Technische Universität Dresden

New Results on Context-Free Tree Languages

Dissertation

zur Erlangung des akademischen Grades

Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt an der

Technischen Universität Dresden

Fakultät Informatik

eingereicht am

20. Dezember 2017

von

Dipl.-Inf. Johannes Osterholzer

geboren am 07. Dezember 1984
in Simbach am Inn

Gutachter:

Prof. Dr.-Ing. habil. Dr. h.c./Univ. Szeged Heiko Vogler, Technische Universität Dresden *(Betreuer)*

Prof. Dr. rer. nat. Andreas Maletti, Universität Leipzig

Verteidigt am: 04. Mai 2018, Dresden

Acknowledgements

It may be a cliché to begin a thesis with the words "This thesis would not have been possible without...", followed by an exhaustive list of relatives, colleagues, friends, acquaintances, spiritual and worldly authorities, deities, favorite pet animals, and so forth. I cannot completely spare the reader such a concatenation, but I will try to keep it brief, and assure it comes from the heart.

First of all, I am indebted to my parents for their continuous support and patience. As compensation, I wish you many pleasant hours with reading this thesis!

Moreover, I want to thank my doctorate supervisor, Heiko Vogler, for his guidance, and for leaving me the freedom to decide on my own where to go next in my research. Thank you very much for this opportunity, Heiko!

I also want to thank Andreas Maletti for agreeing to be second assessor for this work. Thank you for taking the time, Andreas!

My colleagues at the Chair of Foundations of Programming also deserve my gratitude. In particular, I thank Toni Dietze for being a great office-mate and for the many counterexamples he provided, and Tobias Denkinger for his frequent flashes of inspiration. Of course, I must also thank Kerstin Achtruth, for her helpfulness and for her being the heart and soul of the institute.

In the course of Toni's, Luisa's, and my research on inverse homomorphic closure of context-free tree languages, we came into contact with André Arnold. In our email correspondence, which we enjoyed very much, he showed us the flaws in our first proof attempts, and encouraged us to keep trying. Merci beaucoup!

Finally, I want to thank Luisa for being there for me (which may not always have been easy), for mustering the strength to read all of the following pages (and spotting many mistakes), and of course for 3D printing the (objectively) best son (yet?) in the world!

Contents

This book is divided into chapters,
which are divided into sections,
which are divided into paragraphs,
which are divided into sentences,
which are divided into words,
which are divided into letters.

(Carl Linderholm, *Mathematics Made Difficult*)

Introduction

Character is like a tree and
reputation like a shadow.
The shadow is what we think of it;
the tree is the real thing.

(Abraham Lincoln)

Context-Free Grammars

Context-free grammars (cfg) rank among the fundamental models applied in computer science. Given by a finite number of context-free productions such as

$$A \to aAb \qquad \text{and} \qquad A \to \varepsilon\,,$$

they permit describing a possibly infinite set of words, such as the formal language

$$\left\{a^n b^n \,\middle|\, n \in \mathbb{N}\right\},$$

in finite space. They are an *effective* representation: there are algorithms to decide many interesting properties of the languages representable by a cfg – and one can even find efficient algorithms for quite some of those problems. Moreover, the context-free languages thus represented are mathematically well-behaved: given an arbitrary context-free language L and a reasonable operation φ on formal languages, in many cases the image $\varphi(L)$ is also context-free.

The naturalness of the context-free languages is further underscored by the fact that they appear in many different guises – such as ALGOL-like languages, languages defined by (E)BNF definitions, by (simple) phrase-structure grammars, or by pushdown automata, solutions of particular equation systems, and many more.

In summary, since the 1950s there has been much scientific progress on *(i)* complexity of decision problems of cfg, *(ii)* closure properties of cfg, and *(iii)* characterization results for cfg.[1]

Tree Languages

Due to the associativity of monoids, there is not much structure to a word. The word aab is represented likewise by $a \cdot ab$ and $aa \cdot b$. However, many topics in computer science necessitate structured data – be it to represent the syntax of a program, a structured document such as an XML file, or to symbolize the grammatical structure of a natural language sentence. The

[1] And yet, there are still open problems on cfg; compare e.g. [150].

epitomical means to represent such structure is by *trees*. For our example, we obtain two distinct trees

$$
\begin{array}{ccc}
 & \cdot & \\
 / & & \backslash \\
a & & \cdot \\
 & / & & \backslash \\
 & a & & b
\end{array}
\qquad \text{and} \qquad
\begin{array}{ccc}
 & & \cdot \\
 & / & & \backslash \\
 & \cdot & & b \\
 / & & \backslash \\
a & & a
\end{array}\ ,
$$

corresponding to our two conceptions of the structure of aab from above.

From the 1970s onward, tree language theory has evolved as a full-fledged subfield of formal language theory. Many well-known results, e.g. on recognizable sets, have been generalized from words to trees. While some generalizations are straightforward, there are also instances where the increment in structure that comes with trees complicates matters, or where properties that hold in the case of word languages are outright false when generalized to the tree case.[2]

Context-Free Tree Grammars

So how to generalize cfg to the realm of trees? This question has been answered by Rounds, who defined a context-free tree grammar (cftg) to be given by a finite set of productions such as

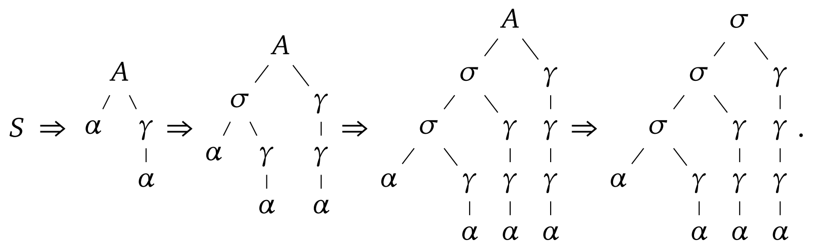
$$
S \to \begin{array}{ccc} & A & \\ / & & \backslash \\ \alpha & & \gamma \\ & & | \\ & & \alpha \end{array}\ ,
\qquad
\begin{array}{ccc} & A & \\ / & & \backslash \\ x_1 & & x_2 \end{array}
\to
\begin{array}{ccccc} & & A & & \\ & / & & \backslash & \\ & \sigma & & & \gamma \\ / & & \backslash & & | \\ x_1 & & x_2 & & x_2 \end{array}\ ,
\qquad \text{and} \qquad
\begin{array}{ccc} & A & \\ / & & \backslash \\ x_1 & & x_2 \end{array}
\to
\begin{array}{ccc} & \sigma & \\ / & & \backslash \\ x_1 & & x_2 \end{array}\ .
$$

As we see, these productions are still context-free in the sense that each left-hand side contains precisely one nonterminal symbol – in this case, S or A. Since in a cftg, nonterminals may also occur as inner nodes of a tree, we must somehow represent a nonterminal's subtrees. This role is fulfilled by the symbols x_1, x_2, …, called *variables*. In our example, S has no variables, it will therefore occur as a leaf. The nonterminal A has two variables x_1 and x_2; thus it will occur as a binary inner node. The right-hand side of a cftg production is a tree made up of nonterminal symbols, terminal symbols – in this case, σ, γ and α –, and the variables from the left-hand side, which may occur as leaves only.

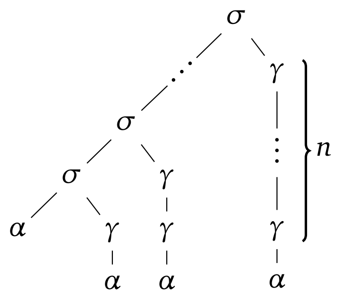
Given this description, the application of a production of form $A(x_1, \ldots, x_n) \to \varrho$ to an occurrence of the nonterminal A in a tree is defined quite naturally: we replace the occurrence of A by the right-hand side ϱ, and substitute the subtrees of A for the respective occurrences of the variables $x_1, \ldots, x_n$.

[2] We will encounter such a property in Chapter 4.

In our example, we obtain thus the derivation



By considering all such derivations, we see that every tree derivable from S that contains only terminal symbols is of the form



for some $n \in \mathbb{N} \setminus \{0\}$. The tree language that consists of all these trees is called context-free, as it is generated by a cftg.

Observe that our grammar is *copying,* or *nonlinear,* in the sense that in the second depicted production, the variable x_2 occurs more than once on its right-hand side. Therefore, the grammar can enumerate the subtrees $\gamma(\alpha)$, $\gamma(\gamma(\alpha))$, ..., in a derivation, and output a copy of each of them in the generated tree. Much of the additional complexity of context-free tree grammars is due to the interplay of copying and nondeterminism.

Rounds proposed context-free tree grammars as a promising model for mathematical linguistics. However, the bulk of research on cftg from the 1970s and 1980s is motivated by the application of cftg to model semantics of computer programs, using *recursive program schemes.* Be that as it may, the possible uses of cftg motivated a similar research program as in the word case: researchers investigated complexity-theoretic traits, closure properties, and characterization results. We will not list all these results here – the reader should consult Chapter 2 for a list of properties important in this work, and for references to literature.

Linear Context-Free Tree Grammars

In recent years, there has been a revival of interest in context-free tree grammars, mainly motivated by tasks from natural language processing (NLP). There, the syntax of a natural language input sentence is modelled by a tree. While former research on cftg was focused on the unrestricted model, which allows copying, current research mostly considers non-copying, or *linear,* cftg.

Linear context-free tree languages appear to be a promising model for NLP, since their yield languages[3] are *mildly context-sensitive.* A class of formal languages is said to be mildly context-sensitive if it allows modelling some non-context-free phenomena that occur in human language, but lacks the full power and complexity of the context-sensitive languages. To demonstrate the applications of linear cftg for natural language processing, let us consider the following research spotlights.

- In [100], Kepser and Rogers compare the power of cftg with tree-adjoining grammars, a model used in natural language processing. They prove that the tree languages of tree-adjoining grammars are precisely those of cftg that are linear and monadic (i.e., where the only variable in a production is x_1). Therefore, established properties and algorithms can be carried over.
- A tree grammar is said to be *lexicalized* if each production contains at least one symbol from a specified set of terminal symbols as a leaf. Lexicalized grammars are desirable since they admit efficient parsing algorithms. Moreover, they describe the (linguistic) context which a terminal symbol may appear in. Engelfriet and Maletti prove in [117] that for every tree-adjoining grammar, there is an equivalent linear cftg that is lexicalized. In fact, they show the stronger property that for every k-adic linear cftg,[4] where $k \in \mathbb{N}$, there is an equivalent lexicalized $(k+1)$-adic linear cftg.
- Kallmeyer shows in [94] that a novel formalism, called k-tree wrapping grammar, is mildly context-sensitive. The proof is by an elaborate transformation into an equivalent linear context-free tree grammar.

The revival of cftg motivates further research of linear and nonlinear cftg. Thus, in this thesis, we present some new results on context-free tree languages, covering each of the areas mentioned above: complexity, characterization, and closure theorems. While some of the theory is somewhat inspired by natural language applications, our focus will be mainly on the underlying mathematics.

Overview

This thesis is structured as follows. Chapter 1 recalls some fundamental notions from mathematics and theoretical computer science. It is probably safe to skip large parts of this chapter and refer to it only when necessary. In Chapter 2, we recall the definitions of context-free tree grammars and pushdown tree automata, as well as some basic properties and known results. Chapter 3 discusses some decision problems of cftg, most importantly their uniform membership problem. Chapter 4 is concerned with closure properties of linear context-free tree languages. It contains a proof of the fact that the linear context-free tree languages are not closed under inverse linear tree homomorphisms. Moreover, we show that the linear monadic context-free tree languages *are* indeed closed under this operation. Chapter 5 is devoted to *synchronous* context-free tree grammars, which induce tree transformations

[3] The word languages obtained by reading off the leaves of each tree from left to right.

[4] A cftg is k-adic if its productions contain only the variables $x_1, \ldots, x_k$.

instead of languages. There, we will characterize a particular restriction of these synchronous grammars by a novel type of pushdown machine. Finally, in Chapter 6, we compare linear monadic and footed cftg, the latter of which are the counterpart of tree-adjoining grammars in the realm of context-free tree grammars.

While Chapters 3 to 6 all depend on the definitions and theorems given in Chapter 2, they are pairwise independent, and it should be possible to read them in any desired order and selection.

Chapter 1

Fundamental Notions and Properties

Es ist schon alles gesagt,
nur noch nicht von allen.

(Karl Valentin)

In this chapter, we will recall some basic mathematical definitions and properties which shall be used in the following. As mentioned in the introduction, most of the following should be known to the reader – we have tried to err on the side of caution and keep this thesis as self-contained as feasible. It should be safe to skip most parts of this chapter, and refer back to it using the index at the end of this work.

We begin in Section 1.1.1, where we recall sets, relations, and functions – mainly to fix notation. Section 1.1.2 is about some notions from (universal) algebra which will play a role in this work. Moreover, we treat the method of proof by induction in Section 1.1.3.

Section 1.2 calls to mind elementary formal language theory. In particular, we recall the classes of recognizable, context-free, indexed, and recursively enumerable languages, as well as some rudimentary complexity theory.

Section 1.3 recollects trees and tree languages – specifically, we recall the class of recognizable tree languages in Section 1.3.2, and some essential definitions on tree homomorphisms and tree transformations in Section 1.3.4. We will often use a slightly non-standard notation for operations on trees, introduced in the context of an algebraic structure called magmoid. Therefore, we recommend that readers unaccustomed to this notation should peruse Section 1.3.1.

Finally, Section 1.4 contains some bare-bones notions from the theory of weighted (tree) languages, which we require later on.

1.1 Mathematical Preliminaries

1.1.1 Sets, Relations, and Functions

Sets

We begin with one of the basic building blocks of modern mathematics. A *set,* as defined by Cantor [30], is a *"collection into a whole [. . .] M of definite and separate objects m of our intuition or our thought. These objects are called the* elements *of M"*, denoted by $m \in M$.

We will gloss over all the well-known problems which arise from this seemingly straightforward definition.[1] Moreover, we take for granted all basic notions of set theory, as, i.a., set union $\cup$, set intersection $\cap$, set difference $\setminus$, the Cartesian product $\times$, the Cartesian power $M^n = M \times \cdots \times M$, the power set $\mathcal{P}(M)$ of a set M, set builder notation $\{x \in M \mid P(x)\}$ or $\{x \mid P(x)\}$ for some property P, set inclusion $\subseteq$, set equality $=$, the empty set $\emptyset$, and so on.

In an expression involving sets, we will assume that $\times$ binds stronger than $\setminus$, and $\setminus$ binds stronger than $\cup$ and $\cap$. So the meaning of the expression

$$A \times B \setminus C \cup D \qquad \text{is} \qquad \big((A \times B) \setminus C\big) \cup D\,.$$

The *cardinality* of a finite set M is the number n of its elements, denoted by $|M| = n$. When M has an infinite amount of elements, we write $|M| = \infty$. A *partitioning* of a set M is a set $\mathcal{M} \subseteq \mathcal{P}(M) \setminus \{\emptyset\}$ such that $M = \bigcup \mathcal{M}$ and for each $P, Q \in \mathcal{M}$, either $P = Q$ or $P \cap Q = \emptyset$. In this situation, the elements of $\mathcal{M}$ are called *partitions.*

Numbers and Booleans

The set of natural numbers $\{0, 1, 2, \dots\}$ is denoted by $\mathbb{N}$, and the set of positive natural numbers $\mathbb{N} \setminus \{0\}$ by $\mathbb{N}_1$. The set of real numbers will be denoted by $\mathbb{R}$. We assume familiarity with the basic arithmetic operations and relations on $\mathbb{N}$ and $\mathbb{R}$. For every $m, n \in \mathbb{N}$, let

$$[m, n] = \big\{ i \in \mathbb{N} \;\big|\; m \leq i \leq n \big\}$$

and let $[n] = [1, n]$. Observe that $[m, n] = \emptyset$ whenever $m > n$, and therefore $[0] = \emptyset$.

The set of *Booleans* is denoted by $\mathbb{B}$ and contains precisely the truth values 0 (false) and 1 (true). Again, we will abstain from recapitulating the well-known logical operations on $\mathbb{B}$. Just to fix notation, logical conjunction is denoted by $\wedge$, disjunction by $\vee$, and negation by $\neg$.

The *factorial* $n!$ of a number $n \in \mathbb{N}$ is defined as

$$n! = \prod_{j=1}^{n} j\,.$$

In particular, $0! = 1! = 1$. For all numbers $n, k \in \mathbb{N}$ with $k \leq n$, their *binomial coefficient* is

$$\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!}\,.$$

[1] However, we recommend [40] as a well-readable and fascinating introduction to the topic.

$n\diagdown k$	0	1	2	3	4	5	6	$\cdots$
0	1							
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Figure 1.1: Pascal's triangle

The binomial coefficient describes (among many others) the number of subsets with cardinality k of a set of n elements. Since it is technically convenient, we let $\binom{n}{k} = 0$ if $k > n$.

It is well-known that the binomial coefficients can be arranged into *Pascal's triangle,* displayed in Figure 1.1. As a consequence, for every $n, k \in \mathbb{N}_1$ with $k \leq n$, we have

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}. \tag{1.1}$$

Relations

A *relation* R between two sets A and B is a tuple (A, B, G) such that $G \subseteq A \times B$. When R is a relation between A and A, we call R a *relation on A.* If $(a, b) \in G$, we will also write aRb and say that a and b are *related* (by R). For R a relation as above, A is called its *domain,*[2] B is its *codomain,* and G is the *graph* of R. It is customary to omit specifying the whole tuple and to write $R \subseteq A \times B$ instead, seemingly identifying R with its graph. As there is hardly danger of confusion, we will follow this custom. However, note that the domain and codomain of a relation are nevertheless important: strictly, two relations R and S are equal only if their respective domains and codomains coincide. In general, we will follow this strict definition of equality of relations. However, there will be some definitions where it is more convenient to construe two relations as equal already when their graphs are so; we will mention these cases explicitly.

In the following, assume sets A, B, and C, and relations $R \subseteq A \times B$ and $S \subseteq B \times C$. The *composition* $R;S \subseteq A \times C$ of R and S is defined by

$$R;S = \left\{(a,c) \in A \times C \;\middle|\; \exists b\colon aRb \wedge bSc\right\}.$$

Sometimes, especially when R and S are functions (see below), we will also write $S \circ R$ instead of $R;S$. Furthermore, the *inverse* of R, denoted $R^{-1} \subseteq B \times A$, is defined by

$$R^{-1} = \left\{(b,a) \in B \times A \;\middle|\; aRb\right\}.$$

[2]There are other definitions which take the domain of R to be the set $\{a \in A \mid \exists b\colon (a,b) \in G\}$.

Finally, the *image* of a set $X \subseteq A$ under R is

$$R(X) = \{ b \in B \mid \exists a \in X : aRb \}$$

and the *preimage* of $Y \subseteq B$ under R is $R^{-1}(Y)$. The *diagonal relation* id_A on A is defined to be

$$\mathrm{id}_A = \{ (a,a) \mid a \in A \} .$$

Let R be a relation on a set A. We say that R is

- *reflexive* if $\mathrm{id}_A \subseteq R$,
- *symmetric* if $R = R^{-1}$,
- *antisymmetric* if $R \cap R^{-1} \subseteq \mathrm{id}_A$,
- *transitive* if $R ; R \subseteq R$, and
- *total* if $R \cup R^{-1} = A \times A$.

Moreover, R is an *equivalence* (relation) if it is reflexive, symmetric and transitive, R is a *partial order* (relation) if it is reflexive, antisymmetric and transitive, and R is a *linear order* (relation) if it is a partial order that is total.

Let $\leq$ be a partial order relation on a set A. We denote by $<$ the relation $\leq \setminus \mathrm{id}_A$, and by $\geq$ and $>$ the respective relations $\leq^{-1}$ and $<^{-1}$. Moreover, we will tacitly use similar notations for partial orders denoted by $\subseteq$, $\sqsubseteq$, $\preceq$, etc. We say that an element $a \in A$ is a *minimal (resp. maximal) element* of A (with respect to $\leq$) if there is no $b \in A$ such that $b < a$ (resp. $b > a$).

Given a relation R on a set A, we let R^+ (resp. R^*) denote the smallest relation $S \supseteq R$ on A that is transitive (resp. reflexive and transitive). We call R^+ the *transitive closure,* and R^* the *reflexive-transitive closure* of the relation R.

Consider sets A and $B \subseteq A$, and a relation R between A^k and A, for some arbitrary number $k \in \mathbb{N}$. We say that B is *closed under R* if $R(B^k) \subseteq B$.

Functions

Let A and B be sets. A *function* (resp. a *partial function*) is a relation $f \subseteq A \times B$ such that for every $a \in A$, there is a unique $b \in B$ (resp. at most one $b \in B$) such that afb. In this case, we write $f(a) = b$. In particular, if f is partial, then writing $f(a)$ comes with the implicit assumption that $f(a)$ is defined. Moreover, instead of $f \subseteq A \times B$, we use the conventional notation $f : A \to B$ (resp. $f : A \twoheadrightarrow B$). The set of all functions of type $A \to B$ is denoted by B^A. Given sets A, B, and $A' \subseteq A$, as well as a function $f : A \to B$, the *restriction* of f to A' is the function

$$f|_{A'} : A' \to B , \qquad a \mapsto f(a) .$$

In this situation, we say that f *extends* $f|_{A'}$, or is an *extension* thereof. Sometimes, functions will also be called *mappings.*

The notions of image and preimage under a (partial) function carry over from relations. A function $f : A \to B$ is called

- *injective*, or an *injection*, if for every a_1, $a_2 \in A$, whenever $f(a_1) = f(a_2)$, then $a_1 = a_2$,
- *surjective*, or a *surjection*, if $f(A) = B$, and
- *bijective*, or a *bijection*, if it is both injective and surjective.

Injectivity and surjectivity apply also to partial functions.

Let $n \in \mathbb{N}$. A function $f : A^n \to A$ is also called an *n-ary operation* on A. If $n = 0$, then f is a *constant,* and we will identify $f : A^0 \to A$ and $f() \in A$. Instead of 1-ary, we will write *unary*, and *binary* instead of 2-ary.

Assume a set I, called an *index set.* An *I-indexed family of elements of A,* or briefly *family,* is a function $f : I \to A$. Instead of writing f, we will use the notation $(f_i \mid i \in I)$, where $f_i = f(i)$. When we write $(f_i \in X_i \mid i \in I)$, or $(f_i : X_i \to Y_i \mid i \in I)$, we mean that for the family $(f_i \mid i \in I)$, the element f_i is an element of the set X_i, or respectively, a function of type $X_i \to Y_i$, for every $i \in I$.

Asymptotic Bounds

We recall the following notation for asymptotic behavior of functions. Consider a function $f : \mathbb{N} \to \mathbb{R}$. We define the set $\mathcal{O}(f)$ to be the set of all functions $g : \mathbb{N} \to \mathbb{R}$ for which there are some $c \in \mathbb{R}$ with $c > 0$ and $n_0 \in \mathbb{N}$, such that for all $n \in \mathbb{N}$,

$$\text{if} \qquad n \geq n_0\,, \qquad \text{then} \qquad g(n) \leq c \cdot f(n)\,.$$

Dually, we let $\Omega(f)$ be the set of all $g : \mathbb{N} \to \mathbb{R}$ for which there are $c \in \mathbb{R}$ with $c > 0$ and $n_0 \in \mathbb{N}$, such that for all $n \in \mathbb{N}$,

$$\text{if} \qquad n \geq n_0\,, \qquad \text{then} \qquad f(n) \leq c \cdot g(n)$$

Note that for every f, $g : \mathbb{N} \to \mathbb{R}$, $f \in \mathcal{O}(g)$ if and only if $g \in \Omega(f)$.

Convention. *Following mathematical custom, we will from now on write $g(n) \in \mathcal{O}(f(n))$ instead of $g \in \mathcal{O}(f)$. When f is an expression containing several variables, we will mention which one is the parameter of f, if not clear.*

Further, when g is a function of type $\mathbb{N} \to \mathbb{N}$, and there is a function $g' : \mathbb{N} \to \mathbb{R}$ with $g'(n) \in \mathcal{O}(f(n))$, then we will write $g(n) \in \mathcal{O}(f(n))$ if the graphs of g and g' are equal. The analogous conventions apply to Ω.

1.1.2 Algebraic Structures

We will now recall some basic definitions from (universal) algebra. For thorough introductions to the topic, consult [76] and [164].

An *algebra* is a tuple $(A, f_1, \ldots, f_n)$, for some $n \in \mathbb{N}$, such that A is a set (the algebra's *carrier set*), and $f_1, \ldots, f_n$ are operations on A. Following the custom from mathematics, we will often denote an algebra briefly by its carrier set A. If, for an algebra A as above, f_i is a k_i-ary operation for every $i \in [n]$, then we say that the *type* of A is $(k_1, \ldots, k_n)$.

Consider an algebra $(A, f_1, \ldots, f_n)$, and an equivalence relation $\equiv$ on A. We call $\equiv$ a *congruence relation* if for every operation $f_i\colon A^{k_i} \to A$, with $i \in [n]$ and $k_i \in \mathbb{N}$, and for every $a_1,\ b_1,\ \ldots,\ a_{k_i},\ b_{k_i} \in A$,

$$\text{if} \qquad a_1 \equiv b_1\,, \quad \ldots, \quad a_{k_i} \equiv b_{k_i}\,, \qquad \text{then} \qquad f_i(a_1, \ldots, a_{k_i}) \equiv f_i(b_1, \ldots, b_{k_i})\,.$$

Consider two algebras $(A, f_1, \ldots, f_n)$ and $(B, g_1, \ldots, g_n)$, where $n \in \mathbb{N}$, such that A and B have the same type. We say that a function $h\colon A \to B$ is a *homomorphism* if for every $i \in [n]$, $f_i\colon A^{k_i} \to A$, and $a_1,\ \ldots,\ a_{k_i} \in A$, we have

$$h\big(f_i(a_1, \ldots, a_{k_i})\big) = g_i\big(h(a_1), \ldots, h(a_{k_i})\big)\,.$$

In the next subsections, we will recall some particular algebras occurring in this thesis.

Monoids

A *monoid* is an algebra $(M, \cdot, 1)$ such that $\cdot$ is a binary operation, $1 \in M$, and

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad \textit{(associativity)}$$

and

$$a \cdot 1 = 1 \cdot a = a \qquad \textit{(neutrality of } 1\textit{)}$$

for every $a,\ b,\ c \in M$. The operation $\cdot$ is called the monoid's *multiplication* or *product,* and 1 its *neutral element* or its *one element.* If the monoid $(M, \cdot, 1)$ additionally fulfills the axiom

$$a \cdot b = b \cdot a \qquad \textit{(commutativity)}$$

for every $a,\ b \in M$, then M is called a *commutative* monoid. In this case, one often writes $(M, +, 0)$ instead of $(M, \cdot, 1)$, refers to the monoid's binary operation as an *addition* or *sum*, and to its neutral element as its *zero.*

Convention. *Whenever we use multiplicative operators like $\cdot$, $\circ$, $\wedge$, $\ldots$, together with additive ones such as $+$, $\cup$, $\vee$, $\ldots$, we will assume that the multiplicative operators bind stronger. That is, the expression*

$$a \cdot b + c \qquad \textit{is to be read} \qquad (a \cdot b) + c\,.$$

Example 1.1. Since addition, as well as multiplication, of natural numbers are associative and commutative operations, the algebras $(\mathbb{N}, +, 0)$ and $(\mathbb{N}, \cdot, 1)$ are commutative monoids. For an archetypical example of a non-commutative monoid, consider the monoid $(A^A, \circ, \mathrm{id}_A)$ of functions on a set A with at least two elements. In fact, also the algebra $(\mathcal{P}(A \times A), ;, \mathrm{id}_A)$ of relations on A forms a non-commutative monoid.

For every set A, we can define the commutative monoid $(\mathcal{P}(A), \cup, \emptyset)$ of subsets of A. When we identify an element of A with the singleton set $\{a\}$, then we can represent every finite subset $B \subseteq A$ by a formal sum

$$B = a_1 + a_2 + \cdots + a_k = \sum_{i=1}^{k} a_i\,,$$

where $a_1, \ldots, a_k$ are the pairwise distinct elements of B, and $+$ denotes the monoid's addition $\cup$. We will make use of this notation later for denoting the productions of formal grammars.

For an example of a congruence relation, consider the additive monoid $(\mathbb{N}, +, 0)$ of natural numbers and define a relation $\equiv$ on $\mathbb{N}$ such that, for every $n, m \in \mathbb{N}$,

$$n \equiv m \qquad \text{if and only if} \qquad (n \text{ and } m \text{ are both even} \quad \vee \quad n \text{ and } m \text{ are both odd.})$$

It is easy to check that $\equiv$ is reflexive, symmetric, and transitive – therefore, $\equiv$ is an equivalence relation. Indeed, it is a congruence on $(\mathbb{N}, +, 0)$ because whether the sum $n + m$ is even depends only on whether the summands n and m are even.

Further, consider the monoid $(A, \oplus, 0)$ such that $A = \{0, 1\}$, and for every $a, b \in A$, we have

$$a \oplus b = 1 \qquad \text{if and only if} \qquad \{a, b\} = \{0, 1\} .$$

The function $h\colon \mathbb{N} \to A$ that maps every even number to 0 and every odd one to 1 is a homomorphism from $(\mathbb{N}, +, 0)$ to $(A, \oplus, 0)$. In fact, it is closely related to the congruence relation $\equiv$ from above, by the homomorphism theorem of universal algebra (cf. e.g. [164, Thm. 2 in Sec. 3.1.2]). $\lhd$

Semirings

A *semiring* is an algebra $(S, +, \cdot, 0, 1)$ such that $(S, +, 0)$ is a commutative monoid, $(S, \cdot, 1)$ is a monoid, and

$$a \cdot (b + c) = a \cdot b + a \cdot c\,, \quad (a + b) \cdot c = a \cdot c + b \cdot c\,, \qquad \textit{(distributivity)}$$

and

$$a \cdot 0 = 0 \cdot a = 0 \qquad \textit{(0 is annihilating)}$$

for every $a, b, c \in S$.

Example 1.2. The epitomical example of a semiring is the algebra $(\mathbb{N}, +, \cdot, 0, 1)$. Clearly, this algebra satisifies the semiring axioms.

Given a set A, we can define a semiring on the set of relations on A. Consider the algebra $(\mathcal{P}(A \times A), \cup, ;, \emptyset, \mathrm{id}_A)$. We already know from Example 1.1 that $(\mathcal{P}(A \times A), ;, \mathrm{id}_A)$ forms a monoid, and it is easy to see that $(\mathcal{P}(A \times A), \cup, \emptyset)$ is a commutative monoid. The proof that our algebra satisifies the remaining semiring axioms is an easy exercise in elementary set theory. $\lhd$

When considering semiring-weighted (tree) languages, it is often necessary that the underlying semiring is complete. We will now recall this notion introduced by Eilenberg [47, Sec. VI.2], cf. also [107].

Let $(S, +, \cdot, 0, 1)$ be a semiring. We say that S has an *infinite sum* $\sum$ if for every index set I, and every I-indexed family $(a_i \mid i \in I)$ of elements of S, the element $\sum(a_i \mid i \in I)$ of S is defined.[3] We will often denote $\sum(a_i \mid i \in I)$ by $\sum_{i \in I} a_i$.

[3]Note that considering *all* possible index sets may lead to set-theoretic antinomies. Yet, we will ignore these antinomies nonchalantly. For a rigorous treatment, see [81].

The semiring S is said to be *complete* if it has an infinite sum $\sum$ that satisfies the following axioms for all index sets I and J, families $(a_i \mid i \in I)$, and $b \in S$:

(i) $\displaystyle\sum_{i\in\emptyset} a_i = 0, \quad \sum_{i\in\{j\}} a_i = a_j, \quad \text{and} \quad \sum_{i\in\{j,k\}} a_i = a_j + a_k \text{ for } j \neq k,$

(ii) $\displaystyle b \cdot \Big(\sum_{i\in I} a_i\Big) = \sum_{i\in I} b \cdot a_i, \quad \text{and} \quad \Big(\sum_{i\in I} a_i\Big) \cdot b = \sum_{i\in I} a_i \cdot b,$

(iii) $\displaystyle\sum_{i\in I} a_i = \sum_{j\in J}\Big(\sum_{i\in I_j} a_i\Big)$ for every family $(I_j \mid j \in J)$ such that $\bigcup_{j\in J} I_j = I$ and $I_j \cap I_k = \emptyset$ for all $j \neq k \in J$.

Such a complete semiring will be denoted by $\big(S, +, \cdot, 0, 1, \sum\big)$. Intuitively, *(i)* demands that infinite sums are an extension of finite sums, *(ii)* is the infinite version of the distributivity law, and *(iii)* demands associativity of infinite sums.

1.1.3 Principles of Induction

As proofs by induction play a prominent role in this work, let us briefly recall some induction principles. One of the most basic such principles is well-founded, or Noetherian, induction (cf. i.a. [164, Sec. 1.3.4]). Given a relation R on a set A, we say that R is *well-founded* if there is no infinite sequence of elements $a_0, a_1, a_2, \ldots \in A$ such that $a_{i+1} R a_i$ for every $i \in \mathbb{N}$. Intuitively, if aRb is interpreted as "a is smaller than b", then R is not allowed to have infinite descending chains.

Then the *principle of well-founded (or Noetherian) induction* can be formulated as follows. Assume a well-founded relation R on a set A. Let $P \subseteq A$ be a property on A.[4] If for every $a \in A$,

$$\text{whenever} \quad \big\{x \in A \mid xRa\big\} \subseteq P \qquad \text{then also} \qquad a \in P\,,$$

then $P = A$. In our intuition, the "induction step" is to show for an arbitrary element a that, using the assumption that all elements x smaller than a satisfy P, then also a must satisfy P. Proving this induction step suffices to prove that all $a \in A$ satisfy P.

In the induction proofs in this thesis, we will not use the full power of well-founded induction. Many times, we will employ *(mathematical) induction* on the set $\mathbb{N}$ of natural numbers, sometimes also called *weak induction*. This kind of induction is an instance of Noetherian induction: just instantiate $A = \mathbb{N}$ and let

$$R = \big\{(n, m) \in \mathbb{N} \times \mathbb{N} \;\big|\; m = n + 1\big\}\,,$$

the successor relation on $\mathbb{N}$, which is clearly well-founded. Observe that then, the induction step of well-founded induction can be proven by considering two cases. In the first case, $a = 0$ and the set $\{x \in A \mid xRa\}$ is empty. So in this case, we must show, without any assumptions, that $0 \in P$, the *induction base*. In the second case, $a = n + 1$ for some $n \in \mathbb{N}$. Thus we must prove that assuming $n \in P$, then also $n + 1 \in P$, the *induction step* of mathematical induction on $\mathbb{N}$.

[4]Here, we identify a property on A with the set $P \subseteq A$ of all elements of A which fulfill the property.

The principle of *complete*, or *strong, induction* on $\mathbb{N}$ follows from well-founded induction as well. Instead of instantiating R with the successor relation, we let

$$R = \left\{(n,m) \in \mathbb{N} \times \mathbb{N} \;\middle|\; n < m\right\},$$

the relation "smaller than".

Thus, in the induction step, when we must prove P for $n+1$, we may assume that P holds for *all* numbers $m \leq n$. The induction base of strong induction is the same as in weak induction. Although this proof technique seems more powerful than weak induction, it is easy to show that both principles are logically equivalent. However, strong induction has the benefit that it will make some of our proofs more concise.[5]

In Section 1.3.1, we will encounter yet another principle of induction which is an instance of Noetherian induction, called *structural induction*.

Remark 1.3. As it can be deduced from the property that is to be proven, we will not state the induction hypothesis explicitly in most induction proofs. $\lhd$

[5] A note for our German readers: the German phrase "vollständige Induktion" designates weak induction, contrary to its literal translation. For an account of weak and strong induction, refer to [2, Sec. 2.4].

1.2 Formal Languages

This section is dedicated to recalling some facts from formal language theory. In particular, after establishing the basic notions and notation for words and word languages, we will call to mind the recognizable, context-free, indexed, and recursively enumerable languages. The latter are important in this work mainly in the context of complexity theory, so we will also give a brief refresher on some of the most important notions from this field.

The literature on formal language theory is rather extensive – but let us recommend [8, 86, 80] as introductions to formal language theory and computational complexity, and [67, 134] as further important references to complexity theory.

1.2.1 Words and Languages

Words

An *alphabet* is a finite nonempty set, its elements called *symbols*. Similarly, an *infinite alphabet* is a countable nonempty set, its elements also referred to as symbols. As the following definitions transfer smoothly to the case of infinite alphabets, we will give them just for alphabets.

Convention. *In this section, let Σ denote an arbitrary alphabet, unless specified otherwise.*

The set Σ^* is the set of all finite *words* over Σ, i.e., of all sequences

$$a_1 \cdots a_n \qquad \text{with } n \in \mathbb{N} \text{ and } a_1, \ldots, a_n \in \Sigma.$$

We will identify sequences of length 1 and mere symbols. Therefore, $\Sigma \subseteq \Sigma^*$. Let $w = a_1 \cdots a_n$ as above. The *length* of w is n, and denoted by $|n|$, while the *reversal* of w is $w^R = a_n \cdots a_1$. The empty word, i.e., the unique word of length 0, is denoted by ε, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. Given two words

$$w = a_1 \cdots a_n \qquad \text{and} \qquad v = b_1 \cdots b_m\,,$$

with $a_1, \ldots, a_n, b_1, \ldots, b_m \in \Sigma$, and $n, m \in \mathbb{N}$, their *concatenation* is

$$w \cdot v = a_1 \cdots a_n b_1 \cdots b_m\,.$$

Often, the operator $\cdot$ is omitted, and we write wv instead.

Remark 1.4. The algebra $(\Sigma^*, \cdot, \varepsilon)$ is a monoid. In fact, it is the *free monoid generated by* Σ – intuitively, the monoid generated by Σ with no identities but those induced by the monoid axioms [164, Sec. 3.2].

As a consequence, every function $h\colon \Sigma \to M$, for some monoid M, extends to a unique homomorphism $\tilde{h}\colon \Sigma^* \to M$. In particular, a homomorphism $\tilde{h}\colon \Sigma^* \to \Delta^*$ between two alphabets Σ and Δ is given uniquely by the images $h(a)$ for every $a \in \Sigma$. It is customary to identify h and $\tilde{h}$. ◁

By iterating concatenation, we can define the *power* of a word. Formally, for every $w \in \Sigma^*$, let $w^0 = \varepsilon$ and $w^{j+1} = w \cdot w^j$ for every $j \in \mathbb{N}$.

We extend the notion of length as follows. For every $A \subseteq \Sigma$ and $w \in \Sigma^*$, let $|w|_A$ denote the number of occurrences of a symbol from A in w, i.e.,

$$|w|_A = \sum_{\substack{i \in [|w|],\\ a_i \in A}} 1\,.$$

If $A = \{a\}$ for some $a \in \Sigma$, we will briefly write $|w|_a$ instead.

Assume words v, $w \in \Sigma^*$. We say that v is a *factor* of w if there are u, $y \in \Sigma^*$ such that $w = uvy$. If additionally $u = \varepsilon$ (resp. $y = \varepsilon$), then v is a *prefix* (resp. a *suffix*) of w. We will write $v \sqsubseteq w$ if v is a prefix of w. It is easy to see that $\sqsubseteq$ is a partial order on Σ^*. Furthermore, we write $v \parallel w$ if v and w are incomparable with respect to the prefix relation, i.e., if neither $v \sqsubseteq w$ nor $w \sqsubseteq v$.

Alternatively, Σ^* can be ordered in the manner of a librarian. Formally, presume a partial order $\leq$ on Σ. Then the *lexicographic order* $\leq_{\text{lex}}$ on Σ^* is defined such that for every v, $w \in \Sigma^*$, $v \leq_{\text{lex}} w$ if either $v \sqsubseteq w$ or there are u, y, $z \in \Sigma^*$ and a, $b \in \Sigma$ such that

$$a < b\,, \qquad v = uay\,, \qquad \text{and} \qquad w = ubz\,.$$

Again, $\leq_{\text{lex}}$ is a partial order, and it is total if $\leq$ is so.

Formal Languages

A *(formal) language (over Σ)* is merely a set $L \subseteq \Sigma^*$ of words over Σ. Hence the language-theoretic operations union $\cup$, intersection $\cap$, and difference $\setminus$ are already defined.

We recall the following operations which are specific to formal languages. Let L, $L' \subseteq \Sigma^*$. Then $L \cdot L' = \{wv \mid w \in L, v \in L'\}$, the *complex product* or *concatenation* of L and L'. Again, we sometimes omit the operator $\cdot$ and write LL' instead. Furthermore, let

$$L^0 = \{\varepsilon\} \qquad \text{and} \qquad L^{i+1} = L \cdot L^i \quad \text{for every } i \in \mathbb{N},$$

and let

$$L^* = \bigcup_{i \in \mathbb{N}} L^i \qquad \text{and} \qquad L^+ = \bigcup_{i \in \mathbb{N}_1} L^i\,.$$

In all these operations, when an operand is a singleton $\{a\}$, we will often omit the braces and write, e.g., a^* instead of $\{a\}^*$, or aL instead of $\{a\}L$.[6]

1.2.2 Recognizable Languages

The first class of formal languages we are going to recall is one of the most basic and important language classes in computer science – the class of languages recognized by finite-state automata. This class of recognizable languages forms the lowest level of the Chomsky hierarchy.

[6]The attentive reader might object that this introduces an ambiguity – is w^n a word or a (singleton) language? However, the intended meaning will always be clear from the context of the expression, and there should be no danger of confusion.

Finite-State Automata

A *finite-state automaton (fsa)* is a tuple $A = (Q, \Sigma, I, F, \delta)$ such that

- Q is a finite set (its elements called *states*),
- Σ is an alphabet,
- I and F are subsets of Q (their elements called *initial* resp. *final states*), and
- $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$, (the *transition table*).

The function δ is extended to $\tilde{\delta}\colon \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$ by setting

$$\tilde{\delta}(P, \varepsilon) = P \qquad \text{and} \qquad \tilde{\delta}(P, aw) = \tilde{\delta}\Big(\bigcup_{q \in P} \delta(q, a), w\Big)$$

for every $P \subseteq Q$, $a \in \Sigma$, and $w \in \Sigma^*$. In this way, we associate to every fsa $A = (Q, \Sigma, I, F, \delta)$ its *recognized language*

$$\mathcal{L}(A) = \left\{ w \in \Sigma^* \;\middle|\; \tilde{\delta}(I, w) \cap F \neq \emptyset \right\}.$$

We say that a language is *recognizable* if it is recognized by some fsa. The class of all recognizable languages (over Σ) is denoted by REC (resp. by $\mathsf{REC}(\Sigma)$).

An fsa $A = (Q, \Sigma, I, F, \delta)$ is called *deterministic* if $|I| \leq 1$, and for every $q \in Q$ and $a \in \Sigma$, the set $\delta(q, a)$ contains at most one element. Similarly, A is *total* if $|I| > 0$, and for every $q \in Q$ and $a \in \Sigma$, there is at least one element in $\delta(q, a)$. Deterministic and total fsa are abbreviated *dfa*. In this case, we denote $A = (Q, \Sigma, q_0, F, \delta)$, where q_0 is the unique element of I, and take δ, and thus also $\tilde{\delta}$, to be functions of type $Q \times \Sigma \to Q$, resp. $Q \times \Sigma^* \to Q$. The following classic theorem of automata theory states that the restriction to dfa has no detriment to the power of recognition.

Theorem 1.5 (Rabin and Scott [137, Thm. 11])**.** *For every $L \in \mathsf{REC}$, there is a deterministic and total finite-state automaton A with $\mathcal{L}(A) = L$.*

Remarks

An early definition of finite-state automata can be found in [137]. The authors of this work cite even earlier formalizations of what is essentially the same model.

There is a plethora of ways to define the recognizable languages, but in this work we will make do with finite-state automata. For the sake of completeness, let us just mention the characterizations of REC by regular grammars [22], rational expressions [101], algebraic recognizability [121], or monadic second-order logic [29].

1.2.3 Context-Free Languages

We continue with the next level of the Chomsky hierarchy – the class of context-free languages.

Context-Free Grammars

A *context-free grammar (cfg)* is a tuple $G = (N, \Sigma, S, P)$ such that

- N is an alphabet (its elements called *nonterminal symbols* or just *nonterminals*),
- Σ is an alphabet disjoint from N (its elements called *terminal symbols* or *terminals*),
- $S \in N$ (the *initial nonterminal*), and
- P is a finite set (its elements called *productions*), where each production is of the form

$$A \to \varrho \qquad \text{for some } A \in N \text{ and } \varrho \in (N \cup \Sigma)^*.$$

For a cfg G as above, we will call every element of $(N \cup \Sigma)^*$ a *sentential form.* Let $p \in P$ be a production of form $A \to \varrho$. The *rewrite relation by p*, denoted by $\Rightarrow_p$, is defined to be the smallest relation on $(N \cup \Sigma)^*$ such that

$$\xi \cdot A \cdot \zeta \;\Rightarrow_p\; \xi \cdot \varrho \cdot \zeta \qquad \text{for every } \xi, \zeta \in (N \cup \Sigma)^*.$$

The *rewrite relation of G* is then $\Rightarrow_G = \bigcup_{p \in P} \Rightarrow_p$. When G is understood, we will sometimes drop the subscript and write $\Rightarrow$ instead of $\Rightarrow_G$. The *language generated by G* is defined to be

$$\mathcal{L}(G) = \left\{ w \in \Sigma^* \mid S \Rightarrow_G^* w \right\}.$$

A language which is generated by some context-free grammar is called *context-free.* The class of all context-free languages (over an alphabet Σ) is denoted by CF (resp. by $\mathsf{CF}(\Sigma)$).

A cfg G is in *(Chomsky) normal form* if each of its productions $A \to \varrho$ satisfies

$$\varrho \in N^2 \cup \Sigma \cup \{\varepsilon\},$$

and $\varrho = \varepsilon$ only if $A = S$.

Theorem 1.6 (Chomsky [32, Thm. 5]). *For every $L \in \mathsf{CF}$, there is a cfg G in Chomsky normal form such that $\mathcal{L}(G) = L$.*

Convention. *We will often denote a finite set of cfg productions $\{A \to \varrho_1, \ldots, A \to \varrho_k\}$ with common left-hand side A by*

$$A \to \varrho_1 + \cdots + \varrho_k, \qquad \textit{or even by} \qquad A \to \sum_{i=1}^{k} \varrho_i.$$

This representation by formal sums has already been introduced in Example 1.1.

Dyck Languages

We continue with defining some very prominent inhabitants of CF – the Dyck languages; cf. [21, Sec. 1.2]. For an alphabet Σ, take some disjoint alphabet $\bar{\Sigma}$ in bijection to Σ, say $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$. Define the *Dyck alphabet* (or *parenthesis alphabet*) $\Gamma = \Sigma \cup \bar{\Sigma}$. Let $\equiv$ denote the least congruence relation on Γ^* such that for every $a \in \Sigma$, we have $a\bar{a} \equiv \varepsilon$. We say that $\bar{a}$ is the *right inverse* of a, and vice versa a the *left inverse* of $\bar{a}$. Let v, $w \in \Gamma^*$. By saying that w *reduces to* v (resp. v is the *reduct* of w), we mean that v is the (unique) shortest word in Γ^* such that $v \equiv w$. Clearly, $v \equiv w$ if and only if v and w have the same reduct.

The *Dyck language over* Γ is defined as

$$D_\Gamma^* = \left\{ w \in \Gamma^* \;\middle|\; w \equiv \varepsilon \right\}.$$

Often, we will also call the elements of Σ *opening,* and those of $\bar{\Sigma}$ *closing parentheses.* Intuitively, D_Γ^* contains then all well-parenthesized words over Γ. It is well-known that D_Γ^* is context-free; in fact, it is generated by the cfg given by the productions

$$S \to \sum_{a \in \Sigma} aS\bar{a}S + \varepsilon .$$

Remark 1.7. The prominence of the Dyck languages is due to the fact that already for $\Gamma = \{a, b, \bar{a}, \bar{b}\}$, D_Γ^* generates the whole class CF under rational transductions [33, 126]. $\lhd$

We need the following lemma on partial cancellability of the Dyck congruence. Pay attention to the quantification of w_1 and w_2 below.

Lemma 1.8. *Consider a Dyck alphabet $\Gamma = \Sigma \cup \bar{\Sigma}$ as defined above, and words $v \in \Gamma^*$, as well as w_1, $w_2 \in \Sigma^* \cup \bar{\Sigma}^*$. If $vw_1 \equiv vw_2$, then $w_1 \equiv w_2$. Similarly, if $w_1 v \equiv w_2 v$, then $w_1 \equiv w_2$.*

Proof. Let $n \in \mathbb{N}$ and $v \in \Gamma^n$. We prove that for every $w_1, w_2 \in \Sigma^* \cup \bar{\Sigma}^*$, whenever $vw_1 \equiv vw_2$, then $w_1 \equiv w_2$. Clearly, this shows the lemma's first implication. The second implication can be proven analogously.

The proof of the above statement is by complete induction on n, and the base case $n = 0$ holds trivially. Suppose the property holds for all $n' \leq n$ for some $n \in \mathbb{N}$, and consider $v \in \Gamma^n$ and $a \in \Gamma$. Assume that $vaw_1 \equiv vaw_2$ for some words w_1, $w_2 \in \Sigma^* \cup \bar{\Sigma}^*$. We make the following case analysis, where we denote the reduct of a word $w \in \Gamma^*$ by $r(w)$.

(I) Let $a \in \Sigma$. Thus a is an opening parenthesis. We proceed by considering all combinations of the following properties.

