top \geq k \,\middle|\, n_w > 0\right\}\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}.\end{aligned}$$

By definition of $\lfloor Z_{w,a} \rfloor^{\top}$, we have

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{ \lfloor Z_{w,a} \rfloor^{\top} \geq k \middle| n_w > 0 \right\} \leq \mathrm{P}_{\langle T,p_T\rangle}\left\{ \frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0 \right\},$$

which, together with (G.3), yields

$$\mathrm{E}_{\langle T,p_T\rangle}\left[\lfloor Z_{w,a} \rfloor^{\top} \right] \leq \mathrm{P}_{\langle T,p_T\rangle}\left\{n_w > 0\right\} \sum_{k\geq 1} \mathrm{P}_{\langle T,p_T\rangle}\left\{ \frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0 \right\}. \tag{G.4}$$

For each term of the summation in the last equation, we have

$$\begin{aligned}
\mathrm{P}_{\langle T,p_T\rangle}\left\{ \frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0 \right\} &= \mathrm{P}_{\langle T,p_T\rangle}\left\{ Z_{w,a} \geq k\sqrt{n_w} \middle| n_w > 0 \right\} \\
&= \mathrm{P}_{\langle T,p_T\rangle}\left\{ \frac{Z_{w,a}}{n_w} \geq \frac{k}{\sqrt{n_w}} \middle| n_w > 0 \right\} \\
&= \mathrm{P}_{\langle T,p_T\rangle}\left\{ \left| \frac{n_w^{(a)}}{n_w} - \frac{n_s^{(a)}}{n_s} \right| \geq \frac{k}{\sqrt{n_w}} \middle| n_w > 0 \right\},
\end{aligned}$$

where the last equality follows from the definition of $Z_{w,a}$. Now, denote by $p_T(a|w)$ the probability of symbol a conditioned on context $\overline{w}$. For the state $s \in S_T$ such that $s \prec w$, we have $p_T(a|w) = p_T(a|s)$, where $p_T(a|s)$ is the probability of symbol a conditioned on state s. Thus, we have

$$\begin{aligned}
&\mathrm{P}_{\langle T,p_T\rangle}\left\{ \frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0 \right\} = \\
&\qquad = \mathrm{P}_{\langle T,p_T\rangle}\left\{ \left| \frac{n_w^{(a)}}{n_w} - p_T(a|w) + p_T(a|s) - \frac{n_s^{(a)}}{n_s} \right| \geq \frac{k}{\sqrt{n_w}} \middle| n_w > 0 \right\} \\
&\qquad \leq \mathrm{P}_{\langle T,p_T\rangle}\left\{ \left| \frac{n_w^{(a)}}{n_w} - p_T(a|w) \right| + \left| p_T(a|s) - \frac{n_s^{(a)}}{n_s} \right| \geq \frac{k}{\sqrt{n_w}} \middle| n_w > 0 \right\},
\end{aligned}$$

and since $n \geq n_w$, we get

$$\begin{aligned}
&\mathrm{P}_{\langle T,p_T\rangle}\left\{ \frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0 \right\} \leq \\
&\qquad \leq \mathrm{P}_{\langle T,p_T\rangle}\left\{ \left| \frac{n_w^{(a)}}{n_w} - p_T(a|w) \right| + \left| p_T(a|s) - \frac{n_s^{(a)}}{n_s} \right| \geq \frac{k}{\sqrt{n}} \middle| n_w > 0 \right\}.
\end{aligned}$$

If $\left| \frac{n_w^{(a)}}{n_w} - p_T(a|w) \right| + \left| p_T(a|s) - \frac{n_s^{(a)}}{n_s} \right| \geq \frac{k}{\sqrt{n}}$, we must have either $\left| \frac{n_w^{(a)}}{n_w} - p_T(a|w) \right| \geq \frac{k}{2\sqrt{n}}$, or $\left| p_T(a|s) - \frac{n_s^{(a)}}{n_s} \right| \geq \frac{k}{2\sqrt{n}}$. Hence, we get

$$\begin{aligned}
\mathrm{P}_{\langle T,p_T\rangle}\left\{ \frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0 \right\} &\leq \mathrm{P}_{\langle T,p_T\rangle}\left\{ \left| \frac{n_w^{(a)}}{n_w} - p_T(a|w) \right| \geq \frac{k}{2\sqrt{n}} \middle| n_w > 0 \right\} \\
&\quad + \mathrm{P}_{\langle T,p_T\rangle}\left\{ \left| p_T(a|s) - \frac{n_s^{(a)}}{n_s} \right| \geq \frac{k}{2\sqrt{n}} \middle| n_w > 0 \right\}.
\end{aligned} \tag{G.5}$$

Since $n_w > 0$ implies $n_s > 0$, the second term of (G.5) is

$$\begin{aligned}
\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|p_T(a|s) - \frac{n_s^{(a)}}{n_s}\right| \geq \frac{k}{2\sqrt{n}}\middle| n_w > 0\right\} = \\
&= \mathrm{P}_{\langle T,p_T\rangle}\left\{\left|p_T(a|s) - \frac{n_s^{(a)}}{n_s}\right| \geq \frac{k}{2\sqrt{n}}\middle| n_w > 0, n_s > 0\right\} \\
&= \frac{\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|p_T(a|s) - \frac{n_s^{(a)}}{n_s}\right| \geq \frac{k}{2\sqrt{n}}\middle| n_s > 0\right\}}{\mathrm{P}_{\langle T,p_T\rangle}\left\{n_w > 0 \middle| n_s > 0\right\}},
\end{aligned}$$

and replacing in (G.5) we get

$$\begin{aligned}
\mathrm{P}_{\langle T,p_T\rangle}\left\{\frac{Z_{w,a}}{\sqrt{n_w}} \geq k\middle| n_w > 0\right\} \quad &\leq \quad \mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| \geq \frac{k}{2\sqrt{n}}\middle| n_w > 0\right\} \\
&+ \quad \frac{\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|p_T(a|s) - \frac{n_s^{(a)}}{n_s}\right| \geq \frac{k}{2\sqrt{n}}\middle| n_s > 0\right\}}{\mathrm{P}_{\langle T,p_T\rangle}\left\{n_w > 0 \middle| n_s > 0\right\}}. \qquad \text{(G.6)}
\end{aligned}$$

We now bound $\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| \geq \epsilon\middle| n_w > 0\right\}$ for any fixed context w with $s \preceq w$. Notice that this includes the case $w = s$, and thus applies to both terms of (G.6). Let p_w and p_{aw} be the probabilities of $\overline{w}$ and $\overline{w}a$ under the stationary distribution of the source T. We have $p_T(a|w) = p_{aw}/p_w$, and therefore

$$\frac{n_w^{(a)}}{n_w} - p_T(a|w) = \frac{n_w^{(a)}/n}{n_w/n} - \frac{p_{aw}}{p_w}.$$

Let $0 < \delta < p_w$. When $p_{aw} - \delta < \frac{n_w^{(a)}}{n} < p_{aw} + \delta$, and $p_w - \delta < \frac{n_w}{n} < p_w + \delta$, we have

$$\frac{p_{aw} - \delta}{p_w + \delta} < \frac{n_w^{(a)}/n}{n_w/n} < \frac{p_{aw} + \delta}{p_w - \delta}.$$

Thus, the event $\left\{\left|\frac{n_w^{(a)}}{n} - p_{aw}\right| < \delta\right\} \cap \left\{\left|\frac{n_w}{n} - p_w\right| < \delta\right\}$ implies

$$\frac{p_{aw}}{p_w} - \left(\frac{p_{aw}}{p_w} - \frac{p_{aw} - \delta}{p_w + \delta}\right) < \frac{n_w^{(a)}/n}{n_w/n} < \frac{p_{aw}}{p_w} + \left(\frac{p_{aw} + \delta}{p_w - \delta} - \frac{p_{aw}}{p_w}\right),$$

or,

$$\frac{p_{aw}}{p_w} - \frac{\delta(p_w + p_{aw})}{p_w(p_w + \delta)} < \frac{n_w^{(a)}/n}{n_w/n} < \frac{p_{aw}}{p_w} + \frac{\delta(p_w + p_{aw})}{p_w(p_w - \delta)}.$$