(P1) There are y_1 and $z_1 \in \Gamma^*$ such that

$$aw_1 = ay_1\bar{a}z_1 \qquad \text{and} \qquad y_1 \equiv \varepsilon .$$

(P2) There are y_2 and $z_2 \in \Gamma^*$ such that

$$aw_2 = ay_2\bar{a}z_2 \qquad \text{and} \qquad y_2 \equiv \varepsilon .$$

Note that if *(P1)* does not hold, then $r(vaw_1) = r(v)\,a\,r(w_1)$, as a is an opening parenthesis that has no matching parenthesis in w_1 (and analogously when *(P2)* does not hold). We continue with the case analysis.

(i) Assume *(P1)* and *(P2)* hold. Then $vaw_1 \equiv vz_1$ and $vaw_2 \equiv vz_2$, and therefore $vz_1 \equiv vz_2$. As $z_1, z_2 \in \Sigma^* \cup \bar{\Sigma}^*$, we can apply the induction hypothesis, and obtain that $z_1 \equiv z_2$. Thus also $w_1 = y_1\bar{a}z_1 \equiv y_2\bar{a}z_2 = w_2$.

(ii) Assume that *(P1)* holds, but *(P2)* does not. Then $r(vaw_1) = r(vz_1)$, and $r(vaw_2) = r(v)\,a\,r(w_2)$. Note that by our assumption on $w_1 = y_1\bar{a}z_1$, we have $z_1 \in \bar{\Sigma}^*$. Then $|r(vz_1)| \leq |r(v)|$, as z_1 contains only closing parentheses. But $|r(v)\,a\,r(w_2)| > |r(v)|$, thus $r(vaw_1) \neq r(vaw_2)$ and hence $vaw_1 \not\equiv vaw_2$, in contradiction to our assumption. So this case does not occur.

(iii) The case that *(P2)* holds, but *(P1)* does not, is precluded by an analogous argument.

(iv) Assume that neither *(P1)* nor *(P2)* hold. Then

$$r(v)\,a\,r(w_1) = r(vaw_1) = r(vaw_2) = r(v)\,a\,r(w_2)\,,$$

and thus $r(w_1) = r(w_2)$. Therefore, $w_1 \equiv w_2$.

(II) Let $a \in \bar{\Sigma}$. Thus a is a closing parenthesis of form $\bar{b}$, for some $b \in \Sigma$. There are two subcases.

(i) There are $y, z \in \Gamma^*$ such that $v\bar{b} = ybz\bar{b}$ and $z \equiv \varepsilon$. Then

$$v\bar{b}w_1 \equiv yw_1 \qquad \text{and} \qquad v\bar{b}w_2 \equiv yw_2\,.$$

By the induction hypothesis, $w_1 \equiv w_2$.

(ii) There are no such words y and z, and therefore $r(v\bar{b}) = r(v)\bar{b}$. Thus

$$r(v\bar{b}w_1) = r(v)\,\bar{b}\,r(w_1) \qquad \text{and} \qquad r(v\bar{b}w_2) = r(v)\,\bar{b}\,r(w_2)\,,$$

and hence $r(w_1) = r(w_2)$. So $w_1 \equiv w_2$. □

Remark 1.9. Due to the assumption in the lemma that $w_1, w_2 \in \Sigma^* \cup \bar{\Sigma}^*$, we even have the stronger property that if $vw_1 \equiv vw_2$, then $w_1 = w_2$, and similarly for composition from the right. We chose to state the lemma the way it is as it expresses a (restricted) cancellation law for the Dyck congruence.

Note that $\equiv$ does not enjoy unrestricted cancellability. For a counterexample, consider $v = a$, $w_1 = \bar{a}a$, and $w_2 = \varepsilon$. Then $vw_1 \equiv vw_2$, but $w_1 \not\equiv w_2$.[7] ◁

[7]This counterexample has been communicated by J.-É. Pin.

Remarks

Context-free grammars were first proposed by Chomsky [31] as a model for linguistics. Soon, they found applications in computer science – i.a., to define the syntax of ALGOL-like languages [74].

As for the recognizable languages, there are quite a number of ways to define the context-free languages. We have already mentioned that CF is the image of a particular Dyck language under rational transductions. This result is closely related to the famous theorem of Chomsky and Schützenberger [33], as well as to the characterization given by Shamir [151].

Moreover, CF is precisely the class of languages recognized by pushdown automata [148], and there is also a characterization of CF by means of logics [108].

1.2.4 Indexed Languages

The *indexed languages* were discovered by Aho [3], when he extended the nonterminals of context-free grammars by a pushdown store.

Indexed Grammars

An *indexed grammar (ixg)* is a tuple $G = (N, \Sigma, \Gamma, S, P)$ such that

- N is an alphabet (of *nonterminals*),
- Σ is an alphabet disjoint from N (of *terminals*),
- Γ is an alphabet disjoint from N and Σ (its elements called *pushdown symbols* or also *flags*),
- $S \in N$ (the *initial nonterminal*), and
- P is a finite set (of *productions*), where each production is of the form

$$A\gamma \to \varrho \qquad \text{for some } A \in N,\ \gamma \in \Gamma \cup \{\varepsilon\}, \text{ and } \varrho \in (N\Gamma^* \cup \Sigma)^*.$$

We say that an ixg G, as given above, is *ε-free* if each of its productions $A\gamma \to \varrho$ satisfies the condition $\varrho \neq \varepsilon$.

To define the rewrite relation of an ixg $G = (N, \Sigma, \Gamma, S, P)$, we require the following auxiliary definitions. Let, for every $\gamma \in \Gamma$,

$$\begin{aligned} a^\gamma &= a && \text{for every } a \in \Sigma \cup \{\varepsilon\}, \\ (A\eta)^\gamma &= A\eta\gamma && \text{for every } A\eta \in N\Gamma^*, \text{ and} \\ (\xi \cdot \zeta)^\gamma &= \xi^\gamma \cdot \zeta^\gamma && \text{for every } \xi,\ \zeta \in (N\Gamma^* \cup \Sigma)^*. \end{aligned}$$

Moreover, let $\xi^\varepsilon = \xi$ and $\xi^{\gamma\eta} = (\xi^\gamma)^\eta$ for every $\xi \in (N\Gamma^* \cup \Sigma)^*$, $\gamma \in \Gamma$, and $\eta \in \Gamma^*$. As a give-away of this definition of "exponentiation", we can use the neater notation A^η instead of $A\eta$.

Given a production $p \in P$ of the form $A^\gamma \to \varrho$ as above, we define the *rewrite relation by* p, denoted by $\Rightarrow_p$, to be the smallest relation on $(N\Gamma^* \cup \Sigma)^*$ such that

$$\xi \cdot A^{\gamma\eta} \cdot \zeta \ \Rightarrow_p \ \xi \cdot \varrho^{\eta} \cdot \zeta \qquad \text{for every } \xi, \zeta \in (N\Gamma^* \cup \Sigma)^* \text{ and } \eta \in \Gamma^*.$$

Again, we set the *rewrite relation of* G to be $\Rightarrow_G = \bigcup_{p \in P} \Rightarrow_p$, omit the subscript G from $\Rightarrow_G$ whenever there is no danger of confusion, and define the *language generated by* G to be

$$\mathcal{L}(G) = \left\{ w \in \Sigma^* \;\middle|\; S \Rightarrow_G^* w \right\}.$$

A language that is generated by an ixg is said to be an *indexed language,* and the class of all indexed languages (over Σ) is denoted by IND (resp. by IND(Σ)).

Remark 1.10. The above definition of indexed grammars is akin to the one given by Hopcroft and Ullman [86, Sec. 14.3], while the definition of "exponentiation" is an idea of Maslov [119].

Note that Aho's original definition of indexed grammars takes flags to be sets of productions themselves [3]. We have avoided this definition, as it makes some constructions a little cumbersome. ◁

An indexed grammar is in *normal form* if each of its productions is of one of the forms

(i) $A \to B_1 \cdots B_n$,

(ii) $A \to a$,

(iii) $A \to B^\gamma$, or

(iv) $A^\gamma \to B$,

for some $n \in \mathbb{N}_1$, $A, B, B_1, \ldots, B_n \in N$, $a \in \Sigma$, and $\gamma \in \Gamma$; moreover, we allow the special production $S \to \varepsilon$. The following theorem shows that this restriction still allows to generate all indexed languages.

Theorem 1.11 (Aho [3, Thm 4.5])**.** *For every* $L \in$ IND*, there is an ixg* G *in normal form such that* $\mathcal{L}(G) = L$.

In particular, if $\varepsilon \notin L$, then G can be chosen ε-free.

Remarks

As mentioned above, the definition of indexed grammars is due to Aho [3]. There is another grammar model which generates the indexed languages, namely the (OI) *macro grammars,* which are the result of augmenting the nonterminals of a cfg with parameters [60]. As macro grammars are closely related to context-free tree grammars, we will not define them seperately.

Moreover, the indexed languages are recognized by nested stack automata [4] and, equivalently [59], by 2-iterated pushdown automata. There is a homomorphic characterization of IND by means of a Chomsky-Schützenberger-like theorem [46], which has recently been rediscovered and extended [62]. Moreover, a very general Chomsky-Schützenberger-like theorem by means of automata with storage has been found, which can also be instantiated to the special case of IND [82].

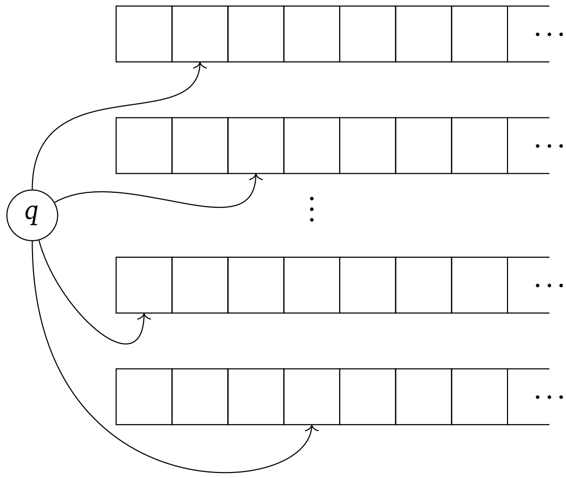


Figure 1.2: A Turing machine

1.2.5 Recursively Enumerable Languages and Complexity Classes

In this section, we recall Turing machines [162], their accepted languages, as well as some basic complexity theory.

Turing Machines

Our basic computational model will be the *multi-tape, off-line Turing machine.* As the algorithms and reductions in this thesis will be given by pseudo-code instead of by specifying a Turing machine, we will refrain from giving a formal definition of this model, and describe its operation in prose. The reader may refer to standard introductions like [86, 134] for a more thorough treatment of the topic.

Let $k \in \mathbb{N}$ with $k \geq 2$. Then a *(k-tape, off-line) Turing machine (tm)* is a machine (as shown in Figure 1.2) with a *finite state control,* and with *k tapes*. A tape consists of an unbounded number of *cells,* and every cell of a tape may contain a symbol from a specified *work alphabet* Σ, or a special *blank symbol.* For each of its tapes, the machine possesses a *read-write head,* or *cursor,* each of which points to one of the cells of the respective tape. We call the first tape the machine's *input tape*, the k-th tape its *output tape,* and the other tapes are its *work tapes.*

So a configuration of a tm M is given by its state, the contents of its tapes, and the positions of its read-write heads. Conditioned on M's current state and the symbols $a_1, \dots, a_k$ under its cursors, M may take a *transition*. In a transition, the state of M can be changed, and the respective symbols a_i under the read-write heads may be overwritten. Moreover, the cursors on each tape may move independently one cell to the left, remain at their position, or move one cell to the right (possibly adding a cell that contains the blank symbol to the tape). We make the assumption that in every transition, the input tape is left unmodified (i.e., the machine is off-line [86, p. 166]), and the output tape cursor never moves to the left. Each tm M possesses a number of designated *final states*, and we will assume that there are no transitions that start in a final state. If for every state and every tuple of symbols under the cursors, there is at most one transition M can take, then we call M *deterministic*. Without this restriction, M is said to be *nondeterministic*.

A *computation* of M is a sequence of consecutive transitions as described above. For an input $w \in \Sigma^*$, the tm M begins its computation in some specified initial state q_0 with w on its input tape, and all other tapes empty (i.e., they have only one cell, which contains the blank symbol). The input tape cursor is on the first symbol of w. We call this configuration the *initial configuration of M with input w*. A configuration of M is said to be *final* if M is in a final state. The *output* of such a configuration is the sequence of symbols on the output tape from the first cell up to, but excluding, the first cell that contains a blank symbol. We say that M *halts* for an input $w \in \Sigma^*$ if there is some number $\ell \in \mathbb{N}$ such that every computation of M that begins in the initial configuration with input w has length at most ℓ.

The *language accepted* by a tm M is the set $L(M)$ of all $w \in \Sigma^*$ such that M reaches a final configuration in a finite number of transitions, starting in its initial configuration with input w. Note that there may be more than one computation starting in this configuration, but for acceptance, the existence of just one computation which reaches a final configuration is sufficient. A language is said to be *recursively enumerable* if it is accepted by a Turing machine. Similarly, the *transformation computed by M* is the relation $T(M)$ that contains all tuples $(w, v) \in \Sigma^* \times \Sigma^*$ such that M reaches, beginning in the initial configuration with input w, a final configuration with output v, in a finite number of transitions. If M is deterministic, then $T(M)$ is a partial function of type $\Sigma^* \rightharpoonup \Sigma^*$.

Let $f : \mathbb{N} \to \mathbb{N}$ and M be a (deterministic) tm. We say that M *operates in (deterministic) time* $f(n)$ if for each $w \in \Sigma^*$, the length of every computation starting in the initial configuration with input w is bounded by $f(|w|)$.[8] Moreover, M *operates in (deterministic) space* $f(n)$ if for each $w \in \Sigma^*$ and every configuration that is reachable by a computation starting with the initial configuration with input w, the number of work tape cells that contain a non-blank symbol is bounded by $f(|w|)$.

The class of all languages which are accepted by a deterministic tm that operates in time (resp. space) $f(n)$ is denoted by $\mathsf{DTIME}(f(n))$ (resp. $\mathsf{DSPACE}(f(n))$), while the class of all languages accepted by any nondeterministic tm that operates in time (resp. space) f will be denoted by $\mathsf{NTIME}(f(n))$ (resp. $\mathsf{NSPACE}(f(n))$).

We are now in a position to define the following basic time and space complexity classes:

$$\mathsf{P} = \bigcup_{k \in \mathbb{N}} \mathsf{DTIME}(n^k), \qquad \mathsf{NP} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}(n^k),$$

$$\mathsf{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathsf{DSPACE}(n^k), \quad \mathsf{NPSPACE} = \bigcup_{k \in \mathbb{N}} \mathsf{NSPACE}(n^k), \quad \mathsf{EXP} = \bigcup_{k \in \mathbb{N}} \mathsf{DTIME}(2^{n^k}).$$

Moreover, if a partial function $\tau \colon \Sigma^* \rightharpoonup \Sigma^*$ is computed by some deterministic tm that operates in space $\log n$, then τ is said to be *computable in logarithmic space*, or briefly *logspace-computable*.

[8] The variable n in $f(n)$ serves as a placeholder for the input word's length. It would be more correct to write "in time f" or "$\lambda n.f(n)$" instead, but we chose to follow the established convention.

We cite the following two theorems, which demonstrate why we can disregard coefficients in the definitions above. Both theorems appear to be folklore; refer to [134, Thm. 2.2 & 2.3] for their proofs.

Theorem 1.12 (*Linear Speedup*). *Let $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$, $f : \mathbb{N} \to \mathbb{N}$, and M be a (deterministic) tm that operates in time $f(n)$. There is a (deterministic) tm M' that operates in time $\lceil \varepsilon f(n) + n + 2 \rceil$ such that $L(M') = L(M)$ and $T(M') = T(M)$.*

Here, $\lceil \cdot \rceil$ denotes the ceiling function: for every $a \in \mathbb{R}$, $\lceil a \rceil$ is the smallest integer z such that $a \leq z$.

Theorem 1.13 (*Tape Compression*). *Let $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$, $f : \mathbb{N} \to \mathbb{N}$, and M be a (deterministic) tm that operates in space $f(n)$. Then there is a (deterministic) tm M' that operates in space $\lceil \varepsilon f(n) + 2 \rceil$ such that $L(M') = L(M)$ and $T(M') = T(M)$.*

As proven by Savitch, nondeterminism can be simulated by deterministic tm with only quadratic increase in required work space.

Theorem 1.14 (Savitch [144]). *Let $f : \mathbb{N} \to \mathbb{N}$ be a proper complexity function*[9] *such that $f(n) \geq \log n$ for every $n \in \mathbb{N}$. Then $\mathsf{NSPACE}(f(n)) \subseteq \mathsf{DSPACE}(f(n)^2)$.*

As an important consequence, $\mathsf{NPSPACE} = \mathsf{PSPACE}$. It is one of the great open questions of computer science whether there is a similar result for time complexity; cf. [1] for an extensive survey article. The relationship between the above complexity classes is summarized by

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE} \subseteq \mathsf{EXP}\,,$$

where the only inclusion that is known to be proper is $\mathsf{P} \subset \mathsf{EXP}$.

Reductions, Hardness, Completeness

Assume languages L_1, L_2 over some common alphabet Σ. We say that L_1 is *logspace-reducible* to L_2, denoted by $L_1 \preceq_{\log} L_2$, if there is a logspace-computable partial function $\tau : \Sigma^* \nrightarrow \Sigma^*$ such that for every $w \in \Sigma^*$, we have $x \in L_1$ if and only if $\tau(x) \in L_2$. It is well-known that the relation $\preceq_{\log}$ is reflexive and transitive [134, Prop. 8.2].

Now, assume a class of languages $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$. A language $L \subseteq \Sigma^*$ is said to be *hard* for $\mathcal{C}$ (or briefly *$\mathcal{C}$-hard*) if for every $L' \in \mathcal{C}$, we have $L' \preceq_{\log} L$. If L is $\mathcal{C}$-hard and $L \in \mathcal{C}$, then L is said to be *complete* for $\mathcal{C}$ (or *$\mathcal{C}$-complete*).

The following lemma helps in proving hardness of a language. It is an easy consequence of the transitivity of $\preceq_{\log}$.

Lemma 1.15. *Let L be $\mathcal{C}$-hard for a class of languages $\mathcal{C}$. Every language L' with $L \preceq_{log} L'$ is $\mathcal{C}$-hard, too.*

[9] Broadly spoken, every "reasonable" function is a proper complexity function. Formally, we demand that f is monotonic and space-constructible (cf. [134, Def. 7.1]).

Remark 1.16. A function is logspace-computable if it is given by an algorithm with a constant number k of integer variables (or *counters*) that range over the set $[n]$ for an input of size n. The "trick" in implementing such an algorithm in logarithmic space is by designing a deterministic Turing machine M with k work tapes, each containing the value of one of the counters, stored as a binary number. Clearly, the work space used by M is then bounded by $k \cdot \log n$, and by Theorem 1.13, we can find a tm M' that computes our function operating in space $\log n$.[10]

Every deterministic tm that operates in logarithmic space does so in polynomial time [134, Prop. 8.1]. Therefore our notion of reducibility is finer than the other popular notion defined by transformations computable in polynomial time. It is still unknown whether the two notions coincide. ◁

Decision Procedures and Decision Problems

The basic object of study in complexity theory is the *decision problem*. A decision problem can be understood as a Boolean predicate on a specified set of *problem instances*.

We specify a decision problem in the well-known format popularized in [67], which consists of two lines. The first line gives an abstract problem instance, while the question in the second line determines the predicate which is to be decided. For example, the uniform membership problem of context-free grammars is defined as follows.

Problem: Context-Free Grammar Uniform Membership
Instance: A cfg $G = (N, \Sigma, S, P)$ and a word $w \in \Sigma^*$
Question: Is $w \in \mathcal{L}(G)$?

So in this case, the set of problem instances contains all tuples (G, w), where G is a cfg over some terminal alphabet Σ and $w \in \Sigma^*$, and the predicate holds for (G, w) if and only if $w \in \mathcal{L}(G)$.

Our aim is to find a Turing machine which decides such a problem in optimal time or space. But in order to present a problem instance as input to a Turing machine, it must be encoded as a word over some alphabet Γ. In the above example, it might for example be convenient to take $\Gamma = \{0, 1, \$\}$, and express every symbol from $\Sigma \cup N$ uniquely by a code over $\{0, 1\}$. The symbol \$ serves as a separator.

The cfg $G = (\{A, B\}, \{a, b\}, A, P)$ with P comprising the productions

$$A \to aBb\,, \qquad B \to \varepsilon + BA$$

might then be represented by the word

$$\$\underbrace{10}_{|N|}\$\underbrace{10}_{|\Sigma|}\$\$\underbrace{10\$001101}_{A\to aBb}\$\$\underbrace{11\$}_{B\to\varepsilon}\$\$\underbrace{11\$1110}_{B\to BA}\$\$\,,$$

where a is encoded as 00, b as 01, A as 10, and B as 11.

[10] The constant space overhead of two cells that is implied by the theorem can be avoided by using the machine's finite state control.

This example shows nicely why, in the following, we will abstain from defining the concrete encoding of a problem. However, we follow the convention of assuming the encoding to be *reasonable.* While it seems hard to define precisely what "reasonable" means in this context, note that in the above example, coding the symbols as unary numbers instead of binary numbers would qualify as *unreasonable,* as such an encoding would take an exponentially larger amount of space.

So, let us assume a reasonable encoding. Then formally, a decision problem is defined as a tuple of languages (I, P) over some fixed alphabet Σ. The language I contains all (encodings of) the problem instances, while $P \subseteq I$ is the set of all (encodings of) instances which satisfy the predicate. A Turing machine *decides* a decision problem if it halts on every input from I, and each $w \in I$ is accepted by the machine if and only if $w \in P$. In this case, we say that the Turing machine implements a *decision procedure* for the problem. Moreover, we say that a decision problem (I, P) is in a complexity class $\mathcal{C}$ (or $\mathcal{C}$-hard, $\mathcal{C}$-complete, etc.) if the language P is so.

In the following, we will not properly define any Turing machines to specify decision procedures. Instead, we follow the established custom to give an algorithm in pseudo-code. Of course, this takes two things for granted: *(i)* that it is clear how to implement the given piece of pseudo-code in a Turing machine; *(ii)* that the implementation of the algorithm by a Turing machine does not drastically worsen the time or space complexity of the procedure. We claim that point *(i)* is satisfied for the given algorithms, and that in principle, it is possible, if tiring, to implement them by a Turing machine. As for point *(ii)*, we will attest to the efficiency of implementation by means of proof or reference.

Remark 1.17. In this work, most algorithms and decision procedures deal with trees (cf. Section 1.3.1 below). We note that assuming a reasonable encoding, operations on trees such as determining the j-th subtree of a node, the label at a given position, or substitution, are logspace-computable; cf. [113, Lem. 2]. ◁

Propositional Satisfiability

The archetypical NP-hard decision problem is the satisfiability problem of propositional logic. Here, we will consider the satisfiability problem of *propositional logic formulas in 3-conjunctive normal form (3-cnf formulas)*. Such a formula is a word of the form

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge \cdots \wedge (L_1^m \vee L_2^m \vee L_3^m) \tag{1.2}$$

over the alphabet $\Gamma = \{0, 1, \neg, \vee, \wedge, (,)\}$, such that $m \in \mathbb{N}_1$, and for every $i \in [m]$ and $j \in [3]$, we have

$$L_j^i \in \{\varepsilon, \neg\} \cdot 1 \cdot \{0, 1\}^* .$$

Intuitively, the words L_j^i are (positive and negated) literals, where a propositional variable v_i with $i \in \mathbb{N}_1$ is represented by its index i in binary notation and without leading zeroes. For brevity's sake, we will identify v_i and the binary representation of i, and say that a formula contains v_i if it contains the representation of i. The set of all propositional variables is denoted $V = \{v_1, v_2, \ldots\}$, and for every $k \in \mathbb{N}$, we let $V_k = \{v_i \mid i \in [k]\}$.

Moreover, we will assume that the propositional variables' indices of the formula in (1.2) are assigned consecutively, i.e., if a 3-cnf formula φ contains the variable $v_n \in V$ for some $n \in \mathbb{N}_1$, then it must also contain each variable $v_1, \ldots, v_{n-1}$. Note that this is no restriction – a Turing machine which relabels the variables' indices in this manner can clearly be implemented in deterministic logarithmic space.[11]

Consider a formula φ of the form in (1.2) over the variables $v_1, \ldots, v_n$, for some $n \in \mathbb{N}$. A *(truth) assignment* for φ is a mapping

$$a\colon V_n \to \mathbb{B}\,.$$

We extend a as follows. Let $a(\neg v_i) = \neg a(v_i)$ for every $i \in [n]$, and let

$$a(\varphi) = \big(a(L_1^1) \vee a(L_2^1) \vee a(L_3^1)\big) \wedge \cdots \wedge \big(a(L_1^m) \vee a(L_2^m) \vee a(L_3^m)\big)\,.$$

A formula φ is called *satisfiable* if there is a truth assignment for φ such that $a(\varphi) = 1$. The satisfiability problem of propositional formulas in 3-conjunctive normal form is specified as follows.

Problem: **3-cnf Formula Satisfiability**
Instance: A 3-cnf formula φ
Question: Is φ satisfiable?

The following theorem is one of the foundational theorems of complexity theory.[12]

Theorem 1.18 (Cook [34, Thm. 1 & 2])**.** *The satisfiability problem of propositional formulas in 3-conjunctive normal form is* NP*-complete.*

[11] The machine might, for $i = 1, 2, \ldots$, search the formula for the i-th smallest index, and rename this index to i.

[12] Actually, Cook proved NP-completeness of the tautology problem of formulas in (3-)disjunctive normal form. However, it is easy to see that this is equivalent to the stated theorem.

1.3 Formal Tree Languages

We will now continue our exposition by recalling some of the theory of formal tree languages. In particular, we will call to mind the definitions of trees, tree languages, and recognizable tree languages. Moreover, we will recollect some helpful notation which was introduced in the context of magmoid theory. For more complete introductions to the topic of formal tree languages, refer to [71, 72, 52].

1.3.1 Trees and Tree Languages

Trees

An alphabet Σ equipped with a function $\mathrm{rk}_\Sigma \colon \Sigma \to \mathbb{N}$ is called a *ranked alphabet*. The rank of a symbol will determine the number of subtrees of an occurrence of the symbol in a tree. Given a ranked alphabet Σ, we will write rk instead of rk_Σ when Σ is obvious. Let $k \in \mathbb{N}$. A symbol $\sigma \in \Sigma$ with $\mathrm{rk}(\sigma) = k$ is sometimes said to be *k-ary*. We let $\Sigma^{(k)} = \mathrm{rk}^{-1}(k)$, the set of k-ary symbols from Σ. Often, we will use the notation $\sigma^{(k)}$ and mean by it that σ is k-ary. The maximal rank of a symbol in Σ is denoted by $\max\mathrm{rk}(\Sigma)$. We will call a ranked alphabet Σ *monadic* if $\Sigma = \Sigma^{(0)} \cup \Sigma^{(1)}$.

Trees will be represented in term notation. That is, a tree is just a particular well-parenthesized word, defined as follows. Let U be a set and let C be the set which consists solely of the three distinct symbols '(', ')', and ','. The set $\mathrm{T}_\Sigma(U)$ of *trees (over Σ indexed by U)* is the smallest set $T \subseteq (\Sigma \cup U \cup C)^*$ such that $U \subseteq T$, and for every $k \in \mathbb{N}$ and $\sigma \in \Sigma^{(k)}$,

$$\text{if} \qquad t_1, \dots, t_k \in T\,, \qquad \text{then} \qquad \sigma(t_1, \dots, t_k) \in T\,.$$

Example 1.19. Let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ be a ranked alphabet and $U = \{x\}$. Then

$$\sigma\Big(\sigma\big(\gamma\big(\alpha()\big), x\big), \gamma(x)\Big)$$

is a tree from $\mathrm{T}_\Sigma(U)$. We can depict this tree as the graph

$$\begin{array}{ccccc} & & \sigma & & \\ & / & & \backslash & \\ & \sigma & & & \gamma \\ / & & \backslash & & | \\ \gamma & & x & & x \\ | & & & & \\ \alpha & & & & \end{array}\;. \qquad \lhd$$

In the following, we will switch between term and graph notation of trees without further ado. By the definition from above, it is clear that the trees considered in this thesis are finite, rooted, and ordered trees in the graph-theoretic sense.

Remark 1.20. It is well-known that $\mathrm{T}_\Sigma(U)$ is the *free Σ-algebra* generated by U. In the nomenclature from Section 1.1.2, a Σ-algebra is an algebra of type $(k_1, \dots, k_n)$, if Σ is a ranked alphabet whose elements are $\sigma_1, \dots, \sigma_n$, when listed in some arbitrary but fixed order, and if $\mathrm{rk}(\sigma_i) = k_i$ for every $i \in [n]$.

The algebra $\mathrm{T}_\Sigma(U)$ is *free* since every function $h\colon U \to A$, where A is an algebra of the same type, admits a unique homomorphic extension $\tilde{h}\colon \mathrm{T}_\Sigma(U) \to A$; cf. [164, Thm. 4 in Sec 1.2.3]. Again, h and $\tilde{h}$ are often identified. $\triangleleft$

The following abbreviations are quite helpful. A tree $\alpha()$, where $\alpha \in \Sigma^{(0)}$, is abbreviated by α, a tree $\gamma(t)$, where $\gamma \in \Sigma^{(1)}$, by γt, and the set $\mathrm{T}_\Sigma(\emptyset)$ by T_Σ. The notation γt suggests a bijection between $\Sigma^* U$ and $\mathrm{T}_\Sigma(U)$ for ranked alphabets Σ with $\Sigma = \Sigma^{(1)}$, and in fact we will sometimes identify such monadic trees with words. As a concrete example, if γ and δ are symbols from $\Sigma^{(1)}$, and α is an element of U, then we will often write $\gamma\delta(\alpha)$ or $\gamma\delta\alpha$ instead of $\gamma(\delta(\alpha))$, and take this tree to be an element of $(\Sigma^{(1)})^* U$.

Let Γ be a ranked alphabet such that $\Gamma = \Gamma^{(k)}$ for some $k \in \mathbb{N}$, and let $T_1, \ldots, T_k \subseteq \mathrm{T}_\Sigma(U)$. Then $\Gamma(T_1, \ldots, T_k)$ denotes the set

$$\left\{\gamma(t_1, \ldots, t_k) \,\middle|\, \gamma \in \Gamma,\ t_1 \in T_1,\ \ldots,\ t_k \in T_k\right\}.$$

Convention. *In the following section, let Σ denote an arbitrary ranked alphabet, and U an arbitrary set, unless specified otherwise.*

We will now list some definitions for trees. Consider, when not defined otherwise, some arbitrary $t \in \mathrm{T}_\Sigma(U)$.

Height and positions. We define the *height* $\mathrm{ht}(t) \in \mathbb{N}$ of t and its set of *positions* $\mathrm{pos}(t) \subseteq \mathbb{N}_1^*$ as follows by induction. For every $u \in U$, let

$$\mathrm{ht}(u) = 0\,, \qquad \mathrm{pos}(u) = \{\varepsilon\}\,,$$

and if $t = \sigma(t_1, \ldots, t_k)$ for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma(U)$, let

$$\mathrm{ht}(t) = 1 + \max_{i \in [k]} \mathrm{ht}(t_i)\,, \qquad \mathrm{pos}(t) = \{\varepsilon\} \cup \bigcup_{i \in [k]} i \cdot \mathrm{pos}(t_i)\,.$$

Note that the latter definitions subsume the case $t = \alpha \in \Sigma^{(0)}$. In this case, we assume that $\max_{i \in [0]} \mathrm{ht}(t_i) = \max \emptyset = 0$.

Sometimes, we will also refer to an element of $\mathrm{pos}(t)$ as a *node* of t. Let $v, w \in \mathrm{pos}(t)$. If $w = vi$ for some $i \in \mathbb{N}$, then we call v the *parent* of w, and w a *child* of v. Hence, every child has at most one parent. Moreover, if $v \sqsubseteq w$ (resp. $v \sqsubset w$), then we call v an *ancestor* (resp. *proper ancestor*) of w, and w a *descendant* (resp. *proper descendant*) of v. We also say that an ancestor *dominates* its descendant. The node ε is called the *root* of t, and a node with no children is called a *leaf (node)* of t.

Labels and subtrees. Let $w \in \mathrm{pos}(t)$. Then we define the *label of t at w*, denoted by $t(w)$, and the *subtree of t at w*, denoted by $t|_w$, as follows by induction. For every $u \in U$, observe that $\mathrm{pos}(u) = \{\varepsilon\}$, and let

$$u(\varepsilon) = u\,, \qquad u|_\varepsilon = u\,.$$

Let $t = \sigma(t_1, \ldots, t_k)$ for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $t_1, \ldots, t_k \in \mathrm{T}_\Sigma(U)$. Define, for every $i \in [k]$ and $w \in \mathrm{pos}(t_i)$,

$$t(\varepsilon) = \sigma\,, \qquad t|_\varepsilon = t\,,$$
$$t(iw) = t_i(w)\,, \qquad t|_{iw} = t_i|_w\,.$$

Let s, $t \in \mathrm{T}_\Sigma(U)$. Then s is called a *subtree* of t if there is some $w \in \mathrm{pos}(t)$ such that $s = t|_w$.

Symbol occurrences and size. For every subset $A \subseteq \Sigma \cup U$, we let

$$\mathrm{pos}_A(t) = \{w \in \mathrm{pos}(t) \mid t(w) \in A\} \qquad \text{and} \qquad |t|_A = |\mathrm{pos}_A(t)|\,.$$

When $A = \{a\}$ for some $a \in \Sigma \cup U$, we will write $\mathrm{pos}_a(t)$, resp. $|t|_a$, instead. If moreover there is precisely one element w in $\mathrm{pos}_A(t)$, then we will write $\mathrm{pos}_A(t) = w$. The *size* of t is defined to be $|t| = |t|_\Sigma$.

Paths and perfect trees. A sequence of nodes $w_1, \ldots, w_n$ of t, where $n \in \mathbb{N}$, is called a *path (from w_1 to w_n)* if for each $i \in [n-1]$, w_{i+1} is a child of w_i. A tree $t \in \mathrm{T}_\Sigma$ is called *perfect* if the paths from the root of t to every leaf of t are all of equal length.

Yield. Let $A \subseteq \Sigma^{(0)} \cup U$. The *A-yield* of a tree $t \in \mathrm{T}_\Sigma(U)$, denoted by $\mathrm{yd}_A(t) \in A^*$, is defined as follows by induction. For every $a \in \Sigma^{(0)} \cup U$, let

$$\mathrm{yd}_A(a) = \begin{cases} a & \text{if } a \in A, \\ \varepsilon & \text{otherwise.} \end{cases}$$

Let $t = \sigma(t_1, \ldots, t_k)$ for some $k \in \mathbb{N}_1$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma(U)$. Then

$$\mathrm{yd}_A(t) = \mathrm{yd}_A(t_1) \cdots \mathrm{yd}_A(t_k)\,.$$

For each $a \in \Sigma^{(0)} \cup U$, we will abbreviate $\mathrm{yd}_{\{a\}}(t)$ by $\mathrm{yd}_a(t)$, and moreover $\mathrm{yd}_{\Sigma^{(0)}}(t)$ by $\mathrm{yd}(t)$.

Replacement. Given s, $t \in \mathrm{T}_\Sigma(U)$ and $w \in \mathrm{pos}(s)$, let $s[t]_w$ denote the (unique) tree $s' \in \mathrm{T}_\Sigma(U)$ such that

$$\mathrm{pos}(s') = \{v \in \mathrm{pos}(s) \mid w \not\sqsubseteq v\} \cup w \cdot \mathrm{pos}(t)\,,$$

and for every $v \in \mathrm{pos}(s')$,

$$s'(v) = \begin{cases} s(v) & \text{if } w \not\sqsubseteq v \\ t(u) & \text{if } v = wu \text{ for some } u \in \mathbb{N}_1^*. \end{cases}$$

Intuitively, we replace the subtree of s at position w by t.

Substitution. Let $u_1, \ldots, u_n \in U$ be pairwise distinct; moreover, let $s_1, \ldots, s_n \in \mathrm{T}_\Sigma(U)$. We define the *substitution of s_i for u_i* $(i \in [n])$, denoted by $t[u_1/s_1, \ldots, u_n/s_n]$, as follows. For $u \in U$, let

$$u[u_1/s_1, \ldots, u_n/s_n] = \begin{cases} s_i & \text{if } u = u_i \text{ for some } i \in [n], \\ u & \text{otherwise.} \end{cases}$$

Let $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma(U)$; then

$$t[u_1/s_1, \ldots, u_n/s_n] = \sigma\big(t_1[u_1/s_1, \ldots, u_n/s_n], \ldots, t_k[u_1/s_1, \ldots, u_n/s_n]\big)\,.$$

Mostly, we will deal with trees indexed by *variables*. Formally, define the sets of variables

$$X = \{x_i \mid i \in \mathbb{N}\} \qquad \text{and} \qquad X_k = \{x_i \in X \mid i \in [k]\} \quad \text{for every } k \in \mathbb{N}\,.$$

Observe that $X_0 = \emptyset$. Sometimes, in particular when it is the only variable that is considered, we will abbreviate x_1 by x. For every $t \in \mathrm{T}_\Sigma(X_n)$ and $t_1, \ldots, t_n \in \mathrm{T}_\Sigma(U)$, $n \in \mathbb{N}$, we will abbreviate the expression

$$t[x_1/t_1, \ldots, x_n/t_n] \qquad \text{by} \qquad t[t_1, \ldots, t_n].$$

Convention. *For each $k \in \mathbb{N}$ and $\sigma \in \Sigma^{(k)}$, we will identify the tree $\sigma(x_1, \ldots, x_k) \in \mathrm{T}_\Sigma(X_k)$ with the symbol σ. Observe that this generalizes the convention of identifying the tree $\alpha()$ and $\alpha \in \Sigma^{(0)}$, which we mentioned above.*

Magmoids

This subsection introduces notation associated with the concept of *magmoids*. Generally spoken, magmoids are algebraic structures with two partial binary operations satisfying certain axioms. The algebraic properties of magmoids have been researched in [12, 15, 16]. Moreover, various magmoids have been used to formalize equational and recognizable classes of tree, graph, and pattern languages, cf. i.a. [9, 23, 25]. We will only concern ourselves with one particular magmoid, namely the *free projective magmoid* $\mathrm{T}(\Sigma)$ *generated by a ranked alphabet* Σ. Its elements are tuples of trees from $\mathrm{T}_\Sigma(X)$. The use of this magmoid allows us to express many properties more concisely and lucidly than with the standard notation introduced above. Compare also Example 1.27 below for an illustration of the following concepts.

Formally, let k, $n \in \mathbb{N}$. Then the set $\mathrm{T}(\Sigma)^n_k$ is given by

$$\mathrm{T}(\Sigma)^n_k = \left\{(k, t_1, \ldots, t_n) \;\middle|\; t_1, \ldots, t_n \in \mathrm{T}_\Sigma(X_k)\right\}.$$

We will write $\langle k; t_1, \ldots, t_n\rangle$ instead of $(k, t_1, \ldots, t_n)$. Let

$$\mathrm{T}(\Sigma) = \bigcup_{n,k \in \mathbb{N}} \mathrm{T}(\Sigma)^n_k.$$

Moreover, let $\mathrm{T}(\Sigma)^n = \bigcup_{k \in \mathbb{N}} \mathrm{T}(\Sigma)^n_k$ for every $n \in \mathbb{N}$ and $\mathrm{T}(\Sigma)_k = \bigcup_{n \in \mathbb{N}} \mathrm{T}(\Sigma)^n_k$ for every $k \in \mathbb{N}$. Observe that due to the definition of $\mathrm{T}(\Sigma)^n_k$, the set $\mathrm{T}(\Sigma)$ is partitioned into the respective sets $\mathrm{T}(\Sigma)^n_k$. Given some $u \in \mathrm{T}(\Sigma)$, we denote the unique numbers n and k such that $u \in \mathrm{T}(\Sigma)^n_k$ by $\mathrm{rk\,sup}(u)$ and $\mathrm{rk\,inf}(u)$, respectively.

Remark 1.21. In the following, we will identify the sets $\mathrm{T}_\Sigma(X_k)$ and $\mathrm{T}(\Sigma)^1_k$ for every $k \in \mathbb{N}$, as well as $\mathrm{T}_\Sigma(X)$ and $\mathrm{T}(\Sigma)^1$, and write t instead of $\langle k; t\rangle$. Although $\mathrm{T}_\Sigma(X_k)$ and $\mathrm{T}(\Sigma)^1_k$ are identified, we will use both notations, and decide from the context which alternative is appropriate. ◁

Vertical concatenation. The first operation we define on $\mathrm{T}(\Sigma)$ is the generalization of tree substitution to tuples. It can also be understood as vertical concatenation. Formally, let n, ℓ, $k \in \mathbb{N}$, and let

$$u = \langle \ell; u_1, \ldots, u_n\rangle \in \mathrm{T}(\Sigma)^n_\ell \qquad \text{and} \qquad v = \langle k; v_1, \ldots, v_\ell\rangle \in \mathrm{T}(\Sigma)^\ell_k.$$

Then we define the element $u \cdot v$ of $\mathrm{T}(\Sigma)^n_k$ by

$$u \cdot v = \big\langle k; u_1[v_1, \dots, v_\ell], \dots, u_n[v_1, \dots, v_\ell]\big\rangle .$$

Note that the operation $\cdot$ is associative [75, Prop. 2.4]. We denote by Id_n the tuple

$$\langle n; x_1, \dots, x_n\rangle \in \mathrm{T}(\Sigma)^n_n .$$

In particular, we have $\mathrm{Id}_0 = \langle 0; \varepsilon\rangle$. Vertical concatenation can be iterated as follows: for every $n \in \mathbb{N}$ and $u \in \mathrm{T}(\Sigma)^n_n$, we let

$$u^0 = \mathrm{Id}_n \qquad \text{and} \qquad u^{j+1} = u \cdot u^j \quad \text{for every } j \in \mathbb{N} .$$

Remark 1.22. The notation $\cdot$ should not be confused with the one for concatenation of words, but it will be clear from the context which operation we mean. Moreover, observe that when restricted to $\mathrm{T}(\Sigma)^1_1$, for a ranked alphabet Σ with $\Sigma = \Sigma^{(1)}$, substitution behaves exactly like word concatenation, so the confounding of notation is justified. $\lhd$

Horizontal concatenation. The second operation on $\mathrm{T}(\Sigma)$, $\otimes$, also called *tensor product*, can be understood as concatenation of tuples, or horizontal concatenation. For every n_1, n_2, k_1, $k_2 \in \mathbb{N}$, and

$$u = \langle k_1; u_1, \dots, u_{n_1}\rangle \in \mathrm{T}(\Sigma)^{n_1}_{k_1} , \qquad v = \langle k_2; v_1, \dots, v_{n_2}\rangle \in \mathrm{T}(\Sigma)^{n_2}_{k_2} ,$$

we define the element $u \otimes v$ of $\mathrm{T}(\Sigma)^{n_1+n_2}_{k_1+k_2}$ by

$$u \otimes v = \big\langle k_1 + k_2; u_1, \dots, u_{n_1}, v'_1, \dots, v'_{n_2}\big\rangle ,$$

where $v'_i = v_i[x_1/x_{k_1+1}, \dots, x_{k_2}/x_{k_1+k_2}]$ for every $i \in [n_2]$. Intuitively, we append v to u and rename the variables in v distinctly. It is not hard to show that $\otimes$ is associative.

Convention. *Let, for the rest of this section, n, $k \in \mathbb{N}$ be arbitrary numbers.*

We recall the following properties of $\mathrm{T}(\Sigma)$. Property (1) is illustrated in Figure 1.3.

Lemma 1.23 (Arnold and Dauchet [15, Prop. 2 & 4])**.**

1. *For every u_1, u_2, v_1, $v_2 \in \mathrm{T}(\Sigma)$, we have*

$$(u_1 \cdot u_2) \otimes (v_1 \cdot v_2) = (u_1 \otimes v_1) \cdot (u_2 \otimes v_2),$$

whenever both sides of the equation are defined.

2. *For every n, $k \in \mathbb{N}$, and $u \in \mathrm{T}(\Sigma)^n_k$, we have* $\mathrm{Id}_n \cdot u = u \cdot \mathrm{Id}_k = u$ *and* $\mathrm{Id}_0 \otimes u = u \otimes \mathrm{Id}_0 = u$.

3. *For every m, $n \in \mathbb{N}$, we have* $\mathrm{Id}_m \otimes \mathrm{Id}_n = \mathrm{Id}_{m+n}$.

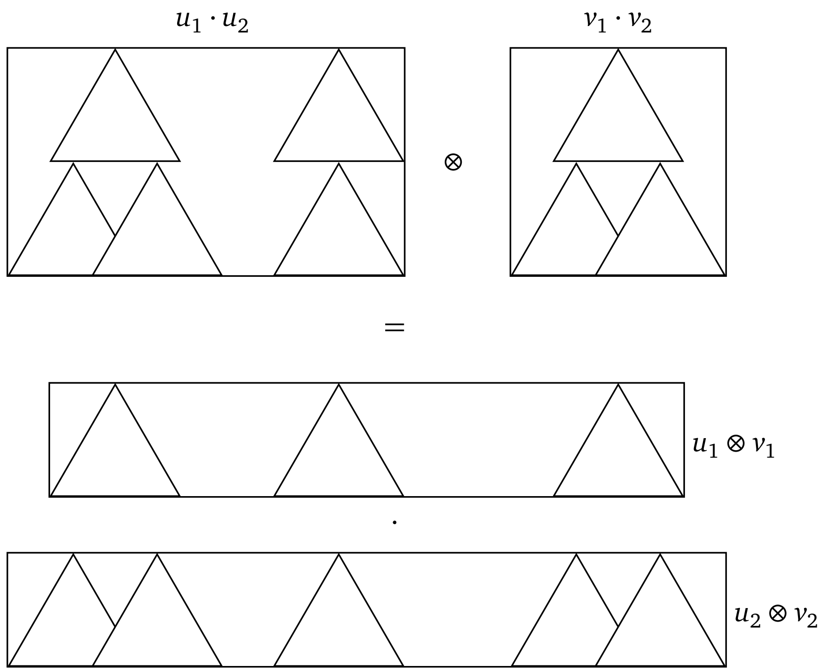


Figure 1.3: Illustration of Lemma 1.23(1)

Torsions. Torsions are particular tuples in $\mathrm{T}(\Sigma)$, which capture the tree-language-theoretic phenomena of copying and deletion. Formally, the set Θ_k^n of *torsions* is

$$\Theta_k^n = \left\{ \langle k; x_{i_1}, \ldots, x_{i_n} \rangle \;\middle|\; i_1, \ldots, i_n \in [k] \right\}.$$

Note that $\Theta_k^n \subseteq \mathrm{T}(\Sigma)_k^n$. We let

$$\Theta_k = \bigcup_{n \in \mathbb{N}} \Theta_k^n, \qquad \Theta^n = \bigcup_{k \in \mathbb{N}} \Theta_k^n, \qquad \text{and} \qquad \Theta = \bigcup_{n, k \in \mathbb{N}} \Theta_k^n$$

for every n, $k \in \mathbb{N}$. A torsion $\vartheta \in \Theta_k^n$, say $\vartheta = \langle k; x_{i_1}, \ldots, x_{i_n} \rangle$, can also be understood as a function $\vartheta \colon [n] \to [k]$ such that

$$\vartheta(\ell) = i_\ell$$

for every $\ell \in [n]$. In fact, we will use these two views of torsions even-handedly without mention.

Clearly, $\mathrm{Id}_n \in \Theta_n^n$. Moreover, we will denote the torsion $\langle n; x_i \rangle \in \Theta_n^1$ by π_i^n, for every $i \in [n]$, and when n is clear from the context, we will write π_i instead. For every $u \in \mathrm{T}(\Sigma)_k^n$, the i-th tree in the tuple u is then $\pi_i \cdot u$.

The following lemma shows the action of a torsion on a tuple of trees. Its proof is trivial, and therefore omitted.

Lemma 1.24. *For every n, ℓ, $k \in \mathbb{N}$, every torsion $\vartheta \in \Theta_\ell^n$, and every tuple $u \in \mathrm{T}(\Sigma)_k^\ell$, we have*

$$\vartheta \cdot u = \langle k; \pi_{\vartheta(1)} \cdot u, \ldots, \pi_{\vartheta(n)} \cdot u \rangle.$$

Torsion-free tuples. Next, we define a particular subset of $\mathrm{T}(\Sigma)$, the set $\widetilde{\mathrm{T}}(\Sigma)$ of torsion-free tuples. For this purpose, we require the following auxiliary definition. Let $A \subseteq \Sigma^{(0)} \cup X$. We extend yd_A from Section 1.3.1 to a function of type $\mathrm{T}(\Sigma) \to A^*$, by setting

$$\mathrm{yd}_A\big(\langle k; t_1, \dots, t_n\rangle\big) = \mathrm{yd}_A(t_1)\cdots \mathrm{yd}_A(t_n)$$

for every $\langle k; t_1, \dots, t_n\rangle \in \mathrm{T}(\Sigma)^n_k$. Then we let

$$\widetilde{\mathrm{T}}(\Sigma)^n_k = \big\{u \in \mathrm{T}(\Sigma)^n_k \;\big|\; \mathrm{yd}_X(u) = x_1 x_2 \cdots x_k\big\}\,.$$

Each tuple $u \in \widetilde{\mathrm{T}}(\Sigma)^n_k$ is said to be *torsion-free*. Clearly, we can decompose every tuple into the product of a torsion-free tuple with some torsion, as the following lemma shows.

Lemma 1.25 (Arnold and Dauchet [15, Prop. 5]). *For every $u \in \mathrm{T}(\Sigma)^n_k$, there are some $m \in \mathbb{N}$, a torsion-free tuple $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)^n_m$, and a torsion $\vartheta \in \Theta^m_k$ such that*

$$u = \tilde{u} \cdot \vartheta\,.$$

In fact, m, $\tilde{u}$, and ϑ are determined uniquely by these conditions.