Since we have $\frac{\delta(p_w + p_{aw})}{p_w(p_w + \delta)} < \frac{\delta(p_w + p_{aw})}{p_w(p_w - \delta)}$, the condition above further implies,

$$\frac{p_{aw}}{p_w} - \frac{\delta(p_w + p_{aw})}{p_w(p_w - \delta)} < \frac{n_w^{(a)}/n}{n_w/n} < \frac{p_{aw}}{p_w} + \frac{\delta(p_w + p_{aw})}{p_w(p_w - \delta)},$$

which in turn gives

$$\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| < \frac{\delta(p_w + p_{aw})}{p_w(p_w - \delta)}\,. \tag{G.7}$$

Now, given $0 < \epsilon \leq 1$, we take $\delta = \epsilon\frac{p_w^2}{p_w+p_{aw}+1}$, which satisfies the condition $0 < \delta < p_w$. Then, $\delta \leq \epsilon\frac{p_w^2}{p_w+p_{aw}+\epsilon p_w}$, and therefore $\frac{\delta(p_w+p_{aw})}{p_w(p_w-\delta)} \leq \epsilon$. Thus, from (G.7) it follows that the event $\left\{\left|\frac{n_w^{(a)}}{n} - p_{aw}\right| < \delta\right\} \cap \left\{\left|\frac{n_w}{n} - p_w\right| < \delta\right\}$ implies $\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| < \epsilon$. Hence, we have

$$\begin{aligned}
&\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| < \epsilon \,\middle|\, n_w > 0\right\} \geq \\
&\qquad\geq \mathrm{P}_{\langle T,p_T\rangle}\left\{\left\{\left|\frac{n_w^{(a)}}{n} - p_{aw}\right| < \delta\right\} \cap \left\{\left|\frac{n_w}{n} - p_w\right| < \delta\right\} \,\middle|\, n_w > 0\right\},
\end{aligned}$$

and by De Morgan's law, we get

$$\begin{aligned}
&\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| > \epsilon \,\middle|\, n_w > 0\right\} \leq \\
&\qquad\leq \mathrm{P}_{\langle T,p_T\rangle}\left\{\left\{\left|\frac{n_w^{(a)}}{n} - p_{aw}\right| > \delta\right\} \cup \left\{\left|\frac{n_w}{n} - p_w\right| > \delta\right\} \,\middle|\, n_w > 0\right\} \\
&\qquad= \frac{\mathrm{P}_{\langle T,p_T\rangle}\left\{\left\{\left|\frac{n_w^{(a)}}{n} - p_{aw}\right| > \delta\right\} \cup \left\{\left|\frac{n_w}{n} - p_w\right| > \delta\right\}\right\}}{\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}},
\end{aligned}$$

which we further bound as

$$\begin{aligned}
&\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| > \epsilon \,\middle|\, n_w > 0\right\} \leq \\
&\qquad\leq \frac{\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n} - p_{aw}\right| > \delta\right\} + \mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w}{n} - p_w\right| > \delta\right\}}{\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}}\,.
\end{aligned}$$

We now apply Theorem G.1, from [40], which gives

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| > \epsilon \,\middle|\, n_w > 0\right\} \leq \frac{2}{\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}} \exp\left\{-\frac{n-1}{2}\left(\frac{\delta}{d} - \frac{3}{n-1}\right)^2\right\},$$

as long as $n \geq 1 + \frac{3d}{\delta}$, where $\delta = \epsilon\frac{p_w^2}{p_w+p_{aw}+1}$ as before, and d is a positive constant. We can rewrite the last equation as

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|\frac{n_w^{(a)}}{n_w} - p_T(a|w)\right| > \epsilon \,\middle|\, n_w > 0\right\} \leq \frac{2}{\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}} \exp\left\{-\frac{n-1}{2}\left(r\epsilon - \frac{3}{n-1}\right)^2\right\}, \tag{G.8}$$

as long as $n \geq 1 + \frac{r'}{\epsilon}$, where r and r' are positive constants. Going back to equation (G.6), for $1 \leq k \leq 2\sqrt{n}$ we get from (G.8) with $\epsilon = \frac{k}{2\sqrt{n}}$, for all $n \geq 1 + r'\frac{2\sqrt{n}}{k}$,

$$\begin{aligned}\mathrm{P}_{\langle T,p_T\rangle}\left\{\frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0\right\} \quad &\leq \quad 2\frac{\exp\left\{-\frac{n-1}{2}\left(r\frac{k}{2\sqrt{n}} - \frac{3}{n-1}\right)^2\right\}}{\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}} \\ &\quad + 2\frac{\exp\left\{-\frac{n-1}{2}\left(r\frac{k}{2\sqrt{n}} - \frac{3}{n-1}\right)^2\right\}}{\mathrm{P}_{\langle T,p_T\rangle}\{n_s > 0\}\,\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0 | \, n_s > 0\}},\end{aligned}$$

which becomes

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{\frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0\right\} \leq \frac{4}{\mathrm{P}_{\langle T,p_T\rangle}\{n_w > 0\}} \exp\left\{-\frac{n-1}{2}\left(r\frac{k}{2\sqrt{n}} - \frac{3}{n-1}\right)^2\right\}. \tag{G.9}$$

The condition $n \geq 1 + r'\frac{2\sqrt{n}}{k}$ holds true for every n greater than a constant N_0 independent of k. On the other hand, for $k > 2\sqrt{n}$, it follows from (G.5) that $\mathrm{P}_{\langle T,p_T\rangle}\left\{\frac{Z_{w,a}}{\sqrt{n_w}} \geq k \middle| n_w > 0\right\} = 0$. Thus, (G.9) holds in fact for every $k \geq 1$ and every $n > N_0$. This concludes the proof of (i). We can now bound the exponent in (G.9) as

$$\frac{n-1}{2}\left(r\frac{k}{2\sqrt{n}} - \frac{3}{n-1}\right)^2 \geq Ck^2 \quad \text{with } C > 0 \text{ for } n > N_0',$$

and going back to (G.4) we get

$$\mathrm{E}_{\langle T,p_T\rangle}\left[\,\lfloor Z_{w,a}\rfloor^\top\,\right] \leq 4\sum_{k\geq 1}\exp\left\{-Ck^2\right\} \leq 4\sum_{k\geq 1}\exp\left\{-Ck\right\},$$

which is a convergent series. □

G.2 Proof of Lemma 5.8

Suppose $\left|m_i - \frac{n_{i-1}^\alpha}{n_{i-1}} m_{i-1}\right| \leq k\sqrt{m_{l+1}}$ for all $l < i \leq t$, where we extend the definition of m_i for $i = t$ as $m_t = n_{\overline{u^t}}$. Then, we have

$$\frac{n_{i-1}^\alpha}{n_{i-1}} m_{i-1} - k\sqrt{m_{l+1}} \leq m_i \leq \frac{n_{i-1}^\alpha}{n_{i-1}} m_{i-1} + k\sqrt{m_{l+1}}.$$

Applying the same inequalities for m_{i-1} we obtain

$$\frac{n_{i-1}^\alpha}{n_{i-1}}\left(\frac{n_{i-2}^\alpha}{n_{i-2}} m_{i-2} - k\sqrt{m_{l+1}}\right) - k\sqrt{m_{l+1}} \leq m_i \leq \frac{n_{i-1}^\alpha}{n_{i-1}}\left(\frac{n_{i-2}^\alpha}{n_{i-2}} m_{i-2} + k\sqrt{m_{l+1}}\right) + k\sqrt{m_{l+1}}.$$

Thus, we have

$$\frac{n_{i-1}^\alpha}{n_{i-1}}\frac{n_{i-2}^\alpha}{n_{i-2}} m_{i-2} - \left(1 + \frac{n_{i-1}^\alpha}{n_{i-1}}\right) k\sqrt{m_{l+1}} \leq m_i \leq \frac{n_{i-1}^\alpha}{n_{i-1}}\frac{n_{i-2}^\alpha}{n_{i-2}} m_{i-2} + \left(1 + \frac{n_{i-1}^\alpha}{n_{i-1}}\right) k\sqrt{m_{l+1}},$$

and, successively applying the same reasoning, we conclude that

$$m_{l+1}\prod_{i=l+1}^{t-1}\frac{n_i^\alpha}{n_i}-\left(1+\sum_{j=l+2}^{t-1}\prod_{i=j}^{t-1}\frac{n_i^\alpha}{n_i}\right)k\sqrt{m_{l+1}}\le m_t\le m_{l+1}\prod_{i=l+1}^{t-1}\frac{n_i^\alpha}{n_i}+\left(1+\sum_{j=l+2}^{t-1}\prod_{i=j}^{t-1}\frac{n_i^\alpha}{n_i}\right)k\sqrt{m_{l+1}}\,.$$