The proof idea is simply to relabel the variables of u from left to right into $x_1, \dots, x_m$, where m is the number of variable occurrences in u, and to store their original values in ϑ. In the following, we will denote the respective unique decomposition $(\tilde{u}, \vartheta)$ of u by $\mathrm{lin}(u)$.

For every $n, k \in \mathbb{N}$, let

$$\widetilde{\mathrm{T}}(\Sigma)_k = \bigcup_{n\in\mathbb{N}} \widetilde{\mathrm{T}}(\Sigma)^n_k\,, \qquad \widetilde{\mathrm{T}}(\Sigma)^n = \bigcup_{k\in\mathbb{N}} \widetilde{\mathrm{T}}(\Sigma)^n_k\,, \qquad \text{and} \qquad \widetilde{\mathrm{T}}(\Sigma) = \bigcup_{n,k\in\mathbb{N}} \widetilde{\mathrm{T}}(\Sigma)^n_k\,.$$

The set $\widetilde{\mathrm{T}}(\Sigma)$ of torsion-free tuples forms a submagmoid[13] of $\mathrm{T}(\Sigma)$ [15, Sec. 3.2]. In particular, it is closed under the operations $\cdot$ and $\otimes$. The magmoid $\widetilde{\mathrm{T}}(\Sigma)$ is interesting because it is *decomposable,* as described in the following lemma. Intuitively, every element of a decomposable magmoid can be uniquely expressed as the tensor product of some elements $u_1, \dots, u_n$, such that $\mathrm{rk}\,\mathrm{sup}(u_i) = 1$ for every $i \in [n]$.

Lemma 1.26 (Arnold and Dauchet [12, Lem. 1.18(a)]). *For every $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$, there are unique $n \in \mathbb{N}$ and $\tilde{u}_1, \dots, \tilde{u}_n \in \widetilde{\mathrm{T}}(\Sigma)^1$ such that*

$$\tilde{u} = \tilde{u}_1 \otimes \cdots \otimes \tilde{u}_n\,.$$

It is easy to show that $\widetilde{\mathrm{T}}(\Sigma)$ is decomposable. On the other hand, $\mathrm{T}(\Sigma)$ is not decomposable. For instance, the tuple $\langle 3; \gamma(x_1), \sigma(x_3, \alpha)\rangle$ has two distinct decompositions, namely $\langle 1; \gamma(x_1)\rangle \otimes \langle 2; \sigma(x_2, \alpha)\rangle$ and $\langle 2; \gamma(x_1)\rangle \otimes \langle 1; \sigma(x_1, \alpha)\rangle$, while the tuple $\langle 2; x_2, x_1\rangle$ cannot be decomposed at all by means of $\otimes$.

[13] Where the notion of submagmoid is defined just as expected.

Nonrenaming horizontal concatenation. Let $u \in \mathrm{T}(\Sigma)^{n_1}_{k_1}$ and $v \in \mathrm{T}(\Sigma)^{n_2}_{k_2}$ for some n_1, n_2, k_1, and $k_2 \in \mathbb{N}$. Then we define the tuple $[u, v] \in \mathrm{T}(\Sigma)$ by setting

$$[u, v] = (u \otimes v) \cdot \vartheta\,, \qquad \text{where } \vartheta = \langle k'; x_1, \ldots, x_{k_1}, x_1, \ldots, x_{k_2}\rangle \text{ and } k' = \max\{k_1, k_2\}\,.$$

So the operator $[\cdot,\cdot]$ behaves like $\otimes$, but without renaming of variables. Clearly, for u, v and k' as given above, we have $[u, v] \in \mathrm{T}(\Sigma)^{n_1+n_2}_{k'}$. As it is obviously associative, we may generalize this operator to a larger number of arguments by setting

$$[u_1, \ldots, u_m] = [u_1, [u_2, \ldots, [u_{m-1}, u_m] \ldots]]$$

for every $m \geq 2$ and $u_1, \ldots, u_m \in \mathrm{T}(\Sigma)$.

Example 1.27. Let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$. By our notational convention, we have that

$$\sigma \in \widetilde{\mathrm{T}}(\Sigma)^1_2\,, \qquad \gamma \in \widetilde{\mathrm{T}}(\Sigma)^1_1\,, \qquad \text{and} \qquad \alpha \in \widetilde{\mathrm{T}}(\Sigma)^1_0\,.$$

Let us set $u = \gamma \otimes \gamma$. Then u is an element of $\mathrm{T}(\Sigma)^2_2$, of the form

$$u = \big\langle 2; \gamma(x_1), \gamma(x_2)\big\rangle\,.$$

Moreover, for every $j \in \mathbb{N}$, the expression

$$\sigma \cdot u^j \cdot [\alpha, \alpha]$$

results in the tree

$$\begin{array}{c}
\sigma \\
\diagup \quad \diagdown \\
j\left\{\begin{array}{cc} \gamma & \gamma \\ \vdots & \vdots \\ \gamma & \gamma \end{array}\right\} j\,. \\
\mid \qquad \mid \\
\alpha \qquad \alpha
\end{array}$$

On the other hand, we obtain the same result by considering the expression

$$\sigma \cdot \vartheta \cdot \gamma^j \cdot \alpha\,,$$

where ϑ is the torsion $\langle 1; x_1, x_1\rangle$. $\lhd$

Linearity and nondeletion. A tuple $u \in \mathrm{T}(\Sigma)^n_k$ with $\mathrm{lin}(u) = (\tilde{u}, \vartheta)$ is called *linear* if ϑ, understood as a function, is injective, and u is *nondeleting* if ϑ is surjective. Moreover, if ϑ is a monotonic function,[14] then we call u *ordered*. Clearly, u is torsion-free if and only if u is linear, nondeleting, and ordered. Note that when restricted to trees, the properties of linearity, nondeletion, and orderedness are equivalent to the classic definitions from tree language theory.

[14] I.e., $i \leq j$ implies $\vartheta(i) \leq \vartheta(j)$ for each $i, j \in [n]$.

Quotient. Let n, m, and $k \in \mathbb{N}$, and consider tuples $u \in \mathrm{T}(\Sigma)^n_k$ and $v \in \mathrm{T}(\Sigma)^n_m$ such that u is linear and nondeleting. Then there is at most one tuple $s \in \mathrm{T}(\Sigma)^k_m$ such that $u \cdot s = v$. If such a tuple s does indeed exist, we will denote s by $u \backslash v$, and call it the *quotient of u with v*.

Positions. Sometimes, it will be necessary to refer uniquely to a node of a tree contained in some tuple. Therefore, we extend Gorn adresses to tuples as follows. Define, for every $u \in \mathrm{T}(\Sigma)^n_k$, the set $\mathrm{pos}(u) \subseteq [n] \times \mathbb{N}_1^*$ by

$$\mathrm{pos}(u) = \{(i, w) \mid i \in [n], w \in \mathrm{pos}(\pi_i \cdot u)\}.$$

To save some parentheses, we will denote each element $(i, w) \in \mathrm{pos}(u)$ by $i.w$. Moreover, the set of all tuple positions $\mathbb{N}_1 \times \mathbb{N}_1^*$ will be denoted by $\mathbb{P}$. We define a right action of $\mathbb{N}_1^*$ on $\mathbb{P}$ in the following manner: for every $i.w \in \mathbb{P}$ and $v \in \mathbb{N}_1^*$, we let $(i.w) \cdot v = i.(w \cdot v)$. As usual, the operator $\cdot$ will often be omitted. It is extended to sets of positions by setting

$$P \cdot W = \{p \cdot w \mid p \in P, w \in W\}$$

for all sets $P \subseteq \mathbb{P}$ and $W \subseteq \mathbb{N}_1^*$. Also in this context, we will abbreviate singleton sets of positions by their sole element. If $u = \langle k; t\rangle$ for some $t \in \mathrm{T}_\Sigma(X_k)$, and there is no risk of confusion, we will identify the positions $1.w \in \mathrm{pos}(u)$ and $w \in \mathrm{pos}(t)$.

Let $u \in \mathrm{T}(\Sigma)^n_k$ for some n, $k \in \mathbb{N}$ and let $i.w$, $j.v \in \mathrm{pos}(u)$. Then

$$u(i.w) = (\pi_i \cdot u)(w), \qquad \text{and} \qquad i.w \sqsubseteq j.v \quad \text{iff} \quad i = j \text{ and } w \sqsubseteq v.$$

Note that the latter definition is equivalent to demanding that for every v, $w \in \mathbb{P}$, we have $v \sqsubseteq w$ if and only if there is some $z \in \mathbb{N}_1^*$ such that $w = vz$. Analogously to the definition for tree positions, we say that v is a *prefix* of w if $v \sqsubseteq w$, and we write $v \parallel w$ if neither $v \sqsubseteq w$ nor $w \sqsubseteq v$, for every tuple position v and $w \in \mathbb{P}$.

Convention. *In this work, we will have no occasion to denote an undefined composition of tuples. So whenever we write $u \cdot v$ for some u, $v \in \mathrm{T}(\Sigma)$, we assume implicitly that there are n, ℓ, and $k \in \mathbb{N}$ such that $u \in \mathrm{T}(\Sigma)^n_\ell$ and $v \in \mathrm{T}(\Sigma)^\ell_k$. This convention allows us to save on a great number of quantifications, thus improving legibility.*

Tree Languages

For a ranked alphabet Σ, a *(formal) tree language (over Σ)* is a subset of T_Σ. There are instances where we want to allow variables, so we will also refer to subsets of $\mathrm{T}_\Sigma(X)$ as tree languages.

A *tree-generator* is a mathematical object G to which a tree language $\mathcal{L}(G)$ is associated. We will consider many tree-generators in this work, such as finite-state tree automata, context-free tree grammars, pushdown tree automata, and so on. Two tree-generators G_1 and G_2 are called *equivalent* if $\mathcal{L}(G_1) = \mathcal{L}(G_2)$.

In the following, we define some operations on tree languages. Let $\alpha \in \Sigma^{(0)}$, $t \in \mathrm{T}_\Sigma$, and $L \subseteq \mathrm{T}_\Sigma$. We define the tree language $t \cdot_\alpha L$ over Σ as follows. If $t = \sigma(t_1, \ldots, t_k)$ for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots t_k \in \mathrm{T}_\Sigma$, then let

$$t \cdot_\alpha L = \begin{cases} L & \text{if } \sigma = \alpha, \\ \{\sigma(s_1, \ldots, s_k) \mid s_1 \in t_1 \cdot_\alpha L, \ldots, s_k \in t_k \cdot_\alpha L\} & \text{otherwise.} \end{cases}$$

Now let $L_1, L_2 \subseteq \mathrm{T}_\Sigma$. The *α-concatenation* of L_1 and L_2 is the tree language

$$L_1 \cdot_\alpha L_2 = \bigcup_{t \in L_1} t \cdot_\alpha L_2 \,.$$

Moreover, the *α-iteration* (or *α-star*) of $L \subseteq \mathrm{T}_\Sigma$ is the tree language over Σ defined by

$$L_\alpha^* = \bigcup_{n\in\mathbb{N}} L_\alpha^n \qquad \text{with} \qquad L_\alpha^0 = \{\alpha\} \quad \text{and} \quad L_\alpha^{i+1} = L_\alpha^i \cdot_\alpha \big(L \cup \{\alpha\}\big) \quad \text{for } i \in \mathbb{N}\,.$$

Remark 1.28. It is also possible to let variables $x_1, x_2, \ldots,$ serve as the points where trees can be substituted, instead of fixing the nullary symbol α for this purpose. In this way, one arrives at the concept of *OI-substitution* [55], which can also be expressed very nicely using magmoids of tuples of tree languages with variables; cf. [16, Ch. IV]. We stuck with using α to keep the notions of recognizable and context-free tree languages as simple as possible – otherwise we would have to define them as tree languages with variables. ◁

Path Languages

In the following, we will define the *path language* $\mathrm{P}_i^k(t)$ of a tree $t \in \mathrm{T}_\Sigma(X_k)$, for every $k \in \mathbb{N}$ and $i \in [0,k]$. Intuitively, a word $w \in \mathrm{P}_i^k(t)$ describes the sequence of symbol labels on a path from the root of t (inclusively) to one of its leaves – either inclusively to a leaf labeled by a symbol if $i = 0$, or exclusively up to an occurrence of the variable x_i if $i > 0$. In addition to this, w encodes the “directions” to take in t: the symbol $\langle\sigma, j\rangle$ informs us that the path continues with the j-th child of the current occurrence of σ.

Formally, given a ranked alphabet Σ, define the *path alphabet*

$$\widehat{\Sigma} = \big\{\langle\sigma, i\rangle \;\big|\; k \in \mathbb{N}_1,\, \sigma \in \Sigma^{(k)},\, i \in [k]\big\} \cup \big\{\langle\alpha, 0\rangle \;\big|\; \alpha \in \Sigma^{(0)}\big\}\,.$$

Moreover, define the family of functions

$$\big(\mathrm{P}_i^k \colon \mathrm{T}_\Sigma(X_k) \to \mathcal{P}(\widehat{\Sigma}^*) \;\big|\; k \in \mathbb{N},\, i \in [0,k]\big)$$

as follows by induction. Let $k \in \mathbb{N}$ and $i \in [0,k]$. For every $\alpha \in \Sigma^{(0)}$, and every $j \in [k]$, let

$$\mathrm{P}_i^k(\alpha) = \begin{cases} \{\langle\alpha, 0\rangle\} & \text{if } i = 0 \\ \emptyset & \text{otherwise} \end{cases} \qquad \text{and} \qquad \mathrm{P}_i^k(x_j) = \begin{cases} \{\varepsilon\} & \text{if } i = j \\ \emptyset & \text{otherwise.} \end{cases}$$

Further, for every $n \in \mathbb{N}$, $\sigma \in \Sigma^{(n)}$, and $t_1, \ldots, t_n \in \mathrm{T}_\Sigma(X_k)$, let

$$\mathrm{P}_i^k\big(\sigma(t_1, \ldots, t_n)\big) = \bigcup_{j \in [n]} \big\{\langle\sigma, j\rangle\big\} \cdot \mathrm{P}_i^k(t_j)\,.$$

The function P_i^k is naturally extended to tree languages $L \subseteq \mathrm{T}_\Sigma(X_k)$ by setting

$$\mathrm{P}_i^k(L) = \bigcup_{t \in L} \mathrm{P}_i^k(t)\,.$$

For every tree language $L \subseteq \mathrm{T}_\Sigma$, the *path language of* L is $\mathrm{P}(L) = \mathrm{P}_0^0(L)$.

Example 1.29. Let $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$ and consider the tree

$$t = \begin{array}{c} \sigma \\ / \quad \backslash \\ \alpha \qquad \sigma \\ \qquad / \quad \backslash \\ \qquad x_1 \quad \alpha \end{array}$$

from $\mathrm{T}_\Sigma(X_2)$. We have

$$\mathrm{P}_1^2(t) = \{\langle\sigma,2\rangle\langle\sigma,1\rangle\}, \quad \mathrm{P}_2^2(t) = \emptyset, \quad \text{and} \quad \mathrm{P}_0^2(t) = \{\langle\sigma,1\rangle\langle\alpha,0\rangle, \langle\sigma,2\rangle\langle\sigma,2\rangle\langle\alpha,0\rangle\}. \quad \triangleleft$$

1.3.2 Recognizable Tree Languages

Next, we recall the class of recognizable tree languages. For this purpose, we introduce finite-state tree automata, which generalize fsa to the realm of trees.

Tree Automata

A *(bottom-up) finite-state tree automaton (fta)* is a tuple $A = (Q, \Sigma, F, \delta)$ such that

- Q is a finite set (its elements called *states*),
- Σ is a ranked alphabet,
- $F \subseteq Q$ (its elements called *final states*), and
- $\delta = \big(\delta_k \colon Q^k \times \Sigma^{(k)} \to \mathcal{P}(Q) \mid k \in [\max \mathrm{rk}(\Sigma)]\big)$ is a family of functions (called the *transition table*).

Using δ, we define the function $\tilde{\delta} \colon \mathrm{T}_\Sigma \to \mathcal{P}(Q)$ by setting, for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and every $t_1, \dots, t_k \in \mathrm{T}_\Sigma$,

$$\tilde{\delta}\big(\sigma(t_1, \dots, t_k)\big) = \bigcup_{q_1 \in \tilde{\delta}(t_1)} \cdots \bigcup_{q_k \in \tilde{\delta}(t_k)} \delta_k(q_1, \dots, q_k, \sigma).$$

Let $A = (Q, \Sigma, F, \delta)$ be an fta. We associate to A the *tree language recognized by A*,

$$\mathcal{L}(A) = \{t \in \mathrm{T}_\Sigma \mid \tilde{\delta}(t) \cap F \neq \emptyset\}.$$

A tree language is called *recognizable* if it is recognized by some fta A, and the class of all tree languages (over some ranked alphabet Σ) is denoted by RECT (resp. by $\mathsf{RECT}(\Sigma)$).

The fta A is said to be *deterministic (resp. total)* if for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $q_1, \dots, q_k \in Q$, the set $\delta_k(q_1, \dots, q_k, \sigma)$ contains at most (resp. at least) one element. A deterministic and total fta will be abbreviated by dfta. In the case of a dfta, its transition table can be assumed to be a family of functions

$$(\delta_k \colon Q^k \times \Sigma^{(k)} \to Q \mid k \in [\max \mathrm{rk}(\Sigma)]).$$

The following theorem states that, also in the case of tree languages, recognizability is equivalent to recognizability by a deterministic and total tree automaton. It can be shown by generalizing the construction for fsa from Theorem 1.5.

Theorem 1.30 (Thatcher and Wright [161, Thm. 1])**.** *For every $L \in \mathsf{RECT}$, there is a dfta A with $\mathcal{L}(A) = L$.*

The recognizable tree languages are intimately related to the context-free languages by the following *yield theorem.* It is based on the observation that for every cfg G, the set of parse trees of G is recognizable.

Theorem 1.31 (Thatcher [159])**.** *Let $L \subseteq \Sigma^*$ for some alphabet Σ. Then $L \in \mathsf{CF}(\Sigma)$ if and only if there are some ranked alphabet Δ with $\Sigma \subseteq \Delta^{(0)}$, and some $L' \in \mathsf{RECT}(\Delta)$ such that $L = \mathrm{yd}_\Sigma(L')$.*

In this theorem, the elements of $\Delta^{(0)} \setminus \Sigma$ perform the role of representing the empty word in a cfg's parse tree. In contrast to the above theorem, the path languages of recognizable tree languages are recognizable.

Theorem 1.32. *For every $L \in \mathsf{RECT}$, we have $\mathrm{P}(L) \in \mathsf{REC}$.*

The theorem appears to be folklore, but compare [140, p. 277] for an early reference.

Remarks

Finite-state tree automata have been discovered independently by Doner [42, 41] and by Thatcher and Wright [159, 161].

Similar to the word case, the class of recognizable tree languages enjoys very broad closure properties. Among others, RECT is closed under union, intersection, inverse tree homomorphisms and linear tree homomorphisms (see Section 1.3.4 below), α-concatenation and α-iteration; cf. e.g. [161, 49]. As a consequence, RECT is also closed under linear bottom-up and top-down tree transformations [49].

Moreover, the class of recognizable tree languages is characterized by a large number of formalisms, of which we will only list a few. The class RECT coincides with the class of tree languages generated by regular tree grammars [26]. A Kleene-type characterization has been given in [161]. In [41], RECT is characterized logically by means of a Büchi-like theorem. There is also a generalization of the Myhill-Nerode theorem to trees; cf. [105] for an elementary proof, and a historical survey on who the result is to be attributed to.

1.3.3 Trees, Tuples, and Structural Induction

Here, we continue the review of induction principles we have begun in Section 1.1.3 and consider two instances of *structural induction.*

Structural Induction on Trees

Say we are to prove that a property P holds for all trees from $\mathrm{T}_\Sigma(U)$, for some ranked alphabet Σ and some set U. For this purpose, we can apply the principle of *structural induction* on trees. It follows from Noetherian induction as follows. Define the relation

$$R = \left\{\left(t_i, \sigma(t_1, \ldots, t_k)\right) \,\middle|\, k \in \mathbb{N},\, i \in [k],\, \sigma \in \Sigma^{(k)},\, t_1, \ldots, t_k \in \mathrm{T}_\Sigma(U)\right\}.$$

Intuitively, R is the relation “is direct subtree of the root of”. Clearly, this relation is well-founded, as we only consider finite trees.

In the induction base that results from this instantiation of R, we are obliged to prove that P holds for all elements of $\Sigma^{(0)} \cup U$. For the resulting induction step, observe that for each element $\sigma(t_1, \ldots, t_k) \in \mathrm{T}_\Sigma(U)$, the set of all $t \in \mathrm{T}_\Sigma(U)$ with $t \, R \, \sigma(t_1, \ldots, t_k)$ is precisely

$$\{t_1, \ldots, t_k\} .$$

So we are required to show that P holds for $\sigma(t_1, \ldots, t_k)$ under the assumption that it holds for $t_1, \ldots, t_k$.

Let us note that the case $k = 0$ is often treated in the induction step instead of the base case; this choice makes some proofs shorter, and is without doubt logically equivalent.

Structural Induction on Tuples of Trees

There will be some instances in this thesis where we prove a property P for all torsion-free tuples from $\widetilde{\mathrm{T}}(\Sigma)$, where Σ is some ranked alphabet. In these cases, we proceed as follows.

First, recall from Lemma 1.26 that for every $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$, there are unique $n \in \mathbb{N}$ and $\tilde{u}_1, \ldots, \tilde{u}_n \in \widetilde{\mathrm{T}}(\Sigma)^1$ such that

$$\tilde{u} = \tilde{u}_1 \otimes \cdots \otimes \tilde{u}_n .$$

We define the relation

$$R = \left\{(\tilde{u}, \sigma \cdot \tilde{u}) \;\middle|\; \tilde{u} \in \widetilde{\mathrm{T}}(\Sigma),\, \sigma \in \Sigma\right\} \cup \left\{(\tilde{u}_i, \tilde{u}_1 \otimes \cdots \otimes \tilde{u}_n) \;\middle|\; n \in \mathbb{N},\, i \in [n],\, \tilde{u}_1, \ldots, \tilde{u}_n \in \widetilde{\mathrm{T}}(\Sigma)^1\right\} .$$

In prose, R is the union of the relations “is the tuple of the direct subtrees of the root of” and “is a tree which appears in the tuple”. Again, it is not hard to see that R is well-founded, and we can apply Noetherian induction.

Thus, the induction base is concerned with proving P for Id_0 and Id_1. In the induction step, we make a case distinction on $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$. If $\mathrm{rk}\,\mathrm{sup}(\tilde{u}) > 1$, we may assume that P is satisfied by all components of the tuple $\tilde{u}$, and we are to show P holds for $\tilde{u}$ itself. In the other case, let $\mathrm{rk}\,\mathrm{sup}(\tilde{u}) = 1$ and let $\tilde{u}$ contain at least one symbol from Σ. Then $\tilde{u}$ is of the form $\sigma \cdot \tilde{v}$ for some symbol $\sigma \in \Sigma$ and tuple $\tilde{v} \in \widetilde{\mathrm{T}}(\Sigma)$. We assume P holds for $\tilde{v}$, and must prove that it holds also for $\tilde{u}$. Note that all other forms $\tilde{u}$ may assume have already been covered in the induction base.

Remark 1.33. Sometimes, it is also necessary to show that P holds also for all tuples, i.e., for all $u \in \mathrm{T}(\Sigma)$. Observe, for this purpose, that every $u \in \mathrm{T}(\Sigma)$ can be written $u = \tilde{u} \cdot \vartheta$ for some unique torsion-free $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$ and some torsion $\vartheta \in \Theta$. Thus, it will often suffice to show that P holds for every such $\tilde{u}$, using the induction principle from above, and then transfer this property to $\tilde{u} \cdot \vartheta$. $\lhd$

1.3.4 Tree Homomorphisms and Tree Transformations

Next, we recall tree transformations, i.e., mappings between tree languages. We start out with a fundamental kind of tree transformation: the tree homomorphism, which replaces every symbol of a tree with some subtree.

Let Σ and Δ be ranked alphabets. A *tree homomorphism* is a mapping $h\colon \Sigma \to \mathrm{T}_\Delta(X)$ such that, for every $k \in \mathbb{N}$, $h(\Sigma^{(k)}) \subseteq \mathrm{T}_\Delta(X_k)$. We extend h to a function $\widehat{h}\colon \mathrm{T}_\Sigma(X) \to \mathrm{T}_\Delta(X)$ by setting

$$\widehat{h}(x_i) = x_i \quad \text{for } i \in \mathbb{N} \qquad \text{and} \qquad \widehat{h}\big(\sigma(t_1,\dots,t_k)\big) = h(\sigma)\big[\,\widehat{h}(t_1),\dots,\widehat{h}(t_k)\big]$$

for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \dots, t_k \in \mathrm{T}_\Sigma(X)$. As $\widehat{h}$ is determined uniquely by these conditions, we will also refer to $\widehat{h}$ as a tree homomorphism, and identify $\widehat{h}$ with h.

We recall the following properties of tree homomorphisms; cf. [49, 18]. Consider a tree homomorphism $h\colon \mathrm{T}_\Sigma(X) \to \mathrm{T}_\Delta(X)$. We say that h is

- *linear* if $h(\sigma)$ is linear,
- *nondeleting* if $h(\sigma)$ is nondeleting,
- *strict* if $h(\sigma) \notin X$, and
- *alphabetic* if $\mathrm{ht}\big(h(\sigma)\big) \leq 1$,

each for every $\sigma \in \Sigma$. Lastly, h is said to be *elementary*[15] if there are n, $k \in \mathbb{N}$, $\sigma \in \Sigma^{(n)}$, $\delta_1 \in \Delta^{(n-k+1)}$, $\delta_2 \in \Delta^{(k)}$, and $\ell \in [n+1]$ such that

$$h(\sigma) = \delta_1\big(x_1,\dots,x_{\ell-1},\delta_2(x_\ell,\dots,x_{\ell+k-1}),x_{\ell+k},\dots,x_n\big)$$

and $h(\omega) = \omega$ for every $\omega \in \Sigma \setminus \{\sigma\}$.

The following decomposition lemma will be very helpful later. It allows expressing a linear tree homomorphism by a linear alphabetic one, together with a sequence of elementary tree homomorphisms.

Lemma 1.34 (Arnold and Leguy [18, Lem. 10]). *Let $h\colon \mathrm{T}_\Sigma(X) \to \mathrm{T}_\Delta(X)$ be a linear tree homomorphism. There are a linear alphabetic tree homomorphism φ, as well as elementary tree homomorphisms $\psi_1, \dots, \psi_k$ for some $k \in \mathbb{N}$ such that $h = \psi_k \circ \dots \circ \psi_1 \circ \varphi$.*

The proof idea is to encode deletion of variables and non-strictness using φ, and then use the homomorphisms ψ_i to "grow", one by one, the nodes of the image $h(\sigma)$ of each $\sigma \in \Sigma$.

Remark 1.35. Every tree homomorphism is extended uniquely to a homomorphism of magmoids [15, Sec. 4.1] by setting

$$h(u) = h(\tilde{u}) \cdot \vartheta\,,$$

for every $u \in \mathrm{T}(\Sigma)$ with $\mathrm{lin}(u) = (\tilde{u}, \vartheta)$. Recall from Lemma 1.26 that for every $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$, there are unique $n \in \mathbb{N}$ and $\tilde{u}_1, \dots, \tilde{u}_n \in \widetilde{\mathrm{T}}(\Sigma)^1$ such that $\tilde{u} = \tilde{u}_1 \otimes \dots \otimes \tilde{u}_n$. In this situation, we let

$$h(\tilde{u}) = h(\tilde{u}_1) \otimes \dots \otimes h(\tilde{u}_n)\,.$$

It is easy to check that by this definition,

$$h(u \cdot v) = h(u) \cdot h(v) \qquad \text{and} \qquad h(u \otimes v) = h(u) \otimes h(v)$$

for every u, $v \in \mathrm{T}(\Sigma)$. Moreover, $h(\vartheta) = \vartheta$ for every torsion $\vartheta \in \Theta$. ◁

[15] Called *élémentaire ordonné* in [18].

In general, a *tree transformation* is any mapping $\mathrm{T}_\Sigma(X) \to \mathcal{P}(\mathrm{T}_\Delta(X))$. Therefore, tree homomorphisms can be understood as tree transformations (by identifying the image of the homomorphism with a singleton set). Tree transformations are often also specified by means of *tree transducers,* such as, e.g., bottom-up or top-down tree transducers [160, 140, 49]. We will not give formal definitions of these models here. However, in Chapter 5, we will acquaint ourselves which a transducer formalism which generalizes top-down tree transducers.

1.4 Weighted Tree Languages and Weighted Tree Transformations

In Chapter 5, we will consider weighted tree languages and tree transformations. Weighted languages are a time-honored subject of formal language theory – in fact, they have already been considered by Chomsky and Schützenberger [33], who counted for each word the number of its derivations by some context-free grammar, and gave thus a *quantitative* version of their famous common theorem. An earlier article by Schützenberger is already concerned with a class of machines which are essentially weighted automata [147]. For a comprehensive and foundational exposition of the theory of weighted automata, we recommended [143]. Refer to [45] for an extensive survey book on weighted (tree) languages.

In this thesis, however, we only require the following definitions. Let Σ and Δ be ranked alphabets, and K be a semiring. A *weighted tree language* over Σ and K, resp. a *weighted tree transformation* over Σ, Δ, and K, is a mapping of type

$$\mathrm{T}_\Sigma \to K\,, \qquad \text{or} \qquad \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \to K\,,$$

respectively. The *support* of a weighted tree transformation $\tau \colon \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \to K$ is the set

$$\mathrm{supp}(\tau) = \left\{(s,t) \in \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \;\middle|\; \tau(s,t) \neq 0\right\}.$$

The support $\mathrm{supp}(L)$ of a weighted tree language L is defined in the same way. If K is the semiring of Booleans $\mathbb{B} = \{0,1\}$, we will identify a weighted tree language L over $\mathbb{B}$ with the tree language $\mathrm{supp}(L)$, and analogously for weighted tree transformations. In this manner, weighted languages are a generalization of the unweighted setting.

Chapter 2

Context-Free Tree Languages

> *context-free (adj.): of, relating to, or being a grammar or language based on rules that describe a change in a string without reference to elements not in the string*
>
> (Merriam-Webster Dictionary)

Following Chomsky, the word languages generated by formal grammars can be categorized into four major classes: the regular, context-free, context-sensitive, and recursively enumerable languages. As tree grammars are a generalization of word grammars, it seems natural to seek a similar hierarchy of tree languages. Indeed, the regular tree languages [26] generalize the regular word languages, and are widely considered as the lowest layer of a Chomsky-like hierarchy of tree languages.

How to fill the next level, of context-free grammars? This question has been answered by Rounds, who introduced *context-free tree grammars (cftg)* [139, 140, 141].[1,2] As already shown in the introduction, a context-free tree grammar is given by a finite set of context-free productions. Each of these productions allows a nonterminal symbol to be rewritten into a tree that may contain terminal and nonterminal symbols; the subtrees of the rewritten nonterminal are represented in a production by symbols called variables.

Because of this form, context-free tree grammars can be understood as a syntactic restriction of *macro grammars* [61, 60], i.e., of context-free word grammars where each nonterminal is equipped with a number of parameters.

Applications of Context-Free Tree Grammars

Context-free tree grammars have mainly been researched due to their applications in the following areas.

[1] In [139, 140], Rounds actually considers a slightly distinct model, called *creative dendrogrammar* in [140], which turns out to be equivalent to context-free tree grammars, as stated in [141].

[2] As a side-note, there does not appear to be an agreed-upon notion of context-sensitive tree grammar. Unrestricted tree grammars and some surprising properties of the recursively enumerable tree languages are presented in [39].

Program Semantics

In the two decades following their discovery, context-free tree grammars were investigated in the context of the theory of *algebraic semantics* of programming languages [127, 75, 19, 78]. There, a functional program with (non-functional) parameters is modeled by a *recursive program scheme*, a variant of a context-free tree grammar. The recursive program scheme generates an infinite tree, whose nodes are labeled with (uninterpreted) atomic operations of the programming language. In fact, this schematic tree can be expressed as the supremum of a context-free tree language with respect to a particular subtree relation. The semantics of a program is obtained by interpreting the schematic tree in a suitable algebra.

But also the uninterpreted tree already gives information on the program's behavior. Moreover, many properties of the tree are decidable, while any nontrivial property of the interpreted program is in general undecidable, due to Rice's theorem [138]. Lastly, a lot of program transformations can be specified entirely on the schematic level. In this respect, formal tree language theory becomes rewarding for research on program semantics.

Mathematical Linguistics

While most syntactic phenomena in natural language can be modelled by context-free word grammars [136, 69], in some human languages there is evidence of phenomena which are not context-free. The most prominent example is a construction from Swiss German [152] that is closely related to the formal language

$$\left\{a^n b^m c^n d^m \;\middle|\; n, m \in \mathbb{N}\right\},$$

which is clearly not context-free.[3]

In the search of a more adequate grammar formalism for natural languages, one might consider using context-sensitive grammars. However, it has been argued that the power of these grammars is too high with respect to the phenomena encountered in natural language syntax [145]; beyond that, the complexity of the word problem of context-sensitive grammars (as well as their undecidable emptiness problem) precludes using them for machine-based language processing tasks.

Therefore, Joshi introduced the notion of *mild context-sensitivity*, to obtain a class of languages "between" the context-free and context-sensitive languages [90]. Roughly, a class of languages is mildly context-sensitive if it has the following properties.

(i) It contains all context-free languages, and some non-context-free languages such as

$$\left\{a^n b^m c^n d^m \;\middle|\; n, m \in \mathbb{N}\right\} \qquad \text{and} \qquad \left\{ww \;\middle|\; w \in \{a, b\}^*\right\}.$$

(ii) All mildly context-sensitive languages have the constant growth property: for every such language L, if L is infinite, then there is a constant $k > 0$ such that for every word $w \in L$, there is another word $v \in L$ with $|w| < |v| \leq |w| + k$, cf. [93].

[3] For more examples, compare [27, 83, 36].

(iii) The membership problem of each mildly context-sensitive language can be solved efficiently, i.e., in deterministic polynomial time.

As mentioned in the introduction, the yield languages[4] of linear context-free tree grammars (where no subtree may be copied in the application of a production) are mildly context-sensitive. Therefore, linear context-free grammars appear to be an interesting model for computational linguistics. In fact, in the recent years, there has been a wide range of results on these grammars that are motivated by language processing.

On the contrary, when one allows also nonlinear context-free tree languages, properties *(ii)* and *(iii)* are violated.[5]

Formal Language Theory

Last but not least, context-free tree grammars are also helpful tools in the study of formal (word) languages. The fruitful interplay between word and tree language theory has been well-known since Thatcher's seminal paper on the interrelationship of recognizable tree languages and context-free word languages [159]. There, it was proven that the set of derivation trees of every context-free grammar is a recognizable tree language, and that each recognizable tree language is such a set of cfg derivation trees, up to a relabeling of symbols.[6] In particular, this means that the yield language of each recognizable tree language is context-free, and vice versa, that for every context-free word language there is a recognizable tree language which has the former as its yield language; cf. Theorem 1.31. This so-called *yield theorem* allows transferring properties of recognizable tree languages to the level of context-free word languages. For example, it is possible to derive in this way the pumping lemma of cfg, leading to decision procedures for their nonemptiness and infiniteness problems.

For the case of context-free tree grammars, a similar yield theorem has been discovered in [141]. Here, the related word grammars are the indexed grammars. Again, this yield theorem allows proving theorems on indexed languages at the level of trees. In [141], the theorem is used to give a decision procedure for the (non-trivial) infiniteness problem of indexed grammars. In a similar vein, we will show in Chapter 3 how to derive a decision procedure for the uniform membership problem of (ε-free) indexed grammars, from a similar procedure for cftg.

Chapter Structure

In the following Section 2.1, we recall the definitions of context-free tree grammars and their languages, as well as some appertaining properties and restrictions. Section 2.1.3 contains a few examples of context-free tree grammars. In particular, we give examples for the various

[4] I.e., the word language that contains the yield of each tree from the tree language.

[5] A counterexample for *(ii)* can be obtained by a straightforward modification of the example cftg given in the Introduction. For *(iii)*, refer to Chapter 3.

[6] It should be noted that the cited paper [159] is concerned with recognizability over unranked trees, and (therefore) with context-free grammars with extended right-hand sides. However, the proof for trees over ranked alphabets and conventional cfg can be recovered in a straightforward manner.

restrictions of the model. Section 2.1.4 is concerned with derivations of context-free tree grammars – we give an alternative characterization of the rewrite relation, and on the basis of this, a production interchange lemma. Subsequently, in Section 2.1.5 we recall the OI and the IO derivation modes, which correspond to leftmost and rightmost derivations of cfg. Of particular interest is a technical lemma on the decomposition of OI derivations (first stated by Fischer, who called it a "parallel derivation lemma"). Since linear context-free tree grammars are of special importance to this thesis, we recall some of their properties in Section 2.1.6. Specifically, we recall a normal form for linear cftg, and shine a light on the relationship between linear and nonlinear cftg.

Section 2.2 is dedicated to a type of pushdown machine for cftg, called here *pushdown tree automaton (pta)*. We recall some properties of pushdown tree automata. Moreover, we describe the construction of an equivalent pta from a cftg, and vice versa. While the equivalence between both formalisms is well-known, the given constructions are more specific. In particular, the connection between our construction of a pta from a cftg and the magmoid notation is quite illuminating. Since in Chapter 3, the transformations' efficiency will be of importance, we will examine their runtime, as well.

Section 2.3 is concerned with the connection between cftg and indexed grammars that has been mentioned above. Moreover, the path languages of cftg are treated. We recall a construction that, given a cftg, produces a cfg which generates the former's path language. As this construction will be used to solve some decision problems later, we will focus on the construction's efficiency.

We recall the most important closure properties of the context-free tree languages in Section 2.4. Again, we put special focus on the particular properties of linear cftg.

Finally, in Section 2.5, we recall the computational complexity of some decision problems of cftg. The chapter ends with Section 2.6, which features some historical remarks on cftg and related formalisms, as well as a noncomprehensive survey of literature on cftg.

Note: Mostly, the results in this chapter have been proven by other authors; they have merely been compiled and sometimes reformulated. Many results have been reproven, or the construction has been restated. Thus, this chapter could have been shorter. This way, however, the thesis is mainly self-contained. To the author's best knowledge, the alternative characterization of the rewrite relation of cftg in Lemma 2.9 is new. Moreover, we present an alternative proof of Theorem 2.22.

2.1 Context-Free Tree Grammars

First, let us recall from [141] the definition of the studied grammar model. A *context-free tree grammar (cftg)* is a tuple $G = (N, \Sigma, \xi_0, P)$ such that

- N is a ranked alphabet (its elements called *nonterminal symbols*),
- Σ is a ranked alphabet disjoint from N (its elements called *terminal symbols*),
- $\xi_0 \in \mathrm{T}(N \cup \Sigma)_0^1$ (the *axiom*),
- P is a finite set (its elements called *productions*), where each production is of the form

$$A(x_1, \ldots, x_n) \to \varrho \qquad \text{for some } n \in \mathbb{N}, A \in N^{(n)}, \text{ and } \varrho \in \mathrm{T}(N \cup \Sigma)_n^1.$$

Using the notation introduced in Section 1.3.1, the above production will often be abbreviated by $A \cdot \mathrm{Id}_n \to \varrho$, or even by $A \to \varrho$ when the rank n is clear from the context.

Assume in the following a cftg $G = (N, \Sigma, \xi_0, P)$. The elements of $\mathrm{T}(N \cup \Sigma)$ will be called the *sentential forms* of G. Let $n \in \mathbb{N}$, $A \in N^{(n)}$, and $\xi_1, \ldots, \xi_n \in \mathrm{T}(N \cup \Sigma)^1$. If the subtree $A(\xi_1, \ldots, \xi_n)$ occurs in a sentential form, we will say that the occurrence of the nonterminal A has the trees $\xi_1, \ldots, \xi_n$ as *parameters.*

Let p be some production from P of form $A \cdot \mathrm{Id}_n \to \varrho$. The *rewrite relation by p* is denoted by $\Rightarrow_p$ and defined as the smallest relation on $\mathrm{T}(N \cup \Sigma)$ such that for every $m, \ell \in \mathbb{N}$, $\xi \in \mathrm{T}(N \cup \Sigma)_{\ell+1}^m$ that contains $x_{\ell+1}$ exactly once, and $\zeta \in \mathrm{T}(N \cup \Sigma)_\ell^n$, we have

$$\xi \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \Rightarrow_p \xi \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta].$$

In this situation, we say that the production p *is applied at position* w, where w is the unique element of $\mathrm{pos}(\xi)$ such that $\xi(w) = x_{\ell+1}$. We will sometimes also write $\overset{w}{\Rightarrow}_p$ to express that p is applied at position w.

Observe that if $\ell = 0$, the definition simplifies to

$$\xi \cdot A \cdot \zeta \Rightarrow_p \xi \cdot \varrho \cdot \zeta\,,$$

analogous to the word case. The *rewrite relation of G* is the relation $\Rightarrow_G$ on $\mathrm{T}(N \cup \Sigma)$ given by $\Rightarrow_G = \bigcup_{p \in P} \Rightarrow_p$. When clear from the context, we will omit the subscript G and write simply $\Rightarrow$ instead of $\Rightarrow_G$. For every $\xi \in \mathrm{T}(N \cup \Sigma)$, let

$$\mathcal{L}(G, \xi) = \left\{ t \in \mathrm{T}(\Sigma) \;\middle|\; \xi \Rightarrow_G^* t \right\}.$$

Every cftg $G = (N, \Sigma, \xi_0, P)$ is associated the *tree language* $\mathcal{L}(G) \subseteq \mathrm{T}(\Sigma)_0^1$ *generated by G,* defined by $\mathcal{L}(G) = \mathcal{L}(G, \xi_0)$. A tree language is said to be *context-free* if it is generated by some cftg, and the class of all context-free tree languages (over some ranked alphabet Σ) is denoted by CFT (resp. by $\mathsf{CFT}(\Sigma)$).

When we discuss complexity-theoretic properties of cftg, we need a notion to measure their size. Formally, the *size* of a cftg $G = (N, \Sigma, \xi_0, P)$, denoted by $|G|$, is

$$|G| = |N| + |\xi_0| + \sum_{(A \to \varrho) \in P} (1 + |\varrho|)\,.$$

Remark 2.1. Note that this notion of size does not take into account the tape space to store the individual symbols in the productions of G. In order to take heed of this additional cost, it suffices to multiply $|G|$ with the factor $\log(|V| + \max \mathrm{rk}(V))$, where V is the ranked alphabet $N \cup \Sigma$ (cf. the discussion in [80, p. 94]). In the context of this work, the rough notion of size defined above will be sufficient. $\lhd$

In allowing an axiom instead of an initial nonterminal symbol, we deviate a little from classical definitions, and from the word case as given in Section 1.2.3. However, this generalization will be technically convenient, and, as the following lemma shows, it does not increase the generative power of cftg.

Lemma 2.2. *For every context-free tree language $L \in \mathsf{CFT}$, there is a cftg $G = (N, \Sigma, S, P)$ with $S \in N^{(0)}$ such that $L = \mathcal{L}(G)$.*

Proof. Let $G = (N, \Sigma, \xi_0, P)$ be a cftg. We construct the cftg $G' = (N', \Sigma, S, P')$, where $N' = N \cup \{S^{(0)}\}$ for some distinct nonterminal S, and P' contains every production from P, as well as $S \to \xi_0$. Clearly, for every $t \in \mathrm{T}(\Sigma)_0^1$, we have

$$\xi_0 \Rightarrow_G^* t \qquad \text{if and only if} \qquad S \Rightarrow_{G'} \xi_0 \Rightarrow_{G'}^* t\,,$$

and therefore $\mathcal{L}(G') = \mathcal{L}(G)$. $\square$

Whenever the axiom of a cftg G is a single nonterminal of rank 0, we will say that G *has an initial nonterminal.*

2.1.1 Particular Restrictions

A cftg $G = (N, \Sigma, \xi_0, P)$ is called *linear* (resp. *nondeleting*), if for every production $A \to \varrho$ in P, the right-hand side ϱ is linear (resp. nondeleting). Linear cftg are abbreviated by l-cftg, and linear and nondeleting cftg by ln-cftg. The respectively generated classes of tree languages (over Σ) are denoted by CFT_ℓ ($\mathsf{CFT}_\ell(\Sigma)$), and $\mathsf{CFT}_{\ell n}$ ($\mathsf{CFT}_{\ell n}(\Sigma)$), and called *linear (and nondeleting) context-free tree languages.* Context-free tree grammars that are not linear will be called *nonlinear* or *copying.*

Moreover, G is said to be a *regular tree grammar (rtg)* if $N = N^{(0)}$. This means that nonterminal symbols may only occur as leaves. The tree languages generated by rtg are precisely the recognizable tree languages, as already mentioned in Section 1.3.2.

As a generalization of the condition for rtg, let $n \in \mathbb{N}$. We say that G is *n-adic* if

$$N = N^{(0)} \cup \cdots \cup N^{(n)}\,,$$

and a language $L \in \mathsf{CFT}$ is *n-adic* if it is generated by some n-adic cftg. The 0-adic cftg are therefore precisely the rtg. We will write "monadic" instead of "1-adic." Note that by this definition, an n-adic grammar (resp. language) is also m-adic, for every $m \geq n$. Later on, we will cover linear monadic cftg, they will be abbreviated by lm-cftg.

A cftg $G = (N, \Sigma, \xi_0, P)$ is called *coregular* [10] if for every production $A \to \varrho$ of G and every $w \in \mathrm{pos}(\varrho)$, $\varrho(w) \in N$ only if $w = \varepsilon$. Intuitively, a nonterminal symbol may only occur at the root node of a production's right-hand side. Coregular cftg are closely related to EDT0L systems [10].[7] The tree languages of coregular cftg have been investigated in [84].

2.1.2 Special Forms

A cftg $G = (N, \Sigma, \xi_0, P)$ is said to be in *normal form* if it has an initial nonterminal and each of its productions is of one of the forms

(i) $A \cdot \mathrm{Id}_n \to B \cdot (C_1 \cdot \mathrm{Id}_n, \dots, C_m \cdot \mathrm{Id}_n)$
for some $n \in \mathbb{N}$, $m \in \mathbb{N}_1$, $A \in N^{(n)}$, $B \in N^{(m)}$, and $C_1, \dots, C_m \in N^{(n)}$,

(ii) $A \cdot \mathrm{Id}_n \to x_i$
for some $n \in \mathbb{N}_1$, $A \in N^{(n)}$, and $i \in [n]$, or

(iii) $A \cdot \mathrm{Id}_n \to \sigma \cdot \vartheta$
for some n, $k \in \mathbb{N}$, $A \in N^{(n)}$, $\sigma \in \Sigma^{(k)}$, and $\vartheta \in \Theta_n^k$.

Productions of form *(i)* are called *nonterminal productions,* those of form *(ii)* are *collapsing productions,* and the productions of form *(iii)* are called *terminal productions.* There is an apparent (but imperfect) analogy to the Chomsky normal form of cfg (see Section 1.2.3): Nonterminal productions correspond to productions of form $A \to BC$, and terminal productions correspond to productions of form $A \to a$. Collapsing productions can be understood as ε-productions $A \to \varepsilon$.

Theorem 2.3 (Maibaum [114, Thm. 14]). *For every $L \in \mathsf{CFT}$, there is a cftg G in normal form such that $\mathcal{L}(G) = L$. Moreover, G can be constructed in logarithmic space.*

The claim on logspace-constructability is easily reobserved. It is important to note that for the theorem to hold, one *must* allow productions of type *(ii)*. In sharp contrast to the word case, collapsing (or ε-) productions cannot be eliminated from cftg [110, 111].

A cftg G is said to be *total* if $\mathcal{L}(G,A) \neq \emptyset$ for every nonterminal A of G. As the following lemma shows, we may always assume that a cftg is total.

Lemma 2.4 (Arnold and Dauchet [11, Annex]). *For every cftg G with $\mathcal{L}(G) \neq \emptyset$, there is an equivalent total cftg G'.*

The proof in [11] assumes that G is in normal form, but with an evident generalization it also goes through without this assumption. The proof idea is to introduce the production $A \to \#$, where $\#$ is some dummy symbol, for every non-productive nonterminal A of G, i.e., with $\mathcal{L}(G,A) = \emptyset$. Of course, care must be taken that this dummy symbol is not produced in the course of a derivation in G' which was blocked before in G. Therefore every nonterminal $A \in N^{(k)}$ is annotated with a set $\alpha \subseteq [k]$ of forbidden indices, which prevents choosing a non-productive nonterminal. Apart from this annotation, the construction does not alter the shape of the productions of G.

[7] EDT0L systems are a well-known type of parallel rewriting system [98]. EDT0L is an abbreviation for *extended deterministic table zero-interaction Lindenmayer system.*

Convention. *Analogously to the word case, we shall often denote a finite set of cftg productions* $\{A \to \varrho_1, \ldots, A \to \varrho_k\}$ *with common left-hand side* A *by*

$$A \to \varrho_1 + \cdots + \varrho_k\,, \qquad \text{or by} \qquad A \to \sum_{i=1}^{k} \varrho_i\,.$$

2.1.3 Examples

In this section, we give a tour of some interesting inhabitants of the class CFT and its various subclasses.

Example 2.5. Let $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$, and define the cftg $G_P = (N, \Sigma, \xi_0, P)$, where $N = \{A^{(1)}\}$,

$$\xi_0 = \begin{array}{c} A \\ | \\ \alpha \end{array}\,,$$

and P contains the productions

$$A(x_1) \;\to\; x_1 \quad + \quad \begin{array}{c} A \\ | \\ \sigma \\ / \;\; \backslash \\ x_1 \;\; x_1 \end{array} \quad .$$

The cftg G_P is monadic, nonlinear, nondeleting, and coregular. Clearly, every derivation of a tree $t \in \mathcal{L}(G)$ is of the form

$$\begin{array}{c} A \\ | \\ \alpha \end{array} \Rightarrow \begin{array}{c} A \\ | \\ \sigma \\ / \;\; \backslash \\ \alpha \;\; \alpha \end{array} \Rightarrow \begin{array}{c} A \\ | \\ \sigma \\ \diagup \qquad \diagdown \\ \begin{array}{c} \sigma \\ / \;\; \backslash \\ \alpha \;\; \alpha \end{array} \quad \begin{array}{c} \sigma \\ / \;\; \backslash \\ \alpha \;\; \alpha \end{array} \end{array} \Rightarrow \cdots \Rightarrow \begin{array}{c} A \\ | \\ t \end{array} \Rightarrow t\,.$$

Hence, it is easy to see that $\mathcal{L}(G_P)$ is the set of all perfect binary trees over Σ. We will see later that $\mathcal{L}(G_P)$ is not a linear context-free tree language. ◁

Example 2.6. Let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$. Define the cftg $G_{\text{lin}} = (N, \Sigma, \xi_0, P)$, where $N = \{A^{(2)}\}$,

$$\xi_0 = \begin{array}{c} A \\ / \;\; \backslash \\ \alpha \;\; \alpha \end{array}\,,$$

and P contains the two productions

$$A(x_1, x_2) \;\to\; \begin{array}{c} \gamma \\ | \\ A \\ / \;\; \backslash \\ \begin{array}{c} \gamma \\ | \\ x_1 \end{array} \;\; \begin{array}{c} \gamma \\ | \\ x_2 \end{array} \end{array} \quad + \quad \begin{array}{c} \sigma \\ / \;\; \backslash \\ x_1 \;\; x_2 \end{array}\,.$$

This cftg is linear and nondeleting. Every derivation in G_{lin} is of the form

$$A(\alpha,\alpha) \Rightarrow \gamma(A(\gamma(\alpha),\gamma(\alpha))) \Rightarrow \gamma(\gamma(A(\gamma(\gamma(\alpha)),\gamma(\gamma(\alpha))))) \Rightarrow \cdots \Rightarrow \gamma^i(A(\gamma^i(\alpha),\gamma^i(\alpha))) \Rightarrow \gamma^i(\sigma(\gamma^i(\alpha),\gamma^i(\alpha))),$$

and therefore $\mathcal{L}(G_{\text{lin}}) = \{\gamma^i(\sigma(\gamma^i\alpha,\gamma^i\alpha)) \mid i \in \mathbb{N}\}$. It is a simple exercise to show that $\mathcal{L}(G_{\text{lin}})$ is not recognizable, e.g. by using the Myhill-Nerode theorem. Hence $\mathcal{L}(G_{\text{lin}})$ is a witness for the proper inclusion $\mathsf{RECT} \subset \mathsf{CFT}_\ell$. ◁

Example 2.7. Let Γ be a ranked alphabet such that $\Gamma^{(0)} = \{o\}$. We define a disjoint alphabet $\bar{\Gamma}$, which is in a bijective correspondence to Γ, as follows. For every $\sigma \in \Gamma^{(k)}$ with $k \in \mathbb{N}_1$, let $\bar{\sigma} \in \bar{\Gamma}^{(1)}$; moreover let $\bar{o} \in \bar{\Gamma}^{(0)}$. We define $\Sigma = \Gamma \cup \bar{\Gamma}$, as well as the cftg $G^D_\Gamma = (N, \Sigma, \xi_0, P)$, where $N = \{A^{(1)}\}$,

$$\xi_0 = o(A(\bar{o})),$$

and P contains the productions

$$A(x_1) \to x_1 \quad + \quad A(A(x_1)) \quad + \sum_{\substack{k>0,\\ \sigma\in\Sigma^{(k)}}} \sigma(A(\bar{\sigma}(x_1)), \ldots, A(\bar{\sigma}(x_1))).$$

The cftg G^D_Γ is monadic, nonlinear, and nondeleting. It generates precisely the tree language of all *Dyck trees* over Σ. A tree $t \in \mathrm{T}_\Sigma$ is called a Dyck tree if every word which can be read along a path from the root of t to one of its leaves is a Dyck word, as described in Section 1.2.3.[8]

Assuming that $\Gamma = \{\sigma^{(2)}, \gamma^{(1)}, o^{(0)}\}$, the following trees are Dyck trees over $\Sigma = \Gamma \cup \bar{\Gamma}$:

[8]Where each symbol $\bar{\sigma} \in \bar{\Gamma}$ is taken to be the right inverse of $\sigma \in \Gamma$.

$$o(\bar{o}) \qquad o(\sigma(\gamma(\bar{\gamma}(\bar{\sigma}(\bar{o}))), \bar{\sigma}(\bar{o}))) \qquad o(\sigma(\sigma(\bar{\sigma}(\bar{\sigma}(\bar{o})), \bar{\sigma}(\bar{\sigma}(\bar{o}))), \bar{\sigma}(\gamma(\bar{\gamma}(\bar{o})))))$$

As shown in [13, Thm. 4.2], the Dyck tree languages have the same significance for CFT as their counterparts in the word case: a tree language L is context-free if and only if there are a recognizable tree language R, a Dyck tree language D and a linear tree homomorphism h such that $L = h(R \cap D)$. Equivalently, every context-free tree language can be represented as the image of a Dyck tree language under some linear and nondeleting top-down tree transducer [13, Thm. 4.2]. $\lhd$

Example 2.8 (Arnold and Dauchet [14]). Let

$$\Delta = \left\{\gamma^{(2)}, \delta_1^{(2)}, \delta_2^{(2)}, a^{(1)}, \#^{(0)}\right\}.$$

Define the cftg $G_{\mathrm{hom}} = (N, \Delta, \xi_0, P)$, where $N = \{A^{(2)}, B^{(1)}, C^{(2)}\}$,

$$\xi_0 = A(\delta_2(\#, \#), a(\#)),$$

and P contains the productions

$$A(x_1, x_2) \to A(C(x_1, \delta_2(x_2, x_2)), a(x_2)) + B(\delta_1(x_1, x_2)),$$

$$C(x_1, x_2) \to x_1 + x_2,$$

$$B(x_1) \to B(\gamma(x_1, x_1)) + \gamma(x_1, x_1).$$

To understand G_{hom}, first observe that the nonterminal C implements nondeterministic choice. Indeed, for every $k \in \mathbb{N}$ and $\xi_1, \ldots, \xi_k \in \mathrm{T}(N \cup \Delta)_0^1$, we have

$$\mathcal{L}\big(G_{\mathrm{hom}}, C(C(\ldots C(C(\xi_1, \xi_2), \xi_3) \ldots, \xi_{k-1}), \xi_k)\big) = \bigcup_{i \in [k]} \mathcal{L}(G_{\mathrm{hom}}, \xi_i).$$

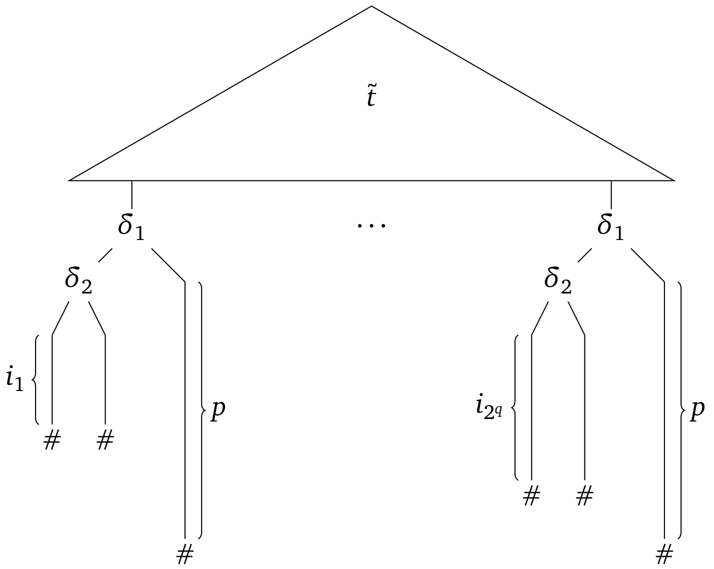
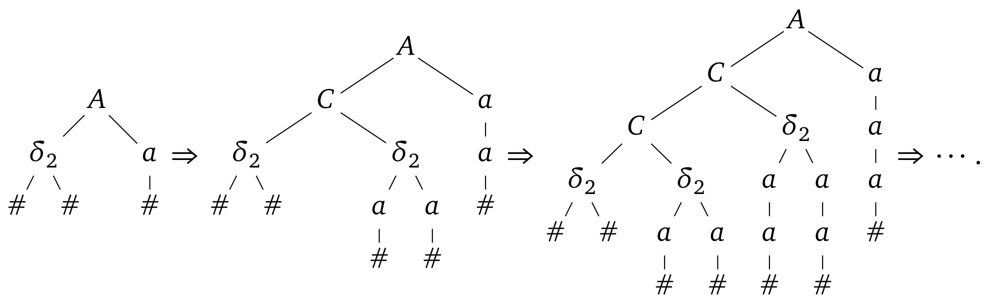


Figure 2.1: A tree $t \in \mathcal{L}(G_{\text{hom}})$

Moreover, the nonterminal B allows generating all perfect binary trees in $\mathrm{T}(\{\gamma\})_1^1$ that have positive height. Following [14], every tree generated by G_{hom} has a derivation which begins with



As we see, in the i-th step above ($i \in \mathbb{N}$), the second subtree of the nonterminal A is $a^{i+1}\#$, while its first subtree is a nondeterministic choice tree that can generate

$$\delta_2(a^0\#, a^0\#), \qquad \delta_2(a^1\#, a^1\#), \qquad \ldots, \qquad \text{or} \qquad \delta_2(a^i\#, a^i\#).$$

Applying the production $A(x_1, x_2) \to B(\delta_1(x_1, x_2))$ to such a sentential form, followed by a sequence of productions for B, we see that

$$\begin{aligned}\mathcal{L}(G_{\text{hom}}) = \bigl\{\tilde{t} \cdot [s_1, \ldots, s_{2^q}] \bigm| \; & q, p \in \mathbb{N}_1, \\ & \tilde{t} \in \widetilde{\mathrm{T}}(\{\gamma\})_{2^q}^1 \text{ is a perfect binary tree of height } q, \\ & s_1, \ldots, s_{2^q} \in F_p\bigr\},\end{aligned}$$

where for every $p \in \mathbb{N}_1$,

$$F_p = \bigl\{\delta_1(\delta_2(a^i\#, a^i\#), a^p\#) \bigm| i \in \mathbb{N}, \, i < p\bigr\}.$$

Compare also Figure 2.1 for a sketch of an element of $\mathcal{L}(G_{\text{hom}})$.

The importance of G_{hom} is that it serves as a counterexample for the closure of CFT under inverse linear tree homomorphisms. Let, in fact,

$$\Sigma = \Delta \setminus \{\delta_1, \delta_2\} \cup \{\sigma^{(3)}\},$$

and consider the linear and nondeleting tree homomorphism $h\colon \mathrm{T}_\Sigma(X) \to \mathrm{T}_\Delta(X)$ such that

$$h\big(\sigma(x_1, x_2, x_3)\big) = \delta_1(\delta_2(x_1, x_2), x_3),$$

and h is the identity on $\Sigma \setminus \{\sigma\}$. Let $L = h^{-1}(\mathcal{L}(G_{\text{hom}}))$. Put simply, L is the result of replacing every subtree $\delta_1(\delta_2(a^i\#, a^i\#), a^p\#)$ in $t \in \mathcal{L}(G_{\text{hom}})$ by $\sigma(a^i\#, a^i\#, a^p\#)$. It has been shown in [14] that L is not a context-free tree language. The intuitive difference between $\mathcal{L}(G_{\text{hom}})$ and L is that in the generation of the former language, G_{hom} can postpone the generation of the subtrees $\delta_2(a^i\#, a^i\#)$ to after the decision for the common subtree $a^p\#$, by using the nonterminal C. However, if there was a cftg generating L, then it would have to generate all subtrees $\sigma(a^i\#, a^i\#, a^p\#)$ simultaneously. The proof in [14] shows that then only a bounded number of distinct subtrees $\sigma(a^i\#, a^i\#, a^p\#)$ can be generated, yielding a contradiction.

In Chapter 4, we will strengthen this nonclosure result, and show that even the class CFT_ℓ is not closed under inverse linear tree homomorphisms. $\lhd$

2.1.4 Elementary Properties of Derivations

The following section is concerned with derivations of cftg. In Lemma 2.9, we begin with an alternative characterization of the rewrite relation of cftg, by means of induction. It appears that this characterization is novel. In Lemma 2.12, we then show when and how the productions in a derivation may be reordered.

Lemma 2.9. *Let $G = (N, \Sigma, \xi_0, P)$ be a cftg. Then $\Rightarrow_G$ is the smallest relation $\Rightarrow'_G$ on $\mathrm{T}(N \cup \Sigma)$ such that*

1. *for every production of form $A \to \varrho$ from P, we have $A \Rightarrow'_G \varrho$, and*
2. *for every ξ, ξ', and $\zeta \in \mathrm{T}(N \cup \Sigma)$, whenever $\xi \Rightarrow'_G \xi'$, then also*
   - *(i) $\sigma \cdot \xi \Rightarrow'_G \sigma \cdot \xi'$ for every $\sigma \in \Sigma$,*
   - *(ii) $A \cdot \xi \Rightarrow'_G A \cdot \xi'$ for every $A \in N$,*
   - *(iii) $\xi \cdot \zeta \Rightarrow'_G \xi' \cdot \zeta$,*
   - *(iv) $\xi \otimes \zeta \Rightarrow'_G \xi' \otimes \zeta$, and*
   - *(v) $\zeta \otimes \xi \Rightarrow'_G \zeta \otimes \xi'$.*

Proof. Denote the smallest relation on $\mathrm{T}(N \cup \Sigma)$ that satisfies the above conditions by $\Rightarrow'_G$. We want to show that $\Rightarrow_G$ and $\Rightarrow'_G$ are equal. The direction $\Rightarrow'_G \subseteq \Rightarrow_G$ is easy to show. We only consider the implication *(2i)*; the other ones are proven similarly. By the definition of $\Rightarrow_G$,

$$\xi = \eta \cdot [\mathrm{Id}_k, A \cdot \zeta] \qquad \text{and} \qquad \xi' = \eta \cdot [\mathrm{Id}_k, \varrho \cdot \zeta]$$

for some k, $n \in \mathbb{N}$, $\eta \in \mathrm{T}(N \cup \Sigma)_{k+1}$ that contains x_{k+1} exactly once, $A \in N^{(n)}$, $\zeta \in \mathrm{T}(N \cup \Sigma)^n_k$, and $\varrho \in \mathrm{T}(N \cup \Sigma)^1_n$. In particular, the production $A \cdot \mathrm{Id}_n \to \varrho$ is contained in P. Let $\sigma \in \Sigma$. Then clearly $\sigma \cdot \eta$ also contains x_{k+1} exactly once, and thus

$$\sigma \cdot \xi = \sigma \cdot \eta \cdot [\mathrm{Id}_k, A \cdot \zeta] \Rightarrow_G \sigma \cdot \eta \cdot [\mathrm{Id}_k, \varrho \cdot \zeta] = \sigma \cdot \xi' .$$

* * *

We still must prove that $\Rightarrow_G \subseteq \Rightarrow'_G$. We will show that for every production $A \cdot \mathrm{Id}_n \to \varrho$ of G, for every m, $\ell \in \mathbb{N}$, $\xi \in \mathrm{T}(N \cup \Sigma)^m_{\ell+1}$ that contains $x_{\ell+1}$ exactly once, and for every $\zeta \in \mathrm{T}(N \cup \Sigma)^n_\ell$, we have

$$\xi \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \Rightarrow'_G \xi \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta] .$$

Let, for this purpose, $\mathrm{lin}(\xi) = (\tilde{\xi}, \vartheta)$ for some $\tilde{\xi} \in \mathrm{T}(N \cup \Sigma)^m_k$ and $\vartheta \in \Theta^k_{\ell+1}$ with $k \in \mathbb{N}$. The proof is by structural induction on $\tilde{\xi}$, as described in Section 1.3.3.

For the induction base, there are two cases, namely $\tilde{\xi} = \mathrm{Id}_0$ and $\tilde{\xi} = \mathrm{Id}_1$. The first case $\tilde{\xi} = \mathrm{Id}_0$ is clearly precluded by the assumption that ξ contains $x_{\ell+1}$. In the second case $\tilde{\xi} = \mathrm{Id}_1$, we obtain that $\vartheta = \langle \ell + 1; x_{\ell+1} \rangle$, and therefore

$$\tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, A \cdot \zeta] = A \cdot \zeta \Rightarrow'_G \varrho \cdot \zeta = \tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta] ,$$

where the relation $\Rightarrow'_G$ holds because of conditions *(1)* and *(2iii)* from above. For the induction step, we distinguish two cases.

(I) For the first case, assume that $\tilde{\xi} \in \widetilde{\mathrm{T}}(N \cup \Sigma)^1_k \setminus \{\mathrm{Id}_1\}$. Then there are $U \in N \cup \Sigma$ and $\tilde{\eta} \in \widetilde{\mathrm{T}}(N \cup \Sigma)_k$ such that $\tilde{\xi} = U \cdot \tilde{\eta}$. By the induction hypothesis,

$$\tilde{\eta} \cdot \vartheta \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \Rightarrow'_G \tilde{\eta} \cdot \vartheta \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta] ,$$

and by condition *(2i)* or *(2ii)*, also

$$\tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \Rightarrow'_G \tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta] .$$

(II) For the second case, assume that $\tilde{\xi} \in \widetilde{\mathrm{T}}(N \cup \Sigma)^m_k$ for some $m \in \mathbb{N}$ with $m > 1$. Since there is precisely one occurrence of $x_{\ell+1}$ in ϑ and $\tilde{\xi}$ is torsion-free, there is a unique component $\tilde{\kappa}$ of $\tilde{\xi}$ into which $x_{\ell+1}$ is substituted.

Formally, there are $\tilde{\eta}_1 \in \widetilde{\mathrm{T}}(N \cup \Sigma)$, $\tilde{\kappa} \in \widetilde{\mathrm{T}}(N \cup \Sigma)^1$, and $\tilde{\eta}_2 \in \widetilde{\mathrm{T}}(N \cup \Sigma)$, as well as $\vartheta_1 \in \Theta_\ell$, $\vartheta_2 \in \Theta_{\ell+1}$, and $\vartheta_3 \in \Theta_\ell$ such that we can write

$$\tilde{\xi} \cdot \vartheta = \big[\tilde{\eta}_1 \cdot \vartheta_1, \tilde{\kappa} \cdot \vartheta_2, \tilde{\eta}_2 \cdot \vartheta_3 \big] ,$$

and such that $x_{\ell+1}$ occurs precisely once in ϑ_2. By the induction hypothesis,

$$\tilde{\kappa} \cdot \vartheta_2 \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \Rightarrow'_G \tilde{\kappa} \cdot \vartheta_2 \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta] .$$

Now, observe that

$$\tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, A \cdot \zeta] = \Big(\tilde{\eta}_1 \cdot \vartheta_1 \otimes \tilde{\kappa} \cdot \vartheta_2 \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \otimes \tilde{\eta}_2 \cdot \vartheta_3 \Big) \cdot [\mathrm{Id}_\ell, \mathrm{Id}_\ell, \mathrm{Id}_\ell]$$

and

$$\tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta] = \Big(\tilde{\eta}_1 \cdot \vartheta_1 \otimes \tilde{\kappa} \cdot \vartheta_2 \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta] \otimes \tilde{\eta}_2 \cdot \vartheta_3\Big) \cdot [\mathrm{Id}_\ell, \mathrm{Id}_\ell, \mathrm{Id}_\ell]\,.$$

By judicious application of the conditions *(2iii)*–*(2v)*, we obtain that

$$\tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \Rightarrow'_G \tilde{\xi} \cdot \vartheta \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta]\,.$$

Since $\Rightarrow_G$ is the union of $\Rightarrow_p$ for all productions $p \in P$, we conclude $\Rightarrow_G \subseteq \Rightarrow'_G$. □

Corollary 2.10. *Let $G = (N, \Sigma, \xi_0, P)$ be a cftg, and assume that $\xi \Rightarrow_G \xi'$ for some ξ, $\xi' \in \mathrm{T}(N \cup \Sigma)$. For every $\tilde{\zeta} \in \widetilde{\mathrm{T}}(N \cup \Sigma)$ such that $\tilde{\zeta} \cdot \xi$ is defined, we have $\tilde{\zeta} \cdot \xi \Rightarrow_G \tilde{\zeta} \cdot \xi'$.*

Proof. By structural induction on $\tilde{\zeta}$, using the conditions *(2i)*, *(2ii)*, *(2iv)* and *(2v)* from the lemma above. □

Remark 2.11. Intuitively, the corollary tells us that the rewrite relation of cftg is compatible with concatenation from the left with torsion-free tuples. In contrast, the relation is preserved under concatenation from the right with *any* tuple, as Condition *(2iii)* shows.

Observe that the corollary holds only for $\tilde{\zeta}$ chosen torsion-free. In fact, assume that a cftg G contains the production $A \to \alpha$. Then $A \Rightarrow_G \alpha$, but if we were to choose an element of $\mathrm{T}(\Sigma)^1_1 \setminus \widetilde{\mathrm{T}}(\Sigma)^1_1$, say $\zeta = \sigma(x_1, x_1)$, the result would be

$$\zeta \cdot A = \sigma(A, A) \not\Rightarrow_G \sigma(\alpha, \alpha) = \zeta \cdot \alpha\,.$$

However, $\sigma(A, A) \Rightarrow^2_G \sigma(\alpha, \alpha)$ *does* hold, of course. ◁

We continue with the announced production interchange lemma, which specifies under which conditions the productions in a derivation may be reordered. This question is nontrivial for cftg, as the features of nonlinearity and deletion may interfere: it may occur that after exchanging productions p_1 and p_2, which appear in this order in a derivation, one can no longer apply p_1, as the site where it is applied has been deleted by p_2. It may also be the case that the nonterminal where p_1 is applied has been copied by p_2, and therefore p_1 must now be applied more than once.

The lemma also treats the special cases of linear and nondeleting grammars. A production interchange lemma for macro grammars has already been given implicitly in [60, Thm. 4.1.2], compare also [114, Thm. 11]. For a similar lemma on linear cftg, see [99, Lem. 4].

Lemma 2.12. *Let $G = (N, \Sigma, \xi_0, P)$ be a cftg, let p_1, $p_2 \in P$ of forms $A_1 \to \varrho_1$ and $A_2 \to \varrho_2$, respectively, and let ξ, ξ_1, $\zeta \in \mathrm{T}(N \cup \Sigma)$, w_1, $w_2 \in \mathbb{P}$ with*

$$\xi \overset{w_1}{\Rightarrow}_{p_1} \xi_1 \overset{w_2}{\Rightarrow}_{p_2} \zeta\,.$$

Moreover, assume that p_2 is not applied in the right-hand side of p_1 – formally, let

$$w_2 \in \mathrm{pos}(\xi_1) \setminus (w_1 \cdot \mathrm{pos}_{N \cup \Sigma}(\varrho_1))\,.$$

Then the following hold.

(i) If $w_1 \parallel w_2$, then there is $\xi_2 \in \mathrm{T}(N \cup \Sigma)$ with

$$\xi \Rightarrow_{p_2} \xi_2 \Rightarrow_{p_1} \zeta\,.$$

(ii) If $w_2 \sqsubseteq w_1$, then there is $\xi_2 \in \mathrm{T}(N \cup \Sigma)$ such that

$$\xi \Rightarrow_{p_2} \xi_2 \,(\Rightarrow_{p_1})^* \,\zeta\,.$$

In particular, if G is linear and nondeleting, then

$$\xi \Rightarrow_{p_2} \xi_2 \Rightarrow_{p_1} \zeta\,.$$

(iii) If $w_1 \sqsubseteq w_2$ and G is linear and nondeleting, then there is $\xi_2 \in \mathrm{T}(N \cup \Sigma)$ such that

$$\xi \Rightarrow_{p_2} \xi_2 \Rightarrow_{p_1} \zeta\,.$$

Proof. Let p_1 be of form $A_1 \to \varrho_1$ and p_2 be of form $A_2 \to \varrho_2$.

(i). Since $w_1 \parallel w_2$, one can write

$$\xi = \eta \cdot [\mathrm{Id}_q, A_1 \cdot \kappa_1, A_2 \cdot \kappa_2]$$

for some $q \in \mathbb{N}$, $\kappa_1, \kappa_2 \in \mathrm{T}(N \cup \Sigma)_q$, and some $\eta \in \mathrm{T}(N \cup \Sigma)_{q+2}$ that contains each of x_{q+1} and x_{q+2} precisely once. Moreover,

$$\xi_1 = \eta \cdot [\mathrm{Id}_q, \varrho_1 \cdot \kappa_1, A_2 \cdot \kappa_2] \qquad \text{and} \qquad \zeta = \eta \cdot [\mathrm{Id}_q, \varrho_1 \cdot \kappa_1, \varrho_2 \cdot \kappa_2]\,.$$

Clearly, the property holds with

$$\xi_2 = \eta \cdot [\mathrm{Id}_q, A_1 \cdot \kappa_1, \varrho_2 \cdot \kappa_2]\,,$$

because then $\xi \Rightarrow_{p_2} \xi_2 \Rightarrow_{p_1} \zeta$.

(ii). As $w_2 \sqsubseteq w_1$, we have

$$\xi = \eta \cdot \big[\mathrm{Id}_q, A_2 \cdot \kappa \cdot [\mathrm{Id}_q, A_1 \cdot \varphi]\big]$$

for some $q \in \mathbb{N}$, $\varphi \in \mathrm{T}(N \cup \Sigma)_q$, and $\eta, \kappa \in \mathrm{T}(N \cup \Sigma)_{q+1}$, where both η and κ contain x_{q+1} precisely once. Then

$$\xi_1 = \eta \cdot \big[\mathrm{Id}_q, A_2 \cdot \kappa \cdot [\mathrm{Id}_q, \varrho_1 \cdot \varphi]\big].$$

Let $\mathrm{lin}(\varrho_2) = (\tilde{\varrho}_2, \vartheta)$, and let $\ell = \mathrm{rk\,inf}(\tilde{\varrho}_2)$. By application of Lemma 1.24,

$$\zeta = \eta \cdot \big[\mathrm{Id}_q, \tilde{\varrho}_2 \cdot [\pi_{\vartheta(1)} \cdot \kappa \cdot [\mathrm{Id}_q, \varrho_1 \cdot \varphi], \ldots, \pi_{\vartheta(\ell)} \cdot \kappa \cdot [\mathrm{Id}_q, \varrho_1 \cdot \varphi]]\big].$$

We let

$$\xi_2 = \eta \cdot \big[\mathrm{Id}_q, \tilde{\varrho}_2 \cdot [\pi_{\vartheta(1)} \cdot \kappa \cdot [\mathrm{Id}_q, A_1 \cdot \varphi], \ldots, \pi_{\vartheta(\ell)} \cdot \kappa \cdot [\mathrm{Id}_q, A_1 \cdot \varphi]]\big].$$

Assume that the unique occurrence of x_{q+1} in κ is in its component $\pi_j \cdot \kappa$, for some $j \in \mathbb{N}$. Then, for every $i \in [\ell]$ with $\vartheta(i) = j$, we have

$$\pi_{\vartheta(i)} \cdot \kappa \cdot [\mathrm{Id}_q, A_1 \cdot \varphi] \Rightarrow_{p_1} \pi_{\vartheta(i)} \cdot \kappa \cdot [\mathrm{Id}_q, \varrho_1 \cdot \varphi].$$

Moreover, for every $i \in [\ell]$ with $\vartheta(i) \neq j$, we obtain

$$\pi_{\vartheta(i)} \cdot \kappa \cdot [\mathrm{Id}_q, A_1 \cdot \varphi] = \pi_{\vartheta(i)} \cdot \kappa \cdot [\mathrm{Id}_q, \varrho_1 \cdot \varphi],$$

since the denoted occurrence of A_1 is deleted. So there is some $\ell' \in \mathbb{N}$ with $\ell' \leq \ell$ that satisfies $\xi_2 \Rightarrow_{p_1}^{\ell'} \zeta$.

In the special case that G is linear and nondeleting, then so is ϑ. Hence there is precisely one $i \in [\ell]$ with $\vartheta(i) = j$, and thus $\xi_2 \Rightarrow_{p_1} \zeta$.

(iii). Since $w_1 \sqsubseteq w_2$, we have

$$\xi = \eta \cdot \big[\mathrm{Id}_q, A_1 \cdot \kappa \cdot [\mathrm{Id}_q, A_2 \cdot \varphi]\big]$$

for some $q \in \mathbb{N}$, $\varphi \in \mathrm{T}(N \cup \Sigma)_q$, and $\eta, \kappa \in \mathrm{T}(N \cup \Sigma)_{q+1}$, where both η and κ contain x_{q+1} precisely once. Moreover, since G is linear and nondeleting, x_{q+1} occurs precisely once in $\eta \cdot [\mathrm{Id}_q, \varrho_1 \cdot \kappa]$. So the tuples

$$\xi_1 = \eta \cdot \big[\mathrm{Id}_q, \varrho_1 \cdot \kappa \cdot [\mathrm{Id}_q, A_2 \cdot \varphi]\big] \qquad \text{and} \qquad \zeta = \eta \cdot \big[\mathrm{Id}_q, \varrho_1 \cdot \kappa \cdot [\mathrm{Id}_q, \varrho_2 \cdot \varphi]\big]$$

satisfy $\xi \Rightarrow_G \xi_1 \Rightarrow_G \zeta$. When we let

$$\xi_2 = \eta \cdot \big[\mathrm{Id}_q, A_1 \cdot \kappa \cdot [\mathrm{Id}_q, \varrho_2 \cdot \varphi]\big],$$

we obtain that $\xi \Rightarrow_{p_2} \xi_2 \Rightarrow_{p_1} \zeta$. □

2.1.5 Derivation Modes

Similar to leftmost and rightmost derivations of cfg, there are two restricted modes of derivation for cftg: the *outside-in (OI)* and the *inside-out (IO)* mode. Intuitively, in an OI derivation a production may only be applied to nonterminals that appear topmost in a sentential form: they must not have a proper ancestor that is also labeled by a nonterminal symbol. Analogously, when the mode is IO, then we can only apply productions to bottommost nonterminals, i.e., to those which are not a proper ancestor to a node labeled by a nonterminal symbol.

Formally, for every production p of a cftg $G = (N, \Sigma, \xi_0, P)$, we define the relations $\overset{\mathrm{OI}}{\Rightarrow}_p$ and $\overset{\mathrm{IO}}{\Rightarrow}_p$ on $\mathrm{T}(N \cup \Sigma)$ just like in the definition of $\Rightarrow_p$ at the beginning of this section, but we demand additionally that

- $\xi \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \overset{\mathrm{OI}}{\Rightarrow}_p \xi \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta]$ only if $\xi(w) \notin N$ for every position w that is a proper prefix of the unique position of $x_{\ell+1}$ in ξ,
- $\xi \cdot [\mathrm{Id}_\ell, A \cdot \zeta] \overset{\mathrm{IO}}{\Rightarrow}_p \xi \cdot [\mathrm{Id}_\ell, \varrho \cdot \zeta]$ only if there is no $w \in \mathrm{pos}(\zeta)$ such that $\zeta(w) \in N$.

Analogously to before, we let $\overset{\text{OI}}{\Rightarrow}_G = \bigcup_{p \in P} \overset{\text{OI}}{\Rightarrow}_p$ and $\overset{\text{IO}}{\Rightarrow}_G = \bigcup_{p \in P} \overset{\text{IO}}{\Rightarrow}_p$ and omit the subscript G when possible. For every $\xi \in \mathrm{T}(N \cup \Sigma)$, let

$$\mathcal{L}_{\text{OI}}(G, \xi) = \left\{ t \in \mathrm{T}(\Sigma) \;\middle|\; \xi \overset{\text{OI}}{\Rightarrow}{}^*_G\, t \right\} \qquad \text{and} \qquad \mathcal{L}_{\text{IO}}(G, \xi) = \left\{ t \in \mathrm{T}(\Sigma) \;\middle|\; \xi \overset{\text{IO}}{\Rightarrow}{}^*_G\, t \right\},$$

and let

$$\mathcal{L}_{\text{OI}}(G) = \mathcal{L}_{\text{OI}}(G, \xi_0) \qquad \text{and} \qquad \mathcal{L}_{\text{IO}}(G) = \mathcal{L}_{\text{IO}}(G, \xi_0).$$

It is well-known that the OI derivation mode comes with no restriction to the generative power of cftg, while there may be some generated trees which cannot be generated under IO derivation mode.

Theorem 2.13 (Fischer [60], Engelfriet and Schmidt [55]). *Let $G = (N, \Sigma, \xi_0, P)$ be a cftg.*

1. *For every ξ, $\zeta \in \mathrm{T}(N \cup \Sigma)$, if $\xi \Rightarrow^*_G \zeta$, then also $\xi \overset{\text{OI}}{\Rightarrow}{}^*_G\, \zeta$. In particular, $\mathcal{L}_{OI}(G) = \mathcal{L}(G)$.*
2. *In contrast, $\mathcal{L}_{IO}(G) \subseteq \mathcal{L}(G)$, and there is a cftg G' with $\mathcal{L}_{IO}(G') \subset \mathcal{L}(G')$.*

Item (1) can be shown using Lemma 2.12, which allows simulating a derivation which is not OI by one that is OI. For the counterexample in item (2) consider, e.g., the cftg G' with nonterminals $A^{(1)}$ and $B^{(0)}$, axiom $A(B)$, and $A \to \alpha$ as its only production.

In this work, we will only consider derivations in unrestricted and in OI mode, but not in the IO mode. OI derivations are important because they are more structured than unrestricted derivations. This is helpful in proofs, or when one wants to count the number of steps in a derivation. For instance, we can give the following technical lemma (called a *parallel derivation lemma* by Fischer), which allows the decomposition of a derivation (and is hence useful for proofs by induction). Compare Example 2.15 after the lemma for intuition.

Lemma 2.14 (Fischer [60, Thm. 4.1.1], Arnold and Leguy [18, Lem. 2]).
Let $G = (N, \Sigma, \xi_0, P)$ be a cftg, $n \in \mathbb{N}$, ξ, $\zeta \in \mathrm{T}(N \cup \Sigma)$, and $t \in \mathrm{T}(\Sigma)$. Then

$$\xi \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^n_G\, t$$

if and only if there are n_1, $n_2 \in \mathbb{N}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$, $\vartheta \in \Theta$, and $v \in \mathrm{T}(\Sigma)$ such that

$$t = \tilde{u} \cdot v, \qquad \xi \overset{\text{OI}}{\Rightarrow}{}^{n_1}_G\, \tilde{u} \cdot \vartheta, \qquad \vartheta \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^{n_2}_G\, v, \qquad \text{and} \qquad n_1 + n_2 = n.$$

Proof. The direction "if" of the equivalence is trivial, therefore we only prove the direction "only if". The proof is by induction on n.

For the induction base, assume that $n = 0$ and thus $\xi \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^0_G\, t$. Hence, $\xi \cdot \zeta = t$. Let $\mathrm{lin}(\xi) = (\tilde{u}, \vartheta)$ and $v = \vartheta \cdot \zeta$, then $\xi \overset{\text{OI}}{\Rightarrow}{}^0_G\, \tilde{u} \cdot \vartheta$ and $\vartheta \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^0_G\, v$. Further, $t = \xi \cdot \zeta = \tilde{u} \cdot \vartheta \cdot \zeta = \tilde{u} \cdot v$.

Assume that the property is already proven for $n \in \mathbb{N}$, and let $\xi \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^{n+1}_G\, t$. We can restrict ourselves to considering the two cases that ξ contains no nonterminal symbol, or that ξ contains an occurence of a nonterminal, and this occurence is rewritten first. This is due to Lemma 2.12*(i)*, which allows us to reorder productions that are applied at independent positions. So consider the following two cases.

(A) There is no occurrence of a nonterminal symbol in ξ.

Let in this case $\mathrm{lin}(\xi) = (\tilde{\xi}, \vartheta)$ and let $\hat{\zeta} = \vartheta \cdot \zeta$. Thus $\xi \cdot \zeta = \tilde{\xi} \cdot \hat{\zeta}$. Since $\tilde{\xi} \in \widetilde{\mathrm{T}}(\Sigma)$, the derivation's first production is applied somewhere in $\hat{\zeta}$. Formally, there is $\hat{\zeta}' \in \mathrm{T}(N \cup \Sigma)$ such that

$$\tilde{\xi} \cdot \hat{\zeta} \overset{\text{OI}}{\Rightarrow}_G \tilde{\xi} \cdot \hat{\zeta}' \overset{\text{OI}}{\Rightarrow}{}^n_G t\,.$$

By the induction hypothesis, there are $n_1, n_2 \in \mathbb{N}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$, $v \in \mathrm{T}(\Sigma)$ and $\tau \in \Theta$ such that

$$\tilde{\xi} \overset{\text{OI}}{\Rightarrow}{}^{n_1}_G \tilde{u} \cdot \tau\,, \qquad \tau \cdot \hat{\zeta}' \overset{\text{OI}}{\Rightarrow}{}^{n_2}_G v\,, \qquad n = n_1 + n_2 \qquad \text{and} \qquad t = \tilde{u} \cdot v\,.$$

In fact, as $\tilde{\xi} \in \widetilde{\mathrm{T}}(\Sigma)$, we have $n_1 = 0$, $\tilde{\xi} = \tilde{u}$, and $\tau = \mathrm{Id}_q$ for some $q \in \mathbb{N}$. Summarized,

$$\xi = \tilde{\xi} \cdot \vartheta \overset{\text{OI}}{\Rightarrow}{}^{n_1}_G \tilde{u} \cdot \tau \cdot \vartheta = \tilde{u} \cdot \vartheta\,, \qquad \vartheta \cdot \zeta = \hat{\zeta} \overset{\text{OI}}{\Rightarrow}_G \hat{\zeta}' = \tau \cdot \hat{\zeta}' \overset{\text{OI}}{\Rightarrow}{}^{n_2}_G v\,, \quad \text{and} \quad t = \tilde{u} \cdot v\,.$$

(B) There is an occurrence of a nonterminal symbol in ξ, and the derivation's first production is applied to this occurrence.

Thus there is $\xi' \in \mathrm{T}(N \cup \Sigma)$ such that

$$\xi \cdot \zeta \overset{\text{OI}}{\Rightarrow}_G \xi' \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^n_G t\,.$$

By the induction hypothesis, there are $n_1, n_2 \in \mathbb{N}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)$, $v \in \mathrm{T}(\Sigma)$, and $\vartheta \in \Theta$ such that

$$\xi' \overset{\text{OI}}{\Rightarrow}{}^{n_1}_G \tilde{u} \cdot \vartheta\,, \qquad \vartheta \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^{n_2}_G v\,, \qquad n = n_1 + n_2 \qquad \text{and} \qquad t = \tilde{u} \cdot v\,.$$

But clearly, then also

$$\xi \overset{\text{OI}}{\Rightarrow}_G \xi' \overset{\text{OI}}{\Rightarrow}{}^{n_1}_G \tilde{u} \cdot \vartheta\,,$$

and thus, the proof is concluded. □

Example 2.15. Consider the following example for direction "only if" of the above lemma. Assume a cftg G with nonterminal and terminal symbols from

$$N = \left\{A^{(2)}, B^{(0)}, C^{(0)}\right\}, \qquad \text{resp. from} \qquad \Sigma = \left\{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}\right\},$$

and with the productions

$$A(x_1, x_2) \to \sigma(x_2, x_2) \qquad \text{and} \qquad C \to \alpha + \beta\,.$$

Note there are no productions for B. Clearly, we have

$$A(B, C) \overset{\text{OI}}{\Rightarrow}{}^3_G \sigma(\alpha, \beta)\,.$$

When we consider the factorization $A(B, C) = \xi \cdot \zeta$ with $\xi = A$ and $\zeta = \langle 0; B, C\rangle$, we obtain that

$$A \cdot \mathrm{Id}_2 \overset{\text{OI}}{\Rightarrow}{}^1_G \sigma(x_2, x_2) = \sigma \cdot \vartheta\,, \qquad \text{where} \quad \vartheta = \langle 2; x_2, x_2\rangle\,.$$

Moreover,

$$\vartheta \cdot \zeta = \langle 0; C, C\rangle \overset{\text{OI}}{\Rightarrow}{}^2_G \langle 0; \alpha, \beta\rangle\,,$$

and $\sigma \cdot \langle 0; \alpha, \beta\rangle = \sigma(\alpha, \beta)$. ◁

Observe that if we did not restrict ourselves to OI derivations in Lemma 2.14, counting the number of steps in a derivation would become challenging. For example, given the two productions $A(x_1) \to \sigma(x_1, x_1)$ and $B \to \alpha$ in some cftg G, we have

$$A(B) \Rightarrow_G A(\alpha) \Rightarrow_G \sigma(\alpha, \alpha)\,,$$

but

$$A(x_1) \Rightarrow_G \sigma \cdot \langle 1; x_1, x_1\rangle \quad \text{and} \quad \langle 1; x_1, x_1\rangle \cdot B = \langle 0; B, B\rangle \Rightarrow^2_G \langle 0; \alpha, \alpha\rangle\,,$$

so the composed derivation at the top takes only two steps, while the decomposed one on the bottom takes three steps overall.

2.1.6 Linear Context-Free Tree Grammars

As linear cftg have a prominent role in this thesis, we recall some of their properties in this section. Moreover, we give an elementary proof of the fact that linearity is a proper restriction on the power of cftg.

Let us start by recalling the relationship between linear and nonlinear cftg. By definition, clearly

$$\mathsf{CFT}_{\ell n}(\Sigma) \subseteq \mathsf{CFT}_{\ell}(\Sigma) \subseteq \mathsf{CFT}(\Sigma)$$

for every ranked alphabet Σ. As the next theorem shows, the first two levels of this hierarchy coincide, in fact.

Theorem 2.16 (Leguy [109, Thm. III.8]). *For every ranked alphabet Σ, $\mathsf{CFT}_{\ell}(\Sigma) = \mathsf{CFT}_{\ell n}(\Sigma)$.*

Proof. The theorem's proof is by introducing, for every $k \in \mathbb{N}$ and nonterminal symbol $A \in N^{(k)}$ of a given l-cftg G, the nonterminals A_ϑ, for each linear torsion $\vartheta \in \Theta^k_\ell$ with $\ell \in [k]$. The productions of the constructed ln-cftg G'' are chosen such that for every $A \in N^{(k)}$, $\tilde{t} \in \widetilde{\mathrm{T}}(\Sigma)^1_k$, and every linear torsion $\vartheta \in \Theta^k_\ell$, we have

$$\tilde{t} \in \mathcal{L}(G', A_\vartheta) \qquad \text{if and only if} \qquad \tilde{t} \cdot \vartheta \in \mathcal{L}(G, A)\,.$$

Observe that this construction implies that the size of G' grows exponentially in the size of G. □

Linear Normal Form

The above theorem leads to a stronger normal form for l-cftg than the one for unrestricted cftg. Formally, we say that an l-cftg is in *linear normal form* if it has an initial nonterminal, and each of its productions is of either form

(i) $A \cdot \mathrm{Id}_n \to B \cdot (U_1 \otimes \cdots \otimes U_m)$
for some $n \in \mathbb{N}$, $m \in \mathbb{N}_1$, $A \in N^{(n)}$, $B \in N^{(m)}$, and $U_1, \ldots, U_m \in N \cup \Theta^1_1$ such that $\{U_1, \ldots, U_m\} \cap N \neq \emptyset$; or

(ii) $A \cdot \mathrm{Id}_n \to \sigma \cdot \mathrm{Id}_n$
for some $n \in \mathbb{N}$, $A \in N^{(n)}$ and $\sigma \in \Sigma^{(n)}$.

Observe that every l-cftg in linear normal form is linear and nondeleting. Moreover, each of its productions' right-hand sides is ordered.

Example 2.17. Consider ranked alphabets $N = \{A^{(3)}, B^{(3)}, C^{(2)}, D^{(0)}\}$ and $\Sigma = \{\sigma^{(2)}\}$. The productions

$$A(x_1, x_2, x_3) \to B(C(x_1, x_2), D, x_3) \qquad \text{and} \qquad C(x_1, x_2) \to \sigma(x_1, x_2)$$

are productions of an l-cftg in linear normal form. ◁

Theorem 2.18 (Stamer [156]). *For every $L \in \mathsf{CFT}_\ell$, there is an l-cftg G in linear normal form such that $\mathcal{L}(G) = L$.*

Proof. The theorem's proof is described thoroughly in [156, Lem. 3.2]. Note that there, a slightly different normal form, called *growing,* is obtained, using transformation rules called T_4, T_5, and T_6. However, it is easy to see that if we only apply the transformation rules T_4 and T_5, then we obtain an l-cftg in linear normal form. □

Remark 2.19. Observe that for every ln-cftg $G = (N, \Sigma, \xi_0, P)$ in linear normal form and every $A \in N^{(k)}$, $k \in \mathbb{N}$, we have $\mathcal{L}(G, A) \subseteq \widetilde{\mathrm{T}}(\Sigma)^1_k$ – only torsion-free trees are generated. ◁

Linear and Nonlinear Context-Free Tree Grammars

The next technical lemma shows that the size of every sentential form generated by cftg in linear normal form is polynomially bounded with respect to its height. In Theorem 2.22 we will use this lemma to give an elementary proof of the fact that linearity is a proper restriction on the power of cftg.

Lemma 2.20. *Let $G = (N, \Sigma, \xi_0, P)$ be an l-cftg in linear normal form such that* $\max \mathrm{rk}(N) = m$. *For every k, $\ell \in \mathbb{N}$, $A \in N^{(k)}$, $B \in N^{(\ell)}$ and $\xi_1, \ldots, \xi_\ell \in \mathrm{T}(N)^1_k$ such that $A \overset{\mathrm{OI}}{\Rightarrow}{}^*_G B(\xi_1, \ldots, \xi_\ell)$, we have for each $j \in [\ell]$ that*

$$|\xi_j| \le \binom{\mathrm{ht}(\xi_j) + m - 1}{\mathrm{ht}(\xi_j) - 1}.$$

Proof. The proof is by a combinatorial argument. We will analyze the derivation of a tree ξ with height $h + 1$, for some $h \in \mathbb{N}$, such that $|\xi|$ is maximal.

Since we are only interested in trees of maximal size, we can assume that $N^{(j)} \neq \emptyset$ for every $j \in [0, m]$, and that P is maximal, in the sense that P contains every possible production of an l-cftg in linear normal form with nonterminals from N and terminals from Σ. Note that there are only finitely many such productions. Note moreover that these assumptions come without loss of generality: by allowing more nonterminals and productions, the maximal size of the generated trees of height $h + 1$ may only rise or stay the same; it will never sink. So the established bound transfers to linear normal form l-cftg whose production sets are not maximal, or which miss nonterminals of some rank.

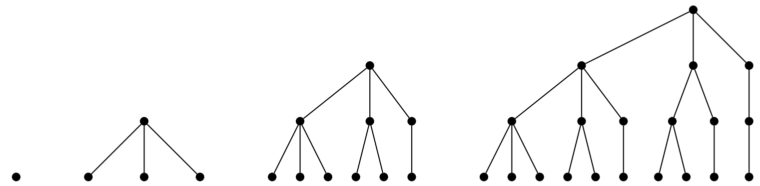


Figure 2.2: Maximal trees for $m = 3$ and $h = 1, \dots, 4$

As G is linear and nondeleting, and the order of the subtrees of a node is neither relevant to h nor to $|\xi|$, the derivation of a maximal tree ξ of height $h+1$ can then be written without loss of generality as

$$\begin{aligned} A_0 \cdot \mathrm{Id}_k &\overset{\text{oi}}{\Rightarrow}{}^*_G A_1 \cdot (\zeta_m \otimes \mathrm{Id}_{\ell_1}) \\ &\overset{\text{oi}}{\Rightarrow}{}^*_G A_2 \cdot (\zeta_m \otimes \zeta_{m-1} \otimes \mathrm{Id}_{\ell_2}) \\ &\vdots \\ &\overset{\text{oi}}{\Rightarrow}{}^*_G A_m \cdot (\zeta_m \otimes \zeta_{m-1} \otimes \cdots \otimes \zeta_1) \\ &= \xi \end{aligned}$$

for some $A_0, \dots, A_m \in N$, $\ell_1, \dots, \ell_{m-1} \in \mathbb{N}$, and trees $\zeta_1, \dots, \zeta_m \in \widetilde{\mathrm{T}}(N)^1$ which are all of height h, and each of which contains the maximal number of nodes.

Note that, in this respect, we can expect ζ_m to be larger than ζ_{m-1}. After all, we can use at most m parameters to build the respective subtrees of ζ_m, but then ζ_m must be stored in one parameter, and we can only use at most the remaining $m-1$ parameters to build the subtrees of ζ_{m-1}. This observation can be applied to every pair of trees ζ_{j+1} and ζ_j, for $j \in [m-1]$. In this situation, we will say that ζ_j is a tree *built using j parameters*, for each $j \in [m]$.

Let us use the above observation to determine the maximal size $M(n,h)$ of a tree ζ of height h built using n parameters. If $h = 1$, then certainly $M(n,h) = 1$. Assume that $h > 1$. Then we build a maximal tree of height h using n parameters as follows:

- We build a maximal subtree of height $h-1$ using n parameters.
- We build a maximal subtree of height $h-1$ using $n-1$ parameters.