Since $\frac{n_i^\alpha}{n_i}\le 1$ for all i, we further bound

$$m_{l+1}\prod_{i=l+1}^{t-1}\frac{n_i^\alpha}{n_i}-(t-l+1)k\sqrt{m_{l+1}}\le m_t\le m_{l+1}\prod_{i=l+1}^{t-1}\frac{n_i^\alpha}{n_i}+(t-l+1)k\sqrt{m_{l+1}}\,.$$

Hence, the event $\left\{\left|m_i-\frac{n_{i-1}^\alpha}{n_{i-1}}m_{i-1}\right|\le k\sqrt{m_{l+1}}\text{ for all }l<i\le t\right\}$ implies that $\left\{\left|m_t-m_{l+1}\prod_{i=l+1}^{t-1}\frac{n_i^\alpha}{n_i}\right|\le k(t-l+1)\sqrt{m_{l+1}}\right\}$. Since $m_t=n_{\overline{u}}$, we have

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{Z_u\ge k(t-l+1)\sqrt{m_{l+1}}\middle|\,A\right\}\le\sum_{i=l+1}^{t}\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|m_i-\frac{n_{i-1}^\alpha}{n_{i-1}}m_{i-1}\right|\ge k\sqrt{m_{l+1}}\middle|\,A\right\},$$

where A denotes the event $\{m_{l+1}>0, n_i>0\ \forall i: l<i<t\}$. Since u^{l+1} is a substring of u^{i-1} for $l<i\le t$, we get $m_{l+1}\ge m_{i-1}$. Thus, we have

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{Z_u\ge k(t-l+1)\sqrt{m_{l+1}}\middle|\,A\right\}\le\sum_{i=l+1}^{t}\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|m_i-\frac{n_{i-1}^\alpha}{n_{i-1}}m_{i-1}\right|\ge k\sqrt{m_{i-1}}\middle|\,A\right\}. \tag{G.10}$$

Each term of the summation in (G.10) is

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|m_i-\frac{n_{i-1}^\alpha}{n_{i-1}}m_{i-1}\right|\ge k\sqrt{m_{i-1}}\middle|\,A\right\}=\frac{\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|m_i-\frac{n_{i-1}^\alpha}{n_{i-1}}m_{i-1}\right|\ge k\sqrt{m_{i-1}}\middle|\,n_{i-1}>0\right\}}{\mathrm{P}_{\langle T,p_T\rangle}\{A|\,n_{i-1}>0\}}. \tag{G.11}$$

Now, $m_i=n_{\overline{u^i}}=n_{\frac{(u_i)}{u^{i-1}}}$ and by Lemma G.2 we get

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{\left|m_i-\frac{n_{i-1}^\alpha}{n_{i-1}}m_{i-1}\right|\ge k\sqrt{m_{i-1}}\middle|\,n_{i-1}>0\right\}\le$$
$$\le\frac{4}{\mathrm{P}_{\langle T,p_T\rangle}\{n_{i-1}>0\}}\exp\left\{-\frac{n-1}{2}\left(r\frac{k}{2\sqrt{n}}-\frac{3}{n-1}\right)^2\right\}.$$

Replacing in (G.11), and then in (G.10), we get

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{Z_u\ge k(t-l+1)\sqrt{m_{l+1}}\middle|\,A\right\}\le\frac{(t-l)4}{\mathrm{P}_{\langle T,p_T\rangle}\{A\}}\exp\left\{-\frac{n-1}{2}\left(r\frac{k}{2\sqrt{n}}-\frac{3}{n-1}\right)^2\right\}.$$

Thus, we have

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{Z_u\ge k\sqrt{m_{l+1}}\middle|\,A\right\}\le\frac{(t-l)4}{\mathrm{P}_{\langle T,p_T\rangle}\{A\}}\exp\left\{-\frac{n-1}{2}\left(\frac{r}{t-l+1}\frac{k}{2\sqrt{n}}-\frac{3}{n-1}\right)^2\right\}.$$

The proof then follows exactly as in Lemma G.2.

□

G.3 Proof of Lemma 5.10

Suppose the claim of Lemma 5.10 is not true and let $z \in \mathcal{A}^*$, $c \in \mathcal{A}$ such that $zc \in U_{k+1}$ is of maximal length among those elements of U_{k+1} that are not leaves of $T_{\mathbf{c}}^{[k]}$. Since the children of zc were added to $T_{\mathbf{c}}^{[k+1]}$ for zc was forgetful, we know that azc is an internal node of $T_{\mathbf{c}}^{[k+1]}$ for every $a \in \mathcal{A}$. If $azc \in U_{k+1}$, azc is a leaf of $T_{\mathbf{c}}^{[k]}$ for azc is longer than zc. Therefore, az is an internal node of $T_{\mathbf{c}}^{[k]}$. If $azc \notin U_{k+1}$, $azcd \notin U'_{k+1}$ for any $d \in \mathcal{A}$. Hence, by definition of U'_{k+1}, either $azcd \in T_{\mathbf{c}}^{[k]}$ or $azcd \in T^{[k+1]}$. In any case we have that $azc \in T_{\mathbf{c}}^{[k]}$, i.e., az is an internal node of $T_{\mathbf{c}}^{[k]}$. We conclude that az is an internal node of $T_{\mathbf{c}}^{[k]}$ for every $a \in \mathcal{A}$, thus $zc \in T_{\mathbf{c}}^{[k]}$ by definition of canonical context tree. Furthermore, since $zc \in U_{k+1}$, $zcd \in U'_{k+1}$ for some $d \in \mathcal{A}$, thus $zcd \notin T_{\mathbf{c}}^{[k]}$ and zc is a state of $T_{\mathbf{c}}^{[k]}$, a contradiction. □

G.4 Proof of Lemma 5.13

We will equate the number of counts given in each iteration of the loop to the increment in $|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|$ with respect to $|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|$. For $u \in \mathcal{A}^*$ we define $V_i(u) = \{uv \in S_{\mathbf{c}}^{[i]} : v \in \mathcal{A} \cup \{\lambda\}\}$, $A_i(u) = \{(uv, w) \in E_{T_{\mathbf{c}}^{[i]}} : v \in \mathcal{A} \cup \{\lambda\}\}$ and, for $a \in \mathcal{A}$, $A_i^{(a)}(u) = \{(uv, w) \in A_i(u) : a = \text{head}(w)\}$. Notice that since $T_{\mathbf{c}}^{[k+1]}$ refines states of $T_{\mathbf{c}}^{[k]}$ in at most one level, $A_k(r)$ and $V_k(r)$ exhaust $E_{T_{\mathbf{c}}^{[k]}}$ and $V_{T_{\mathbf{c}}^{[k]}}$ respectively as r varies in R_{k+1}. Of course, $A_{k+1}(r)$ and $V_{k+1}(r)$ do also exhaust $E_{T_{\mathbf{c}}^{[k+1]}}$ and $V_{T_{\mathbf{c}}^{[k+1]}}$ respectively as r varies in R_{k+1}.