  $\vdots$
- We build a maximal subtree of height $h-1$ using 1 parameter.

As all these trees are subtrees of the root node, the result is a maximal tree of height h built using n parameters. Therefore, we obtain the recurrence

$$M(n,h) = \begin{cases} 1 & \text{if } h = 1 \\ 1 + \sum_{j=1}^{n} M(j, h-1) & \text{otherwise.} \end{cases}$$

Consider Figure 2.2 for examples of maximal trees of height 1, ..., 4, using 3 parameters, constructed by the above method. As the trees' labels are irrelevant in this context, they have been omitted.

In the following, we will prove by induction on h that for every $n \in \mathbb{N}$,

$$M(n,h) = \binom{h+n-1}{h-1}.$$

If $h = 1$, then $M(n,h) = 1$ by definition, and obviously,

$$\binom{h+n-1}{h-1} = \frac{n!}{n!} = 1\,.$$

Otherwise, assume that $h > 1$, and the proposition has already been proven for $h-1$. Then we can show, using the identity (1.1) of Pascal's triangle, that

$$\begin{aligned}
\binom{h+n-1}{h-1} &= \binom{h+n-2}{h-2} + \binom{h+n-2}{h-1} \\
&= \binom{h+n-2}{h-2} + \binom{h+n-3}{h-2} + \binom{h+n-3}{h-1} \\
&\;\vdots \\
&= \binom{h+n-2}{h-2} + \binom{h+n-3}{h-2} + \cdots + \binom{h+n-(n+2)}{h-2} + \binom{h+n-(n+2)}{h-1} \\
&= \sum_{j=0}^{n} \binom{h+j-2}{h-2} && (2.1) \\
&= 1 + \sum_{j=1}^{n} M(j,h-1) && (2.2) \\
&= M(n,h)\,.
\end{aligned}$$

The identity (2.1) holds because the last summand from the line above reduces to zero. Equation (2.2) is valid since $\binom{h-2}{h-2} = 1$, and because of the induction hypothesis.

* * *

Now assume that $A \overset{\text{OI}}{\Rightarrow}{}^{*}_{G} B(\xi_1, \ldots, \xi_\ell)$ as stated in the lemma. Then we obtain for every $j \in [\ell]$ that

$$|\xi_j| \le M\big(m, \mathrm{ht}(\xi_j)\big) = \binom{\mathrm{ht}(\xi_j)+m-1}{\mathrm{ht}(\xi_j)-1},$$

and this concludes the lemma's proof. □

Remark 2.21. One can establish a bound for the size of trees generated by linear coregular cftg in a similar manner to Lemma 2.20. ◁

Theorem 2.22 (Leguy [109, Prop. IV.47]). *For every Σ with $\Sigma^{(0)} \neq \emptyset$ and $\Sigma \neq \Sigma^{(0)} \cup \Sigma^{(1)}$, there is a tree language $L \in \mathsf{CFT}(\Sigma) \setminus \mathsf{CFT}_\ell(\Sigma)$.*

Proof. The property has originally been proven by Leguy. In the following, we give a witness for L, together with an elementary proof based on a growth argument.

We will first prove the lemma for the ranked alphabet $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$, and generalize to arbitrary ranked alphabets later. Recall for this purpose the cftg G_P from Example 2.5, and let $L_P = \mathcal{L}(G_P)$ – the language of all perfect binary trees over $\{\sigma^{(2)}, \alpha^{(0)}\}$.

Let us assume that there is an ln-cftg $G = (N, \Sigma, S, P)$ in linear normal form such that $\mathcal{L}(G) = L_P$. We will lead this assumption to contradiction.

Consider an OI derivation $S \overset{\text{OI}}{\Rightarrow}{}^*_G t$ for some $t \in L_P \setminus \{\alpha\}$. Then

$$S \overset{\text{OI}}{\Rightarrow}{}^*_G Z(\xi,\zeta) \overset{\text{OI}}{\Rightarrow}_G \sigma(\xi,\zeta) \overset{\text{OI}}{\Rightarrow}{}^*_G \sigma(t',t') \tag{2.3}$$

for some $Z \in N^{(2)}$, ξ, $\zeta \in \mathrm{T}(N)^1_0$, and $t' \in L_P$. Clearly, we have $\mathcal{L}(G,\xi) = \mathcal{L}(G,\zeta) = \{t'\}$, as otherwise we could derive a tree outside of L_P.

We can give the following two bounds for $|t'|$ depending on $|\xi|$.[9] The number $\max \mathrm{rk}(N)$ will be abbreviated by m.

(A) $|t'| \in \Omega\Big(2^{\sqrt[m]{|\xi|}}\Big)$.

By Lemma 2.20,

$$\begin{aligned} |\xi| &\leq \binom{\mathrm{ht}(\xi)+m-1}{\mathrm{ht}(\xi)-1} \\ &= \frac{(\mathrm{ht}(\xi)+m-1)!}{(\mathrm{ht}(\xi)-1)! \cdot m!} \\ &= \frac{(\mathrm{ht}(\xi)+m-1)\cdot(\mathrm{ht}(\xi)+m-2)\cdots(\mathrm{ht}(\xi)+m-m)}{m!}, \end{aligned}$$

and therefore $|\xi| \in \mathcal{O}(\mathrm{ht}(\xi)^m)$. Dually,

$$\mathrm{ht}(\xi) \in \Omega(\sqrt[m]{|\xi|}).$$

As G contains no productions of form $A \to x_i$, we have $\mathrm{ht}(\xi) \leq \mathrm{ht}(t')$. Therefore $\mathrm{ht}(\xi) \in \mathcal{O}(\mathrm{ht}(t'))$, and hence $\mathrm{ht}(t') \in \Omega(\mathrm{ht}(\xi))$. Moreover, by the well-known property of perfect binary trees, $|t'| = 2^{\mathrm{ht}(t')} - 1$. We obtain

$$|t'| \in \Omega\big(2^{\mathrm{ht}(t')}\big) \subseteq \Omega\big(2^{\mathrm{ht}(\xi)}\big) \subseteq \Omega\Big(2^{\sqrt[m]{|\xi|}}\Big).$$

(B) $|t'| \in \mathcal{O}\big(|\xi|\big)$.

Let M be the set of all $A \in N$ such that $\mathcal{L}(G,A)$ is finite. Clearly, for every nonterminal A that occurs in ξ, we have that $A \in M$, as otherwise the property $\mathcal{L}(G,\xi) = \{t'\}$ would be violated (observe that G is nondeleting). Let

$$\mu = \max\Big\{ |s| \;\Big|\; s \in \mathcal{L}(G,A), A \in M \Big\}.$$

[9] Analogous bounds can be given depending on $|\zeta|$, but considering ξ suffices for our purposes.

Then $|t'| \leq \mu \cdot |\xi|$. As μ depends only on G, we obtain the bound $|t'| \in \mathcal{O}(|\xi|)$.

* * *

But clearly, the conjunction of **(A)** and **(B)** results in a contradiction, since, for every $m > 0$, the function $n \mapsto 2^{\sqrt[m]{n}}$ grows faster than $n \mapsto n$. To see this, differentiate both functions with respect to n. So the assumed ln-cftg G cannot generate the tree language L_P, and by this contradiction, L_P is not a linear context-free tree language.

* * *

It remains to show that the claim holds for other ranked alphabets than Σ. For this purpose, assume a ranked alphabet Δ such that $\Delta^{(0)}$ and $\Delta^{(k)}$ are nonempty, for some $k > 1$. Choose some symbols $\delta \in \Delta^{(k)}$ and $\beta \in \Delta^{(0)}$, and denote by L_P^k the language of all perfect k-ary trees over $\{\delta, \beta\}$. It is easy to see that L_P^k is context-free, by a straightforward modification of the cftg G_P from Example 2.5.

Consider the linear tree homomorphism $h\colon \mathrm{T}_{\{\delta,\beta\}}(X) \to \mathrm{T}_\Sigma(X)$ given by

$$h\colon \delta \mapsto \sigma(x_1, x_2)\,, \qquad \beta \mapsto \alpha\,.$$

It is easy to see that $h(L_P^k) = L_P$. Assume that $L_P^k \in \mathsf{CFT}_\ell(\Delta)$. Since the class CFT_ℓ is closed under linear tree homomorphisms (see Theorem 2.34 further below), this implies that also $L_P \in \mathsf{CFT}_\ell(\Sigma)$, in contradiction to the above. So $L_P^k \in \mathsf{CFT}(\Delta) \setminus \mathsf{CFT}_\ell(\Delta)$. $\square$

2.2 Pushdown Tree Automata

Next, we recall the definition of (restricted) pushdown tree automata from [79]. Compare Remark 2.23 below for a note on nomenclature.

A *pushdown tree system (pts)* is a tuple $M = (Q, \Sigma, \Gamma, q_0, R)$, where

- Q is a ranked alphabet (its elements called *states*) such that $Q = Q^{(1)}$,
- Σ is a ranked alphabet disjoint from Q,
- Γ is a nonempty set disjoint from Q and Σ (its elements called *pushdown symbols*),
- $q_0 \in Q$ (the *initial state*), and
- R is a set (its elements called *rules*), where each rule is of the form

$$q(ux) \to \varrho \tag{2.4}$$

  for some $q \in Q$, $u \in \Gamma \cup \{\varepsilon\}$, and $\varrho \in \mathrm{T}_\Sigma(Q(\Gamma^* X_1))$.

If Γ and R are finite, then we call the pts M from above a *pushdown tree automaton (pta)*.

Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pts, and let $r \in R$ be a rule of form $q(ux) \to \varrho$ as in (2.4). The *rewrite relation by* r is denoted by $\Rightarrow_r$ and defined to be the smallest relation on $\mathrm{T}_\Sigma(Q(\Gamma^*))$ such that for every $\xi \in \mathrm{T}_\Sigma(Q(\Gamma^*) \cup X_1)$ that contains x precisely once, and every $\eta \in \Gamma^*$, we have

$$\xi\big[x/q(u\eta)\big] \Rightarrow_r \xi\big[x/\varrho[x/\eta]\big].$$

Here, the subterm $\varrho[x/\eta]$ is to be understood as an instance of tree substitution, due to the isomorphism between words and monadic trees.

In the situation above, we say that *the rule r was applied at position w*, denoted by $\overset{w}{\Rightarrow}_r$, where w is the unique position in ξ that is labeled with x. Moreover, the *rewrite relation of M*, denoted by $\Rightarrow_M$, is $\Rightarrow_M = \bigcup_{r \in R} \Rightarrow_r$. Finally, the *tree language accepted by M* is defined as

$$\mathcal{L}(M) = \left\{t \in \mathrm{T}_\Sigma \;\middle|\; q_0(\varepsilon) \Rightarrow_M^* t\right\}.$$

Again, we require a measure of size. Let for this purpose $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pta. For every number $\ell \in \mathbb{N}$, every tree $\tilde{t} \in \widetilde{\mathrm{T}}(\Sigma)^1_\ell$, and every $q_1(\eta_1), \ldots, q_\ell(\eta_\ell) \in Q(\Gamma^*)$, let

$$\big\|\tilde{t}[q_1(\eta_1), \ldots, q_\ell(\eta_\ell)]\big\| = |\tilde{t}| + \sum_{i \in [\ell]} \big(1 + |\eta_i|\big).$$

The *size* of M, denoted by $|M|$, is then defined by

$$|M| = |Q| + |\Gamma| + \sum_{(l \to r) \in R} \big(\|l\| + \|r\|\big).$$

Remark 2.23. The notion of pushdown tree system will be used in some proofs in Chapter 3. Apart from this, we will only be concerned with pushdown tree automata.

Pushdown tree automata have been introduced by Guessarian [79]. However, her definition of the general model uses *tree pushdowns* instead of pushdown words. Our notion of pta corresponds therefore (aside from syntactic differences) to the *restricted pushdown tree automata* of Guessarian.

Note that, in contrast to fta, we have chosen to present pta in a term-rewriting style instead of giving a transition table. Thus, it would be more correct to speak of *regular tree grammars with pushdown storage* (an instance of the concept of *grammars with storage* [51, 59]). However, our nomenclature coincides with Guessarian's, who also defined pta in term-rewriting style. The given notation is slightly more economical, since Guessarian chose to denote pta by transducers which compute a partial identity (cf. also Chapter 5 for further remarks). ◁

The following lemma allows us to interchange the order of rules that are applied in a derivation to independent nodes.

Lemma 2.24. *Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pts, let r_1, $r_2 \in R$, and let ξ, ξ_1, $\zeta \in \mathrm{T}_\Sigma(Q(\Gamma^*))$, $w_1 \in \mathrm{pos}(\xi)$, and $w_2 \in \mathrm{pos}(\xi_1)$ such that*

$$\xi \overset{w_1}{\Rightarrow}_{r_1} \xi_1 \overset{w_2}{\Rightarrow}_{r_2} \zeta .$$

If $w_1 \parallel w_2$, then there is some $\xi_2 \in \mathrm{T}_\Sigma(Q(\Gamma^*))$ such that

$$\xi \Rightarrow_{r_2} \xi_2 \Rightarrow_{r_1} \zeta .$$

Proof. Analogous to the proof of Lemma 2.12*(i)*. □

Corollary 2.25. *Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pts. It is no restriction to only consider derivations of the form*

$$\xi_0 \Rightarrow_{r_1} \xi_1 \Rightarrow_{r_2} \cdots \Rightarrow_{r_n} \xi_n ,$$

with $n \in \mathbb{N}$, $r_1, \ldots, r_n \in R$, and $\xi_0, \ldots, \xi_n \in \mathrm{T}_\Sigma(Q(\Gamma^*))$, and such that for each $i \in [n]$, r_i is applied at the smallest position $w \in \mathrm{pos}(\xi_{i-1})$ with respect to $\leq_{lex}$ that is labeled by an element of $Q(\Gamma^*)$.

Derivations of this form will be called *leftmost derivations.* A pts $M = (Q, \Sigma, \Gamma, q_0, R)$ is said to be in *normal form* if each of its rules is of either form

(i) $q(x) \to \sigma\big(p_1(x), \ldots, p_k(x)\big)$,

(ii) $q(x) \to p(\gamma x)$, or

(iii) $q(\gamma x) \to p(x)$

for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, q, p, $p_1, \ldots, p_k \in Q$, and $\gamma \in \Gamma$. A rule of type *(i)* will be called a *copy rule*, one of type *(ii)* a *push rule,* while rules of type *(iii)* are called *pop rules.*

Lemma 2.26. *For every pta M, there is a pta M' in normal form such that $\mathcal{L}(M) = \mathcal{L}(M')$. Moreover, the construction of M' is computable in logarithmic space.*

Proof. As the normal form is fairly well-known from indexed grammars [86, Sec. 14.3], we only sketch the construction, in order to show that it can be implemented in logarithmic space.

Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pta, and consider a rule r of the form $q(ux) \to \varrho$, as given in (2.4). We will simulate r by a pop rule, which consumes u if $u \neq \varepsilon$, a sequence of copy rules, which read the tree on the right-hand side symbol by symbol, and then a sequence of push rules, which are responsible for pushing new symbols onto the pushdowns successively. As rules of the form $q(x) \to p(x)$ are not allowed in the normal form, we introduce a dummy pushdown symbol E.

Formally, construct the pta $M = (Q', \Sigma, \Gamma', q_0, R')$, where $\Gamma' = \Gamma \cup \{E\}$ for some distinct symbol E, and Q' contains

- all states from Q,
- the state r_w, for every rule $r \in R$ of the form $q(ux) \to \varrho$ and every $w \in \mathrm{pos}_{Q(\Gamma^* X_1) \cup \Sigma}(\varrho)$,
- the state q_v, for every state $q \in Q$, and every word $v \in \Gamma^*$ that occurs as prefix of a pushdown in a rule of R.

Moreover, R' contains the following rules.

- For every state $q \in Q'$, R' contains the rules $q(x) \to q(Ex)$ and $q(Ex) \to q(x)$.
- For every rule r of the form $q(ux) \to \varrho$, R' contains
  - the rule $q(u'x) \to r_\varepsilon(x)$, where $u' = u$ if $u \in \Gamma$ and $u' = E$ otherwise,
  - for every $w \in \mathrm{pos}_\Sigma(\varrho)$, the rule
    $$r_w(x) \to \sigma\big(r_{w1}(x), \dots, r_{wk}(x)\big),$$
    where $\sigma = \varrho(w)$ and $k = \mathrm{rk}(\sigma)$, and
  - for every $w \in \mathrm{pos}_{Q(\Gamma^* X_1)}(\varrho)$, the rule
    $$r_w(x) \to q_v(Ex),$$
    where $\varrho(w) = q(vx)$.
- For every state of the form $q_v \in Q'$, R' contains
  - the rule $q_\varepsilon(x) \to q(Ex)$, and
  - if $v = v'\gamma$ for some $v' \in \Gamma^*$ and $\gamma \in \Gamma$, the rule $q_v(x) \to q_{v'}(\gamma x)$.

It is easy to see that M' can be constructed from M using a constant number of loops, employing binary counters which range over the length of the representation of M. Therefore, following Remark 1.16, the construction of M' is logspace-computable. We omit the straightforward proof of equivalence. □

The following theorem generalizes the relationship between cfg and pushdown automata. Again, the theorem is well-known. We restate the underlying construction for two reasons – in order to show that it can be performed in logarithmic space, and because it has an interesting connection to the magmoid notation.

Theorem 2.27 (Guessarian [79]). *Let $L \subseteq \mathrm{T}_\Sigma$ be a tree language. The following are equivalent:*

1. *There is a cftg G such that $\mathcal{L}(G) = L$.*
2. *There is a pta M such that $\mathcal{L}(M) = L$.*

The respective constructions are logspace-computable.

Proof. We begin with the implication (1) ⇒ (2). The construction has the following intuition. Given a cftg G in normal form, we have to simulate G by a pushdown tree automaton M. For every nonterminal A of G, M contains a state q_A. The pushdown symbols of M are *tuples* of nonterminals from G. For an example, consider the derivation

$$A \Rightarrow_G \begin{array}{c} B \\ \diagup \quad \diagdown \\ C_1 \;\cdots\; C_m \end{array} \Rightarrow_G \begin{array}{c} D \\ \diagup \qquad\qquad \diagdown \\ \begin{array}{c} E_1 \\ \diagup \; \diagdown \\ C_1 \cdots C_m \end{array} \cdots \begin{array}{c} E_n \\ \diagup \; \diagdown \\ C_1 \cdots C_m \end{array} \end{array}$$

in G. We simulate this derivation in M by

$$q_A(\varepsilon) \Rightarrow_M q_B\big([C_1,\ldots,C_m]\big) \Rightarrow_M q_D\big([E_1,\ldots,E_n][C_1,\ldots,C_m]\big).$$

Now assume that the derivation in G continues

$$\begin{array}{c} D \\ \diagup \qquad\qquad \diagdown \\ \begin{array}{c} E_1 \\ \diagup \; \diagdown \\ C_1 \cdots C_m \end{array} \cdots \begin{array}{c} E_n \\ \diagup \; \diagdown \\ C_1 \cdots C_m \end{array} \end{array} \Rightarrow_G \begin{array}{c} E_i \\ \diagup \quad \diagdown \\ C_1 \;\cdots\; C_m \end{array},$$

by the collapsing production $D \to x_i$ of G, where $i \in [n]$. In order to simulate this derivation in M, we have to extract the i-th nonterminal within the symbol on the pushdown's top. For this purpose, M contains the special states $p_1, \ldots, p_\ell$, for some $\ell \in \mathbb{N}$. The corresponding derivation in M is then

$$q_D\big([E_1,\ldots,E_n][C_1,\ldots,C_m]\big) \Rightarrow_M p_i\big([E_1,\ldots,E_n][C_1,\ldots,C_m]\big) \Rightarrow_M q_{E_i}\big([C_1,\ldots,C_m]\big).$$

Terminal productions of G are simulated similarly. This construction is essentially the one presented in [59, Lem. 5.6].

* * *

For the construction's formal definition, assume a cftg $G = (N, \Sigma, S, P)$, chosen without loss of generality to be in normal form and with an initial nonterminal S. We construct a pta $M = (Q, \Sigma, \Gamma, q_S, R)$, where

$$Q = \big\{q_A \;\big|\; A \in N\big\} \cup \big\{p_i \;\big|\; i \in [\max \mathrm{rk}(N)]\big\},$$

and

$$\Gamma = \big\{\zeta \in \mathrm{T}(N) \;\big|\; \text{there is some nonterminal production } A \to B \cdot \zeta \text{ in } P\big\}\,,$$

while the set R is built as follows. For every nonterminal production of form $A \to B \cdot \zeta$ in P, where $\zeta \in \mathrm{T}(N)$, R contains the rule

$$q_A(x) \to q_B(\zeta x)\,.$$

For every terminal production $A \to \sigma \cdot \vartheta$ in P, where $\mathrm{rk}(\sigma) = k$, R contains the rule

$$q_A(x) \to \sigma\big(p_{\vartheta(1)}(x), \ldots, p_{\vartheta(k)}(x)\big)\,.$$

For every collapsing production $A \to x_i$ in P, R contains the rule

$$q_A(x) \to p_i(x)\,.$$

Finally, for every symbol $\zeta \in \Gamma$, and every $i \in \mathbb{N}$ such that $\pi_i \cdot \zeta$ is defined, R contains the rule

$$p_i(\zeta x) \to q_{\pi_i \cdot \zeta}(x)\,.$$

Following the reasoning of Remark 1.16, we see that M can be constructed from G in logarithmic space.

As the result is well-known, we will not prove the construction's correctness. Let us just remark that the proof is based on showing for every $k, \ell \in \mathbb{N}$, $A \in N^{(k)}$, $\tilde{t} \in \mathrm{T}(\Sigma)^1_\ell$, and $\vartheta \in \Theta^\ell_k$, that

$$A \overset{\text{OI}}{\Rightarrow}{}^*_G \tilde{t} \cdot \vartheta \qquad \text{if and only if} \qquad q_A(\varepsilon) \Rightarrow^*_M \tilde{t} \cdot \big[p_{\vartheta(1)}(\varepsilon), \ldots, p_{\vartheta(\ell)}(\varepsilon)\big]\,.$$

* * *

Let us continue with the other direction (2) ⇒ (1) of the proof. The construction's idea has been given by Rounds, when he proved that creative dendrogrammars can be assumed without loss of generality to possess only one state [140, Thm. 7]. We will give a short example after the definition.

Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pta in normal form. Without loss of generality, we assume that $Q = \{1, \ldots, n\}$ for some $n \in \mathbb{N}$. Let $\Gamma' = \Gamma \cup \{\gamma_0\}$ for some distinct symbol γ_0. We construct the cftg $G = (N, \Sigma, \xi_0, P)$, where

$$N = N^{(n)} \cup \{Z^{(0)}\}\,, \qquad N^{(n)} = \big\{\gamma^q \;\big|\; \gamma \in \Gamma', q \in Q\big\}\,, \qquad \xi_0 = \gamma_0^{q_0}(Z, \ldots, Z)\,,$$

and P is defined as follows. For every push rule $q(x) \to p(\gamma x)$ in R, and every $\delta \in \Gamma'$, we insert into P the production

$$\delta^q \cdot \mathrm{Id}_n \to \gamma^p(\delta^1 \cdot \mathrm{Id}_n, \ldots, \delta^n \cdot \mathrm{Id}_n)\,.$$

For every pop rule $q(\gamma x) \to p(x)$ in R, we add to P the production

$$\gamma^q \cdot \mathrm{Id}_n \to x_p\,.$$

Finally, for every copy rule $q(x) \to \sigma(p_1(x), \ldots, p_k(x))$ in R, and every $\gamma \in \Gamma'$, let P contain the production

$$\gamma^q \cdot \mathrm{Id}_n \to \sigma(\gamma^{p_1} \cdot \mathrm{Id}_n, \ldots, \gamma^{p_k} \cdot \mathrm{Id}_n).$$

Observe that there is neither a production for the nonterminal Z, nor is there a production of form $\gamma_0^q \cdot \mathrm{Id}_n \to x_i$, for any $q \in Q$ and $i \in [n]$. It is easy to see that G is logspace-computable from M.

Again, we omit the proof of correctness, which rests upon the property that

$$q(\gamma) \Rightarrow_M^* \tilde{t} \cdot \big[\vartheta(1)(\varepsilon), \ldots, \vartheta(\ell)(\varepsilon)\big] \qquad \text{if and only if} \qquad \gamma^q \overset{\text{OI}}{\Rightarrow}{}_G^* \tilde{t} \cdot \vartheta$$

for every $q \in Q$, $\gamma \in \Gamma'$, $\ell \in \mathbb{N}$, $\tilde{t} \in \widetilde{\mathrm{T}}(\Sigma)_\ell^1$, and $\vartheta \in \Theta_n^\ell$.

* * *

Consider the following example. Assume a pta M with the two states q and p, the single pushdown symbol δ, and the rules

$$q(x) \to p(\delta x), \qquad p(x) \to \sigma\big(q(x), p(x)\big), \qquad q(\delta x) \to \alpha, \qquad \text{and} \qquad p(\delta x) \to \beta.$$

Moreover, consider the derivation

$$q(\varepsilon) \Rightarrow_M p(\delta) \Rightarrow_M \sigma\big(q(\delta), p(\delta)\big) \Rightarrow_M \sigma\big(\alpha, p(\delta)\big) \Rightarrow_M \sigma(\alpha, \beta).$$

in M. We construct a cftg G with nonterminals δ^q, δ^p, γ_0^q, γ_0^p, and Z. All nonterminals but Z are of rank 2, and Z has rank 0.

Consider $z \in \{q, p\}$ and $\gamma \in \{\delta, \gamma_0\}$. The nonterminal γ^z simulates a configuration of M in state z, and with the symbol γ on top of the pushdown. The first subtree of γ^z is used to encode the behavior of M when it pops γ and ends up in state q. Analogously, the second subtree encodes the behavior of M when γ is popped and M ends up in state p. We use the symbol γ_0 to represent the bottom of the pushdown, and Z due to technical convenience. The cftg G contains, among others, the productions

$$\gamma_0^q \cdot \mathrm{Id}_2 \to \delta^p\big(\gamma_0^q(x_1, x_2), \gamma_0^p(x_1, x_2)\big), \qquad \delta^p \cdot \mathrm{Id}_2 \to \sigma\big(\delta^q(x_1, x_2), \delta^p(x_1, x_2)\big),$$

as well as $\delta^q \cdot \mathrm{Id}_2 \to \alpha$ and $\delta^p \cdot \mathrm{Id}_2 \to \beta$. We can then perform the derivation

$$\gamma_0^q(Z, Z) \Rightarrow_G \delta^p\big(\gamma_0^q(Z, Z), \gamma_0^p(Z, Z)\big) \Rightarrow_G \sigma\Big(\delta^q\big(\gamma_0^q(Z, Z), \gamma_0^p(Z, Z)\big), \delta^p\big(\gamma_0^q(Z, Z), \gamma_0^p(Z, Z)\big)\Big) \Rightarrow_G^* \sigma(\alpha, \beta)$$

in G, simulating the one of M from above. □

2.3 Yield and Path Languages

Similarly to the relationship between the recognizable tree languages and the context-free languages stated in Theorem 1.31, the following theorem shows us the correspondence between the context-free tree languages and the indexed languages.

Theorem 2.28 (Rounds [140, p. 286]). *Let $L \subseteq \Sigma^*$ for some alphabet Σ. The following are equivalent:*

1. *There are a terminal ranked alphabet Δ and a cftg G over Δ such that $\Sigma \subseteq \Delta^{(0)}$ and $L = \mathrm{yd}_\Sigma(\mathcal{L}(G))$.*
2. *There is an indexed grammar G' such that $L = \mathcal{L}(G')$.*

The respective constructions are logspace-computable. The theorem holds in particular if $\Sigma = \Delta^{(0)}$ and the ixg in item (2) is demanded to be ε-free.

Proof. Recall from Lemma 2.26 and Theorem 2.27 that item *(1)* above is equivalent to the existence of some pta M in normal form over the specified ranked alphabet Δ such that $L = \mathrm{yd}_\Sigma(\mathcal{L}(M))$.

The theorem's validity can then be proven as follows. Consider a pta $M = (Q, \Delta, \Gamma, q_0, R)$ in normal form and an ixg $G = (N, \Sigma, \Omega, S, P)$ in normal form. We say that M and G are *related* if $Q = N$, $\Gamma = \Omega$, $q_0 = S$, and for every q, $p \in Q$, $\gamma \in \Gamma$, and $\alpha \in \Delta^{(0)}$, we have

- the production $q^\gamma \to p$ is in P if and only if the rule $q(\gamma x) \to p(x)$ is in R,
- the production $q \to p^\gamma$ is in P if and only if the rule $q(x) \to p(\gamma x)$ is in R,
- for every $k \in \mathbb{N}_1$ and $p_1, \ldots, p_k \in Q$, the production $q \to p_1 \cdots p_k$ is in P if and only if there is some $\delta \in \Delta^{(k)}$ such that R contains the rule $q(x) \to \delta(p_1(x), \ldots, p_k(x))$, and
- for every $a \in \Sigma \cup \{\varepsilon\}$, the production $q \to a$ is in P if and only if there is some $\alpha \in \Delta^{(0)}$ such that $\mathrm{yd}_\Sigma(\alpha) = a$ and the rule $q(x) \to \alpha$ is in R.

Let M and G be related, as above. By a straightforward induction argument, one can show for every $n \in \mathbb{N}$, $q \in Q$, $\eta \in \Gamma^*$, and $w \in \Sigma^*$, that

$$q^\eta \Rightarrow_G^n w \qquad \text{if and only if} \qquad \exists t \in \mathrm{T}_\Delta : q(\eta) \Rightarrow_M^n t \;\wedge\; \mathrm{yd}(t) = w\,.$$

Therefore, $\mathcal{L}(G) = \mathrm{yd}(\mathcal{L}(M))$. By reading the above definition as a construction, it is moreover easy to obtain from a given pta M a related ixg G, and vice versa, in logarithmic space. Moreover, if $\Sigma = \Delta^{(0)}$, then G contains no productions with right-hand side ε; and if G is ε-free, then we can choose Δ such that $\Sigma = \Delta^{(0)}$. This concludes the theorem's proof. □

In analogy to Theorem 1.32, the path languages of context-free tree languages are context-free. We reprove the theorem to substantiate the given resource bound, which has not been stated explicitly.

Theorem 2.29 (Rounds [141, p. 115]). *Let Σ be a ranked alphabet. For every cftg G over Σ, there is a cfg $\widehat{G}$ with $\mathcal{L}(\widehat{G}) = \mathrm{P}(\mathcal{L}(G))$. Moreover, $\widehat{G}$ can be constructed from G in time exponential in the size of G for arbitrary alphabets Σ, and even in space logarithmic in the size of G if Σ is monadic.*

Proof. Assume a cftg $G = (N, \Sigma, S, P)$ in normal form. Depending on Σ, we proceed as follows.

1. If Σ is not monadic, then we will remove all useless productions from P in the first part of the construction. A production $A \cdot \mathrm{Id}_n \to \varrho$ is said to be *useless* if $\mathcal{L}(G, \varrho) = \emptyset$. As noted by Rounds, it can be decided whether $\mathcal{L}(G, \varrho) = \emptyset$ using an algorithm by Aho [3, Alg. 1] (cf. also Theorem 2.37 and Chapter 3), and it is easy to see that this algorithm can be executed in time exponential in the size of G. Let P' be the set of all productions from P which are not useless. Clearly, the cftg $G' = (N, \Sigma, S, P')$ satisfies $\mathcal{L}(G') = \mathcal{L}(G)$.
2. If Σ is monadic, then let $P' = P$.

We construct the context-free grammar $\widehat{G} = (\widehat{N}, \widehat{\Sigma}, \langle S, 0\rangle, \widehat{P})$, where $\widehat{N}$ is the path alphabet associated to N as defined in Section 1.3.1, and $\widehat{P}$ is the smallest set that contains the following productions.

(i) For every production
$$A \cdot \mathrm{Id}_n \to B\big(C_1 \cdot \mathrm{Id}_n, \dots, C_m \cdot \mathrm{Id}_n\big)$$
in P', $\widehat{P}$ contains the productions
$$\langle A, 0\rangle \to \langle B, 0\rangle + \sum_{j \in [m]} \langle B, j\rangle \langle C_j, 0\rangle\,,$$
as well as
$$\langle A, i\rangle \to \sum_{j \in [m]} \langle B, j\rangle \langle C_j, i\rangle$$
for every $i \in [n]$.

(ii) For every production $A \cdot \mathrm{Id}_n \to x_i$ in P', $\widehat{P}$ contains the production $\langle A, i\rangle \to \varepsilon$.

(iii) For every terminal production $A \cdot \mathrm{Id}_n \to \alpha$ in P', where $\alpha \in \Sigma^{(0)}$, $\widehat{P}$ contains the production $\langle A, 0\rangle \to \langle \alpha, 0\rangle$.

(iv) Finally, for every terminal production $A \cdot \mathrm{Id}_n \to \sigma \cdot \vartheta$ in P', where $\sigma \in \Sigma^{(k)}$ for some $k > 0$ and $\vartheta \in \Theta_n^k$, $\widehat{P}$ contains the production $\langle A, \vartheta(j)\rangle \to \langle \sigma, j\rangle$ for every $j \in [k]$.

Now we see why removing useless productions was necessary for non-monadic Σ. For example, for a cftg G with axiom S and the only productions
$$S \to A(\alpha, B)\,, \qquad A(x_1, x_2) \to \sigma(x_1, x_2)\,,$$
the first of which is useless, we have $\mathcal{L}(G) = \emptyset$, but $\mathrm{P}(\mathcal{L}(\widehat{G})) = \{\langle \sigma, 1\rangle \langle \alpha, 0\rangle\}$. Intuitively, the paths are generated independently in $\widehat{G}$, while in G they must be derived simultaneously, and the derivation fails if some path of a tree cannot be derived.

Since we want to show where the assumptions on P' become important, we give a detailed proof of correctness. We will use the following facts.

(A) *Let $n \in \mathbb{N}$, $m \in \mathbb{N}_1$, $i \in [0,n]$, and $u \in \mathrm{T}(\Sigma)^1_m$, $v \in \mathrm{T}(\Sigma)^m_n$. Consider some $w \in \widehat{\Sigma}^*$. Then $w \in \mathrm{P}^n_i(u \cdot v)$ if and only if one of the following holds.*

- *We have $i = 0$, and $w \in \mathrm{P}^m_0(u)$.*
- *There are $j \in [m]$, $w_1 \in \mathrm{P}^m_j(u)$, and $w_2 \in \mathrm{P}^n_i(\pi_j \cdot v)$ such that $w = w_1 w_2$.*

(B) *Let n, $m \in \mathbb{N}$, $i \in [n]$, $u \in \mathrm{T}(\Sigma)^1_m$, and $\vartheta \in \Theta^m_n$. Then*

$$\mathrm{P}^n_i(u \cdot \vartheta) = \bigcup_{j \in \vartheta^{-1}(i)} \mathrm{P}^m_j(u)\,.$$

In particular, $\mathrm{P}^m_j(u) \subseteq \mathrm{P}^n_{\vartheta(j)}(u \cdot \vartheta)$ for each $j \in [m]$.

(C) *Let $t \in \mathrm{T}(\Sigma)^1_n$ for some $n \in \mathbb{N}$, and let $m \in \mathbb{N}$. Then $\mathrm{P}^n_0(t) \subseteq \mathrm{P}^m_0(t \cdot s)$ for every $s \in \mathrm{T}(\Sigma)^n_m$. In particular, $\mathrm{P}^n_0(t) = \mathrm{P}^m_0(t \cdot \vartheta)$ for every $\vartheta \in \Theta^n_m$.*

The proof consists of two parts.

* * *

In the first part, we show for every ℓ, $n \in \mathbb{N}$, $A \in N^{(n)}$ and $t \in \mathrm{T}(\Sigma)^1_n$, that

$$A \cdot \mathrm{Id}_n \overset{\mathrm{oi}}{\Rightarrow}{}^{\ell}_{G}\, t \qquad \text{implies} \qquad \forall i \in [0,n],\, w \in \mathrm{P}^n_i(t)\colon \langle A, i\rangle \Rightarrow^*_{\widehat{G}} w\,.$$

The proof is by complete induction on the length ℓ of the derivation. We must analyze the following cases.

(I) Let

$$A \cdot \mathrm{Id}_n \overset{\mathrm{oi}}{\Rightarrow}_G B(C_1 \cdot \mathrm{Id}_n, \dots, C_m \cdot \mathrm{Id}_n) \overset{\mathrm{oi}}{\Rightarrow}{}^{\ell}_G\, t$$

for some $\ell \in \mathbb{N}$, where the derivation begins with some nonterminal production of G. Then there are ℓ_1, ℓ_2, $k \in \mathbb{N}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)^1_k$, $\vartheta \in \Theta^k_m$, and $v \in \mathrm{T}(\Sigma)$ such that

$$B \overset{\mathrm{oi}}{\Rightarrow}{}^{\ell_1}_G\, \tilde{u} \cdot \vartheta\,, \qquad \vartheta \cdot [C_1, \dots, C_m] \overset{\mathrm{oi}}{\Rightarrow}{}^{\ell_2}_G\, v\,, \qquad t = \tilde{u} \cdot v\,, \qquad \text{and} \qquad \ell = \ell_1 + \ell_2\,.$$

Let $w \in \mathrm{P}^n_i(t)$ for some $i \in [0,n]$. By property **(A)**, we have to distinguish two cases. In the first case, $i = 0$ and

$$w \in \mathrm{P}^m_0(\tilde{u}) \subseteq \mathrm{P}^m_0(\tilde{u} \cdot \vartheta)\,.$$

By the induction hypothesis, $\langle B, 0\rangle \Rightarrow^*_{\widehat{G}} w$, and by construction of $\widehat{G}$, then also $\langle A, 0\rangle \Rightarrow^*_{\widehat{G}} w$.

In the other case, there are $j \in [k]$, $w_1 \in \mathrm{P}^k_j(\tilde{u})$, and $w_2 \in \mathrm{P}^n_i(\pi_j \cdot v)$ such that $w = w_1 w_2$. By fact **(B)**, $w_1 \in \mathrm{P}^m_{\vartheta(j)}(\tilde{u} \cdot \vartheta)$. Moreover, observe that

$$\pi_j \cdot \vartheta \cdot [C_1, \dots, C_m] = \pi_j \cdot [C_{\vartheta(1)}, \dots, C_{\vartheta(k)}] = C_{\vartheta(j)}\,,$$

and thus $C_{\vartheta(j)} \overset{\mathrm{oi}}{\Rightarrow}{}^{\ell'}_G\, \pi_j \cdot v$ for some $\ell' \leq \ell_2$. So we can apply the induction hypothesis and obtain that

$$\big\langle B, \vartheta(j)\big\rangle \Rightarrow^*_{\widehat{G}} w_1 \qquad \text{and} \qquad \big\langle C_{\vartheta(j)}, i\big\rangle \Rightarrow^*_{\widehat{G}} w_2\,,$$

and thus, by construction of $\widehat{G}$, $\langle A, i\rangle \Rightarrow^*_{\widehat{G}} w_1 w_2 = w$.

(II) Let
$$A \cdot \mathrm{Id}_n \overset{\mathrm{OI}}{\Rightarrow}_G x_j$$
for some $j \in [n]$, by some collapsing production of G. If $i = j$, then $\mathrm{P}_i^n(x_j) = \{\varepsilon\}$, and $\langle A, i\rangle \Rightarrow_{\widehat{G}} \varepsilon$. Otherwise, $\mathrm{P}_i^n(x_j) = \emptyset$, and there is nothing to show.

(III) Let
$$A \cdot \mathrm{Id}_n \overset{\mathrm{OI}}{\Rightarrow}_G \sigma \cdot \vartheta$$
for some $k \in \mathbb{N}_1$, $\sigma \in \Sigma^{(k)}$, and $\vartheta \in \Theta_n^k$. Then $\mathrm{P}_0^n(\sigma \cdot \vartheta) = \emptyset$, and
$$\mathrm{P}_i^n(\sigma \cdot \vartheta) = \big\{\langle \sigma, j\rangle \mid \vartheta(j) = i\big\}$$
for each $i \in [n]$. In the latter case, we obtain that $\langle A, i\rangle \Rightarrow_{\widehat{G}} w$ for each $w \in \mathrm{P}_i^n(\sigma \cdot \vartheta)$, by construction of $\widehat{G}$.

(IV) It remains to consider the case
$$A \cdot \mathrm{Id}_n \overset{\mathrm{OI}}{\Rightarrow}_G \alpha$$
for some $\alpha \in \Sigma^{(0)}$. Clearly, $\mathrm{P}_i^n(\alpha) = \emptyset$ for each $i \in [n]$. Further, $\mathrm{P}_0^n(\alpha) = \{\langle \alpha, 0\rangle\}$, and $\langle A, 0\rangle \Rightarrow_{\widehat{G}} \langle \alpha, 0\rangle$.

* * *

The second part of the proof is to show for every ℓ, $n \in \mathbb{N}$, $i \in [0, n]$, and $w \in \widehat{\Sigma}^*$, that whenever $\langle A, i\rangle \Rightarrow_{\widehat{G}}^{\ell} w$, then there is some $t \in \mathcal{L}(G, A \cdot \mathrm{Id}_n)$ such that
$$w \in \mathrm{P}_i^n(t) \qquad \text{and} \qquad i \neq 0 \quad \text{implies} \quad \mathrm{pos}_{x_i}(t) \neq \emptyset\,.$$
We proceed by complete induction on the length ℓ of the derivation in $\widehat{G}$, and make a case analysis on the derivation's initial production.

(I) Let w_1, $w_2 \in \widehat{\Sigma}^*$ such that
$$\langle A, i\rangle \Rightarrow_{\widehat{G}} \langle B, j\rangle\langle C_j, i\rangle \Rightarrow_{\widehat{G}}^{\ell} w_1 w_2$$
by some production $\langle A, i\rangle \to \langle B, j\rangle\langle C_j, i\rangle$ of $\widehat{G}$ constructed according to rule *(i)* from above. In particular, assume that $\langle B, j\rangle \Rightarrow_{\widehat{G}}^{*} w_1$ and $\langle C_j, i\rangle \Rightarrow_{\widehat{G}}^{*} w_2$. By construction, we know that there is a production of form
$$A \cdot \mathrm{Id}_n \to B\big(C_1 \cdot \mathrm{Id}_n, \ldots, C_m \cdot \mathrm{Id}_n\big) \tag{2.5}$$
in P, for some nonterminals $C_1, \ldots, C_{j-1}, C_{j+1}, \ldots, C_m \in N$ with $m \in \mathbb{N}_1$.

By the induction hypothesis, there are $u \in \mathcal{L}(G, B \cdot \mathrm{Id}_m)$ and $v \in \mathcal{L}(G, C_j \cdot \mathrm{Id}_n)$ such that
$$w_1 \in \mathrm{P}_j^m(u)\,, \qquad \mathrm{pos}_{x_j}(u) \neq \emptyset\,, \qquad w_2 \in \mathrm{P}_i^n(v) \qquad \text{and} \qquad \mathrm{pos}_{x_i}(v) \neq \emptyset\,.$$
Let $\mathrm{lin}(u) = (\tilde{u}, \vartheta)$, and let $k = \mathrm{rk}\,\mathrm{inf}(\tilde{u})$. We have the following two subcases.

(a) If Σ is monadic, then $k = 1$ and $\vartheta = \langle m; x_j \rangle$. Since $\vartheta \cdot [C_1, \ldots, C_m] = C_j$, we can apply Lemma 2.14, and conclude that $\tilde{u} \cdot v \in \mathcal{L}(G, A \cdot \mathrm{Id}_n)$. Clearly, $\tilde{u} \cdot v$ contains x_i, since v does.

Let $v' = [z_1, \ldots, z_{j-1}, v, z_{j+1}, \ldots, z_m]$ for arbitrary trees $z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_m \in \mathrm{T}(\Sigma)^1_n$. Then $u \cdot v' = \tilde{u} \cdot v$. Moreover, we can apply fact **(A)** and obtain that $w_1 w_2 \in \mathrm{P}^n_i(u \cdot v')$.

(b) Let Σ be non-monadic. Then we know that the production in (2.5) is not useless, and therefore $\mathcal{L}(G, C_{\vartheta(q)}) \neq \emptyset$ for every $q \in [k]$. Choose some tuple $v' \in \mathrm{T}(\Sigma)^k_n$ where for each $q \in [k]$,

$$\pi_q \cdot v' \in \mathcal{L}(G, C_{\vartheta(j)}), \qquad \text{and in particular,} \qquad \pi_q \cdot v' = v \quad \text{if} \quad \vartheta(q) = j\,.$$

By Lemma 2.14, we see that $\tilde{u} \cdot v' \in \mathcal{L}(G, A \cdot \mathrm{Id}_n)$. Further, $\tilde{u} \cdot v'$ contains x_i, because v does so, and since ϑ contains x_j, v occurs in v' at least once. By fact **(B)**, there is some $q \in \vartheta^{-1}(j)$ such that $w_1 \in \mathrm{P}^k_q(\tilde{u})$. Using property **(A)**, we conclude that $w_1 w_2 \in \mathrm{P}^n_i(\tilde{u} \cdot v')$.

(II) Let

$$\langle A, 0 \rangle \Rightarrow_{\widehat{G}} \langle B, 0 \rangle \Rightarrow^*_{\widehat{G}} w$$

by one of the productions built according to rule *(i)*. By the induction hypothesis, we see that there is some $u \in \mathrm{T}(\Sigma)^1_m$ such that $w \in \mathrm{P}^m_0(u)$. Let $\mathrm{lin}(u) = (\tilde{u}, \vartheta)$, and let $k = \mathrm{rk}\,\mathrm{inf}(\tilde{u})$. Whether Σ is monadic or non-monadic, we can again use the procedures described in case **(I)** to find some

$$v' \in \mathcal{L}\big(G, \vartheta \cdot [C_1, \ldots, C_m]\big)\,.$$

By property **(C)**, $w \in \mathrm{P}^m_0(u)$ implies that $w \in \mathrm{P}^k_0(\tilde{u})$, and therefore $w \in \mathrm{P}^n_0(\tilde{u} \cdot v')$.

(III) Let

$$\langle A, i \rangle \Rightarrow_{\widehat{G}} \varepsilon$$

by some production introduced to $\widehat{P}$ by rule *(ii)*. Consequently, there is the production $A \cdot \mathrm{Id}_n \to x_i$ in P, and hence $x_i \in \mathcal{L}(G, A \cdot \mathrm{Id}_n)$. Observe that $\mathrm{P}^n_i(x_i) = \{\varepsilon\}$, and clearly x_i occurs in x_i.

(IV) Let

$$\langle A, 0 \rangle \Rightarrow_{\widehat{G}} \langle \alpha, 0 \rangle$$

by some production created by rule *(iii)*. Then G contains the production $A \cdot \mathrm{Id}_n \to \alpha$, and thus $\alpha \in \mathcal{L}(G, A \cdot \mathrm{Id}_n)$. Note that $\mathrm{P}^n_0(\alpha) = \{\langle \alpha, 0 \rangle\}$.

(V) Let

$$\langle A, i \rangle \Rightarrow_{\widehat{G}} \langle \sigma, j \rangle\,,$$

using a production created by rule *(iv)*. Then there is a terminal production $A \cdot \mathrm{Id}_n \to \sigma \cdot \vartheta$ in P, such that $\vartheta(j) = i$. So $\sigma \cdot \vartheta \in \mathcal{L}(G, A \cdot \mathrm{Id}_n)$. Further, $\langle \sigma, j \rangle \in \mathrm{P}^k_j(\sigma)$ by definition, and by fact **(B)**, $\langle \sigma, j \rangle \in \mathrm{P}^n_{\vartheta(j)}(\sigma \cdot \vartheta)$. Since $\vartheta(j) = i$, this means that $\langle \sigma, j \rangle \in \mathrm{P}^n_i(\sigma \cdot \vartheta)$. Moreover, we can conclude that $\sigma \cdot \vartheta$ contains x_i.

* * *

It remains to show that $\mathcal{L}(\widehat{G}) = \mathrm{P}(\mathcal{L}(G))$. The first part of the proof implies that for every $t \in \mathcal{L}(G)$, we have $\mathrm{P}(t) \subseteq \mathcal{L}(\widehat{G})$, and therefore

$$\mathrm{P}(\mathcal{L}(G)) = \bigcup_{t \in \mathcal{L}(G)} \mathrm{P}(t) \subseteq \mathcal{L}(\widehat{G}).$$

The proof's second part shows that for every $w \in \mathcal{L}(\widehat{G})$, there is some $t \in \mathcal{L}(G)$ such that $w \in \mathrm{P}(t)$. Hence

$$\mathcal{L}(\widehat{G}) \subseteq \bigcup_{t \in \mathcal{L}(G)} \mathrm{P}(t) = \mathrm{P}(\mathcal{L}(G)).$$

Thus the proof is concluded. □

2.4 Closure Properties

The class of context-free tree languages exhibits many closure properties similar to the context-free word languages – after all, cftg are a reasonable generalization of cfg to trees. Let us list some easy properties, whose proofs can be adapted straightforwardly from the word case.

Theorem 2.30. *The class* CFT *is closed under*

1. *union [141, p. 113],*
2. *α-concatenation and α-iteration [114, Thm. 15], and*
3. *application of linear tree homomorphisms [141, p. 114].*

Of course, as a consequence of Theorem 2.29, the negative closure results on cfg [22, Thm. 3.2] transfer to cftg.

Theorem 2.31. *The class* CFT *is* not *closed under*

1. *complement, nor under*
2. *intersection.*

Closure under intersection with recognizable tree languages does hold, but the proof is nontrivial: the state behavior of the tree automaton is encoded by duplicating the cftg's parameters. Compare Theorem 2.27 for a similar construction.

Theorem 2.32 (Rounds [141, p. 114], [140, Thm. 7])**.** *The class* CFT *is closed under intersection with recognizable tree languages.*

More precisely, for every n, $q \in \mathbb{N}$, if G is an n-adic cftg and A an fta with q states, then there is a $(q \cdot n)$-adic cftg G' such that $\mathcal{L}(G') = \mathcal{L}(G) \cap \mathcal{L}(A)$.

In fact, the theorem's first line follows quite straightforwardly from Theorem 2.27. Given some pta M with n states, and an fta A with q states, one can construct a pta M' with $\mathcal{L}(M') = \mathcal{L}(M) \cap \mathcal{L}(A)$ by a product construction, yielding $n \cdot q$ states for M'. The pta M' can then be turned into an equivalent cftg G' with maximal nonterminal rank $n \cdot q$.[10]

However, CFT is not closed under arbitrary homomorphisms [55].[11] Moreover, although conjectured by Maibaum as a generalization of the result for CF [114, p. 435], the class CFT is not closed under inverse tree homomorphisms [14, p. 195], and neither is it closed with the restriction to inverse *linear* tree homomorphisms (compare Example 2.8).

Theorem 2.33 (Arnold and Dauchet [14])**.** *There are a cftg G and a linear tree homomorphism h such that the tree language $h^{-1}(\mathcal{L}(G))$ is not context-free.*

In the case of linear context-free tree grammars, most positive closure results from above can be adopted by a close look at the proofs for the general case, checking that the constructed cftg is again linear if the input is so.

[10] However, note that the conversion of cftg into pta in Theorem 2.27 is not optimal with respect to number of states; therefore the theorem does not follow in its entirety.

[11] The IO-context-free tree languages, however, are closed under arbitrary homomorphisms [39, p. 342]!

Theorem 2.34. *The class* CFT_ℓ *is*

1. *closed under union,*
2. *closed under* α*-concatenation and* α*-iteration [84, Lem. 23 & 25], and*
3. *closed under application of linear tree homomorphisms [99, Thm. 14].*

The proof for closure under intersection with recognizable tree languages must be modified, as a nonlinear cftg is constructed in the general case. Here, one can use the straightforward generalization of the proof idea for the word case [22, Thm. 8.1] – compare e.g. [99, Cor. 16] for a complete proof, as well as [158, Thm. 5.4.2]. We merely present the construction behind the proof, as its runtime will be of interest later.

Theorem 2.35. *The class* CFT_ℓ *is closed under intersection with recognizable tree languages.*

In fact, for every $n \in \mathbb{N}$*, every* n*-adic l-cftg and fta* M*, there is an* n*-adic l-cftg* G' *such that* $\mathcal{L}(G') = \mathcal{L}(G) \cap \mathcal{L}(M)$*.*

Proof. Consider some l-cftg $G = (N, \Sigma, S, P)$, given without loss of generality in linear normal form. Moreover, let $M = (Q, \Sigma, F, \delta)$ be a finite-state tree automaton. We assume that $F = \{q_0\}$ for some $q_0 \in Q$. It is easy to see that this assumption does not impact generality either. Define the ranked alphabet N' such that for each $n \in \mathbb{N}$,

$$(N')^{(n)} = \big\{A^{q}_{q_1\cdots q_n} \,\big|\, A \in N^{(n)},\, q, q_1, \ldots, q_n \in Q\big\}\,.$$

For every $n \in \mathbb{N}$ and every q, q_1, …, $q_n \in Q$, we define a function

$$\varphi^{q}_{q_1\cdots q_n} \colon \mathrm{T}(N)^1_n \to \mathcal{P}\big(\mathrm{T}(N')^1_n\big)$$

as follows by structural induction. For every $i \in [n]$, let

$$\varphi^{q}_{q_1\cdots q_n}(x_i) = \begin{cases} \{x_i\} & \text{if } q = q_i \\ \emptyset & \text{otherwise.} \end{cases}$$

For every $m \in \mathbb{N}$, $A \in N^{(m)}$, and ξ_1, …, $\xi_m \in \mathrm{T}(N)$, let

$$\varphi^{q}_{q_1\cdots q_n}\big(A(\xi_1, \ldots, \xi_m)\big) = \Big\{A^{q}_{p_1\cdots p_m}\big(\xi'_1, \ldots, \xi'_m\big) \,\Big|\, p_1, \ldots, p_m \in Q,$$
$$\xi'_1 \in \varphi^{p_1}_{q_1\cdots q_n}\big(\xi_1\big), \ldots, \xi'_m \in \varphi^{p_m}_{q_1\cdots q_n}\big(\xi_m\big)\Big\}\,.$$

Moreover, let $G' = (N', \Sigma, S^{q_0}_{\varepsilon}, P')$ be a cftg, with its set of productions P' given as follows. For every production in P of form

$$A \cdot \mathrm{Id}_n \to B \cdot (U_1 \otimes \cdots \otimes U_m)\,,$$

every q, q_1, …, $q_n \in Q$, and every $\xi \in \varphi^{q}_{q_1\cdots q_n}\big(B \cdot (U_1 \otimes \cdots \otimes U_m)\big)$, insert into P' the production

$$A^{q}_{q_1\cdots q_n} \cdot \mathrm{Id}_n \to \xi\,.$$

Furthermore, for every production in P of form

$$A \cdot \mathrm{Id}_n \to \sigma \cdot \mathrm{Id}_n\,,$$

and every q, q_1, …, $q_n \in Q$ such that $q \in \delta_n(q_1, \ldots, q_n, \sigma)$, insert the production

$$A^{q}_{q_1 \cdots q_n} \cdot \mathrm{Id}_n \to \sigma \cdot \mathrm{Id}_n$$

into P'. We omit the straightforward proof that then $\mathcal{L}(G') = \mathcal{L}(G) \cap \mathcal{L}(A)$. □

Example 2.36. Essentially, the family of functions φ defined in the above proof annotates nonterminals with consistent state behavior. For an illustration, consider the production

$$A(x_1, x_2, x_3) \to B\big(C(x_1, x_2), D, x_3\big)$$

from Example 2.17. Given an fta M with state set Q, we would construct from this production all productions of form

$$A^{q}_{q_1 q_2 q_3}(x_1, x_2, x_3) \to B^{q}_{p_1 p_2 q_3}\big(C^{p_1}_{q_1 q_2}(x_1, x_2), D^{p_2}_{\varepsilon}, x_3\big)\,,$$

for each state q_1, q_2, q_3, p_1, and $p_2 \in Q$. ◁

As linear cftg are substantially simpler than the general model, it stands to reason that CFT_ℓ is closed under inverse linear tree homomorphisms. However, in Chapter 4, we will demonstrate that even CFT_ℓ is not closed under this operation.

2.5 Complexity of Decision Problems

We list the following decision problems of cftg, in the format introduced in Section 1.2.5. Let us begin with the *nonemptiness problem of cftg.* Let in the following Σ be some ranked alphabet.

Problem: Nonemptiness of Context-Free Tree Grammars over Σ
Instance: A cftg $G = (N, \Sigma, \xi_0, P)$.
Question: Is $\mathcal{L}(G) \neq \emptyset$?

As the following theorem shows, we cannot expect to solve this problem efficiently in all instances.

Theorem 2.37 (Tanaka and Kasai [157])**.** *For every non-monadic ranked alphabet Σ such that $\Sigma^{(0)} \neq \emptyset$, the nonemptiness problem of context-free tree grammars over Σ is* EXP*-complete.*

In fact, the original of the above theorem is stated for indexed grammars. Hardness is shown by giving a reduction from the EXP-complete winning strategy problem of pebble games to indexed grammar nonemptiness. The authors construct, given an instance of a pebble game, an indexed grammar G such that $\mathcal{L}(G)$ is nonempty if and only if the game allows a winning strategy for the first player. The result can be transfered from ixg to cftg using Theorem 2.28.

Let G be a cftg with terminal alphabet Σ. The *(non-uniform) membership problem of G* is given as follows.

Problem: Non-Uniform Membership of a Context-Free Tree Grammar G
Instance: A tree $t \in \mathrm{T}_\Sigma$.
Question: Is $t \in \mathcal{L}(G)$?

Here, we fix a cftg G, which therefore does not contribute to the size of the problem's input. Intuitively, in a decision procedure we can apply any transformation to G without having to account for its runtime!

Theorem 2.38 (Rounds [142])**.** *For every cftg G, the non-uniform membership problem of G is in* NP*. Moreover, there are a ranked alphabet Σ and a cftg G' over Σ whose non-uniform membership problem is* NP*-hard.*

Compare also Section 3.3 for an alternative proof of this theorem. If we take the cftg G to be part of the input instead, we obtain the *uniform membership problem of cftg.* Obviously, uniform membership is at least as hard as the non-uniform membership problem.

Problem: Uniform Membership of Context-Free Tree Grammars over Σ
Instance: A cftg $G = (N, \Sigma, \xi_0, P)$ and a tree $t \in \mathrm{T}_\Sigma$.
Question: Is $t \in \mathcal{L}(G)$?

Moreover, we may ask the question whether a cftg generates an infinite number of trees – the *infiniteness problem of cftg.*

Problem: **Infiniteness of Context-Free Tree Grammars over Σ**
Instance: A cftg $G = (N, \Sigma, \xi_0, P)$.
Question: Is $|\mathcal{L}(G)| = \infty$?

The complexity of these two problems will be established in Chapter 3. When we restrict the inputs to the above problems to *linear* cftg (resp. to ln-cftg), we obtain, respectively, the nonemptiness, non-uniform membership, uniform membership, and infiniteness problem of l-cftg (resp. of ln-cftg) over Σ. Their complexity will also be treated in Chapter 3.

2.6 Chapter Conclusion

In the concluding section of this chapter, we will give a brief survey of literature on cftg. Although we try to mention most interesting results, we make no claim to completeness. We will go into the origins of cftg, survey several characterization results, and other properties of cftg. Moreover, we will summarize what is known about some particular restrictions of cftg, and mention a number of generalizations of the formalism.

Origins

As described in this chapter's introduction, the concept of context-free tree grammar is already implicit in the definition of the macro (word) grammar, discovered by Fischer [61, 60]. These are context-free grammars where every nonterminal is allowed a fixed number of parameters. As an example, the macro grammar given by the productions

$$S \to A(\varepsilon, \varepsilon), \qquad A(x_1, x_2) \to aA(bx_1,\, bx_1x_1x_2) + x_2$$

generates the word language $\{a^n b^{n^2} \mid n \in \mathbb{N}\}$.[12] If the right-hand sides of macro grammars are restricted to be terms encoding trees, then one obtains precisely the context-free tree grammars. Moreover, the class of languages generated by macro grammars is exactly the class IND [60, Thm. 4.2.8].

This, however, is not the original definition of the formalism that was given by Rounds in [139, 140]. The context-free tree grammars described in [139], and called *creative dendrogrammars* in [140], are rewrite systems with two types of productions, called *index-creating* and *index-erasing*. In the nomenclature of [51, 59], creative dendrogrammars can be understood as regular tree grammars with a tree pushdown storage. However, as a consequence of [140, Thm. 7], creative dendrogrammars are indeed equivalent to cftg as defined in this work.[13]

In [114], Maibaum states the independent discovery of cftg. Moreover, the definition of cftg is essentially given in Nivat's work on program schemes [127].

Characterization Results

Many early characterization results on context-free tree languages are motivated by algebraic semantics. For instance, an equational (or fixed-point) characterization of CFT (resp. of the macro languages) is given in [44, 127, 114, 55].[14] In addition, [55] contains an analogous result on IO context-free tree languages. In fact, the solutions of equation systems in OI and IO mode given in [55] differ only in the employed type of tree language substitution. Moreover, the second part of this article [56] includes Mezei-Wright-like theorems,[15] which show that the solutions of context-free equation systems are the homomorphic images of

[12]Hint: for every $n \in \mathbb{N}$, $(1+n)^2 = 1 + 2n + n^2$.

[13]As a side-note, creative dendrogrammars are very close to the pushdown transducers we will introduce in Chapter 5.

[14]However, note that the characterization given in [114] is incorrect, as remarked in [55].

[15]Named after the authors of the seminal article [121].

solutions in associated regular equation systems. Such theorems are useful as they allow the transfer of theories.

Two distinct pushdown machine characterizations of CFT have been given by Guessarian [79] and by Schimpf and Gallier [146]. While the model of Guessarian (a restriction of which we have recalled in Section 2.2) recognizes trees top-down, the one of Schimpf and Gallier processes them in a bottom-up manner. Since every context-free tree languages is the output language of some *macro tree transducer* [58], the pushdown machine characterizations presented in [59] apply also to CFT.

In [64], Fujiyoshi states a pushdown machine characterization of the linear monadic context-free tree languages; cf. also [65] for an implicit statement of the characterization. The presented pushdown automaton is closely related to linear indexed grammars [68]. Kanazawa proposes how to generalize such a characterization to the class CFT_ℓ [96], by using tree tuples on the pushdown.

Rounds's yield theorem, which illuminates the connection between the classes CFT and IND, has already been mentioned above [140, p. 286]. In [13], a Chomsky-Schützenberger-style characterization of CFT has been presented; compare Example 2.7 above. In [97], a similar theorem is proven for the class CFT_ℓ, by means of a very intricate analysis of multi-dimensional trees.

Kepser and Rogers show the equivalence of linear monadic cftg with (a variant of) tree-adjoining grammars, a formalism well-known in computational linguistics [100]; compare also [70] for a direct proof of one direction of the equivalence, as well as Chapter 6.

Theorems and Properties

Maibaum proves a pumping lemma for cftg in [115]. One interesting difference of this pumping lemma to the one for cfg is that it captures to a certain extent the interplay of nondeterminism and copying.[16]

In [11], Arnold and Dauchet prove a copying theorem for CFT. A copying theorem characterizes the power of a formalism to generate identical subwords or subtrees, cf. [57]. The copying theorem for CFT states that if a cftg G can generate all trees of the form $\sigma(t,t)$ with $t \in L$, for some tree language $L \in \mathsf{CFT}$, then L is a coregular context-free tree language.[17] As a corollary, the tree language $\{\sigma(t,t) \mid t \in \mathrm{T}_\Sigma\}$ is not context-free, except when Σ is monadic.

Most well-known closure properties of CFT had already been established by Rounds [141, 140]. In [14], Arnold and Dauchet prove that closure of CFT under inverse linear tree homomorphisms does not hold in general. Compare Example 2.8 above for a brief description of their counterexample. In his thesis [109], Leguy uses similarly constructed grammars to distinguish the power of many restrictions of cftg. Moreover, several transformations are given to simplify cftg, both for the OI and the IO case.

While, as stated below Theorem 2.3, collapsing productions cannot be removed from a cftg in general, a partial solution is given in [85]. There, the authors show how to construct, for every cftg G and every given $n \in \mathbb{N}$, an equivalent cftg G'. In a derivation of G', productions

[16] For a very elegant pumping lemma for indexed languages, see [155].
[17] In fact, then also $L(G)$ is coregular.

of form $A \to x_i$ need only be applied to nodes whose distance from the root is greater than n. In other words, such productions can be forbidden when considering only the n upper levels of a sentential form. As a corollary, one obtains an alternative decision procedure for the membership problem of cftg.

Dauchet and Tison analyze in [39] the structural complexity of classes of tree languages. Pertaining to cftg, they show that every recursively enumerable tree language can be expressed as the image of the intersection of two context-free tree languages under a linear alphabetic tree homomorphism.[18]

In his dissertation [156], Stamer introduces a new type of tree automaton, called *restarting tree automaton*. He proves that every linear context-free tree language can be recognized by a restarting tree automaton. However, there are some tree languages accepted by a restarting tree automaton which are not context-free. The work contains some very detailed constructions of particular normal forms of linear cftg.

A work of Nederhof, Teichmann and Vogler [123] pertains to a generalization of Chomsky's theorem on non-self-embedding cfg [32].[19] The article contains the definition of what it means for an ln-cftg to be non-self-embedding. Moreover, the authors show that the tree language of each non-self-embedding ln-cftg is recognizable, by an elaborate construction of an equivalent regular tree grammar. The second author's PhD thesis [158] also provides material on weighted approximation of context-free tree languages.

Particular Restrictions

Next, we recall some interesting restrictions of cftg. *Greibach* cftg are defined in [18]. They generalize Greibach cfg [77], inasmuch the root of the right-hand side of every production of a Greibach cftg must be a terminal symbol. However, in contrast to cfg, there are cftg for which there is *no* equivalent Greibach cftg. As shown in [18], the class of tree languages of Greibach cftg is closed under inverse linear tree homomorphisms, in contrast to the whole class CFT. In [63], Fujiyoshi proves that for linear monadic cftg, there is indeed a Greibach normal form – i.e., for every lm-cftg there is an equivalent Greibach lm-cftg.[20]

As defined above, in a coregular cftg nonterminals may only occur in a production's right-hand side at its root. In [10], it is proven that the tree languages of coregular cftg are precisely the images of the recognizable word languages (understood as monadic tree languages) under deterministic top-down tree transducers. This theorem is used to derive some closure properties, and a connection to EDT0L systems is revealed. For similar results on the word level, compare [44, 54]. Coregular cftg are also studied in [84], where they are called *top-context-free*. In particular, the tree languages of coregular cftg which are recognizable are characterized. The article also gives an extensive overview of closure properties of linear, coregular, and unrestricted cftg.

[18] This generalizes the famous result of Ginsburg, Greibach and Harrison [73, Thm. 3.1]. However, Dauchet and Tison also prove that, suprisingly, for every recursively enumerable tree language L, there are recognizable tree languages R_1 and R_2, tree homomorphisms φ_1 and φ_2, and a linear alphabetic tree homomorphism ψ such that $L = \psi(\varphi_1(R_1) \cap \varphi_2(R_2))$.

[19] See also Teichmann's PhD thesis [158].

[20] It is easy to see that this property does no longer hold if one does not demand monadicity – consider the production $A(x_1, x_2) \to A(a(x_1), b(x_2))$.

Fujiyoshi and Kasai define *spinal-formed* cftg in [65]. They show that the yield languages of spinal-formed cftg are precisely those of tree-adjoining grammars. Moreover, they prove that spinal-formed cftg are expressively equivalent to linear monadic cftg, and they give a characterization of the tree languages of spinal-formed cftg by linear tree pushdown automata.

Straight-line context-free tree grammars are cftg which generate precisely one tree, in precisely one derivation. They are interesting because they are a space-efficient representation of trees. In [89], Jeż and Lohrey show how to compute, given a tree $t \in \mathrm{T}_{\Sigma}$, a small straight-line cftg which generates t. Although the problem to determine the smallest such grammar for t cannot be solved efficiently, their solution is only larger by a factor of $\mathcal{O}(\log|t|)$, if the ranked alphabet Σ is fixed.

Generalizations

Engelfriet and Vogler study *macro tree transducers* in [58]. Macro tree transducers can be understood as context-free tree grammars whose derivations are controlled by an input tree. In this manner, they define a tree transformation, translating the input tree into the tree(s) derived from it. Intuitively, macro tree transducers are the common generalization of cftg and top-down tree transducers. The cited article contains composition and decomposition results on macro tree transducers. Moreover, macro tree transducers with *regular lookahead* (cf. [50]) are considered, and it is shown that for most restrictions of macro tree transducers, the addition of regular lookahead does not increase the transducers' power. Engelfriet and Vogler continue the investigation of macro tree transducers in [59]; there, they present (several types of) pushdown machines which characterize the tree transformations of macro tree transducers. The work is based on the concept of grammars with storage [51], and most equivalence proofs are by means of simulation of one storage type by another storage type. Using this method, the authors can also show a characterization result for compositions of macro tree transducers by machines with iterated pushdowns.

Bozapalidis defines weighted context-free tree languages in [24]. They are given by particular equation systems which resemble the systems given in [55], but where each summand of an equation's right-hand side is associated with a weight from an underlying semiring K. The solution of such a system is the least fixed point of an associated mapping. The article contains normal form results, as well as a Kleene-like theorem for the class of weighted context-free tree languages. Moreover, it gives a Mezei-Wright-like result for equational elements of well ω-additive K-Γ-algebras. The latter can be understood as algebras with operators indexed by a ranked alphabet Γ, with a semiring K which acts on them, and which possess well-behaved countably infinite sums. As an application, the article closes with a discussion of additive recursive program schemes.

In his dissertation [9], Arnold introduces context-free grammars over a particular kind of magmoid, called *magmoid with torsion.* The representative example for this type of magmoid is $\mathrm{T}(\Sigma)$, the free projective magmoid generated by a ranked alphabet Σ. Let us call the introduced grammar formalism *magmoid grammar.* Magmoid grammars over $\mathrm{T}(\Sigma)$ as underlying magmoid are merely context-free tree grammars. Another instance of a magmoid with torsion is the magmoid $k\text{-dil}\,\mathrm{T}(\Sigma)DT$, for some number $k \in \mathbb{N}$ and ranked alphabet Σ. Intuitively, each nonterminal of a magmoid grammar over $k\text{-dil}\,\mathrm{T}(\Sigma)DT$ generates a

k-tuple of trees. If for every production of such a magmoid grammar, only linear k-tuples of trees occur in its right-hand side, then the grammar is called *linear.* Every context-free tree language can be generated by a linear magmoid grammar over $k\text{-dil}\,\mathrm{T}(\Sigma)DT$. Moreover, the class of tree languages generated by linear magmoid grammars over $k\text{-dil}\,\mathrm{T}(\Sigma)DT$ is closed under inverse linear tree homomorphisms, in contrast to the class of context-free tree languages. The dissertation contains a wide range of further results on grammars and equation systems over magmoids.

Engelfriet, Maletti, and Maneth propose in [53] a common extension of multiple context-free grammars [149] and linear and nondeleting cftg, called *multiple context-free tree grammar (mcftg)*. In the derivation semantics of mcftg, the application of a production to a sentential form rewrites several nonterminal occurrences simultaneously with ln-cftg productions. However, a set of nonterminal occurrences can only be rewritten if the respective nonterminals are *linked* in the sentential form; in particular, one may not parallely rewrite nonterminals which were introduced into the sentential form by distinct productions.[21] In this manner, an mcftg derives a tree language. In fact, the authors present two further semantics for mcftg, namely a fixed-point and a derivation tree semantics, and prove all three semantics to be equivalent. Then, they generalize the lexicalization result of l-cftg from [117] to mcftg.[22] Moreover, they relate the power of mcftg to that of multi-component tree-adjoining grammars, deterministic finite-copying macro tree transducers, and multiple context-free word grammars.

The final generalization of cftg we are going to discuss are *higher-order grammars (hog)*. For this purpose, observe that a cftg can be understood as a first-order nondeterministic functional program, i.e., as a hog of order 1: each production $A \cdot \mathrm{Id}_n \to \varrho$ corresponds to a function which maps an n-tuple of trees $\xi \in \mathrm{T}(N \cup \Sigma)^n$ to the tree $\varrho \cdot \xi$. So we can express functions of type

$$\mathrm{T} \times \cdots \times \mathrm{T} \to \mathrm{T}\,,$$

where T abbreviates $\mathrm{T}(N \cup \Sigma)^1$. We obtain a hog of order 2 by allowing functions to have functions such as the above as arguments. That is, we can express functions of type

$$\mathrm{T}^{\mathrm{T} \times \cdots \times \mathrm{T}} \times \cdots \times \mathrm{T}^{\mathrm{T} \times \cdots \times \mathrm{T}} \times \mathrm{T} \times \cdots \times \mathrm{T} \to \mathrm{T}\,.$$

Higher order grammars of order 3 are obtained by allowing functions of the above type as arguments, and by iterating this process, we obtain hog of arbitrary order.

Higher-order grammars have been introduced by Damm in [37]. The article's motivation is algebraic semantics of higher-order functional programs. For this purpose, a Mezei-Wright-like theorem is proven, as well as a Kleene-type result. Moreover, the OI and the IO hierarchies are studied. These are the hierarchies of classes of languages generated by hog of order 1, 2, 3, ..., respectively in OI and IO derivation mode. It is proven that both hierarchies are indeed proper. The main theorem in [38] shows that for every $n \in \mathbb{N}_1$, hog of order n are expressively equivalent to the *n-iterated pushdown automata* of Maslov.

[21] By this description, the *synchronous cftg* of [124] turn out to be a special case of mcftg. We will get back to synchronous cftg in Chapter 5.

[22] Compare the discussion of the result in the chapter's introduction. For the result for mcftg, lexical symbols need not necessarily be of rank 0.

In recent years, there has been renewed interest in higher-order grammars. It has been observed that the higher-order grammars introduced by Damm fulfill by their definition a property called *safety* [102]. If one drops the restriction to this property, one obtains *unsafe* hog. Language-theoretic properties of unsafe hog are studied in [103]. For survey articles on recent developments (with focus on applications of higher-order grammars to model checking), see [128] and [163].

Chapter 3

Decision Problems of Context-Free Tree Grammars

She wanted to know what is the worst. Not the best, the worst. By which she meant the truth.

(Philip Roth)

In the previous chapter, we have listed a number of applications of context-free tree grammars. Of course, for many such applications, one is not just interested in theoretical possibility, but in practical feasibility. Therefore, in this chapter, we will cover the decision problems of cftg, and their computational complexity.

The following is already known about the complexity of cftg and related formalisms. In [3, Alg. 1], Aho presents an algorithm which can be used to solve the nonemptiness problem of indexed grammars in exponential time. In fact, as proven by Tanaka and Kasai [157], this algorithm is optimal, as the problem is EXP-complete. As shown by Rounds [142], the non-uniform membership problem of indexed grammars is NP-complete. The proof for NP-hardness is by reduction from the satisfiability problem of propositional logic. The upper bound is established by an analysis of Aho's proof that the indexed languages are context-sensitive [3, Thm. 5.1]. All these results can be transfered to cftg using their close correspondence to ixg; cf. Theorem 2.28. Inaba and Maneth show in [87] that the class of output languages of compositions of macro tree transducers is contained in NP. Observe that every context-free tree language is the output language of some macro tree transducer, so the proof applies also to CFT. Mehlhorn presents a restriction on macro grammars which allows efficient parsing in [120]. Further, in [20], Asveld determines the time and space complexity of IO macro grammars.

In this chapter, we will analyze some decision problems of cftg. First of all, we tackle the uniform membership problem of cftg, and prove that it is PSPACE-complete. To facilitate the proof of containment in PSPACE, we introduce in Section 3.1 some properties for pta, and show that if a pta satisfies these properties, then its derivations can be represented in polynomial space. In fact, the idea of the construction is already implicit in the Turing machine presented by Aho in [3, Thm. 5.1]. Note however that there, no proof of correctness is given. In contrast, by providing automata-theoretic constructions on the level of pta, we prove the construction correct. The construction is applied in Section 3.2, where we prove that the problem is indeed PSPACE-complete. As a byproduct of the construction, we can state an alternative proof for the NP-completeness of the non-uniform membership problem

Table 3.1: Decision Problems of Context-Free Tree Grammars

	nonemptiness	membership	unif. membership	infiniteness
cftg	EXP-complete	NP-complete	PSPACE-complete	EXP-complete
l-cftg	NP-hard	in P	NP-hard	NP-hard
ln-cftg	in P	in P	?	in P

in Section 3.3. In Section 3.4, we show that the infiniteness problem of cftg is EXP-complete. The proof of hardness is by a reduction from the nonemptiness problem, and the upper bound is a consequence of Theorem 2.29.

As the above problems are computationally hard, it is interesting to determine whether the restriction to linear cftg makes matters better. In Section 3.5, we prove that nonemptiness, infiniteness, and uniform membership of l-cftg are still NP-hard, while non-uniform membership of l-cftg is solvable in deterministic polynomial time. The problem behind NP-hardness is the phenomenon of deletion. In fact, if one considers ln-cftg, then nonemptiness and infiniteness can also be decided in P. Table 3.1 summarizes the complexity-theoretic results for (the restrictions of) cftg.

As exhibited in Theorem 2.29, if Σ is monadic, then the tree language of a cftg G over Σ can be represented faithfully by a context-free grammar $\widehat{G}$ which is constructible from G in polynomial time. Moreover, it is well-known that the nonemptiness, infiniteness, and (uniform) membership problems of cfg are decidable in deterministic polynomial time [22]. On the other hand, if $\Sigma^{(0)} = \emptyset$, then the tree language of G is always empty, and the respective decision problems are trivial. By this motivation, we will call a ranked alphabet Σ *nontrivial* if Σ is not monadic and $\Sigma^{(0)} \neq \emptyset$.

The next convention will spare us the necessity of quite a number of quantifications.

Convention. *We are going to assume for this chapter that $M = (Q, \Sigma, \Gamma, q_0, R)$ is an arbitrary pta in normal form, unless stated otherwise. Moreover, we will denote the set of pop rules of M by $R_\downarrow$, the set of its push rules by $R_\uparrow$, and the set of its copy rules by R_Σ.*

Note: The results from Sections 3.1 to 3.3 have been first reported in [130]. However, most proofs have been rewritten and extended. The material presented on the infiniteness problem of cftg has not yet been published. Most of the theorems from Section 3.5 have been discovered together with Florian Starke, during the supervision of his “Belegarbeit” thesis. The results on infiniteness of l(n)-cftg are new.

3.1 Space- and Time-Efficient Pushdown Tree Automata

To obtain efficient algorithms, we must make sure that a derivation of a pta is as short as possible, meaning that a minimal number of rules are applied. Moreover, the pushdowns that occur within the derivation should be of minimal size. In the following, we will present certain normal forms of pta which facilitate these goals.

3.1.1 Derivations

In this chapter, we will often deal with derivations of pts as independent mathematical objects. Therefore, we reify this notion. Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pts. We say that a sequence $r_1 \cdots r_n$ of rules $r_1, \ldots, r_n \in R$ is a *derivation (in M)* if there are $\xi_0, \ldots, \xi_n \in \mathrm{T}_\Sigma(Q(\Gamma^*))$ such that

$$\xi_0 \Rightarrow_{r_1} \xi_1 \Rightarrow_{r_2} \cdots \Rightarrow_{r_n} \xi_n\,,$$

and each rule r_i is applied at the minimal position of ξ_{i-1} with respect to $\leq_{\mathrm{lex}}$ that is labeled by an element of $Q(\Gamma^*)$. In this case, we also say that $r_1 \cdots r_n$ is a *derivation of ξ_n from ξ_0 (in M),* and write $\xi_0 \Rightarrow_{r_1 \cdots r_n} \xi_n$. Given $\xi, \zeta \in \mathrm{T}_\Sigma(Q(\Gamma^*))$, we denote by $\mathcal{D}_M(\xi, \zeta)$ the set of all derivations of ζ from ξ in M. By Corollary 2.25, we have that $\xi \Rightarrow_M^* \zeta$ if and only if $\mathcal{D}_M(\xi, \zeta) \neq \emptyset$. Moreover, we let

$$\mathcal{D}_M = \bigcup \left\{ \mathcal{D}_M(q(\eta), \xi) \;\middle|\; q(\eta) \in Q(\Gamma^*),\ \xi \in \mathrm{T}_\Sigma(Q(\Gamma^*)) \right\}.$$

3.1.2 Succinct Pushdown Tree Automata

We begin with a construction which allows us to bound the length of derivations in M, by avoiding turns. A *turn* of M is a derivation of the form

$$q_0(\eta) \Rightarrow_M q_1(\kappa_1\eta) \Rightarrow_M q_2(\kappa_2\eta) \Rightarrow_M \cdots \Rightarrow_M q_n(\kappa_n\eta) \Rightarrow_M q_{n+1}(\eta)\,,$$

for some $n \in \mathbb{N}$, $q_0, \ldots, q_{n+1} \in Q$ and $\kappa_1, \ldots, \kappa_n, \eta \in \Gamma^*$. Intuitively, in a turn M pushes some symbols onto the pushdown η, never touching η, and only to pop them again before arriving in state q_{n+1}. In order to obtain derivations that are as short as possible, we would like to be able to cut short such turns.

For an example, consider the derivation

$$q(\gamma) \Rightarrow_M q_1(\delta\gamma) \Rightarrow_M q_2(\tau\delta\gamma) \Rightarrow_M q_3(\delta\gamma) \Rightarrow_M q_4(\gamma) \Rightarrow_M p(\varepsilon)\,.$$

We could avoid the turn $q(\gamma) \Rightarrow_M^+ q_4(\gamma)$ if there was some rule $q(\gamma x) \to p(x)$ in R, because then $q(\gamma) \Rightarrow_M p(\varepsilon)$. We say that a pta is *succinct* if it has such rules to cut short turns. Formally, a pta M is *succinct* if for every q_1, q_2, $q_3 \in Q$ and $\gamma \in \Gamma$ such that $q_1(\varepsilon) \Rightarrow_M q_2(\gamma) \Rightarrow_M q_3(\varepsilon)$, and for every rule $q_3(ux) \to \varrho$ in R, the rule $q_1(ux) \to \varrho$ is also in R.

Lemma 3.1. *For every pta M, an equivalent succinct pta M' is constructible in polynomial time.*

Algorithm 1 Construction of a succinct pta

$R' \leftarrow R$
while there are $q_1, q_2, q_3 \in Q, \gamma \in \Gamma$, and $(q_3(ux) \to \varrho) \in R'$
 such that $q_1(\varepsilon) \Rightarrow_{M'} q_2(\gamma) \Rightarrow_{M'} q_3(\varepsilon)$ and $(q_1(ux) \to \varrho) \notin R'$
do
 insert $(q_1(ux) \to \varrho)$ into R'
end while

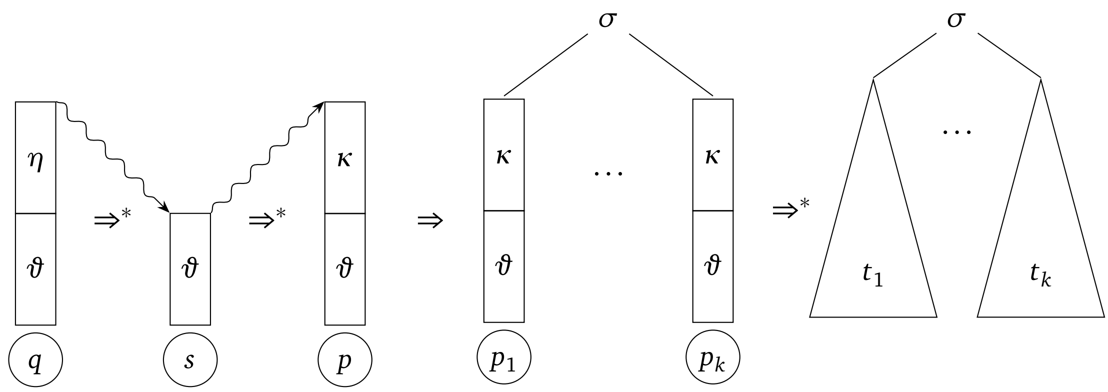


Figure 3.1: A succinct derivation in M

Proof. We may assume without loss of generality that M is in normal form. Define the pta $M' = (Q, \Sigma, \Gamma, q_0, R')$, with R' constructed according to Algorithm 1. Clearly, the algorithm's while loop respects the invariant $\mathcal{L}(M') = \mathcal{L}(M)$, as only alternative derivations of trees from $t \in \mathcal{L}(G)$ are added. Moreover, after termination of the algorithm, M' is obviously succinct, and it is easy to check that M' is in normal form.

Observe that the number of rules of a pta in normal form over the terminal alphabet Σ is bounded by

$$2 \cdot |Q|^2 \cdot |\Gamma| + |\Sigma| \cdot |Q|^{\max \mathrm{rk}(\Sigma)+1} .$$

As in every iteration of the while loop a rule is inserted into R', the algorithm terminates eventually. Since Σ is fixed, the number of iterations of the while loop is polynomial in the input. □

A derivation is called *succinct* if it contains no turns. Formally, $d \in \mathcal{D}_M$ is *succinct* if there are $e_1 \in R_{\downarrow}^*$, $e_2 \in R_{\uparrow}^*$, $r \in R_{\Sigma}$, $k \in \mathbb{N}$ and $d_1, \ldots, d_k \in \mathcal{D}_M$ such that

$$d = e_1 e_2 r d_1 \ldots d_k$$

and for every $i \in [k]$, d_i is succinct. Compare Figure 3.1 for the structure of a succinct derivation. For every $q(\eta) \in Q(\Gamma^*)$ and $t \in \mathrm{T}_{\Sigma}$, the set of all succinct derivations in $\mathcal{D}_M(q(\eta), t)$ is denoted by $\mathcal{DS}_M(q(\eta), t)$, and the set of all succinct elements of $\mathcal{D}_M$ by $\mathcal{DS}_M$.

The following lemma shows that in a succinct pta M, we can restrict ourselves to succinct derivations.

Lemma 3.2. *Let M be succinct, $q(\eta) \in Q(\Gamma^*)$, and $t \in \mathrm{T}_\Sigma$. If $q(\eta) \Rightarrow_M^* t$, then there is a succinct derivation $d \in \mathcal{DS}_M(q(\eta), t)$.*

Proof. Let $t \in \mathrm{T}_\Sigma$, and assume a succinct pta M, as well as $q(\eta) \in Q(\Gamma^*)$ such that $q(\eta) \Rightarrow_M^* t$. The proof of the lemma is by structural induction on t, so let t be of the form $\sigma(t_1, \dots, t_k)$ for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \dots, t_k \in \mathrm{T}_\Sigma$.

By Corollary 2.25, we can restrict ourselves to considering just leftmost derivations, hence there are $p_1, \dots, p_k \in Q$, $\kappa \in \Gamma^*$ and some derivation $d \in (R_\uparrow \cup R_\downarrow)^* \cdot R_\Sigma$ such that

$$q(\eta) \Rightarrow_d \sigma\big(p_1(\kappa), \dots, p_k(\kappa)\big) \Rightarrow_M^* \sigma\big(t_1, p_2(\kappa), \dots, p_k(\kappa)\big) \Rightarrow_M^* \cdots \Rightarrow_M^* \sigma(t_1, \dots, t_k)\,.$$

By the induction hypothesis, there is a succinct $d_i \in \mathcal{DS}_M(p_i(\kappa), t_i)$ for every $i \in [k]$. We show that in the derivation d, it is actually not necessary to apply a push rule right before a pop rule. For that purpose, assume that $d = d' r_1 r_2 r_3 d''$ for some d', $d'' \in R^*$, $r_1 \in R_\uparrow$, $r_2 \in R_\downarrow$, and $r_3 \in R$. Then there are q_1, q_2 and $q_3 \in Q$, as well as $\tau \in \Gamma^*$, $\gamma \in \Gamma$, and $\zeta \in \mathrm{T}_\Sigma(Q(\Gamma^*))$ such that

$$q(\eta) \Rightarrow_{d'} q_1(\tau) \Rightarrow_{r_1} q_2(\gamma\tau) \Rightarrow_{r_2} q_3(\tau) \Rightarrow_{r_3} \zeta \Rightarrow_{d''} \sigma\big(p_1(\kappa), \dots, p_k(\kappa)\big)\,.$$

In particular, $q_1(\varepsilon) \Rightarrow q_2(\gamma) \Rightarrow q_3(\varepsilon)$. As M is succinct, there is a rule $r \in R$ such that $q_1(\tau) \Rightarrow_r \zeta$ and thus

$$q(\eta) \Rightarrow_{d'} q_1(\tau) \Rightarrow_r \zeta \Rightarrow_{d''} \sigma\big(p_1(\kappa), \dots, p_k(\kappa)\big)\,.$$

As derivations are of finite length, a finite number of such elimination steps shows that there is some $\tilde{d} \in R_\downarrow^* \cdot R_\uparrow^* \cdot R_\Sigma$ with

$$q(\eta) \Rightarrow_{\tilde{d}} \sigma\big(p_1(\kappa), \dots, p_k(\kappa)\big)\,,$$

and hence $\tilde{d} d_1 \dots d_k \in \mathcal{DS}_M(q(\eta), t)$. $\square$

3.1.3 Subdivisions of Symbols and Compact Systems

While using succinct pta already allows us to avoid unnecessary turns within a derivation, this is not yet sufficient for an efficient algorithm. We also need to make sure that the pushdowns that occur in a derivation are as small as possible. Consider for example the succinct derivation

$$q(\varepsilon) \Rightarrow_M q'(\gamma) \Rightarrow_M q''(\delta\gamma) \Rightarrow_M \sigma\big(u(\delta\gamma), p(\delta\gamma)\big) \Rightarrow_M^2 \sigma\big(\alpha, p(\gamma)\big) \Rightarrow_M \sigma\big(\alpha, p(\varepsilon)\big)\,.$$

In this derivation, there is no point where the pushdown symbols δ and γ are used separately from each other: the subderivation $u(\delta\gamma) \Rightarrow_M \alpha$ discards both of them, while in the subderivation $p(\delta\gamma) \Rightarrow_M p(\gamma) \Rightarrow_M p(\varepsilon)$, the pushdown γ that remains after popping δ is also popped afterwards, without being copied inbetween.

So if we had a symbol $\overbracket{\delta\gamma}$ in Γ, and the appropriate rules in R, then we could emulate this derivation by

$$q(\varepsilon) \Rightarrow_M q''(\overbracket{\delta\gamma}) \Rightarrow_M \sigma\big(u(\overbracket{\delta\gamma}), p(\overbracket{\delta\gamma})\big) \Rightarrow_M^2 \sigma\big(\alpha, p(\varepsilon)\big)\,.$$

Here, all occurring pushdowns are of size at most 1, instead of 2. Moreover, since we do not have to push and pop as many symbols, the derivation is also shorter. In the following, we will show how to construct from M an equivalent pts (called *compact*) that allows compressing pushdowns in the way described above.

First, however, we must introduce some notation. Define the infinite alphabet $\mathcal{S}(\Gamma)$ by

$$\mathcal{S}(\Gamma) = \left\{ \overline{\gamma_1 \cdots \gamma_n} \;\middle|\; n \in \mathbb{N}_1, \gamma_1, \ldots, \gamma_n \in \Gamma \right\}.$$

Here, $\overline{\gamma_1 \cdots \gamma_n}$ is *one* atomic symbol of $\mathcal{S}(\Gamma)$.

Using symbols from $\mathcal{S}(\Gamma)$ allows us to denote subdivisions of a word from Γ^*. Formally, assume a word $\eta = \gamma_1 \ldots \gamma_n$ from Γ^+, where $n \in \mathbb{N}_1$ and $\gamma_i \in \Gamma$ for every $i \in [n]$. Let $m \in \mathbb{N}_1$, and furthermore let $k_0, \ldots, k_m \in \mathbb{N}$ with $0 = k_0 < \cdots < k_m = n$. Then the $\{k_0, \ldots, k_m\}$*-subdivision of* η is the word

$$\overline{\gamma_{k_0+1} \cdots \gamma_{k_1}} \cdots \overline{\gamma_{k_{m-1}+1} \cdots \gamma_{k_m}} \in \mathcal{S}(\Gamma)^+ .$$

Moreover, the $\emptyset$*-subdivision of* ε is ε. A word $\eta' \in \mathcal{S}(\Gamma)^*$ is called a *subdivision* of a word $\eta \in \Gamma^*$, denoted by $\eta' \preceq \eta$, if η' is an E-subdivision of η for some $E \subseteq \mathbb{N}$. This E is unique; we denote it by $E(\eta')$. In this situation, the length $|\eta'|$ of η' (as an element of $\mathcal{S}(\Gamma)^*$) satisfies

$$|\eta'| = \begin{cases} |E(\eta')| - 1 & \text{if } \eta \in \Gamma^+, \text{ and} \\ 0 & \text{if } \eta = \varepsilon. \end{cases}$$

Convention. *Whenever E is denoted by $\{k_0, \ldots, k_m\}$, we make the implicit assumption that the elements are ordered, viz., $k_0 < k_1 < \cdots < k_m$.*

Define the injection $\iota \colon \Gamma^* \to \mathcal{S}(\Gamma)^*$ by

$$\iota(\varepsilon) = \varepsilon \qquad \text{and} \qquad \iota(\eta) = \overline{\eta} \quad \text{for every} \quad \eta \in \Gamma^+ .$$

Consider $\eta \in \Gamma^*$ and $\eta', \eta'' \in \mathcal{S}(\Gamma)^*$ with $\eta', \eta'' \preceq \eta$. We write

$$\eta' \preceq \eta'' \qquad \text{if} \qquad E(\eta') \supseteq E(\eta''),$$

and denote the unique $\kappa \preceq \eta$ with $E(\kappa) = E(\eta') \cup E(\eta'')$ by $\eta' \curlywedge \eta''$. By this definition, clearly

$$\eta' \curlywedge \eta'' \preceq \eta' \qquad \text{and} \qquad \eta' \curlywedge \eta'' \preceq \eta'' .$$

Intuitively, $\eta' \curlywedge \eta''$ is the coarsest subdivision of η that refines both η' and η''. It is easy to see that the operation $\curlywedge$ is associative. Regarding the length of $\eta' \curlywedge \eta''$ as an element of $\mathcal{S}(\Gamma)^*$, we obtain the following bound.

Lemma 3.3. *Let $\eta \in \Gamma^*$ and $\eta', \eta'' \preceq \eta$. Then*

(i) $|\eta' \curlywedge \eta''| \leq |\eta'| + |\eta''| - 1$ *if $\eta \in \Gamma^+$, and*

(ii) $|\eta' \curlywedge \eta''| = 0$ *if $\eta = \varepsilon$.*

Proof. If $\eta \in \Gamma^+$, then

$$|\eta' \curlywedge \eta''| = |E(\eta' \curlywedge \eta'')| - 1 \leq |E(\eta')| + |E(\eta'')| - 3 = |\eta'| + |\eta''| - 1\,,$$

as $E(\eta')$ and $E(\eta'')$ have at least two indices in common. The property is trivial if $\eta = \varepsilon$. $\square$

If $\eta' \in \mathcal{S}(\Gamma)^+$ is the $\{k_0, \dots, k_m\}$-subdivision of $\eta \in \Gamma^+$, then the $\{|\eta| - k_m, \dots, |\eta| - k_1\}$-subdivision of the reversal η^R of η will be denoted by $(\eta')^R$. If $\eta = \varepsilon$ instead, then $(\eta')^R = \varepsilon$.[1]

Example 3.4. Consider the word

$$\eta = abbabbababbaab\,,$$

and its two subdivisions

$$\eta' = \overline{abba}\ \overline{bbabab}\ \overline{baab} \qquad \text{and} \qquad \eta'' = \overline{abbabb}\ \overline{abab}\ \overline{ba}\ \overline{ab}\ .$$

Clearly, η' is the $\{0, 4, 10, 14\}$-subdivision of η, and η'' is its $\{0, 6, 10, 12, 14\}$-subdivision. Therefore,

$$\eta' \curlywedge \eta'' = \overline{abba}\ \overline{bb}\ \overline{abab}\ \overline{ba}\ \overline{ab}\ ,$$

the $\{0, 4, 6, 10, 12, 14\}$-subdivision of η. We have that $\eta' \curlywedge \eta'' \preceq \eta'$, since the former's "blocks" are finer than the latter's – formally, $\{0, 4, 6, 10, 12, 14\} \supseteq \{0, 4, 10, 14\}$. Finally,

$$\eta^R = baabbababbabba \qquad \text{and} \qquad (\eta'')^R = \overline{ba}\ \overline{ab}\ \overline{baba}\ \overline{bbabba}\ ,$$

and $(\eta'')^R$ is the $\{0, 2, 4, 8, 14\}$-subdivision of η^R, because

$$\{0, 2, 4, 8, 14\} \ = \ \{14 - 14, 14 - 12, 14 - 10, 14 - 6, 14 - 0\}\,. \qquad \triangleleft$$

The following lemma describes the interplay of concatenation and subdivision.

Lemma 3.5. *Let $\eta \in \Gamma^*$ and let $\eta' \preceq \eta$ such that $\eta' = \eta'_1\eta'_2$ for some η'_1, $\eta'_2 \in \mathcal{S}(\Gamma)^*$. For every $\eta'' \preceq \eta'$, there are η''_1 and $\eta''_2 \in \mathcal{S}(\Gamma)^*$ such that*

$$\eta''_1 \preceq \eta'_1\,, \qquad \eta''_2 \preceq \eta'_2 \qquad \text{and} \qquad \eta'' = \eta''_1\eta''_2\,.$$

Proof. Since $E(\eta') \subseteq E(\eta'')$, the factorization of η' can be transferred to η''. $\square$

We are now in a position to define the *compact pts $M^\sharp = (Q, \Sigma, \Gamma_\sharp, q_0, R_\sharp)$ of M,* where $\Gamma_\sharp = \mathcal{S}(\Gamma)$, and $R_\sharp$ contains the following rules:

(i) For every q_1, $q_2 \in Q$ and $\eta \in \Gamma^+$ such that $q_1(\varepsilon) \Rightarrow_{r_1 \dots r_k} q_2(\eta)$ for some $k \in \mathbb{N}_1$ and $r_1, \dots, r_k \in R_\uparrow$, the set $R_\sharp$ contains the rule

$$q_1(x) \to q_2(\overline{\eta}\ x)\,.$$

The resulting rule will be denoted by $\overline{r_1 \dots r_k}$.

[1] Note that this definition clashes with the general definition of the reversal w^R of a word w. However, we will not consider the reversal of a word over $\mathcal{S}(\Gamma)$ in this chapter, so the notation should lead to no confusion.

(ii) For every q_1, $q_2 \in Q$ and $\eta \in \Gamma^+$ such that $q_1(\eta) \Rightarrow_{r_1 \dots r_k} q_2(\varepsilon)$ for some $k \in \mathbb{N}_1$ and $r_1, \dots, r_k \in R_\downarrow$, the set $R_\sharp$ contains the rule

$$q_1(\ulcorner \eta \urcorner\, x) \to q_2(x)\,.$$

The resulting rule is denoted by $\ulcorner r_1 \dots r_k \urcorner$.

(iii) For every rule $r \in R_\Sigma$, $R_\sharp$ contains a rule denoted by $\ulcorner r \urcorner$, which is identical to r.