We claim that the number of counts given by an invocation to $\mathtt{P}(r, c)$ equals $|A_{k+1}^{(c)}(r)| - |A_k^{(c)}(r)|$. When the condition of Step 1 holds true, we describe $\alpha - 1$ counts. There are α edges from children of r to children of cr in $A_{k+1}(r)$ and one edge from r to cr in $A_k(r)$, i.e., the number of given counts coincides with the increment $|A_{k+1}^{(c)}(r)| - |A_k^{(c)}(r)|$. Now, when the condition of Step 5 is satisfied, we have for each $csu \in W$, that there is an increment of $\alpha - 1$ in the number of edges that depart from s to children of csu in $A_{k+1}(r)$ with respect to the one single edge from $\sigma_{\mathbf{c}}^{[k]}(\overline{s})$ to csu in $A_k(r)$. The increment coincides with the number of counts given in Step 9. On the other hand, in Step 12, where csv is a state of $T_{\mathbf{c}}^{[k+1]}$ that is not in W', csv is also a state of $T_{\mathbf{c}}^{[k]}$. There is one edge from s to csv in $A_{k+1}(r)$ and also one edge from $\sigma_{\mathbf{c}}^{[k]}(\overline{s})$ to csv in $A_k(r)$. Thus, there is no increment in the number of edges. Finally, in Step 14, we have that cs is a leaf of $T_{\mathbf{c}}^{[k+1]}$. Then, as mentioned, either $cs \in T_{\mathbf{c}}^{[k]}$ or $s \in T_{\mathbf{c}}^{[k]}$, for otherwise, by Corollary 5.11, their parents cr, r, would belong to $T_{\mathbf{c}}^{[k]}$ and the condition of Step 1 would hold true. When $cs \in T_{\mathbf{c}}^{[k]}$, there is one edge from s to cs in $A_{k+1}(r)$ and also one edge from $\sigma_{\mathbf{c}}^{[k]}(\overline{s})$ to cs in $A_k(r)$. If, on the other hand, $s \in T_{\mathbf{c}}^{[k]}$, there is one edge from s to cs in $A_{k+1}(r)$ and also one edge from s to $\sigma_{\mathbf{c}}^{[k]}(\overline{cs})$ in $A_k(r)$. The claim is proved.

We now analyze **`RefineTypeClass`**. When $r \in U_{k+1}$, the number of counts described in Step 4 is $|A_{k+1}(r) \setminus A_{k+1}^{(d)}(r)| - |A_k(r) \setminus A_k^{(d)}(r)|$. For the symbols $d \in \mathcal{A}$ of Step 3, we have that dr is not an internal node of $T_{\mathbf{c}}^{[k]}$ but, since $r \in U_{k+1}$, dr is an internal node of $T_{\mathbf{c}}^{[k+1]}$. Then, dr is a leaf of $T_{\mathbf{c}}^{[k]}$ and the full set of children of dr are leaves of $T_{\mathbf{c}}^{[k+1]}$. There are α edges from the children of r to the children of dr in $A_{k+1}^{(d)}(r)$ and one single edge from r to dr

in $A_k^{(d)}(r)$. Hence, the number of counts described in Step 4 is $|A_{k+1}(r)| - |A_k(r)| - (\alpha - 1)$. Since $|V_{k+1}(r)| - |V_k(r)| = \alpha - 1$ we have that the total number of counts described is $(|A_{k+1}(r)| - |V_{k+1}(r)|) - (|A_k(r)| - |V_k(r)|)$. We now consider the case where $r \notin U_{k+1}$ and the children of r belong to $T_{\mathbf{c}}^{[k]}$. When $cs \notin T_{\mathbf{c}}^{[k+1]}$, the decoder computes state transition counts in Step 9. In this case, for every state $s \in S_{\mathbf{c}}^{[k+1]}$ child of r, there is one edge from s to $s' = \sigma_{\mathbf{c}}^{[k+1]}(\overline{cs})$ in $A_{k+1}(r)$ and, since also $s \in S_{\mathbf{c}}^{[k]}$, there is also one edge from $\sigma_{\mathbf{c}}^{[k]}(\bar{s})$ to $\sigma_{\mathbf{c}}^{[k]}(\overline{cs})$ in $A_k(r)$. Hence, $|A_{k+1}^{(c)}(r)| = |A_k^{(c)}(r)|$. For the remaining values of c, we use $|A_{k+1}^{(c)}(r)| - |A_k^{(c)}(r)|$ counts in Step 11. Since the children of r belong to $T_{\mathbf{c}}^{[k]}$, we have $|V_{k+1}(r)| = |V_k(r)|$. Thus, the total number of counts is $(|A_{k+1}(r)| - |V_{k+1}(r)|) - (|A_k(r)| - |V_k(r)|)$. Finally, when the algorithm skips to Step 12, we have that all states s that are children of r do not belong to $T_{\mathbf{c}}^{[k]}$, and do not belong to U'_{k+1}, for $r \notin U_{k+1}$ in Step 2. Hence, by the definition of U'_{k+1}, all states s that are children of r belong to $T^{[k+1]} \backslash T_{\mathbf{c}}^{[k]}$ and, thus, $|s| = k+1$ and $\overline{as}$ is sufficiently long to determine a state in $T_{\mathbf{c}}^{[k+1]}$ for every symbol a. There are α edges in $A_{k+1}(r)$ departing from each of the α children of r, for a total of α^2 edges in $A_{k+1}(r)$. On the other hand, r is a leaf of maximal length in $T_{\mathbf{c}}^{[k]}$ and therefore there are α edges departing from r in $A_k(r)$. Since $|V_{k+1}(r)| = \alpha$ and $|V_k(r)| = 1$, we have $(|A_{k+1}(r)| - |V_{k+1}(r)|) - (|A_k(r)| - |V_k(r)|) = (\alpha - 1)^2$.

Over all, the number of counts described is $\left(|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|\right) - \left(|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|\right) - B_{k+1}(\alpha - 1)^2$ where B_{k+1} is the number of elements r in R_{k+1} with $|r| = k$. Clearly, the children in $T_{\mathbf{c}}^{[k+1]}$ of such elements of R_{k+1} are the nodes in $T^{[k+1]} \backslash T^{[k]}$ and we can write $B_{k+1} = \left(|S_{T^{[k+1]}}| - |S_{T^{[k]}}|\right)/(\alpha - 1)$. It follows that the number of counts described can be written as $\left(|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|\right) - \left(|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|\right) - (\alpha - 1)\left(|S_{T^{[k+1]}}| - |S_{T^{[k]}}|\right)$.

The total number of counts described by **`EncodeTypeClass`** is, therefore,

$$\sum_{k=h+1}^{d-1} \left(|E_{T_{\mathbf{c}}^{[k+1]}}| - |V_{T_{\mathbf{c}}^{[k+1]}}|\right) - \left(|E_{T_{\mathbf{c}}^{[k]}}| - |V_{T_{\mathbf{c}}^{[k]}}|\right) - (\alpha - 1)\left(|S_{T^{[k+1]}}| - |S_{T^{[k]}}|\right), \tag{G.12}$$

where we recall that $d = \max\{|s| : s \in S_T\}$ and $h = \min\{|s| : s \in S_T\}$. The telescopic sum in (G.12) reduces to

$$(|E_{T_{\mathbf{c}}}| - |V_{T_{\mathbf{c}}}|) - \left(|E_{T_{\mathbf{c}}^{[h+1]}}| - |V_{T_{\mathbf{c}}^{[h+1]}}|\right) - (\alpha - 1)\left(|S_T| - |S_{T^{[h+1]}}|\right). \tag{G.13}$$

Now, since $T^{[h+1]}$ is FSM, $V_{T_{\mathbf{c}}^{[h+1]}} = V_{T^{[h+1]}} = S_{T^{[h+1]}}$, $E_{T_{\mathbf{c}}^{[h+1]}} = E_{T^{[h+1]}}$, and $|E_{T^{[h+1]}}| = \alpha|S_{T^{[h+1]}}|$, so that $|E_{T_{\mathbf{c}}^{[h+1]}}| - |V_{T_{\mathbf{c}}^{[h+1]}}| = (\alpha - 1)|S_{T^{[h+1]}}|$ and (G.13) becomes

$$(|E_{T_{\mathbf{c}}}| - |V_{T_{\mathbf{c}}}|) - (\alpha - 1)|S_T|. \tag{G.14}$$

□

Appendix H

Probability of context tree estimation error

H.1 Proof of Lemma 5.17

Consider a fixed state $s \in S_T$, and a sequence x^n with $\hat{T} = \hat{T}(x^n)$ such that s is an internal node of $\hat{T}$. Let $W = \{sw \in \hat{T} : w \in \mathcal{A}^+\}$, and S_W the set of leaves descending from s in $\hat{T}$. Define the context trees $T' = T \cup W$, and $\hat{T}' = \hat{T} \backslash W$. The context trees T' and T differ only in that s is refined by a subtree W in T' and the same occurs between $\hat{T}'$ and $\hat{T}$. Thus,

$$\begin{aligned} -\log \hat{\mathrm{P}}_{\hat{T}}(x^n) + \log \hat{\mathrm{P}}_{\hat{T}'}(x^n) &= -\sum_{su \in S_W} \sum_{a \in \mathcal{A}} n_{su}^{(a)}(x^n) \log \frac{n_{su}^{(a)}(x^n)}{n_{su}(x^n)} + \sum_{a \in \mathcal{A}} n_s^{(a)}(x^n) \log \frac{n_s^{(a)}(x^n)}{n_s(x^n)} \\ &= -\log \hat{\mathrm{P}}_{T'}(x^n) + \log \hat{\mathrm{P}}_{T}(x^n) \,. \end{aligned} \tag{H.1}$$