Remark 3.6. In general, $M^\sharp$ is an infinite object. However, with the help of the concept of subdivision, we will be able to analyze the shape of derivations of $M^\sharp$ formally, and give bounds for their size and length. Later, in Section 3.1.4, we will show how to represent $M^\sharp$ finitely, and also how to transfer the established bounds to this representation. $\triangleleft$

First of all, we demonstrate that $M^\sharp$ and M recognize the same tree language.

Lemma 3.7. $\mathcal{L}(M^\sharp) = \mathcal{L}(M)$.

Proof. For the direction $\mathcal{L}(M^\sharp) \supseteq \mathcal{L}(M)$, observe that for every push rule $q(x) \to p(\gamma x)$ in R, there is a corresponding rule $q(x) \to p(\ulcorner \gamma \urcorner\, x)$ in $R_\sharp$, and analogously for rules from $R_\downarrow$ and R_Σ. Thus, for every derivation d of some $t \in \mathrm{T}_\Sigma$ from $q_0(\varepsilon)$ in M, one can easily construct a derivation of t from $q_0(\varepsilon)$ in $M^\sharp$.

The reverse direction $\mathcal{L}(M^\sharp) \subseteq \mathcal{L}(M)$ is a consequence of the following stronger property. We prove that, for every $n \in \mathbb{N}$, $q(\eta) \in Q(\Gamma_\sharp^*)$ and $t \in \mathrm{T}_\Sigma$,

$$q(\eta) \Rightarrow^n_{M^\sharp} t \qquad \text{implies} \qquad q(h(\eta)) \Rightarrow^*_M t\,,$$

where $h\colon \Gamma_\sharp^* \to \Gamma^*$ is the homomorphism given by $h(\ulcorner \eta \urcorner) = \eta$ for every $\ulcorner \eta \urcorner \in \Gamma_\sharp$.

The proof is by complete induction on n. The case $n = 0$ is vacuous, so assume that $q(\eta) \Rightarrow_r \xi \Rightarrow^n_{M^\sharp} t$ for some rule $r \in R_\sharp$ and $\xi \in \mathrm{T}_\Sigma(Q(\Gamma_\sharp^*))$. We make a case analysis on the form of r.

(I) If r is a copy rule of the form $q(x) \to \sigma(p_1(x), \dots, p_k(x))$, then

$$\xi = \sigma\big(p_1(\eta), \dots, p_k(\eta)\big)\,,$$

and for each $i \in [k]$, there is some $n_i \le n$ such that $p_i(\eta) \Rightarrow^{n_i}_{M^\sharp} t|_i$. By the induction hypothesis, $p_i(h(\eta)) \Rightarrow^*_M t|_i$, and as the copy rule $q(x) \to \sigma(p_1(x), \dots, p_k(x))$ is in R by construction of $M^\sharp$,

$$q\big(h(\eta)\big) \Rightarrow_M \sigma\big(p_1(h(\eta)), \dots, p_k(h(\eta))\big) \Rightarrow^*_M t\,.$$

(II) If r is a push rule of the form $q(x) \to p(\ulcorner \kappa \urcorner\, x)$, for some $\kappa \in \Gamma^+$, then $\xi = p(\ulcorner \kappa \urcorner\, \eta)$. By the induction hypothesis, $p(h(\ulcorner \kappa \urcorner\, \eta)) \Rightarrow^*_M t$. Furthermore, by construction of r, we have $q(\varepsilon) \Rightarrow^*_M p(\kappa)$. Thus also

$$q\big(h(\eta)\big) \Rightarrow^*_M p\big(\kappa h(\eta)\big) = p\big(h(\ulcorner \kappa \urcorner\, \eta)\big) \Rightarrow^*_M t\,.$$

(III) Finally, assume that r is a pop rule of the form $q(\overbracket{\kappa}\, x) \to p(x)$, for some $\kappa \in \Gamma^+$. Then $\eta = \overbracket{\kappa}\, \tau$ for some $\tau \in \Gamma_\sharp^*$ and $\xi = p(\tau)$. By the induction hypothesis, $p(h(\tau)) \Rightarrow_M^* t$. Additionally, by construction of r, we have that $q(\kappa) \Rightarrow_M^* p(\varepsilon)$. Hence

$$q\big(h(\eta)\big) = q\big(\kappa h(\tau)\big) \Rightarrow_M^* p\big(h(\tau)\big) \Rightarrow_M^* t\,.$$

This concludes the case distinction. Consider now some tree $t \in \mathcal{L}(M^\sharp)$. Then $q_0(\varepsilon) \Rightarrow_{M^\sharp}^* t$. By the property just shown, we obtain that $q_0(\varepsilon) = q_0(h(\varepsilon)) \Rightarrow_M^* t$. Hence $t \in \mathcal{L}(M)$, and therefore $\mathcal{L}(M^\sharp) \subseteq \mathcal{L}(M)$. □

By the notation for the rules of $M^\sharp$, we have $R_\sharp \subseteq \mathcal{S}(R)$. Therefore, the notion of subdivision, and of the relation $\preceq$, carry over to derivations of $M^\sharp$. Moreover, it is easy to see that $R_\sharp$ is closed under the operation $\curlywedge$: for every $r_1, r_2 \in R_\sharp$, also $r_1 \curlywedge r_2 \in R_\sharp$.

As shown in the following lemma, in a derivation d in $M^\sharp$, a subdivision of an occurring pushdown determines a corresponding subdivision d' of d, and vice versa.

Lemma 3.8. *Let $q, p \in Q$, $\eta \in \Gamma^*$, and $d \in R^*$. Moreover, let $d' \preceq d$ and $\eta' \preceq \eta$.*

(i) If $d \in R_\downarrow^*$ with $q(\eta) \Rightarrow_d p(\varepsilon)$, then $q(\eta') \Rightarrow_{d'} p(\varepsilon)$ if and only if $E(\eta') = E(d')$.

(ii) If $d \in R_\uparrow^*$ with $q(\varepsilon) \Rightarrow_d p(\eta)$, then $q(\varepsilon) \Rightarrow_{d'} p(\eta')$ if and only if $E((\eta')^R) = E(d')$.

Proof. For statement *(i)*, let $\eta = \gamma_1 \cdots \gamma_n$ and $d = r_1 \cdots r_n$ for some $n \in \mathbb{N}$, and $q_0, \ldots, q_n \in Q$ such that

$$q_0(\gamma_1 \cdots \gamma_n) \Rightarrow_{r_1} q_1(\gamma_2 \cdots \gamma_n) \Rightarrow_{r_2} \cdots \Rightarrow_{r_n} q_n(\varepsilon)\,.$$

Assume $\eta' \preceq \eta$ and $d' \preceq d$. The equivalence is trivial for $n = 0$, so assume $n > 0$. Let $E(\eta') = \{k_0, \ldots, k_m\}$ and $E(d') = \{l_0, \ldots, l_{m'}\}$ for some $m, m' \in \mathbb{N}$. By definition of $M^\sharp$,

$$\begin{aligned} q_0\Big(\overbracket{\gamma_{k_0+1}\cdots\gamma_{k_1}} \cdots \overbracket{\gamma_{k_{m-1}+1}\cdots\gamma_{k_m}}\Big) &\Rightarrow_{\underbracket{r_{l_0+1}\cdots r_{l_1}}} q_{k_1}\Big(\overbracket{\gamma_{k_1+1}\cdots\gamma_{k_2}} \cdots \overbracket{\gamma_{k_{m-1}+1}\cdots\gamma_{k_m}}\Big) \\ &\;\;\vdots \\ &\Rightarrow_{\underbracket{r_{l_{m'-1}+1}\cdots r_{l_{m'}}}} q_{k_m}(\varepsilon) \end{aligned}$$

if and only if $\{k_0, \ldots, k_m\} = \{l_0, \ldots, l_{m'}\}$, which is equivalent to $E(\eta') = E(d')$.

Statement *(ii)* is proven analogously, but we must take care that the pushdown η is written from right to left. Let $\eta = \gamma_n \cdots \gamma_1$ and $d = r_1 \cdots r_n$ for some $n \in \mathbb{N}$, and $q_0, \ldots, q_n \in Q$ with

$$q_0(\varepsilon) \Rightarrow_{r_1} q_1(\gamma_1) \Rightarrow_{r_2} \cdots \Rightarrow_{r_n} q_n(\gamma_n \cdots \gamma_1)\,.$$

Let $\eta' \preceq \eta$ and $d' \preceq d$. Again, the case $n = 0$ is easy, so let $n > 0$. By definition of $M^\sharp$,

$$\begin{aligned} q_0(\varepsilon) &\Rightarrow_{\underbracket{r_{l_0+1}\cdots r_{l_1}}} q_{k_1}\Big(\overbracket{\gamma_{k_1}\cdots\gamma_{k_0+1}}\Big) \\ &\;\;\vdots \\ &\Rightarrow_{\underbracket{r_{l_{m'-1}+1}\cdots r_{l_{m'}}}} q_{k_m}\Big(\overbracket{\gamma_{k_m}\cdots\gamma_{k_{m-1}+1}} \cdots \overbracket{\gamma_{k_1}\cdots\gamma_{k_0+1}}\Big) \end{aligned}$$

if and only if $\{k_0, \ldots, k_m\} = \{l_0, \ldots, l_{m'}\}$, and the latter is equivalent to $E\big((\eta')^R\big) = E(d')$. □

Let $M = (Q, \Sigma, \Gamma, q_0, R)$ be a pts. We will now introduce a restricted mode of derivation for M, which disallows pushdowns whose size exceeds a certain bound. Let $\mu \in \mathbb{N}$ and $\xi \in \mathrm{T}_\Sigma(Q(\Gamma^*))$. We say that ξ *has μ-bounded pushdowns* if for every subtree of ξ of form $q(\eta) \in Q(\Gamma^*)$, we have that $|\eta| \leq \mu$. Put simply, the size of every pushdown occurring in ξ is at most μ.

Let moreover $\xi, \zeta \in \mathrm{T}_\Sigma(Q(\Gamma^*))$. For every $r \in R$, we write $\xi \overset{(\mu)}{\Longrightarrow}_r \zeta$ if $\xi \Rightarrow_r \zeta$ and both ξ and ζ have μ-bounded pushdowns. Moreover, we define

$$\overset{(\mu)}{\Longrightarrow}_M = \bigcup_{r \in R} \overset{(\mu)}{\Longrightarrow}_r \qquad \text{and} \qquad \overset{(\mu)}{\Longrightarrow}_d = \overset{(\mu)}{\Longrightarrow}_{r_1} ; \cdots ; \overset{(\mu)}{\Longrightarrow}_{r_n}$$

for every derivation $d = r_1 \cdots r_n$, where $n \in \mathbb{N}$ and $r_1, \ldots, r_n \in R$. Observe that in the latter case, where we apply the composition of relations, *all* intermediate trees produced by d are required to have μ-bounded pushdowns.

We will continue with showing that in a derivation $q_0(\eta) \overset{(\mu)}{\Longrightarrow}_d t$ of a tree $t \in \mathcal{L}(M)$, μ can be bounded by a polynomial in $|t|$. First we require the following auxiliary lemma, which states how much μ must grow in order to further subdivide a pushdown.

Lemma 3.9. *Let M be succinct, and consider $q(\eta) \in Q(\Gamma^*)$, $\eta' \preceq \eta$, $d \in \mathcal{DS}_M$, $d' \preceq d$, $t \in \mathrm{T}_\Sigma$, and $\mu \in \mathbb{N}$ with*

$$q(\eta) \Rightarrow_d t \qquad \textit{and} \qquad q(\eta') \overset{(\mu)}{\Longrightarrow}_{d'} t\,.$$

For every subdivision $\eta'' \preceq \eta'$, there is a derivation $d'' \preceq d'$ such that

$$q(\eta'') \overset{(\mu')}{\Longrightarrow}_{d''} t\,, \qquad \textit{where} \qquad \mu' = \mu + |\eta''| - |\eta'|\,.$$

Proof. Consider some $\eta'' \preceq \eta'$, and let $\mu' = \mu + |\eta''| - |\eta'|$. We will show that the stated property holds by structural induction on t. For this purpose, let $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, as well as $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$ such that $t = \sigma(t_1, \ldots, t_k)$. As $d \in \mathcal{DS}_M$, there are some $e_1 \in R_\downarrow^*$, $e_2 \in R_\uparrow^*$, $r \in R_\Sigma$, and $d_1, \ldots, d_k \in \mathcal{DS}_M$ such that

$$d = e_1 e_2 r d_1 \cdots d_k\,.$$

In particular, there are u, p, $p_1, \ldots, p_k \in Q$, and $\tau \in \Gamma^*$ such that

$$q(\eta_1\eta_2) \Rightarrow_{e_1} u(\eta_2) \Rightarrow_{e_2} p(\eta_3\eta_2) \Rightarrow_r \sigma\big(p_1(\tau), \ldots, p_k(\tau)\big) \Rightarrow_{d_1} \cdots \Rightarrow_{d_k} t\,,$$

for some $\eta_1, \eta_2, \eta_3 \in \Gamma^*$ with $\eta = \eta_1\eta_2$ and $\tau = \eta_3\eta_2$. By definition of $M^\sharp$, we have

$$d' = e_1' e_2' \overset{\sqcap}{r}\, d_1' \cdots d_k'$$

for some $e_1' \preceq e_1$, $e_2' \preceq e_2$, and $d_1' \preceq d_1, \ldots, d_k' \preceq d_k$. Furthermore,

$$q(\eta') \overset{(\mu)}{\Longrightarrow}_{e_1'e_2'} p(\tau')\,,$$

where $\eta' = \eta'_1\eta'_2$, $\tau' = \eta'_3\eta'_2$, and $\eta'_i \preceq \eta_i$ for every $i \in [3]$. Observe that $|\tau'| \leq \mu$. As $\eta'' \preceq \eta'$, by Lemma 3.5 there are $\eta''_1 \preceq \eta'_1$ and $\eta''_2 \preceq \eta'_2$ such that $\eta'' = \eta''_1\eta''_2$. Note that $|\eta''_1| \geq |\eta'_1|$. Let e''_1 be the $E(\eta''_1)$-subdivision of η_1. Then, by Lemma 3.8, $q(\eta''_1\eta''_2) \Rightarrow_{e''_1} u(\eta''_2)$. In fact,

$$\begin{aligned} |\eta''| &= |\eta'| + |\eta''| - |\eta'| \\ &\leq \mu + |\eta''| - |\eta'|, && \text{(as } |\eta'| \leq \mu) \end{aligned}$$

and hence

$$q(\eta''_1\eta''_2) \overset{(\mu')}{\Longrightarrow}_{e''_1} u(\eta''_2)\,.$$

Moreover, as $u(\eta'_2) \Rightarrow_{e'_2} p(\eta'_3\eta'_2)$, we also have $u(\eta''_2) \Rightarrow_{e'_2} p(\eta'_3\eta''_2)$. Let $\tau'' = \eta'_3\eta''_2$. Then

$$\begin{aligned} |\tau''| &= |\eta'_3| + |\eta''_2| \\ &= |\eta'_3| + |\eta'_2| + |\eta'_1| + |\eta''_2| - (|\eta'_2| + |\eta'_1|) \\ &= |\tau'| + |\eta'_1| + |\eta''_2| - (|\eta'_1| + |\eta'_2|) && \text{(as } \tau' = \eta'_3\eta'_2) \\ &\leq \mu + |\eta''_1| + |\eta''_2| - (|\eta'_1| + |\eta'_2|) && \text{(as } |\tau'| \leq \mu \text{ and } |\eta'_1| \leq |\eta''_1|) \\ &= \mu + |\eta''| - |\eta'|, && \text{(as } \eta'' = \eta''_1\eta''_2 \text{ and } \eta' = \eta'_1\eta'_2) \end{aligned}$$

and thus

$$u(\eta''_2) \overset{(\mu')}{\Longrightarrow}_{e'_2} p(\eta'_3\eta''_2)\,.$$

Since $\tau'' \preceq \tau'$, the induction hypothesis implies that for every $i \in [k]$, there is some $d''_i \preceq d'_i$ such that

$$p_i(\tau'') \overset{(\mu'')}{\Longrightarrow}_{d''_i} t_i \qquad \text{and} \qquad \mu'' = \mu + |\tau''| - |\tau'|\,.$$

Hence

$$\begin{aligned} \mu'' &= \mu + |\tau''| - |\tau'| \\ &= \mu + |\eta'_3| + |\eta''_2| - |\eta'_3| - |\eta'_2| && \text{(as } \tau'' = \eta'_3\eta''_2 \text{ and } \tau' = \eta'_3\eta'_2) \\ &= \mu + |\eta''_2| - |\eta'_2| \\ &\leq \mu + |\eta''_1| + |\eta''_2| - |\eta'_1| - |\eta'_2| && \text{(as } |\eta''_1| \geq |\eta'_1|) \\ &= \mu + |\eta''| - |\eta'| && \text{(as } \eta'' = \eta''_1\eta''_2 \text{ and } \eta' = \eta'_1\eta'_2) \\ &= \mu'\,. \end{aligned}$$

Thus for each $i \in [k]$, we have $p_i(\tau'') \overset{(\mu')}{\Longrightarrow}_{d''_i} t_i$. We set

$$d'' = e''_1 e'_2 \overset{\sqcap}{r}\, d''_1 \cdots d''_k\,,$$

yielding $q(\eta'') \overset{(\mu')}{\Longrightarrow}_{d''} t$. □

Convention. *In the following, we denote the number $2 \cdot |t|$ by $\mu(t)$, for every tree $t \in \mathrm{T}_\Sigma$.*

We can now prove the polynomial size bound of pushdowns occurring in derivations of $M^\sharp$.

Lemma 3.10. *Let M be succinct. For every $q(\eta) \in Q(\Gamma^*)$, $t \in \mathrm{T}_\Sigma$ and $d \in \mathcal{DS}_M(q(\eta), t)$, there are $\eta' \preceq \eta$ and $d' \preceq d$ such that $q(\eta') \xRightarrow{(\mu(t))}_{d'} t$.*

Proof. The proof is by structural induction on t, therefore let $t = \sigma(t_1, \dots, t_k)$ for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$ and $t_1, \dots, t_k \in \mathrm{T}_\Sigma$. Moreover, let

$$d = e_1 e_2 r d_1 \cdots d_k$$

such that $e_1 \in R_\downarrow^*$, $e_2 \in R_\uparrow^*$, $r \in R_\Sigma$, and $d_1, \dots, d_k \in \mathcal{DS}_M$. Thus there are η_1, η_2, η_3, and $\tau \in \Gamma^*$ with $\eta = \eta_1\eta_2$ and $\tau = \eta_3\eta_2$, as well as u, p, $p_1, \dots, p_k \in Q$, satisfying

$$q(\eta_1\eta_2) \Rightarrow_{e_1} u(\eta_2) \Rightarrow_{e_2} p(\eta_3\eta_2) \Rightarrow_r \sigma\big(p_1(\tau), \dots, p_k(\tau)\big) \Rightarrow_{d_1} \cdots \Rightarrow_{d_k} t\,.$$

By the induction hypothesis, there are subdivisions $\tau'_1, \dots, \tau'_k \preceq \tau$ and respective derivations $d'_1 \preceq d_1, \dots, d'_k \preceq d_k$ such that

$$|\tau'_i| \le \mu(t_i) \qquad \text{and} \qquad p_i(\tau'_i) \xRightarrow{(\mu(t_i))}_{d'_i} t_i$$

for every $i \in [k]$. Let

$$\tau' = \tau'_1 \curlywedge \cdots \curlywedge \tau'_k \curlywedge \big(\iota(\eta_3)\iota(\eta_2)\big)\,.$$

In particular, if $k = 0$, then $\tau' = \iota(\eta_3)\iota(\eta_2)$. If $\tau = \varepsilon$, then $\tau' = \varepsilon$ and $|\tau'| \le \mu(t)$. If otherwise $\tau \ne \varepsilon$, then

$$\begin{aligned}
|\tau'| &\le \Big(\sum_{i\in[k]} |\tau'_i|\Big) + 2 - k && \text{(applying Lemma 3.3 k times)}\\
&\le \Big(\sum_{i\in[k]} \mu(t_i)\Big) + 2 - k && \text{(as $|\tau'_i| \le \mu(t_i)$)}\\
&= \mu(t) - k && \text{(since $\mu(t) = 2\cdot(|t_1| + \cdots + |t_k| + 1)$)} \qquad (3.1)\\
&\le \mu(t)\,.
\end{aligned}$$

By this case distinction,

$$p(\tau') \xRightarrow{(\mu(t))}_{\underset{r}{\sqcap}} \sigma(p_1(\tau'), \dots, p_k(\tau'))\,.$$

Let $j \in [k]$. Because $\tau' \preceq \tau'_j$, by Lemma 3.9, there is some $d''_j \preceq d'_j$ such that $p_j(\tau') \xRightarrow{(\mu')}_{d''_j} t_j$, and where $\mu' = \mu(t_j) + |\tau'| - |\tau'_j|$. If $\tau = \varepsilon$, then $\mu' = \mu(t_j) \le \mu(t)$. Otherwise,

$$\begin{aligned}
\mu' &= \mu(t_j) + |\tau'| - |\tau'_j|\\
&\le \mu(t_j) + \Big(\sum_{i\in[k]} |\tau'_i|\Big) + 2 - |\tau'_j| && \text{(applying Lemma 3.3 as above)}\\
&\le \Big(\sum_{i\in[k]} \mu(t_i)\Big) + 2 && \text{(as $|\tau'_i| \le \mu(t_i)$ for $i \in [k]\setminus\{j\}$)}\\
&= \mu(t)\,.
\end{aligned}$$

By these two cases, also

$$p_j(\tau') \overset{(\mu(t))}{\Longrightarrow}_{d''_j} t_j\,.$$

By definition of τ', Lemma 3.5 implies that there are some $\eta'_2 \preceq \eta_2$ and $\eta'_3 \preceq \eta_3$ such that $\tau' = \eta'_3\eta'_2$. Set $\eta' = \iota(\eta_1)\eta'_2$. If $k = 0$, then by the definition of τ', we have $|\eta'_2| \leq 1 < \mu(t)$. If $k > 0$, then by (3.1) from above, $|\eta'_2| \leq |\tau'| < \mu(t)$. Thus in both cases $|\eta'| \leq \mu(t)$. Hence

$$q(\eta') \overset{(\mu(t))}{\Longrightarrow}_{\iota(e_1)} u(\eta'_2)\,.$$

Moreover, as $\eta'_3 \preceq \iota(\eta_3)$, by Lemma 3.8, there is some $e'_2 \preceq e_2$ with

$$u(\eta'_2) \overset{(\mu(t))}{\Longrightarrow}_{e'_2} p(\eta'_3\eta'_2)\,.$$

We let

$$d' = \iota(e_1)e'_2\, \overset{\sqcap}{r}\, d''_1 \cdots d''_k\,,$$

then $q(\eta') \overset{(\mu(t))}{\Longrightarrow}_{d'} t$, and the proof is concluded. □

The following lemma shows that since only bounded pushdowns are required for derivations in $M^\sharp$ (as demonstrated in Lemma 3.10), the corresponding derivations are bounded in their length.

Lemma 3.11. *Let M be succinct. For every $t \in \mathcal{L}(M)$, there is a derivation $d' \in \mathcal{D}_{M^\sharp}(q_0(\varepsilon), t)$ with $|d'| \leq \mu(t)^2 + \mu(t)$.*

Proof. Let $t \in \mathcal{L}(M)$, let $d \in \mathcal{DS}_M(q_0(\varepsilon), t)$, and consider the derivation d' as constructed in Lemma 3.10. We prove for every $w \in \mathrm{pos}(t)$, and every factor d'' of d', where $d'' \in \mathcal{D}_{M^\sharp}(q(\eta'), t|_w)$ for some $q(\eta') \in Q(\Gamma^*_\sharp)$, that

$$|d''| \leq (\mu(t) + 1) \cdot \mu(t|_w)\,.$$

The proof is by well-founded induction using the relation "is child node of" on $\mathrm{pos}(t)$. For this purpose, let $t|_w = \sigma(t_1, \ldots, t_k)$ for some $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$. We know that d'' is of the form

$$e_1 e_2\, \overset{\sqcap}{r}\, d'_1 \cdots d'_k$$

for some $e_1 \in (R_\sharp)^*_\downarrow$, $e_2 \in (R_\sharp)^*_\uparrow$, $r \in R_\Sigma$, $u, p_1, \ldots, p_k \in Q$, κ', $\tau' \in \Gamma^*_\sharp$, and $d'_i \in \mathcal{D}_{M^\sharp}(p_i(\tau'), t_i)$, for $i \in [k]$, and

$$q(\eta') \overset{(\mu(t))}{\Longrightarrow}_{e_1} u(\kappa') \overset{(\mu(t))}{\Longrightarrow}_{e_2} p(\tau') \overset{(\mu(t))}{\Longrightarrow}_{\overset{\sqcap}{r}} \sigma\big(p_1(\tau'), \ldots, p_k(\tau')\big)\,.$$

As the pushdowns η' and τ' are bounded in their size by $\mu(t)$,

$$|e_1 e_2\, \overset{\sqcap}{r}\,| \leq 2 \cdot \mu(t) + 1\,.$$

By the induction hypothesis, $|d_i'| \le (\mu(t)+1)\cdot\mu(t_i)$, so we obtain

$$\begin{aligned}|d''| &\le 2\cdot(\mu(t)+1) + \sum_{i\in[k]}\big((\mu(t)+1)\cdot\mu(t_i)\big)\\ &= (\mu(t)+1)\cdot\big(2+\sum_{i\in[k]}\mu(t_i)\big)\\ &= (\mu(t)+1)\cdot\mu(t|_w)\,.\end{aligned}$$

The lemma follows from the property when we choose $w=\varepsilon$ and $d''=d'$. □

3.1.4 Representing $M^\sharp$ by a Finite Object

In this section, we show how to construct from M a finite representation $M^\dagger$ of $M^\sharp$. Let $\Gamma_\dagger = \mathcal{P}(Q\times Q)$ and define a mapping $h\colon \Gamma \to \Gamma_\dagger$ such that, for every $\gamma\in\Gamma$,

$$h(\gamma) = \big\{(q,p) \mid q(\gamma x)\to p(x) \text{ in } R\big\}\,.$$

Define the pta $M^\dagger = (Q,\Sigma,\Gamma_\dagger,q_0,R_\dagger)$, where $R_\dagger$ is the smallest set R' such that

(i) $R_\Sigma \subseteq R'$,

(ii) for every rule $q(x)\to p(\gamma x)$ in R, R' contains the rule

$$q(x)\to p\big(h(\gamma)x\big)\,,$$

(iii) for every rule $q(x)\to p(Ux)$ and $p(x)\to u(Vx)$ in R', where $U, V\in\Gamma_\dagger$, R' also contains the rule

$$q(x)\to u\big((V\circ U)x\big)\,,$$

(iv) for every $U\in\Gamma_\dagger$ and $(q,p)\in U$, R' contains the rule

$$q(Ux)\to p(x)\,.$$

Note that $R_\dagger$ is given effectively by these conditions.

Remark 3.12. The size of $M^\dagger$ is in general exponential in $|M|$, due to rule *(iii)* from above. The principal reason for this is that, given some set Q, the relation monoid $(\{\mathrm{id}_A \mid A\subseteq Q\},\circ,\mathrm{id}_Q)$ of partial identities on Q can be generated by the elements of

$$G = \{\mathrm{id}_Q\}\cup\big\{\mathrm{id}_Q\setminus\{(q,q)\}\;\big|\; q\in Q\big\}\,.$$

Clearly, if Q is a finite set of cardinality $n\in\mathbb{N}$, then the monoid has 2^n elements, while $|G| = n+1$. ◁

We show that $M^\dagger$ is indeed a faithful representation of $M^\sharp$. For this purpose, extend h to $\tilde{h}\colon \Gamma^+ \to \Gamma_\dagger$ by defining

$$\tilde{h}(\gamma_1 \cdots \gamma_k) = h(\gamma_1) \circ \cdots \circ h(\gamma_k)$$

for every $k \in \mathbb{N}_1$ and $\gamma_1, \ldots, \gamma_k \in \Gamma$. Further, extend $\tilde{h}$ to $\hat{h}\colon \Gamma_\sharp^* \to \Gamma_\dagger^*$ by defining

$$\hat{h}(\overline{\eta_1} \cdots \overline{\eta_k}) = \tilde{h}(\eta_1) \cdots \tilde{h}(\eta_k)$$

for every $k \in \mathbb{N}$ and $\eta_1, \ldots, \eta_k \in \Gamma^+$. We identify h, $\tilde{h}$, and $\hat{h}$ from this point onwards. There is the following close relation between $M^\sharp$ and $M^\dagger$.

Lemma 3.13. *For every n, $\mu \in \mathbb{N}$, $q(\eta) \in Q(\Gamma_\sharp^*)$, and for every $t \in \mathrm{T}_\Sigma$,*

$$q(\eta) \overset{(\mu)}{\Longrightarrow}{}^n_{M^\sharp} t \qquad \textit{if and only if} \qquad q\big(h(\eta)\big) \overset{(\mu)}{\Longrightarrow}{}^n_{M^\dagger} t\,.$$

Proof. First we will prove the direction “only if” of the equivalence, using complete induction on n. If $n = 0$, the implication is vacuously true. Hence assume that

$$q(\eta) \overset{(\mu)}{\Longrightarrow}_r \xi \overset{(\mu)}{\Longrightarrow}{}^n_{M^\sharp} t$$

for some $n \in \mathbb{N}$, $r \in R_\sharp$ and $\xi \in \mathrm{T}_\Sigma(Q(\Gamma_\sharp^*))$. We proceed by a case analysis on r.

(I) If r is a copy rule of form $q(x) \to \sigma(p_1(x), \ldots, p_k(x))$, then $t = \sigma(t_1, \ldots, t_k)$ for some trees $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$, $\xi = \sigma(p_1(\eta), \ldots, p_k(\eta))$, and for every $i \in [k]$, we have

$$p_i(\eta) \overset{(\mu)}{\Longrightarrow}{}^{n_i}_{M^\sharp} t_i$$

for some $n_i \in \mathbb{N}$ such that $n = \sum_{i=1}^k n_i$. Thus by the induction hypothesis,

$$p_i(h(\eta)) \overset{(\mu)}{\Longrightarrow}{}^{n_i}_{M^\dagger} t_i\,,$$

and, by construction,

$$q(h(\eta)) \overset{(\mu)}{\Longrightarrow}_r \sigma\big(p_1(h(\eta)), \ldots, p_k(h(\eta))\big) \overset{(\mu)}{\Longrightarrow}{}^n_{M^\dagger} t\,.$$

(II) If r is a push rule of form $q(x) \to p(\overline{\gamma_k \cdots \gamma_1}\, x)$ for some $\gamma_1, \ldots, \gamma_k \in \Gamma$, and $k \in \mathbb{N}_1$, then $|\eta| < \mu$ and $\xi = p(\overline{\gamma_k \cdots \gamma_1}\, \eta)$. By construction of $R_\sharp$, there is a derivation

$$q(\eta) \Rightarrow^k_M p(\gamma_k \cdots \gamma_1 \eta)$$

by a sequence of push rules $q_{i-1}(x) \to q_i(\gamma_i x)$ from R, where $i \in [k]$, such that $q = q_0$ and $q_k = p$. Thus $R_\dagger$ contains the rules

$$q_{i-1}(x) \to q_i(h(\gamma_i)x) \quad \text{for each } i \in [k], \qquad \text{and} \qquad q(x) \to p\Big(\big(h(\gamma_k) \circ \cdots \circ h(\gamma_1)\big)\, x\Big).$$

By the definition of h,

$$h(\overline{\gamma_k \cdots \gamma_1}\, \eta) = \big(h(\gamma_k) \circ \cdots \circ h(\gamma_1)\big)\, h(\eta)\,,$$

and the length of this word is at most μ. Thus by the induction hypothesis,

$$p\big((h(\gamma_k) \circ \cdots \circ h(\gamma_1))\, h(\eta)\big) \overset{(\mu)}{\Longrightarrow}{}^n_{M^\dagger} t\,.$$

Therefore, $q(h(\eta)) \overset{(\mu)}{\Longrightarrow}{}^{n+1}_{M^\dagger} t$.

(III) Finally, if r is a pop rule of form $q(\overline{\gamma_1 \cdots \gamma_k}\, x) \to p(x)$ with $\gamma_1, \ldots, \gamma_k \in \Gamma$, and $k \in \mathbb{N}_1$, then $\eta = \overline{\gamma_1 \cdots \gamma_k}\, \kappa$ for some $\kappa \in \Gamma_\sharp^*$, and $\xi = p(\kappa)$. By construction of $M^\sharp$, there are rules $q_{i-1}(\gamma_i x) \to q_i(x)$ in R, for every $i \in [k]$, such that $q = q_0$ and $q_k = p$. Thus $(q_{i-1}, q_i) \in h(\gamma_i)$, and in particular,

$$(q, p) \in \big(h(\gamma_1) \circ \cdots \circ h(\gamma_k)\big).$$

Observe again that $h(\gamma_1) \circ \cdots \circ h(\gamma_k) = h(\overline{\gamma_1 \cdots \gamma_k})$. Therefore $q(h(\overline{\gamma_1 \cdots \gamma_k})x) \to p(x)$ is a rule in $R_\dagger$. By the induction hypothesis, $p(h(\kappa)) \overset{(\mu)}{\Longrightarrow}{}^{n}_{M^\dagger} t$, and hence $q(h(\eta)) \overset{(\mu)}{\Longrightarrow}{}^{n+1}_{M^\dagger} t$.

* * *

It remains to show the direction "if". Again, the case $n = 0$ holds trivially. We continue by assuming that

$$q(h(\eta)) \overset{(\mu)}{\Longrightarrow}_r \xi \overset{(\mu)}{\Longrightarrow}{}^{n}_{M^\dagger} t$$

for some $n \in \mathbb{N}$, $r \in R_\dagger$ and $\xi \in \mathrm{T}_\Sigma(Q(\Gamma_\dagger^*))$. We perform a case analysis on r.

(I) The case that r is a copy rule of form $q(x) \to \sigma(p_1(x), \ldots, p_k(x))$ is analogous to before. We have $t = \sigma(t_1, \ldots, t_k)$ for some $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$, $\xi = \sigma(p_1(h(\eta)), \ldots, p_k(h(\eta)))$, and for every $i \in [k]$, we have $p_i(h(\eta)) \overset{(\mu)}{\Longrightarrow}{}^{n_i}_{M^\dagger} t_i$ for some $n_i \in \mathbb{N}$ such that $n = \sum_{i=1}^k n_i$. Thus by the induction hypothesis, $p_i(\eta) \overset{(\mu)}{\Longrightarrow}{}^{n_i}_{M^\sharp} t_i$, and, by construction,

$$q(\eta) \overset{(\mu)}{\Longrightarrow}_r \sigma\big(p_1(\eta), \ldots, p_k(\eta)\big) \Rightarrow^n_{M^\sharp} t\,.$$

(II) Consider the case that r is a push rule of form $q(x) \to p(Ux)$. Thus $\xi = p(Uh(\eta))$, and $|h(\eta)| < \mu$. By construction, there are $k \in \mathbb{N}_1$, and rules $q_{i-1}(x) \to q_i(\gamma_i x)$ in R, for every $i \in [k]$, such that $q = q_0$, $q_k = p$, and

$$U = h(\gamma_k) \circ \cdots \circ h(\gamma_1)\,.$$

Thus, $q(\varepsilon) \Rightarrow^*_M p(\gamma_k \cdots \gamma_1)$, and hence the rule $q(x) \to p(\overline{\gamma_k \cdots \gamma_1}\, x)$ is in $R_\sharp$. Moreover, by the induction hypothesis, we have that

$$p(\overline{\gamma_k \cdots \gamma_1}\, \eta) \overset{(\mu)}{\Longrightarrow}{}^{n}_{M^\sharp} t\,,$$

and therefore $q(\eta) \overset{(\mu)}{\Longrightarrow}{}^{n+1}_{M^\sharp} t$.

(III) Finally, let r be a pop rule of form $q(Ux) \to p(x)$. We conclude that $\eta = \overline{\gamma_1 \cdots \gamma_k}\, \kappa$ for some $k \in \mathbb{N}_1$, $\gamma_1, \ldots, \gamma_k \in \Gamma$ such that $h(\overline{\gamma_1 \cdots \gamma_k}) = U$, and for some $\kappa \in \Gamma_\sharp^*$. Therefore, by definition of h,

$$U = \big\{(u, v) \in Q \times Q \,\big|\, u(\gamma_1 \cdots \gamma_k) \Rightarrow^k_M v(\varepsilon) \text{ using only pop rules}\big\}\,.$$

By definition of $M^\dagger$, we have $(q, p) \in U$, and hence there is a rule $q(\overline{\gamma_1 \cdots \gamma_k}\, x) \to p(x)$ in $R_\sharp$. By the induction hypothesis, $p(\kappa) \overset{(\mu)}{\Longrightarrow}{}^{n}_{M^\sharp} t$, and thus $q(\eta) \overset{(\mu)}{\Longrightarrow}{}^{n+1}_{M^\sharp} t$. ☐

By choosing $q = q_0$ and $\eta = h(\eta) = \varepsilon$, we obtain the following easy corollary to Lemma 3.13.

Corollary 3.14. $\mathcal{L}(M^\dagger) = \mathcal{L}(M^\sharp)$.

If M is succinct, then the Lemmas 3.10, 3.11 and 3.13 imply that while $|M^\dagger|$ may be exponential in $|M|$, nevertheless we know that for every $t \in \mathcal{L}(M)$, there is a derivation d of t in $M^\dagger$ such that both the length of d, as well as the size of every pushdown occurring in d, are bounded by a polynomial in $|t|$. The existence of such derivations will be exploited by the following decision procedures.

Algorithm 2 Nondeterministic decision procedure for uniform membership

Input: pta $M = (Q, \Sigma, \Gamma, q_0, R)$, $t \in \mathrm{T}_\Sigma$
Output: "Yes" if $t \in \mathcal{L}(M)$, "No" otherwise

```
ξ ← q_0(ε)
loop
   select leftmost w ∈ pos(ξ) such that ξ(w) = q(η) for some q(η) ∈ Q(Γ_†^*)
   either
      choose a rule q(x) → σ(p_1(x), ..., p_k(x)) in R
      ξ ← ξ[σ(p_1(η), ..., p_k(η))]_w
   or
      choose a rule q(x) → p(γx) in R and set u ← p, U ← h(γ)
      repeat n times for some n ∈ ℕ
         choose a rule u(x) → v(γx) in R and set u ← v, U ← h(γ) ∘ U
      end repeat
      ξ ← ξ[u(Uη)]_w
   or if η = Uκ for some U ∈ Γ_†, κ ∈ Γ_†^*
      choose some (u, p) ∈ U such that u = q
      ξ ← ξ[p(κ)]_w
   end either
   if ξ = t then return "Yes" else if ξ ∈ T_Σ then return "No" endif
end loop
```

3.2 The Uniform Membership Problem

Using the machinery developed in the previous section, we now turn our attention to the uniform membership problem of cftg.

Theorem 3.15. *Let Σ be a nontrivial ranked alphabet. Then the uniform membership problem of cftg over Σ is* PSPACE*-complete.*

The theorem is a direct consequence of Lemmas 3.16 and 3.17, which we will prove in the following.

3.2.1 Upper Bound

Employing $M^\dagger$, we can now investigate the complexity of the uniform membership problem of cftg. We begin with the upper bound.

Lemma 3.16. *For every ranked alphabet Σ, the uniform membership problem for cftg over Σ is in* PSPACE.

Proof. Let $t \in \mathrm{T}_\Sigma$ and let G be a cftg over Σ. Construct a succinct pta $M = (Q, \Sigma, \Gamma, q_0, R)$ with $\mathcal{L}(M) = \mathcal{L}(G)$. By Theorem 2.27 and Lemma 3.1, this construction can be performed in time (and thus space) polynomial in $|G|$.

Algorithm 2 contains a nondeterministic procedure which decides whether $t \in \mathcal{L}(M)$ in space restricted to $2 \cdot |t|^2 \cdot |Q|^2$. There, h denotes the mapping $h\colon \Gamma^+ \to \Gamma_\dagger$ from the definition of $M^\dagger$. The decision procedure tries to find a derivation d' in the compact pta $M^\dagger$. However, d' is constructed "on-the-fly." In each loop of the algorithm, the leftmost occurrence of some $q(\eta) \in Q(\Gamma_\dagger^*)$ in ξ is selected, and a rule r is chosen. On the one hand, if r is a copy or pop rule of $M^\dagger$, then it is applied to $q(\eta)$. On the other hand, we may choose a nonzero number of push rules of M with compatible states, apply h to the symbols they push, and combine the results by the product of binary relations. Clearly, this procedure captures exactly the derivations in $M^\dagger$.

If $t \in \mathcal{L}(M)$, then by Lemma 3.2, there is a succinct derivation $d \in \mathcal{DS}_M(q_0(\eta), t)$, and, by Lemmas 3.10 and 3.13, a derivation $d' \preceq d$ in $M^\dagger$ that has $(2 \cdot |t|)$-bounded pushdowns. Each pushdown symbol that occurs in d' is a subset of $Q \times Q$, and can thus be stored within space $|Q|^2$. As the number of elements of $Q(\Gamma_\dagger^*)$ that may occur in an intermediate tree ξ in the derivation d is bounded by $|t|$, ξ can be stored in space $2 \cdot |t|^2 \cdot |Q|^2$. By Theorem 1.14, the procedure is also computable in deterministic space polynomial in $|t|$ and $|M|$. □

3.2.2 Lower Bound

Lemma 3.17. *Let Σ be a nontrivial ranked alphabet. Then the uniform membership problem of cftg over Σ is* PSPACE*-hard.*

Proof. Recall the following decision problem. Let Δ be an alphabet. Then the *intersection nonemptiness problem of dfa* is defined as follows.

Problem:	**DFA Intersection Nonemptiness**
Instance:	Deterministic and total fsa $A_1, \ldots, A_k$ over Δ, for some $k \in \mathbb{N}$
Question:	Is $\bigcap_{i=1}^{k} \mathcal{L}(A_i) \neq \emptyset$?

As shown by Kozen [104], the intersection nonemptiness problem is PSPACE-complete.[2] We will show that this problem is logspace-reducible to the uniform membership problem of cftg.

So let us assume we are given as input k deterministic finite-state automata $A_1, \ldots, A_k$, where for each $i \in [k]$, A_i is of form $(Q_i, \Delta, q_0^i, F_i, \delta_i)$. Moreover, we demand that the automata's state sets Q_i are pairwise disjoint, and that Σ and Δ are disjoint. This assumption comes with no loss of generality, as non-distinct symbols can simply be renamed.

Since Σ is nontrivial, there are $\alpha \in \Sigma^{(0)}$ and $\sigma \in \Sigma^{(n)}$ for some $n \in \mathbb{N}$ with $n \geq 2$. We construct the pta $M = (Q, \Sigma, \Delta \cup \{\#\}, q_0, R)$ where $\#$ is a distinct symbol,

$$Q = \{q_0\} \cup \{u_0, \ldots, u_k\} \cup \bigcup_{i=1}^{k} Q_i \,,$$

with $q_0, u_0, \ldots, u_k$ distinct states, and R is defined as follows.

[2] Actually, Kozen proved that the intersection *emptiness* problem, which asks whether $\bigcap_{i=1}^{k} \mathcal{L}(A_i) = \emptyset$ instead, is PSPACE-complete. However, the class PSPACE is closed under complement, see Theorem 1.14.

(i) The rule $q_0(x) \to u_k(\#x)$ is in R.

(ii) For every $b \in \Delta$, R contains the rule $u_k(x) \to u_k(bx)$.

(iii) For every $i \in [k]$, the rule

$$u_i(x) \to \sigma\big(q_0^i(x), u_{i-1}(x), u_0(x), \dots, u_0(x)\big)$$

is in R.

(iv) Moreover, for every $i \in [k]$, $b \in \Delta$, q, $p \in Q_i$ such that $\delta_i(q,b) = p$, and $f \in F_i$, the rule set R contains

$$q(bx) \to p(x) \qquad \text{and} \qquad f(\#x) \to \alpha\,.$$

(v) Finally, for every $\gamma \in \Delta \cup \{\#\}$, R contains the rule $u_0(\gamma x) \to \alpha$.

We construct the tree $t = s_k$, where

$$s_0 = \alpha \qquad \text{and} \qquad s_{j+1} = \sigma(\alpha, s_j, \underbrace{\alpha, \dots, \alpha}_{n-2}) \quad \text{for every } j \in \mathbb{N}\,.$$

Hence, t is of the form

$$t = \sigma\big(\alpha, \sigma(\alpha, \cdots \sigma(\alpha, \dots, \alpha) \cdots, \alpha, \dots, \alpha), \alpha, \dots, \alpha\big)$$

such that σ occurs exactly k times in t. Both M and t are logspace-computable from the input, since the construction requires only a constant number of loops with binary counters.

* * *

The construction's idea is as follows. With the rule created in *(ii)*, we can guess some arbitrary word w from Δ^* on the pushdown of u_k. The rules in *(iii)* create k copies of our guess w; the configurations $u_0(x)$ are just for padding. Finally, the rules in *(iv)* independently simulate the state behaviour of the automata $A_1, \dots, A_k$ on w. The derivation terminates (deriving t) if and only if $w \in \mathcal{L}(A_1) \cap \cdots \cap \mathcal{L}(A_k)$.

* * *

The above intuition will now be put into formal terms. We will show that

$$t \in \mathcal{L}(M) \qquad \text{if and only if} \qquad \exists w \in \Delta^* \colon w \in \bigcap_{i=1}^{k} \mathcal{L}(A_i)\,,$$

which implies correctness of the construction. The following two observations are easy to see, and will be helpful in the proof.

(A) *For every $i \in [k]$ and $w \in \Delta^*$, we have*

$$q_0^i(w\#) \Rightarrow_M^* \alpha \qquad \textit{if and only if} \qquad w \in \mathcal{L}(A_i)\,.$$

(B) *Moreover,* $u_0(w\#) \Rightarrow_M \alpha$ *for every* $w \in \Delta^*$.

First, let us prove the direction "only if" of the stated equivalence. Assume that $t \in \mathcal{L}(M)$. Then there is some $w \in \Delta^*$ such that

$$\begin{aligned}
q_0(\varepsilon) &\Rightarrow_M u_k(\#) \Rightarrow_M^* u_k(w\#) \\
&\Rightarrow_M \sigma\big(q_0^k(w\#), u_{k-1}(w\#), u_0(w\#), \ldots, u_0(w\#)\big) \\
&\Rightarrow_M^* \sigma\Big(\alpha, \sigma\big(q_0^{k-1}(w\#), u_{k-2}(w\#), u_0(w\#), \ldots, u_0(w\#)\big), u_0(w\#), \ldots, u_0(w\#)\Big) \\
&\ \vdots \\
&\Rightarrow_M^* \sigma\Big(\alpha, \sigma\big(\alpha, \cdots \sigma(q_0^1(w\#), u_0(w\#), \ldots, u_0(w\#)) \cdots, u_0(w\#), \ldots, u_0(w\#)\big), \\
&\qquad\qquad u_0(w\#), \ldots, u_0(w\#)\Big) \\
&\Rightarrow_M^* \sigma\Big(\alpha, \sigma\big(\alpha, \cdots \sigma(\alpha, \alpha, \ldots, \alpha) \cdots, \alpha, \ldots, \alpha\big), \alpha, \ldots, \alpha\Big) = t
\end{aligned}$$

By observation **(A)**, we obtain $w \in \mathcal{L}(A_1) \cap \cdots \cap \mathcal{L}(A_k)$.

* * *

For the direction "if", assume that there is some $w \in \mathcal{L}(A_1) \cap \cdots \cap \mathcal{L}(A_k)$. It is easy to see that, then, M has a derivation as displayed above, and thus $t \in \mathcal{L}(M)$. □

3.2.3 Uniform Membership of ε-free Indexed Grammars

As is the case for earlier research, see e.g. [141], new results on cftg also lead to new theorems for indexed grammars. Let us consider, e.g., given an alphabet Σ, the uniform membership problem of ε-free indexed grammars over Σ, which is specified as follows.

Problem:	**Uniform Membership of ε-free Indexed Grammars over Σ**
Instance:	An ε-free ixg G over Σ and a word $w \in \Sigma^*$
Question:	Is $w \in \mathcal{L}(G)$?

Theorem 3.18. *For every alphabet Σ, the uniform membership problem of ε-free indexed grammars over Σ is* PSPACE*-complete.*

Proof. Hardness of the problem is a direct consequence of Lemma 3.17, together with Theorem 2.28.

To see that the problem can be decided in polynomial space, consider an ε-free ixg $G = (N, \Sigma, \Gamma, S, P)$ and a word $w \in \Sigma^*$. We demand that G is in normal form, and for every production of G of the form $A \to B_1 \cdots B_k$ for some A, B_1, ..., $B_k \in N$, we have $k = 2$. Our demand comes with no loss of generality, and can be enforced in polynomial time, using the construction from the proof of [3, Thm. 3.1].

By Theorem 2.28, there is a cftg G' over some ranked alphabet Δ such that $\Sigma = \Delta^{(0)}$ and $\mathrm{yd}(\mathcal{L}(G')) = \mathcal{L}(G)$. Moreover, by our above demand for G, we can choose Δ such that

$\Delta = \Delta^{(0)} \cup \Delta^{(2)}$. This means that for every tree $t \in \mathrm{T}_\Delta$ with $\mathrm{yd}(t) = w$, we have $|t| = 2 \cdot |w| - 1$, by the well-known formula for the size of a full binary tree with n leaves.

Our decision procedure consists of guessing one of these trees, say t, and then of checking whether $t \in \mathcal{L}(G')$. This procedure can be executed in nondeterministic polynomial space due to Lemma 3.16. Using Theorem 1.14, we obtain a decision procedure that can be performed in deterministic polynomial space. □

Remark 3.19. If ε-productions are allowed, the problem becomes more complex: the uniform membership problem of indexed grammars *with* ε-productions is EXP-complete [157].