Since $\hat{T}$ is the estimated context tree for x^n, by (5.15) we have

$$-\log \hat{\mathrm{P}}_{\hat{T}}(x^n) + \log \hat{\mathrm{P}}_{\hat{T}'}(x^n) \leq (K_{\hat{T}'} - K_{\hat{T}}) f(n) \,. \tag{H.2}$$

Also by linearity of the penalization coefficient,

$$K_{\hat{T}'} - K_{\hat{T}} = K_T - K_{T'} = \beta(|S_T| - |S_{T'}|) = -\beta(|S_W| - 1) \,, \tag{H.3}$$

and replacing (H.3) in (H.2), we get

$$-\log \hat{\mathrm{P}}_{T'}(x^n) + \log \hat{\mathrm{P}}_{T}(x^n) \leq -\beta(|S_W| - 1) f(n) \,. \tag{H.4}$$

Define the probability distribution Q_T over $\mathcal{A}^n$, as that induced by the probability assignment obtained by replacing the conditional probability $p_T(\cdot|s)$ in the model $\langle T, p_T \rangle$, by the empirical distribution in s of x^n, i.e.,

$$\log Q_T(y^n) = \sum_{t \in S_T \backslash \{s\}} \sum_{a \in \mathcal{A}} n_t^{(a)}(y^n) \log p_T(a|t) + \sum_{a \in \mathcal{A}} n_s^{(a)}(y^n) \log \frac{n_s^{(a)}(x^n)}{n_s(x^n)} \,. \tag{H.5}$$

Similarly, define $Q_{T'}$ such that

$$\log Q_{T'}(y^n) = \sum_{t \in S_T \backslash \{s\}} \sum_{a \in \mathcal{A}} n_t^{(a)}(y^n) \log p_T(a|t) + \sum_{su \in S_W} \sum_{a \in \mathcal{A}} n_{su}^{(a)}(y^n) \log \frac{n_{su}^{(a)}(x^n)}{n_{su}(x^n)} \,. \tag{H.6}$$

Let $\mathcal{T}_{s,T'}(x^n)$ be the *(s,T')-type class* of x^n, defined as the set of sequences of $\mathcal{A}^n$ with the same symbol occurrence counts as x^n with respect to the states descending from s in T',

$$\mathcal{T}_{s,T'}(x^n) = \{y^n \in \mathcal{A}^n : n_{su}^{(a)}(y^n) = n_{su}^{(a)}(x^n) \text{ for all } su \in S_W, \text{ and all } a \in \mathcal{A}\} \,.$$

From (H.5) and (H.6) we get

$$-\log Q_{T'}(y^n) + \log Q_T(y^n) = -\sum_{su\in S_W}\sum_{a\in\mathcal{A}} n_{su}^{(a)}(y^n)\log\frac{n_{su}^{(a)}(x^n)}{n_{su}(x^n)} + \sum_{a\in\mathcal{A}} n_s^{(a)}(y^n)\log\frac{n_s^{(a)}(x^n)}{n_s(x^n)}\,. \tag{H.7}$$

Hence, $-\log\frac{Q_{T'}(y^n)}{Q_T(y^n)}$ is constant within $\mathcal{T}_{s,T'}(x^n)$, or equivalently $\frac{Q_{T'}(y^n)}{Q_T(y^n)} = \frac{Q_{T'}(x^n)}{Q_T(x^n)}$ for all $y^n \in \mathcal{T}_{s,T'}(x^n)$. Therefore,

$$-\log\frac{\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_{T'}(y^n)}{\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_T(y^n)} = -\log\frac{Q_{T'}(y^n)}{Q_T(y^n)}\frac{\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_T(y^n)}{\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_T(y^n)} = -\log\frac{Q_{T'}(x^n)}{Q_T(x^n)}\,. \tag{H.8}$$

By (H.1) and (H.7), we have

$$-\log Q_{T'}(x^n) + \log Q_T(x^n) = -\log\hat{\mathrm{P}}_{T'}(x^n) + \log\hat{\mathrm{P}}_T(x^n)\,, \tag{H.9}$$

and, thus, from (H.4) and (H.8), we get

$$-\log\left(\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_{T'}(y^n)\right) + \log\left(\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_T(y^n)\right) \le -\beta(|S_W|-1)f(n)\,. \tag{H.10}$$

Since Q_T maximizes the probability of the sub-sequence of x^n that occurs in state s, we have $Q_T(y^n) \ge \mathrm{P}_{\langle T,p_T\rangle}(y^n)$ for every $y^n \in \mathcal{T}_{s,T'}(x^n)$. Thus,

$$-\log\left(\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_{T'}(y^n)\right) + \log\mathrm{P}_{\langle T,p_T\rangle}\left\{\mathcal{T}_{s,T'}(x^n)\right\} \le -\beta(|S_W|-1)f(n)\,, \tag{H.11}$$

where we use the notation $\mathrm{P}_{\langle T,p_T\rangle}\{\Omega\}$ for $\Omega\in\mathcal{A}^n$ as a shorthand for $\mathrm{P}_{\langle T,p_T\rangle}\{X^n\in\Omega\}$. Since also $\sum_{y^n\in\mathcal{T}_{s,T'}(x^n)} Q_{T'}(y^n) \le 1$, we further get

$$\log\mathrm{P}_{\langle T,p_T\rangle}\left\{\mathcal{T}_{s,T'}(x^n)\right\} \le -\beta(|S_W|-1)f(n)\,,$$

or,

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{\mathcal{T}_{s,T'}(x^n)\right\} \le 2^{-\beta(|S_W|-1)f(n)}\,. \tag{H.12}$$

Let $O_{s,W}^n \subset \mathcal{A}^n$ be the set of sequences whose estimated context tree refines s with the subtree W, and let $\mathcal{P}_{s,T'}$ be the set of possible $(s,T')-$type classes of sequences from $\mathcal{A}^n$. Then, we have

$$O_{s,W}^n \subset \bigcup_{\mathcal{T}\in\mathcal{P}_{s,T'}}\ \bigcup_{x^n\in O_{s,W}^n\cap\mathcal{T}} \mathcal{T}_{s,T'}(x^n)\,.$$

Since there are at most $n^{\alpha|S_W|}$ elements in $\mathcal{P}_{s,T'}$, we get, by (H.12),

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{O_{s,W}^n\right\} \le n^{\alpha|S_W|}2^{-\beta(|S_W|-1)f(n)} = 2^{|S_W|(\alpha\log n-\beta f(n))+\beta f(n)}\,. \tag{H.13}$$

Let $O_{s,k}^n = \bigcup_{|S_W|=k} O_{s,W}^n$ be the set of sequences whose estimated context tree refines s with $k \ge \alpha$ states. The number of nodes in the subtree that refines s is $\frac{\alpha k-1}{\alpha-1} \le 2k$. By using

a natural code [56, 88], a full tree can be described with as many bits as the number of nodes. Thus, the number of different subtrees that refine s with k states is bounded by 2^{2k}. Then, we get from (H.13)

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{O^n_{s,k}\right\} \le 2^{k(\alpha\log n-\beta f(n)+2)+\beta f(n)}\,. \tag{H.14}$$

Notice that when $\beta f(n) > \alpha\log n + 2$, the exponent in the last equation is a decreasing function of k, and since $k \ge \alpha$, we have

$$\mathrm{P}_{\langle T,p_T\rangle}\left\{O^n_{s,k}\right\} \le 2^{\alpha(\alpha\log n-\beta f(n)+2)+\beta f(n)} = n^{\alpha^2}2^{\beta(1-\alpha)f(n)+2\alpha}\,. \tag{H.15}$$

Let O^n be the set of sequences whose estimated context tree refines as least one state of T. Then, $O^n \subset \bigcup_{s\in S_T}\bigcup_{k\ge\alpha} O^n_{s,k}$ and by (H.15) we have

$$\mathrm{P}_{\langle T,p_T\rangle}\{O^n\} \le \sum_{s\in S_T}\sum_{k\ge\alpha} n^{\alpha^2}2^{\beta(1-\alpha)f(n)+2\alpha}\,. \tag{H.16}$$

The number of states in $\hat{T}(x^n)$ is bounded by n and we conclude that

$$\mathrm{P}_{\langle T,p_T\rangle}\{O^n\} \le |S_T|n^{\alpha^2+1}2^{\beta(1-\alpha)f(n)+2\alpha}\,. \tag{H.17}$$

□

H.2 Proof of Lemma 5.18

Let w be an internal node of T, and let $W = \{wu \in T : u \in \mathcal{A}^+\}$ be the subtree of T descending from w (excluding w). Similarly, let $W_F = \{wu \in T_{\text{suf}} : u \in \mathcal{A}^+\}$ be the subtree of the FSM closure T_{suf} of T descending from w. Let also $S_W = S_T \cap W$ be the set of states of T descending from w, and $S_{W_F} = S_{T_{\text{suf}}} \cap W_F$ be the set of states of T_{suf} descending from w.

Consider a sequence x^n such that w is a state in $\hat{T} = \hat{T}(x^n)$, and define the context trees $\hat{T}' = \hat{T} \cup W_F$, $T' = T \cup W_F$ and $T'' = T\backslash W_F$. The context trees T'' and T' differ only in that state w of T'' is refined by a subtree W_F in T', and the same occurs between $\hat{T}$ and $\hat{T}'$. Thus,

$$-\log\hat{\mathrm{P}}_{\hat{T}}(x^n) + \log\hat{\mathrm{P}}_{\hat{T}'}(x^n) = -\log\hat{\mathrm{P}}_{T''}(x^n) + \log\hat{\mathrm{P}}_{T'}(x^n)\,. \tag{H.18}$$

Since $\hat{T}$ is the estimated context tree for x^n, by (5.15), we have

$$-\log\hat{\mathrm{P}}_{\hat{T}}(x^n) + \log\hat{\mathrm{P}}_{\hat{T}'}(x^n) \le (K_{\hat{T}'} - K_{\hat{T}})f(n)\,. \tag{H.19}$$

Also by linearity of the penalization coefficient,

$$K_{\hat{T}'} - K_{\hat{T}} = K_{T'} - K_{T''} = \beta(|S_{T'}| - |S_{T''}|) = \beta(|S_{W_F}| - 1)\,, \tag{H.20}$$

and replacing (H.20) in (H.19), and using (H.18), we get

$$\Omega_w(x^n) \triangleq -\log\hat{\mathrm{P}}_{T''}(x^n) + \log\hat{\mathrm{P}}_{T'}(x^n) \le \beta(|S_{W_F}| - 1)f(n)\,. \tag{H.21}$$

Notice that, since T' and T'' differ only in that state w of T'' is refined by the set S_{W_F} of states in T', we have

$$\begin{aligned} -\log \hat{\mathrm{P}}_{T''}(x^n) + \log \hat{\mathrm{P}}_{T'}(x^n) &= -\sum_{a\in\mathcal{A}} n_w^{(a)} \log \frac{n_w^{(a)}}{n_w} + \sum_{wu\in S_{W_F}} \sum_{a\in\mathcal{A}} n_{wu}^{(a)} \log \frac{n_{wu}^{(a)}}{n_{wu}} \\ &= -\sum_{wu\in S_{W_F}} \sum_{a\in\mathcal{A}} n_{wu}^{(a)} \log \frac{n_w^{(a)}}{n_w} + \sum_{wu\in S_{W_F}} \sum_{a\in\mathcal{A}} n_{wu}^{(a)} \log \frac{n_{wu}^{(a)}}{n_{wu}} \\ &= \sum_{wu\in S_{W_F}} \sum_{a\in\mathcal{A}} n_{wu}^{(a)} \log \frac{n_{wu}^{(a)}/n_{wu}}{n_w^{(a)}/n_w} \,. \end{aligned} \tag{H.22}$$

Define $U_{w,\epsilon}^n$ as

$$U_{w,\epsilon}^n = \left\{ x^n \in \mathcal{A}^n : \frac{\Omega_w(x^n)}{n} \le \epsilon(w) \right\} . \tag{H.23}$$

Since $f(n) = o(n)$, it is clear from (H.21) that, if $\epsilon(w) > 0$ for all $w \in \mathcal{I}(T)$, for n sufficiently large, we have

$$U^n \subset \bigcup_{w\in\mathcal{I}(T)} U_{w,\epsilon}^n \,. \tag{H.24}$$

Consider the empirical distribution over $S_{T_{\mathrm{suf}}} \times S_{T_{\mathrm{suf}}}$ defined as

$$\hat{\mathrm{P}}_{F,x^n}(s,z) = \begin{cases} \frac{n_s^{(a)}}{n}, & \text{if } f(s,a) = z, \\ 0, & \text{otherwise,} \end{cases}$$

where $f : S_{T_{\mathrm{suf}}} \times \mathcal{A} \to S_{T_{\mathrm{suf}}}$ is the next-state function of T_{suf}. For a distribution Q over $S_{T_{\mathrm{suf}}} \times S_{T_{\mathrm{suf}}}$, let $\bar{Q}$ denote its left marginal, and define

$$Q(s|z) = \frac{Q(z,s)}{\bar{Q}(z)}, \qquad \bar{Q}(z) \neq 0 \,.$$

Furthermore, for $w \in \mathcal{I}(T)$ define

$$Q(s|w) = \frac{\sum_{z\in S_{W_F}} Q(z,s)}{\sum_{z\in S_{W_F}} \bar{Q}(z)}, \qquad \sum_{z\in S_{W_F}} \bar{Q}(z) \neq 0 \,.$$

Let Γ denote the set of distributions Q over $S_{T_{\mathrm{suf}}} \times S_{T_{\mathrm{suf}}}$ that satisfy

$$\epsilon(Q) \triangleq \sum_{z\in S_{W_F}} \sum_{s\in S_{T_{\mathrm{suf}}}} Q(z,s) \log \frac{Q(s|z)}{Q(s|w)} \le \epsilon(w) \,.$$

By (H.22) and the definition of $U_{w,\epsilon}^n$ it follows that $x^n \in U_{w,\epsilon}^n$ if and only if $\hat{\mathrm{P}}_{F,x^n} \in \Gamma$. Let $p_F(\cdot|\cdot)$ denote the next-state probability mass functions conditioned on the states of T_{suf}, induced by the symbol conditional probability mass functions on states of T, $p_T(\cdot|\cdot)$. This is,

$$p_F(z|s) \triangleq \begin{cases} p_T(a|s'), & \text{if } s' \in S_T, s' \preceq s, \text{ and } f(s,a) = z, \\ 0, & \text{otherwise.} \end{cases}$$

Let also $\mathrm{P}_F(\cdot)$ denote the probability assignment induced by $p_F(\cdot|\cdot)$. Denote by Γ_0 the set of distributions in the closure of Γ (relative to the set of all distributions over $S_{T_{\text{suf}}} \times S_{T_{\text{suf}}}$) with identical left and right marginal distributions. By [14, Lemma 2(a)],

$$\limsup_{n\to\infty} n^{-1} \log \mathrm{P}_F \left\{ \hat{\mathrm{P}}_{F,x^n} \in \Gamma \right\} \leq -D \,, \tag{H.25}$$

where

$$D = \min\{D(Q||\mathrm{P}_F) : Q \in \Gamma_0\} \,,$$

and

$$D(Q||P_F) = \sum_{s,z \in S_{T_{\text{suf}}}} Q(s,z) \log \frac{Q(s,z)}{\bar{Q}(s) p_F(z|s)} \,.$$

The unique distribution Q^0 over $S_{T_{\text{suf}}} \times S_{T_{\text{suf}}}$ with two identical marginal distributions for which $D(Q^0||\mathrm{P}_F) = 0$ is

$$Q^0(s,z) = P_F^0(s) \times p_F(z|s) \,,$$

where P_F^0 is the unique stationary distribution defined by $p_F(\cdot|\cdot)$. Clearly $\epsilon(Q^0) > 0$, for otherwise $Q^0(.|z) = p_F(.|z)$ for all $z \in S_{W_F}$ and the states in S_W of T would have identical conditional distributions, and thus T would not be minimal. Hence, taking $0 < \epsilon(w) < \epsilon(Q^0)$, $Q^0 \notin \Gamma_0$ and $D > 0$. Thus, by (H.25), $\mathrm{P}_{\langle T,p_T\rangle} \left\{ x^n \in U^n_{w,\epsilon} \right\} \leq 2^{-nD}$ with $D > 0$, and by (H.24)

$$\mathrm{P}_{\langle T,p_T\rangle} \{X^n \in U^n\} \leq |S_T| 2^{-nD}$$

□

Bibliography

[1] Paul H. Algoet and Thomas M. Cover. A sandwich proof of the Shannon-McMillan-Breiman theorem. *The Annals of Probability*, 16(2):899–909, 1988. 126

[2] Robert B. Ash. *Information Theory*. John Wiley & Sons, Inc., 1967. 1, 2, 18, 19, 26

[3] Dror Baron and Yoram Bresler. An $O(n)$ semi-predictive universal encoder via the BWT. *IEEE Trans. Inform. Theory*, 50(5):928–937, May 2004. 10, 21, 40, 42, 43, 114

[4] Andrew R. Barron. *Logically Smooth Density Estimation*. PhD thesis, Stanford University, Stanford, CA, September 1985. 125

[5] Claude Berge. *Graphs*. North-Holland, Amsterdam, 1985. 56, 86, 87

[6] Patrick Billingsley. Statistical methods in Markov chains. *Annals Math. Stat.*, 32:12–40, 1961. 11, 52, 55

[7] Lawrence D. Brown. *Fundamentals of statistical exponential families with applications in statistical decision theory*. Institute of Mathematical Statistics, Hayward, CA, 1986. 13

[8] Peter L. Buhlmann and Abraham J. Wyner. Variable length Markov chains. *Annals of Statistics*, 27:480–513, 1998. 7

[9] Michael Burrows and David J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report SRC Research Report 124, Digital Systems Research Center, Palo Alto, CA, May 1994. 10

[10] John G. Cleary and Ian H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Trans. Commun.*, 32 (4):396–402, April 1984. 10

[11] Thomas M. Cover. Enumerative source encoding. *IEEE Trans. Inform. Theory*, IT-19:73–77, January 1973. 7, 12, 97

[12] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, second edition, 1991. 2, 4, 115, 124

[13] Imre Csiszár. The method of types. *IEEE Trans. Inform. Theory*, 44(6):2505–2523, October 1998. 11

[14] Imre Csiszár, Thomas M. Cover, and Byoung-Seon Choi. Conditional limit theorems under Markov conditioning. *IEEE Trans. Inform. Theory*, IT-33(6):788–801, November 1987. 11, 203

[15] Imre Csiszár and János Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic, New York, 1981. 6, 11, 51

[16] Imre Csiszár and Paul C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4), 2004. 112

[17] Imre Csiszár and Zsolt Talata. Context tree estimation for not necessarily finite memory processes, via BIC and MDL. *IEEE Trans. Inform. Theory*, 52(3):1007–1016, March 2006. 112, 126

[18] Lee D. Davisson. Universal noiseless coding. *IEEE Trans. Inform. Theory*, IT-19(6):783–795, November 1973. 3, 4

[19] Lee D. Davisson. Minimax noiseless universal coding for Markov sources. *IEEE Trans. Inform. Theory*, IT-29(2):211–215, March 1983. 4

[20] Lee D. Davisson and Alberto Leon-Garcia. A source matching approach to finding minimax codes. *IEEE Trans. Inform. Theory*, IT-26:166–174, March 1980. 3

[21] Lee D. Davisson, Giuseppe Longo, and Andrea Sgarro. The error exponent for the noiseless encoding of finite ergodic Markov sources. *IEEE Trans. Inform. Theory*, IT-27(4):431–438, July 1981. 11

[22] Nicolaas Govert de Bruijn and Tanja van Aardenne-Ehrenfest. Circuits and trees in oriented linear graphs. *Simon Stevin*, 4:203–217, 1951. 11, 67, 69, 70, 189

[23] Michelle Effros. PPM performance with BWT complexity: A fast and effective data compression algorithm. *Proceedings of the IEEE*, 88(11):1703–1712, November 2000. 10, 21, 47, 145

[24] Michelle Effros, Karthik Visweswariah, Sanjeev Kulkarni, and Sergio Verdú. Universal lossless source coding with the Burrows-Wheeler transform. *IEEE Trans. Inform. Theory*, 48:1061–1081, May 2002. 10

[25] Shimon Even. *Algorithmic Combinatorics*. Macmillan, 1973. 190

[26] Robert M. Fano. The transmission of information. Technical Report Technical Report No. 65, Research Laboratory of Electronics, M.I.T., Cambridge, MA, USA, 1949. 2

[27] Meir Feder, Neri Merhav, and Michael Gutman. Universal prediction of individual sequences. *IEEE Trans. Inform. Theory*, 38:1258–1270, July 1992. 26

[28] William Feller. *Probability theory and its applications*, volume 1. John Wiley & Sons, Inc., New York, third edition, 1968. 18

[29] Boris M. Fitingof. Optimal coding in the case of unknown and changing message statistics. *Problems of Information Transmission*, 2(2):1–7, 1966. 4

[30] Boris M. Fitingof. The compression of discrete information. *Problems of Information Transmission*, 3(3):22–29, 1967. 4

[31] Robert G. Gallager. Source coding with side information and universal coding. Unpublished manuscript, October 1974. 3

[32] Robert Giegerich and Stefan Kurtz. From Ukkonen to McCreight and Weiner: A unifying view to linear-time suffix tree construction. *Algorithmica*, 19:331–353, November 1997. 10, 21, 22, 30, 33, 40, 42, 43, 130

[33] Solomon W. Golomb. Run-length encodings. *IEEE Trans. Inform. Theory*, IT-12:399–401, July 1966. 98

[34] Leo A. Goodman. Exact probabilities and asymptotic relationships for some statistics from m-th order Markov chains. *Annals of Mathematical Statistics*, 29:476–490, 1958. 11, 49, 52, 55, 60, 130

[35] Philippe Jacquet and Wojciech Szpankowski. Markov types and minimax redundancy for Markov sources. *IEEE Trans. Inform. Theory*, 50(7):1393–1402, July 2004. 11, 52, 55

[36] John C. Kieffer. Sample converses in source coding theory. *IEEE Trans. Inform. Theory*, 37(2):263–268, 1991. 125

[37] Donald E. Knuth. *The Art of Computer Programming. Fundamental Algorithms*, volume 1. Addison-Wesley, Reading, MA, third edition, 1997. 21, 37, 42, 43

[38] Donald E. Knuth. *The Art of Computer Programming. Sorting and Searching*, volume 3. Addison-Wesley, Reading, MA, second edition, 1997. 21

[39] Andrey N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems in Information Transmission*, 1:1–7, 1965. 4

[40] Ioannis Kontoyiannis, Luis A. Lastras-Montaño, and Sean P. Meyn. Relative entropy and exponential deviation bounds for general Markov chains. In *Proc. 2005 International Symposium on Information Theory*, pages 1563– 1567, Adelaide, Australia, September 2005. 101, 191, 194

[41] Rafail E. Krichevskii. The relation between redundancy coding and the reliability of information from a source. *Problems of Information Transmission*, 4(3):37–45, 1968. 4

[42] Rafail E. Krichevskii and Victor K. Trofimov. The performance of universal encoding. *IEEE Trans. Inform. Theory*, IT-27:199–207, Mar 1981. 4, 5, 12, 37, 38

[43] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951. 4

[44] N. Jesper Larsson. Extended application of suffix trees to data compression. In *Proc. 1996 Data Compression Conference*, pages 190–199, Snowbird, Utah, USA, April 1996. 10, 21, 40

[45] N. Jesper Larsson. The context trees of block sorting compression. In *Proc. 1998 Data Compression Conference*, pages 189–198, Snowbird, Utah, USA, March 1998. 10, 20

[46] Thomas J. Lynch. Sequence time coding for data compression. *Proceedings of the IEEE*, 54(10):1490–1491, Oct. 1966. 12

[47] Álvaro Martín, Neri Merhav, Gadiel Seroussi, and Marcelo J. Weinberger. Twice-universal simulation of Markov sources and individual sequences. In *Proc. 2007 International Symposium on Information Theory*, Nice, France, June 2007. 14, 15, 119, 122, 126, 131

[48] Álvaro Martín, Gadiel Seroussi, and Marcelo J. Weinberger. Linear time universal coding and time reversal of tree sources via FSM closure. *IEEE Trans. Inform. Theory*, 50(7):1442–1468, July 2004. 8, 15, 17, 27, 28, 29, 37, 44

[49] Álvaro Martín, Gadiel Seroussi, and Marcelo J. Weinberger. Type classes of tree models. In *Proc. 2007 International Symposium on Information Theory*, Nice, France, June 2007. 15, 49

[50] Álvaro Martín, Gadiel Seroussi, and Marcelo J. Weinberger. Enumerative coding for tree sources. In Peter Grünwald, Petri Myllymäki, Ioan Tabus, Marcelo J. Weinberger, and Bin Yu, editors, *Festschrift in Honor of Jorma Rissanen on the Occasion of his 75th Birthday*, 38, pages 93–116. Tampere University of Technology, Tampere International Center for Signal Processing, Tampere, 2008. 15, 97

[51] Edward M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272, 1976. 21

[52] Brockway McMillan. Two inequalities implied by unique decipherability. *IEEE Trans. Inform. Theory*, IT-2(4):115–116, December 1956. 2

[53] Neri Merhav and Meir Feder. Universal prediction. *IEEE Trans. Inform. Theory*, 44:2124–2147, October 1998. 5

[54] Neri Merhav and Marcelo J. Weinberger. On universal simulation of information sources using training data. *IEEE Trans. Inform. Theory*, 50(1):5–20, January 2004. 13, 124, 131

[55] Donald R. Morrison. Patricia - practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 15(4):514–534, 1968. 21

[56] Ragnar Nohre. *Some Topics in Descriptive Complexity*. PhD thesis, Department of Computer Science, The Technical University of Linkoping, Sweden, 1994. 10, 37, 40, 113, 126, 201

[57] Jorma Rissanen. Generalized Kraft inequality and arithmetic coding. 20(3):198–203, May 1976. 2

[58] Jorma Rissanen. A universal data compression system. *IEEE Trans. Inform. Theory*, IT-29:656–664, September 1983. 6, 7, 9, 19, 20, 23

[59] Jorma Rissanen. Universal coding, information, prediction, and estimation. *IEEE Trans. Inform. Theory*, IT-30:629–636, July 1984. 4

[60] Jorma Rissanen. Complexity of strings in the class of Markov sources. *IEEE Trans. Inform. Theory*, IT-32(4):526–532, July 1986. 8, 78

[61] Jorma Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, September 1986. 9, 37

[62] Jorma Rissanen. Fisher information and stochastic complexity. *IEEE Trans. Inform. Theory*, 42(1):40–47, 1996. 12

[63] Jorma Rissanen and Glen G. Langdon. Universal modeling and coding. *IEEE Trans. Inform. Theory*, IT-27:12–23, January 1981. 17, 18

[64] Boris Y. Ryabko. Encoding a source with unknown but ordered probabilities. *Problems of Information Transmission*, pages 134–138, oct 1979. 3

[65] Boris Y. Ryabko. Twice-universal coding. *Problems of Information Transmission*, 20:173–177, July/September 1984. 4, 9, 37

[66] Johan P.M. Schalkwijk. An algorithm for source coding. *IEEE Trans. Inform. Theory*, 18(3):395–399, May 1972. 12

[67] Gadiel Seroussi. On universal types. *IEEE Trans. Inform. Theory*, 52(1):171–189, January 2006. 7, 13, 14, 15, 119, 120, 122, 124, 125, 126, 131

[68] Gadiel Seroussi, Nicolás Fraiman, Alix Lhéritier, and Alfredo Viola. Lossless compression for sparse finite memory sources. In *Information Theory and applications (ITA'08)*, San Diego, CA, USA, January 2008. 8

[69] Gadiel Seroussi and Marcelo J. Weinberger. On tree sources, finite state machines, and time reversal. In *Proc. International Symposium on Information Theory*, Whistler, BC, Canada, September 1995. 8, 26

[70] Claude E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. J.*, 27:379–423, 623–656, 1948. 1, 2

[71] Yuri M. Shtarkov. Universal sequential coding of single messages. *Problems of Inform. Trans.*, 23:175–186, July 1987. 12

[72] Valeri T. Stefanov. Noncurved exponential families associated with observations over finite-state Markov chains. *Scand. J. Statist*, 18:353–356, 1991. 13

[73] Joe Suzuki. A CTW scheme for some FSM models. In *Proc., 1995 IEEE International Symposium on Information Theory*, page 389, Sep 1995. 8

[74] Wojciech Szpankowski. *Average Case Analysis of Algorithms on Sequences*. John Wiley & Sons, Inc., New York, 2001. 21

[75] Jun'ichi Takeuchi and Andrew R. Barron. Asymptotically minimax regret by Bayes mixtures. In *Proc. 1998 International Symposium on Information Theory*, page 318, Cambridge, MA, U.S.A., August 1998. 13

[76] Jun'ichi Takeuchi and Tsutomu Kawabata. Exponential curvature of Markov models. In *Proc. 2007 International Symposium on Information Theory*, Nice, France, June 2007. 13

[77] Sandeep Tata, Richard A. Hankins, and Jignesh M. Patel. Practical suffix tree construction. In *Proc. 13th International Conference on Very Large Data Bases*, pages 36–47, 2004. 130

[78] Tjalling J. Tjalkens, Paul A.J. Volf, and Frans M.J. Willems. A context-tree weighting method for text generating sources. *Data Compression Conference*, 0:472, 1997. 129

[79] Victor K. Trofimov. Redundancy of universal coding of arbitrary Markov sources. *Problems of Information Transmission*, 10(4):16–24, 1974. 4

[80] Paul A.J. Volf and Frans M. J. Willems. Context-tree weighting for extended tree sources. In *Proc. of the 17th Symposium on Information Theory in the Benelux*, pages 95–101, Enschede, The Netherlands, May 1996. 8

[81] Marcelo J. Weinberger and Meir Feder. Predictive stochastic complexity and model estimation for finite-state processes. *Journal of Statistical Planning and Inference*, 39:353–372, 1994. 26, 27

[82] Marcelo J. Weinberger, Abraham Lempel, and Jacob Ziv. A sequential algorithm for the universal coding of finite-memory sources. *IEEE Trans. Inform. Theory*, 38:1002–1014, May 1992. 8, 9, 19, 20, 43

[83] Marcelo J. Weinberger, Neri Merhav, and Meir Feder. Optimal sequential probability assignment for individual sequences. *IEEE Trans. Inform. Theory*, 40(2):384–396, March 1994. 6, 11, 12, 50, 86

[84] Marcelo J. Weinberger, Jorma Rissanen, and Meir Feder. A universal finite memory source. *IEEE Trans. Inform. Theory*, 41:643–652, May 1995. 1, 7, 9, 19, 20, 25, 113

[85] Peter Weiner. Linear pattern matching algorithms. In *Proc. 14th IEEE Annual Symposium on Switching and Automata Theory*, pages 1–11, 1973. 21

[86] Peter Whittle. Some distribution and moment formulae for the Markov chain. *J. Roy. Statist. Soc. Ser. B*, 17(3):235–242, 1955. 11, 49, 52, 55

[87] Frans M. J. Willems. The context-tree weighting method: Extensions. *IEEE Trans. Inform. Theory*, 44:792–798, March 1998. 9, 10

[88] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Trans. Inform. Theory*, IT-41:653–664, May 1995. 5, 8, 9, 10, 26, 37, 126, 201

[89] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. Context weighting for general finite-context sources. *IEEE Trans. Inform. Theory*, 42(5):1514–1520, Sep 1996. 8

[90] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. Context-tree maximizing. In *Proc. 2000 Conference on Information Sciences and Systems*, pages TP6–7–TP6–12, Princeton, New Jersey, USA, March 2000. 10, 20, 37

[91] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Inform. Theory*, IT-24:530–536, September 1978. 13, 119