Theorems 2.28 and 2.37 imply that the nonemptiness problem of ε-free indexed grammars remains EXP-complete, however. ◁

Algorithm 3 Nondeterministic decision procedure for membership of cftg

Input: $t \in \mathrm{T}_\Sigma$
Output: "Yes" if $t \in \mathcal{L}(M)$, "No" otherwise
choose some $d \in R_\dagger^*$ with $|d| \leq \mu(t)^2 + \mu(t)$ and $\mu(t)$-bounded pushdowns
if $q_0(\eta) \Rightarrow_d t$ **then** return "Yes" **else** return "No" **endif**

3.3 The Non-Uniform Membership Problem

In this section, we show that the pta $M^\dagger$ may also be useful for other means, by presenting an alternative proof of the NP upper bound of non-uniform membership of a cftg. Note that this bound is already known: the class of output languages of compositions of macro tree transducers, a proper superclass of the context-free tree languages, is in NP [87, Thm. 8].

Moreover, by Theorem 2.28, the following upper bound is as well a consequence of the containment of the indexed languages in NP [142]. Note however that the proof in [142] rests on the correctness of the Turing machine from [3].

Lemma 3.20. *For every ranked alphabet Σ and every cftg G over Σ, the non-uniform membership problem of G is in* NP.

Proof. Let G be a cftg over Σ. We construct an equivalent succinct pta M, as well as $M^\dagger$ as defined above. As G is not part of the input, $M^\dagger$ is constructible in constant time. Consider the nondeterministic decision procedure in Algorithm 3. By Lemma 3.13, $\mathcal{L}(M^\dagger) = \mathcal{L}(M^\sharp)$, and moreover $\mathcal{L}(M^\sharp) = \mathcal{L}(M)$. So if the procedure returns "Yes", then there is some $d \in \mathcal{D}_{M^\dagger}(q_0(\varepsilon), t)$, and hence $t \in \mathcal{L}(M)$. Conversely, if $t \in \mathcal{L}(M)$, then there is some $d' \in \mathcal{D}_{M^\sharp}(q_0(\varepsilon), t)$, and by Lemma 3.11, we may assume that $|d'| \leq \mu(t)^2 + \mu(t)$. Lemma 3.13 implies that there is a $d \in \mathcal{D}_{M^\dagger}(q_0(\varepsilon), t)$ with the same length bound. Therefore the procedure returns "Yes". □

Hardness of the problem can be demonstrated in the same manner as for indexed grammars [142, Prop. 1], by constructing a cftg G such that $\mathcal{L}(G)$ encodes the set of all satisfiable propositional formulas in 3-conjunctive normal form. As the construction in [142] is in fact for one-way nondeterministic stack languages, we restate it here for cftg. A similar, but not identical, construction is given in [95].

Lemma 3.21. *There are a ranked alphabet Σ and a context-free tree grammar G over Σ such that the membership problem of G is* NP*-hard.*

Proof. We will construct a cftg G such that the satisfiability problem of propositional logic formulas in 3-conjunctive normal form can be reduced to $\mathcal{L}(G)$. Let $n \in \mathbb{N}$. Recall that a 3-cnf formula φ with variables $v_1, \ldots, v_n$ is considered to be a word

$$(L_1^1 \vee L_2^1 \vee L_3^1) \wedge \cdots \wedge (L_1^m \vee L_2^m \vee L_3^m),$$

for some $m \in \mathbb{N}_1$, over the alphabet $\Gamma = \{0, 1, \neg, \vee, \wedge, (,)\}$, where each factor L_i^j is a positive literal v_k or a negative literal $\neg v_k$, for some $k \in [n]$. Without loss of generality, we can assume that φ contains each variable from V_n at least once.

Consider the ranked alphabet

$$\Sigma = \{\wedge^{(2)}, \vee^{(3)}, \neg^{(1)}, \gamma^{(1)}, \alpha^{(0)}\} .$$

We will define a partial function $e\colon \Gamma^* \rightarrow \mathrm{T}_\Sigma$, such that every 3-cnf formula φ is encoded by a tree $e(\varphi) \in \mathrm{T}_\Sigma$. For every $i \in [n]$, let

$$e(v_i) = \gamma^i(\alpha), \quad \text{and} \quad e(\neg v_i) = \neg(\gamma^i(\alpha)) .$$

Moreover, let

$$e\big((L_1 \vee L_2 \vee L_3)\big) = \vee\big(e(L_1), e(L_2), e(L_3)\big)$$

for every L_1, L_2, $L_3 \in \{\varepsilon, \neg\} \cdot 1 \cdot \{0,1\}^*$, and let

$$e(C \wedge \varphi) = \wedge(e(C), e(\varphi))$$

for every 3-cnf formula φ, and every $C \in (\Gamma \setminus \{\wedge\})^*$. Observe that e is well-defined on its domain; in particular it is defined for every 3-cnf formula.

For example, the formula

$$\varphi = (v_1 \vee \neg v_1 \vee v_2) \wedge (v_2 \vee v_3 \vee v_1) \wedge (v_1 \vee \neg v_1 \vee v_4)$$

is encoded by

$$e(\varphi) = \wedge\Big(\vee(\gamma\alpha, \neg\gamma\alpha, \gamma\gamma\alpha), \wedge\Big(\vee(\gamma\gamma\alpha, \gamma\gamma\gamma\alpha, \gamma\alpha), \vee(\gamma\alpha, \neg\gamma\alpha, \gamma\gamma\gamma\gamma\alpha)\Big)\Big).$$

Since variable indices are assigned consecutively, the length of a 3-cnf formula φ is at least as large as the largest variable index in φ, even although the indices are encoded in binary notation. Vice versa, every index i of a variable v_i in φ is bounded by $|\varphi|$. Hence $e(\varphi)$ is logspace-constructible from φ.

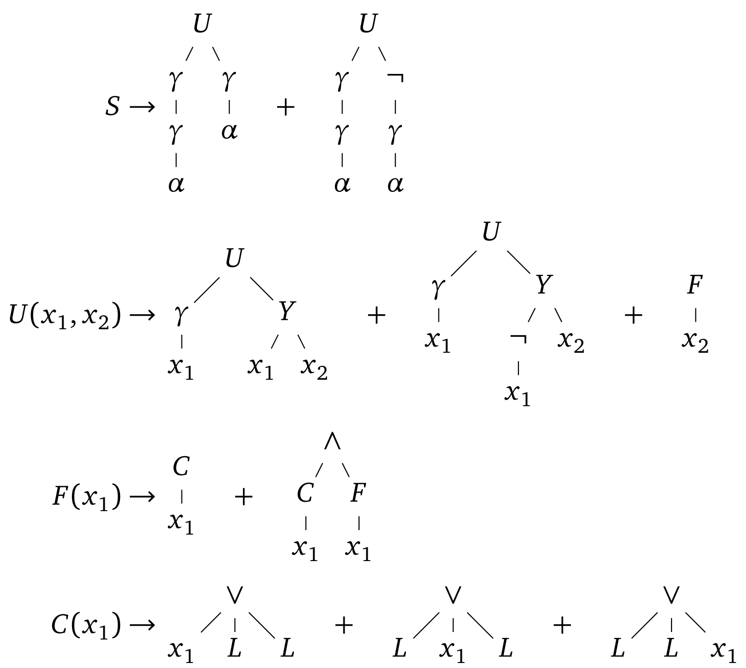
We will now construct a cftg G such that for every 3-cnf formula φ,

$$e(\varphi) \in \mathcal{L}(G) \qquad \text{if and only if} \qquad \varphi \text{ is satisfiable.}$$

Thus, the satisfiability problem of 3-cnf formulas is logspace-reducible to the non-uniform membership problem of G; the latter is therefore NP-hard by Theorem 1.18.

Let $G = (N, \Sigma, S, P)$, where $N = \{S^{(0)}, L^{(0)}, W^{(0)}, C^{(1)}, F^{(1)}, U^{(2)}, Y^{(2)}\}$, and let P contain the productions as depicted in Figure 3.2.

The cftg G can be understood as follows. In a derivation of G, the nonterminal U is responsible for guessing a variable assignment. In every step, U's first parameter holds the variable whose truth value is to be determined next. The second parameter accumulates all literals guessed to be true up to that very moment. Eventually, the variable assignment is passed on to the nonterminal F, which derives the encoding of a satisfiable 3-cnf formula by endowing each clause with (at least) one satisfied literal. The productions for the nonterminal Y implement nondeterministic choice, and allow deriving each of the literals guessed before. The role of the nonterminals L and W is to guess the other literals in each clause, which



$$L \to W \;+\; \neg(W) \qquad W \to \gamma(W) + \alpha \qquad Y(x_1, x_2) \to x_1 + x_2$$

Figure 3.2: Productions of the cftg G from Lemma 3.21

need not necessarily be satisfied. Compare Figure 3.3 for a derivation of $e(\varphi)$ as given in the example from above.

* * *

We will now show for every 3-cnf formula φ that, indeed, φ is satisfiable if and only if $e(\varphi) \in \mathcal{L}(G)$. For this purpose, let $n \in \mathbb{N}$ and assume that φ is a 3-cnf formula that contains exactly the variables $v_1, \ldots, v_n$. Let φ be of the form

$$(L_1^1 \vee L_1^2 \vee L_1^3) \wedge \cdots \wedge (L_m^1 \vee L_m^2 \vee L_m^3) \tag{3.2}$$

for some $m \in \mathbb{N}_1$.

For the direction "only if", assume that φ is satisfiable. Thus there is an assignment $a \colon V_n \to \mathbb{B}$ such that $a(\varphi) = 1$. Hence for every $i \in [m]$ there is some $j_i \in [3]$ with $a(L_i^{j_i}) = 1$. Denote $e(L_i^{j_i})$ by t_i. Define $s_1, \ldots, s_n \in \mathrm{T}_\Sigma$ such that for every $i \in [n]$,

$$s_i = \begin{cases} \gamma^i \alpha & \text{if } a(v_i) = 1 \\ \neg\gamma^i \alpha & \text{otherwise.} \end{cases}$$

Observe that

$$\{t_1, \ldots, t_m\} \subseteq \{s_1, \ldots, s_n\}.$$

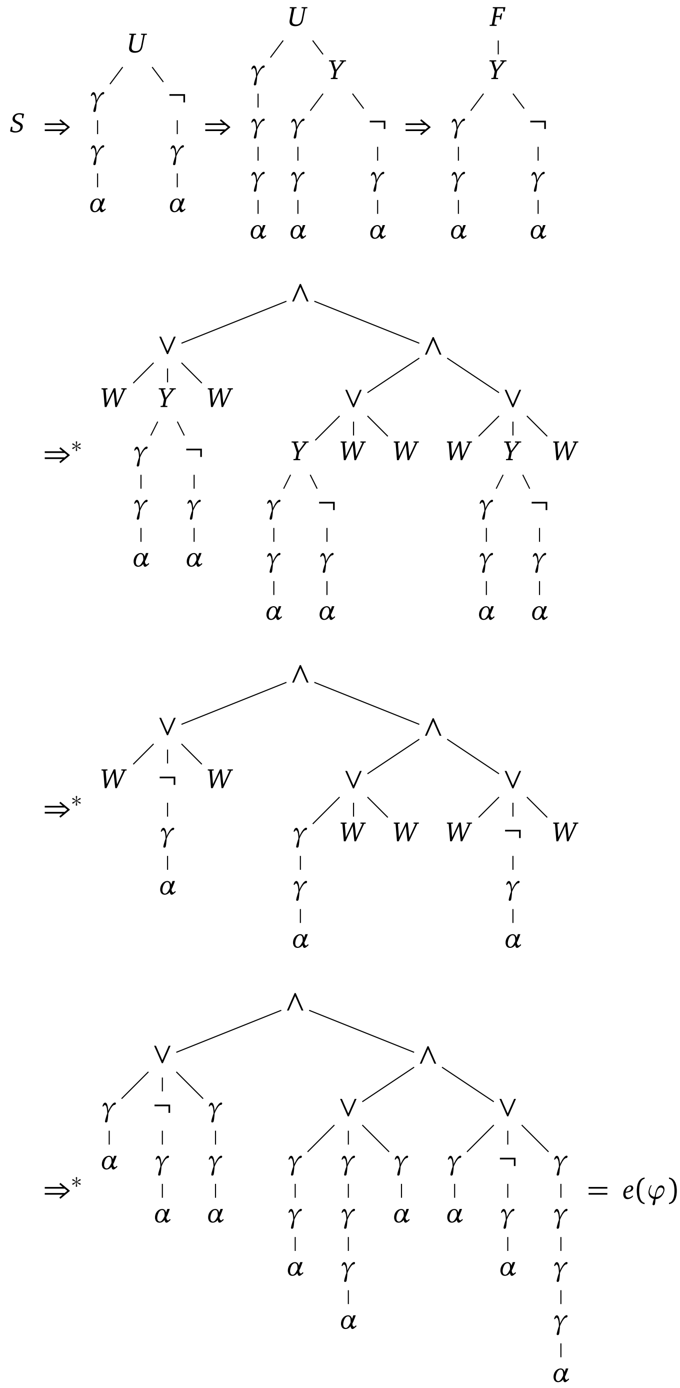


Figure 3.3: Example derivation of $e(\varphi)$

Then

$$S \Rightarrow_G^* F(\xi) \Rightarrow_G^* \underbrace{\wedge(C(\xi), \wedge(\cdots \wedge (C(\xi), C(\xi))\cdots))}_{m \text{ times } C(\xi)}$$

with $\xi = Y(s_n, Y(\cdots Y(s_2, s_1)\cdots))$. Note that $\xi \Rightarrow_G^* t_i$ for every $i \in [m]$. In fact, if $j_i = 1$, then

$$C(\xi) \Rightarrow_G \vee(\xi, W, W) \Rightarrow_G^* \vee(t_i, t', t''),$$

where t', resp. t'', encode L_i^2 and L_i^3. The cases $j_i \in \{2,3\}$ are analogous, hence $S \Rightarrow_G^* e(\varphi)$ and $e(\varphi) \in \mathcal{L}(G)$.

* * *

For the direction "if", consider some $t \in \mathcal{L}(G)$ and a 3-cnf formula φ with variables precisely from V_n for some $n \in \mathbb{N}$, of the form given in (3.2). Assume that $e(\varphi) = t$. Due to the definition of G, we have

$$\begin{aligned} S \overset{\text{OI}}{\Rightarrow}{}_G^* F\Big(\underbrace{Y(t_n, Y(\cdots Y(t_2, t_1)\cdots))}_{\xi}\Big) &\overset{\text{OI}}{\Rightarrow}{}_G^* \underbrace{\wedge(C(\xi), \wedge(\cdots \wedge (C(\xi), C(\xi))\cdots))}_{m \text{ times } C(\xi)} \\ &\overset{\text{OI}}{\Rightarrow}{}_G^* \wedge(\zeta_1, \wedge(\cdots \wedge (\zeta_{m-1}, \zeta_m))) \overset{\text{OI}}{\Rightarrow}{}_G^* t \end{aligned}$$

for some $m \in \mathbb{N}_1$, where

$$t_i \in \{\neg\gamma^i\alpha, \gamma^i\alpha\} \qquad \text{and} \qquad \zeta_j \in \big\{\vee(\xi, W, W), \vee(W, \xi, W), \vee(W, W, \xi)\big\}$$

for every $i \in [n]$ and $j \in [m]$.

Define the variable assignment $a\colon V_n \to \mathbb{B}$ such that

$$a(v_i) = 1 \qquad \text{if and only if} \qquad t_i = \gamma^i\alpha.$$

Consider the clause $C_j = (L_j^1 \vee L_j^2 \vee L_j^3)$, for each $j \in [m]$, and assume that $\zeta_j = \vee(\xi, W, W)$. Then $e(L_j^1) = t_i$ for some $i \in [n]$, and by definition of a, we have $a(C_j) = 1$. The other cases $\zeta_j = \vee(W, \xi, W)$ and $\zeta_j = \vee(W, W, \xi)$ are analogous. Therefore φ is satisfiable, as a satisfies all its clauses. □

From Lemmas 3.20 and 3.21, we obtain the following theorem as a direct corollary.

Theorem 3.22. *There are a ranked alphabet Σ and a context-free tree grammar G over Σ such that the membership problem of G is* NP*-complete.*

3.4 The Infiniteness Problem

In this section, we prove the following theorem on the infiniteness problem of cftg.

Theorem 3.23. *For every nontrivial ranked alphabet Σ, the infiniteness problem of cftg over Σ is* EXP*-complete.*

The theorem is a direct consequence of Lemmas 3.24 and 3.25 below.

Lemma 3.24. *For every nontrivial ranked alphabet Σ, the infiniteness problem of cftg over Σ is* EXP*-hard.*

Proof. By Theorem 2.37, the nonemptiness problem of cftg over Σ is EXP-hard. We will reduce this problem to the infiniteness problem of cftg.

For this purpose, let $G = (N, \Sigma, S, P)$ be a cftg in normal form, and let $T \subseteq P$ be the set of terminal productions of G. Let $U^{(1)} \notin N$ be a fresh nonterminal symbol. For every terminal production

$$A \cdot \mathrm{Id}_n \to \sigma \cdot \vartheta$$

in T, construct the production

$$A \cdot \mathrm{Id}_n \to U \cdot \sigma \cdot \vartheta\,,$$

moreover construct the two productions

$$U(x_1) \to U(\gamma(x_1, \alpha, \dots, \alpha)) \;+\; x_1\,,$$

where γ and α are some fixed terminal symbols from $\Sigma \setminus \Sigma^{(0)}$ and $\Sigma^{(0)}$, respectively. The set of all such constructed productions is denoted by T'. Let moreover $N' = N \cup \{U\}$, and define the cftg $G' = (N', \Sigma, S, P \setminus T \cup T')$. It is easy to see that

$$A \cdot \mathrm{Id}_n \Rightarrow_G \sigma \cdot \vartheta \qquad \text{if and only if} \qquad \forall i \in \mathbb{N} \colon A \cdot \mathrm{Id}_n \Rightarrow^*_{G'} \big(\gamma(x_1, \alpha, \dots, \alpha)\big)^i \cdot \sigma \cdot \vartheta$$

for every $A \in N$, $\sigma \in \Sigma$, and $\vartheta \in \Theta$. Clearly, this implies that $\mathcal{L}(G)$ is nonemtpy if and only if $\mathcal{L}(G')$ is infinite. Therefore, the infiniteness problem of cftg is EXP-hard. □

Lemma 3.25. *For every ranked alphabet Σ, the infiniteness problem of cftg over Σ is in* EXP*.*

Proof. The decidability of the infiniteness problem of cftg has been proven by Rounds [141]. All we have to do is argue why this method can be performed in exponential time. The proof idea in [141] is based on the observation that a tree language is infinite if and only if its path language is infinite.

So let G be a cftg over the ranked alphabet Σ, and recall its path language $\mathrm{P}(\mathcal{L}(G))$ as defined in Section 2.3. By Theorem 2.29, $\mathrm{P}(\mathcal{L}(G))$ is a context-free word language, and a cfg $\widehat{G}$ which generates this language can be computed from G in time exponential in the size of G. As the infiniteness problem of cfg can be decided in polynomial time (cf. e.g. [86, Thm. 6.6]), this proves that infiniteness of cftg can be decided in exponential time. □

3.5 Linear Context-Free Tree Grammars

We turn our attention to the decision problems of linear, and linear and nondeleting, cftg. Surprisingly, there still remain hard problems if copying is disallowed. We begin with the following auxiliary lemma, which will aid us in the treatment both of nonemptiness and of uniform membership.

Lemma 3.26. *Let Σ be a nontrivial ranked alphabet. For every 3-cnf formula φ, we can construct in logarithmic space an l-cftg G_φ and a tree $t_\varphi \in \mathrm{T}_\Sigma$ such that*

$$\mathcal{L}(G_\varphi) = \begin{cases} \{t_\varphi\} & \text{if } \varphi \text{ is satisfiable,} \\ \emptyset & \text{otherwise.} \end{cases}$$

Proof. Consider a 3-cnf formula

$$\varphi = \big(L_1 \vee L_2 \vee L_3\big) \wedge \cdots \wedge \big(L_{3(m-1)+1} \vee L_{3(m-1)+2} \vee L_{3m}\big)\,,$$

for some $m \in \mathbb{N}_1$. Again, we assume that the propositional variables' indices are assigned consecutively – say φ contains precisely the variables $v_1, \ldots, v_n$, for some $n \in \mathbb{N}$.

Chosse $\alpha \in \Sigma^{(0)}$ and $\sigma \in \Sigma^{(k)}$ for some $k > 1$. There are such symbols, since Σ is nontrivial. Let $q = 3m$, and let $G_\varphi = (N, \Sigma, \xi_0, P)$ such that

$$N = \big\{A_1^{(q)}, \ldots, A_{n+1}^{(q)}, C^{(3)}, T^{(0)}, F^{(0)}\big\}\,.$$

The grammar's axiom is given by

$$\xi_0 = A_1(F, \ldots, F)\,,$$

and the productions in P are constructed as follows.

(i) For every $i \in [n]$, P contains the productions

$$A_i \cdot \mathrm{Id}_q \to A_{i+1} \cdot u_1 \quad + \quad A_{i+1} \cdot u_2\,,$$

where $u_1, u_2 \in \mathrm{T}(N)_q^q$ are such that for every $j \in [q]$,

$$\pi_j \cdot u_1 = \begin{cases} T & \text{if } L_j = v_i, \\ F & \text{if } L_j = \neg v_i, \\ x_j & \text{otherwise,} \end{cases} \qquad \text{and} \qquad \pi_j \cdot u_2 = \begin{cases} F & \text{if } L_j = v_i, \\ T & \text{if } L_j = \neg v_i, \\ x_j & \text{otherwise.} \end{cases}$$

(ii) Moreover, P contains the production

$$A_{n+1} \cdot \mathrm{Id}_q \to t_m \cdot \underbrace{(C \otimes \cdots \otimes C)}_{m}\,,$$

where

$$t_1 = x_1 \qquad \text{and} \qquad t_{i+1} = \sigma \cdot (x_1 \otimes t_i \otimes \underbrace{\alpha \otimes \cdots \otimes \alpha}_{k-2}) \quad \text{for each } i \in \mathbb{N}_1\,.$$

Compare Figure 3.4 for an example of the tree t_3, where $k = 3$.

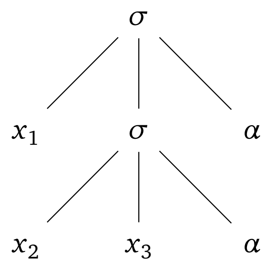


Figure 3.4: The tree t_3, for $k = 3$

(iii) Finally, P contains the productions

$$C \cdot \mathrm{Id}_3 \to x_1 + x_2 + x_3 \qquad \text{and} \qquad T \to \alpha\,.$$

Note that there is no production for the nonterminal F.

Construct the tree

$$t_\varphi = t_m \cdot \underbrace{(\alpha \otimes \cdots \otimes \alpha)}_{m}\,.$$

It is easy to see that $\mathcal{L}(G_\varphi) \subseteq \{t_\varphi\}$, and that G_φ and t_φ are logspace-constructible from the formula φ. It remains to show that $t_\varphi \in \mathcal{L}(G_\varphi)$ if and only if φ is satisfiable. As this can be demonstrated by analyzing the derivations in G_φ, in the very same manner as in Lemma 3.21, we omit the formal proof.

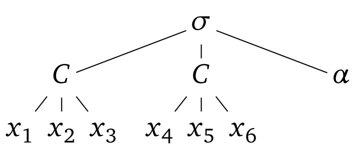
Intuitively, the productions introduced by rule *(i)* guess an assignment for the variables in φ, one after each other. After guessing, we introduce in rule *(ii)* an instance of C for every clause of φ. Then G can derive t_φ if each instance of C can project to an occurrence of T. Vice versa, if G can derive t_φ, then clearly there is a satisfying assignment for φ. $\square$

Example 3.27. Consider the 3-cnf formula

$$\varphi = (v_1 \vee v_1 \vee v_2) \wedge (\neg v_2 \vee \neg v_1 \vee \neg v_2)\,.$$

Clearly, φ is satisfiable – consider e.g. the variable assignment a with $a(v_1) = 1$ and $a(v_2) = 0$. Let $\Sigma = \{\sigma^{(3)}, \alpha^{(0)}\}$. When we construct the l-cftg G_φ over Σ according to the definition above, we obtain the productions

$$A_1 \cdot \mathrm{Id}_6 \to A_2(T, T, x_3, x_4, F, x_6) + A_2(F, F, x_3, x_4, T, x_6)\,,$$
$$A_2 \cdot \mathrm{Id}_6 \to A_3(x_1, x_2, T, F, x_5, F) + A_3(x_1, x_2, F, T, x_5, T)\,,$$

$A_3 \cdot \mathrm{Id}_6 \to$  , along with $C \cdot \mathrm{Id}_3 \to x_1 + x_2 + x_3$ and $T \to \alpha$. The derivation

$$A_1(F, \dots, F) \Rightarrow A_2(T, T, F, F, F, F) \Rightarrow A_3(T, T, F, T, F, T)$$

$$\Rightarrow \sigma\big(C(T, T, F),\ C(T, F, T),\ \alpha\big) \Rightarrow^* \sigma(\alpha, \alpha, \alpha)$$

generates t_φ, affirming that φ is satisfiable. ◁

The following two theorems are direct consequences of Lemma 3.26.

Theorem 3.28. *For every nontrivial ranked alphabet Σ, the nonemptiness problem of l-cftg over Σ is* NP*-hard.*

Theorem 3.29. *For every nontrivial ranked alphabet Σ, the uniform membership problem of l-cftg over Σ is* NP*-hard.*

Moreover, we can also prove a lower bound for infiniteness of l-cftg.

Theorem 3.30. *For every nontrivial ranked alphabet Σ, the infiniteness problem of l-cftg over Σ is* NP*-hard.*

Proof. Consider an l-cftg G. We can use a similar technique as in the proof of Lemma 3.24 to construct an l-cftg G' such that $\mathcal{L}(G)$ is nonempty if and only if $\mathcal{L}(G')$ is infinite. As deciding nonemptiness of G is NP-hard (Theorem 3.28), infiniteness of G' is NP-hard, too. □

Unfortunately, we could not find a nontrivial upper bound for any of the three problems above. We conjecture *(i)* that they are not PSPACE-hard, since it seems difficult to encode a PSPACE-hard problem (such as quantified Boolean formula validity) without unbounded copying, and *(ii)* that they are in fact NP-complete.

In contrast, the non-uniform membership problem of l-cftg is solvable efficiently.

Theorem 3.31. *The non-uniform membership problem of l-cftg is in* P*.*

Proof. Consider an l-cftg $G = (N, \Sigma, S, P)$ and a tree $t \in \mathrm{T}_\Sigma$. We can assume without loss of generality that G is in linear normal form, by Theorem 2.18. The normal form construction's runtime does not have to be attributed for, as G is not part of the problem's input.

Note that $\{t\}$ is a tree language recognizable by a dfta with $|t|$ states. Thus we can construct an ln-cftg $G' = (N', \Sigma, S', P')$ such that $\mathcal{L}(G') = \mathcal{L}(G) \cap \{t\}$, using the method from Theorem 2.35. Note that this method preserves the productions' shapes, therefore G' is also in linear normal form. With the abbreviation $m = \max \mathrm{rk}(N)$, we obtain that

$$|N'| \leq |N| \cdot |t|^{m+1} \qquad \text{and} \qquad |P'| \leq |P| \cdot |t|^{2m+1},$$

so $|G'|$ is polynomial in $|G|$. Applying Theorem 3.33, we may decide whether $\mathcal{L}(G') \neq \emptyset$ in time polynomial in $|G'|$, and therefore also in $|G|$. Since $\mathcal{L}(G') \neq \emptyset$ if and only if $t \in \mathcal{L}(G)$, the theorem is proven. □

Corollary 3.32. *The non-uniform membership problem of ln-cftg is in* P.

If we demand that the input l-cftg is additionally nondeleting, then the nonemptiness problem becomes feasible.

Theorem 3.33. *For every ranked alphabet Σ, the nonemptiness problem of ln-cftg over Σ is in* P.

Proof. Consider an ln-cftg $G = (N, \Sigma, S, P)$ with initial nonterminal S. Note we cannot assume that G is in linear normal form, as eliminating the torsions from G would cause a superpolynomial size increase of factor $\max \mathrm{rk}(N)!$.

Instead, we proceed as follows. Define, for every $Q \subseteq N$, the function $h_Q \colon \mathrm{T}(N \cup \Sigma)^1 \to \mathbb{B}$ such that

$$\begin{aligned} h_Q \colon \sigma(\xi_1, \ldots, \xi_k) &\mapsto h_Q(\xi_1) \wedge \cdots \wedge h_Q(\xi_k) \\ A(\xi_1, \ldots, \xi_k) &\mapsto \begin{cases} h_Q(\xi_1) \wedge \cdots \wedge h_Q(\xi_k) & \text{if } A \in Q, \\ 0 & \text{otherwise,} \end{cases} \\ x_i &\mapsto 1 \end{aligned}$$

for every $k \in \mathbb{N}$, $\sigma \in \Sigma^{(k)}$, $A \in N^{(k)}$, and $i \in \mathbb{N}$.

We use the following iterative procedure to determine nonemptiness of G.[3] Let $Q_0 = \emptyset$. Moreover, for every $i \in \mathbb{N}$, let

$$Q_{i+1} = Q_i \cup \left\{ A \in N \,\middle|\, (A \to \varrho) \in P,\, h_{Q_i}(\varrho) = 1 \right\}.$$

Since

$$Q_0 \subseteq Q_1 \subseteq Q_2 \subseteq \cdots \qquad \text{and} \qquad \bigcup_{i \in \mathbb{N}} Q_i \subseteq N\,,$$

there is some minimal $\ell \in \mathbb{N}$ such that

$$Q_\ell = \bigcup_{i \in \mathbb{N}} Q_i\,.$$

It is easy to see that Q_ℓ is computable in time polynomial in $|G|$. By straightforward induction arguments, one can show that for every $k \in \mathbb{N}$ and $A \in N^{(k)}$

$$A \in Q_\ell \qquad \text{if and only if} \qquad \mathcal{L}(G, A \cdot \mathrm{Id}_k) \neq \emptyset\,.$$

In particular, $S \in Q_\ell$ if and only if $\mathcal{L}(G) \neq \emptyset$. □

Theorem 3.34. *For every ranked alphabet Σ, the infiniteness problem of ln-cftg over Σ is in* P.

Proof. Consider an ln-cftg G over Σ. As we have shown above, the nonemptiness problem of ln-cftg over Σ is in P. Applying this knowledge to the construction in Theorem 2.29, we see that the cfg $\widehat{G}$ that generates the path language of G can be computed in deterministic polynomial time. Proceeding as in Lemma 3.25, we conclude that infiniteness of G is in P. □

[3] The procedure is similar to the "marking" procedure for deciding nonemptiness of cfg; see [22, Thm. 5.2].

3.6 Chapter Conclusion

In this chapter, we analyzed the complexity of decision problems of context-free tree grammars. As most problems for the unrestricted grammar model turn out to be computationally hard, we turned special attention to the decision problems of linear and linear and nondeleting cftg. Surprisingly, even if we only consider linear cftg, deletion is still powerful enough to make the emptiness problem NP-hard. Unfortunately, we could not find a (nontrivial) upper bound for the complexity of the emptiness problem of l-cftg. We conjecture the problem is solvable in nondeterministic polynomial time. Using the proof idea of Lemma 3.25, this would imply that also the infiniteness problem of l-cftg is in NP.

The complexity of uniform membership of ln-cftg is also left as an open problem. In all our ideas for an algorithm which decides whether $t \in \mathcal{L}(G)$ for a tree t and an ln-cftg G, the algorithm's worst-case runtime was in $\Omega\big(|t|^{1+m}\big)$, where m is the maximal rank of a nonterminal of G.

Chapter 4

Linear Context-Free Tree Languages and Inverse Linear Tree Homomorphisms

Contradictio est regula veri,
non contradictio falsi.

(Georg Wilhelm Friedrich Hegel)

The modular design of syntax-based natural language processing systems requires that the utilized class of tree languages $\mathcal{C}$ possesses a specific set of closure properties. In particular, for translation tasks it is important that $\mathcal{C}$ is closed under application of linear extended tree transducers (l-xtt).[1] This transducer model was first described by Rounds [140] (under the name *finite-state transformation with templates*), and further investigated, i.a., in [17, 66, 118]. Unfortunately, the closure under l-xtt does *not* hold when $\mathcal{C}$ is the class of context-free tree languages. This is due to a theorem of Arnold and Dauchet, who proved that the context-free tree languages are not closed under inverse linear tree homomorphisms; see Theorem 2.33. Trivially, every inverse linear tree homomorphism can be computed by an l-xtt. The proof of Theorem 2.33 works by constructing a *nonlinear* cftg G, and the preimage of the tree language of G under a certain tree homomorphism is shown to be non-context-free.

This proof suggests the assumption that the non-closure of CFT under inverse linear tree homomorphisms depends on the nonlinearity of the involved cftg – and that the situation could be remedied if we were to restrict $\mathcal{C}$ to the class CFT_ℓ. In this chapter, we will show that even in this restricted case, closure cannot be obtained: there are an l-cftg G_{ex} and a linear tree homomorphism h such that $L = h^{-1}(\mathcal{L}(G_{\text{ex}}))$ is *not* a context-free tree language. Since $\mathsf{CFT}_\ell \subseteq \mathsf{CFT}$, this property implies that CFT_ℓ is not closed under inverse linear tree homomorphisms.

The intuition behind our proof is as follows. Every tree t in L is of the form

for some $n \geq 1$ and monadic trees $u_1, v_1, \ldots, u_n, v_n$. Here, t is depicted such that its root is the leftmost symbol σ. The "horizontal" branch of t, labeled $\sigma \cdots \sigma\#$, is its *spine*. The

[1]Compare, e.g., the desired property (e) in the survey article [116, Sec. 3]. There, the weighted setting is considered, and $\mathcal{C}$ is the class of recognizable weighted tree languages.

subtrees u_i, v_i, called *chains* in the following, are built up over a parenthesis alphabet, such that the chains u_i contain only opening parentheses, the chains v_i only closing parentheses, and $u_1^R v_1 \cdots u_n^R v_n$ is a well-parenthesized word.

If one were to cut such a tree t into two parts t_1 and t_2, right through an edge between two σs, then one could observe that there are some chains u_j in t_1 which contain opening parentheses which are not closed in t_1, but only in t_2. A similar observation holds of course for some chains v_j in t_2. These chains u_j and v_j will be called *critical chains*, and their "unclosed" parts *defects*.

We assume that there is some (not necessarily linear) cftg G with $\mathcal{L}(G) = L$, and show that if G exists, then it can be assumed to be of a special normal form. We analyze the derivations of such a G in normal form. A derivation of a tree t as above begins with a subderivation

$$A(\#, \ldots, \#, \#) \Rightarrow_G^* B(s_1, \ldots, s_p, \#),$$

where A and B are nonterminals of G, and $s_1, \ldots, s_p$ are chains over the parenthesis alphabet. After that, the derivation continues with

$$B(s_1, \ldots, s_p, \#) \Rightarrow_G C(s'_1, \ldots, s'_p, D(s'_{p+1}, \ldots, s'_{2p}, \#)),$$

for some nonterminals C and D and $s'_1, \ldots, s'_{2p} \in \{s_1, \ldots, s_p\}$. Finally, C and D derive some terminal trees t_1 and t_2, respectively. So a derivation of t in G "cuts" t into two pieces as described above!

If G exists, it must therefore prepare the defects of t_1 and t_2 such that they "fit together", and it can only do so in the initial subderivation $A(\#, \ldots, \#) \Rightarrow_G^* B(s_1, \ldots, s_p, \#)$. But there are only finitely many parameters of A in which the defects could be prepared. We give a sequence of trees in L such that the number of their defects is strictly increasing, no matter how they are cut apart. Then there is some tree t in this sequence whose defects cannot be prepared fully. Hence it is possible to show by a pumping argument that if $t \in \mathcal{L}(G)$, then there is also a tree $t' \in \mathcal{L}(G)$ whose respective parts do not fit together, and therefore $t' \notin L$. Thus the existence of G is ruled out.

We conclude the chapter with a positive result: the tree languages of linear *monadic* cftg *are* closed under inverse linear tree homomorphisms. This fact, together with closure under intersection with recognizable tree languages and under application of linear tree homomorphisms (see Theorems 2.34 and 2.35), shows that the class of tree languages of lm-cftg is closed under application of l-xtt. The suitability of lm-cftg is further underscored by their expressive equivalence to the well-known linguistic formalism of *tree-adjoining grammars* [100, 70]. Our proof is based on the Greibach normal form of lm-cftg [63]. In fact, the closure of Greibach cftg under inverse linear tree homomorphisms was already proven by Arnold and Leguy [18], but their construction results in a nonlinear cftg of higher nonterminal rank.

This chapter is organized as follows. After establishing some specific notation in Section 4.1.1, we define the tree language L in Section 4.1.2. In Section 4.1.2, the grammar G_{ex} is introduced, while Section 4.1.2 contains the definition of the homomorphism h and some easy observations on L. In Section 4.1.3 we work out a normal form for the assumed cftg G, which allows us to define the concept of *derivation trees* of G in Section 4.1.4. This concept facilitates the analysis of the derivations in G. Section 4.1.5 contains some properties about factorizations of Dyck words, which formalize the idea of cutting t into two. Finally, in Section 4.1.6 we give a counterexample, and rule out the existence of G. Section 4.2 is about the positive result for lm-cftg.

Note: This chapter is based on a revised version of a technical report in collaboration with Toni Dietze and Luisa Herrmann [131]. Parts of the report have been published as a conference paper with the same coauthors [132]. The revised report has been accepted for publication in the journal *Information and Computation* [133].

Our research of inverse homomorphic closure of linear cftg has greatly benefited from our email correspondence with André Arnold. The idea for the intermediate normal form of G in Lemma 4.11 is due to him, and he showed us how to significantly improve the presentation of the results in Sections 4.1.5 and 4.1.6.

Large parts of the revised report mentioned above have been included into this chapter verbatim. Note however that Observations 3.4 and 6.1 from the report have been given proofs, and thus promoted to Lemmas 4.6 and 4.19, respectively. Moreover, Lemmas 4.4, 4.5 and 4.15 have now explicit formal proofs, instead of the brief sketches given in the report. Lastly, we give a complete proof of Lemma 4.30.

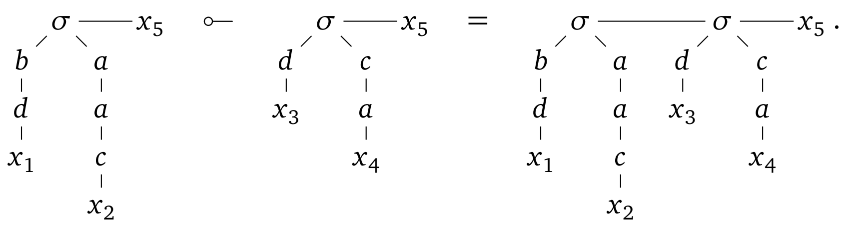
Table 4.1: Magmoid notation (where $[U, x_{n+1}] = \{[u, x_{n+1}] \mid u \in U\}$)

	tuples of trees	tuples of chains
general	$\mathrm{T}(\Sigma)^n_k$	$\mathrm{C}(\Sigma)_n = [\mathrm{T}(\Sigma)^n_n, x_{n+1}]$
torsion-free	$\widetilde{\mathrm{T}}(\Sigma)^n_k$	$\widetilde{\mathrm{C}}(\Sigma)_n = [\widetilde{\mathrm{T}}(\Sigma)^n_n, x_{n+1}]$
torsions	Θ^n_k	$\widehat{\Theta}_n = [\Theta^n_n, x_{n+1}]$

4.1 Linear Context-Free Tree Languages and Inverse Linear Tree Homomorphisms

4.1.1 Notation

As discussed in the introduction, we will consider trees made up of a spine, drawn horizontally, from which spring several monadic subtrees, called the tree's chains, drawn vertically. The following special notation will be helpful to denote and handle such *spine-trees.* We define an operator $\circ\!\!-$, which intuitively concatenates two spine-trees "along their spine." As an example for this operation,



Formally, consider an arbitrary ranked alphabet Σ. For every n, $k \in \mathbb{N}$, $s \in \mathrm{T}(\Sigma)^1_{n+1}$ and $t \in \mathrm{T}(\Sigma)^1_k$, let

$$s \circ\!\!- t = s \cdot [\mathrm{Id}_n,\, t]\,.$$

Of course, this definition only captures the above intuition if x_{n+1} is situated precisely on the spine of s. So whenever we use $\circ\!\!-$, we will take care that x_{n+1} is in this position. We assume that $\cdot$ binds stronger than $\circ\!\!-$. So, for instance, $t \cdot u \circ\!\!- s \cdot v$ means $(t \cdot u) \circ\!\!- (s \cdot v)$.

In the same vein, when we substitute an n-tuple u of chains into a spine-tree $t \in \mathrm{T}(\Sigma)^1_{n+1}$, we would like the variable x_{n+1} on t's spine to remain unaffected. This is obtained by adjoining x_{n+1} to u: for every $n \in \mathbb{N}$, let

$$\mathrm{C}(\Sigma)_n = \left\{[u, x_{n+1}] \,\middle|\, u \in \mathrm{T}(\Sigma)^n_n\right\}, \quad \widetilde{\mathrm{C}}(\Sigma)_n = \widetilde{\mathrm{T}}(\Sigma)^{n+1}_{n+1} \cap \mathrm{C}(\Sigma)_n\,, \quad \text{and} \quad \widehat{\Theta}_n = \Theta^{n+1}_{n+1} \cap \mathrm{C}(\Sigma)_n\,.$$

Remark 4.1. To prevent a possible source of confusion: the mnemonic "C" refers here to "chains", and *not* to "contexts", as sometimes used in the literature on tree languages. In our nomenclature, the latter are torsion-free trees, i.e., elements of $\widetilde{\mathrm{T}}(\Sigma)$. ◁

The magmoid-notation we use in this chapter is summarized in Table 4.1.

4.1.2 The Tree Language L

We start out by introducing the l-cftg G_{ex}. The preimage L of $\mathcal{L}(G_{\mathrm{ex}})$ under a particular linear tree homomorphism h, introduced afterwards, will be shown to be non-context-free.

The Grammar G_{ex}

Let

$$\Delta = \{\delta_1^{(2)}, \delta_2^{(2)}, \#^{(0)}\} \cup \Gamma\,, \qquad \text{where} \qquad \Gamma = \{a^{(1)}, b^{(1)}, c^{(1)}, d^{(1)}\}\,.$$

Consider the l-cftg $G_{\mathrm{ex}} = (N_{\mathrm{ex}}, \Delta, \xi_{\mathrm{ex}}, P_{\mathrm{ex}})$ with set of nonterminal symbols $N_{\mathrm{ex}} = \{A^{(3)}\}$, axiom

$$\xi_{\mathrm{ex}} = \delta_1\big(\#, A(c\#, d\#, \delta_2(\#, \#))\big)\,,$$

and productions in P_{ex} given by

$$A \;\to\; A(ax_1, bx_2, x_3) \;+\; A\big(ccx_1, d\#, A(c\#, ddx_2, x_3)\big) \;+\; \delta_2\big(cx_1, \delta_1(dx_2, x_3)\big)\,.$$

Note that G_{ex} is nondeleting and ordered, but neither Greibach nor coregular. The axiom and productions of G_{ex} are depicted in Figure 4.1.

Example 4.2. The following is an example derivation of a tree in $\mathcal{L}(G_{\mathrm{ex}})$.

$$\begin{aligned}
\xi_{\mathrm{ex}} = \;& \delta_1(\#, x) \circ\!\!- A(c\#, d\#, x) \circ\!\!- \delta_2(\#, \#)\\
\Rightarrow^*_{G_{\mathrm{ex}}} & \delta_1(\#, x) \circ\!\!- A(a^2c\#, b^2d\#, x) \circ\!\!- \delta_2(\#, \#)\\
\Rightarrow_{G_{\mathrm{ex}}} & \delta_1(\#, x) \circ\!\!- A(c^2a^2c\#, d\#, x) \circ\!\!- A(c\#, d^2b^2d\#, x) \circ\!\!- \delta_2(\#, \#)\\
\Rightarrow^*_{G_{\mathrm{ex}}} & \delta_1(\#, x) \circ\!\!- A(ac^2a^2c\#, bd\#, x) \circ\!\!- A(a^2c\#, b^2d^2b^2d\#, x) \circ\!\!- \delta_2(\#, \#)\\
\Rightarrow^*_{G_{\mathrm{ex}}} & \delta_1(\#, x) \circ\!\!- \delta_2(cac^2a^2c\#, x) \circ\!\!- \delta_1(dbd\#, x)\\
& \qquad \circ\!\!- \delta_2(ca^2c\#, x) \circ\!\!- \delta_1(db^2d^2b^2d\#, x) \circ\!\!- \delta_2(\#, \#)\,.
\end{aligned}$$

The resulting tree is depicted in Figure 4.2. ◁

The Homomorphism h and Its Preimage

Let $\Sigma = \{\sigma^{(3)}, \#^{(0)}\} \cup \Gamma$ and let $h\colon \mathrm{T}(\Sigma) \to \mathrm{T}(\Delta)$ be the linear tree homomorphism with

$$h\big(\sigma(x_1, x_2, x_3)\big) = \delta_1\big(x_1, \delta_2(x_2, x_3)\big) \quad \text{and} \quad h(\omega) = \omega \text{ for each } \omega \in \Sigma \setminus \{\sigma\}\,.$$

Note that h is surjective on $\mathcal{L}(G_{\mathrm{ex}})$, and, in addition, injective, nondeleting, strict, and elementary.

The tree language that is the homomorphic preimage of the language of G_{ex} is denoted $L = h^{-1}(\mathcal{L}(G_{\mathrm{ex}}))$. Since h is the identity on all symbols but σ, it is easy to see that every tree $t \in L$ is of the form

$$\sigma\big(\#, u_1\#, x\big) \circ\!\!- \sigma\big(v_1\#, u_2\#, x\big) \circ\!\!- \cdots \circ\!\!- \sigma\big(v_{n-1}\#, u_n\#, x\big) \circ\!\!- \sigma\big(v_n\#, \#, \#\big)$$

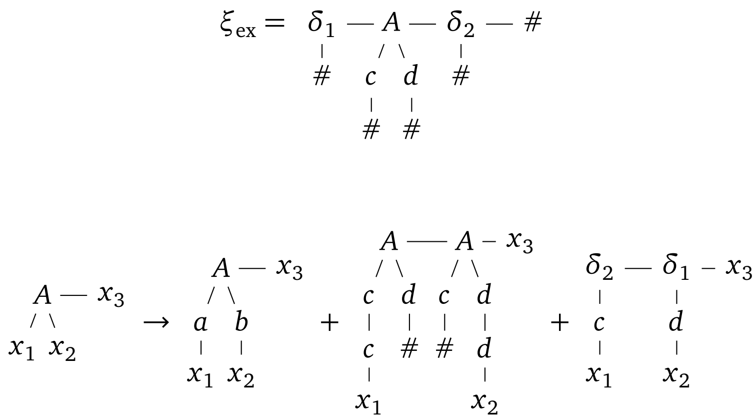


Figure 4.1: Axiom and productions of G_{ex}

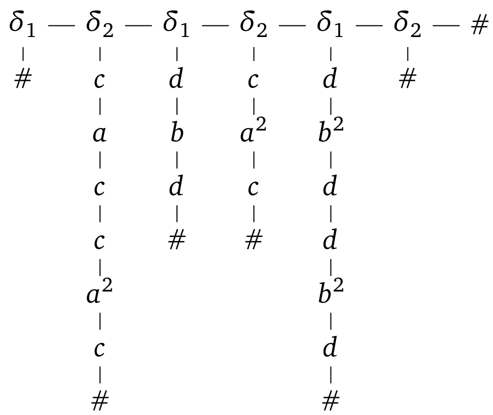


Figure 4.2: A derived tree of G_{ex}

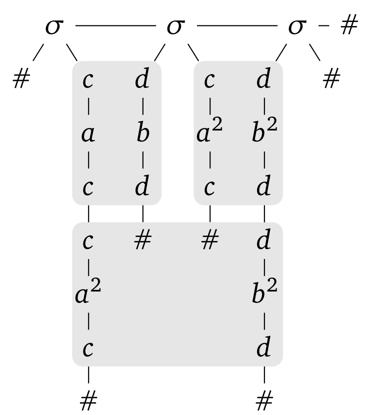


Figure 4.3: Its preimage under h

for some $n \geq 1$, and $u_i \in (ca^*c)^+$, $v_i \in (db^*d)^+$, for each $i \in [n]$. In general, given a tree t of the form

$$\sigma\big(v_1\#, u_1\#, x\big) \circ\!\!- \cdots \circ\!\!- \sigma\big(v_n\#, u_n\#, \zeta\big) \qquad \text{with } n \geq 1\,, \quad \zeta \in \{\#\} \cup X\,, \tag{4.1}$$

where $v_i \in \{b,d\}^*$ and $u_i \in \{a,c\}^*$, $i \in [n]$, we will call the monadic subtrees u_j (resp. v_j) of t the *a-chains* (resp. the *b-chains*) of t. A *chain* is either an a- or a b-chain. The rightmost root-to-leaf path in t (that is labeled $\sigma \cdots \sigma\zeta$) will be referred to as t's *spine*.

We introduce a notation to read off the chains of a tree as above, and arrange them in a word. Formally, for every tree t of the form as in (4.1), we let

$$\iota(t) = v_1 u_1^R v_2 u_2^R \cdots v_n u_n^R\,.$$

A similar, but slightly more general, notation is defined for sentential forms of G_{ex}, which may also contain variables. It will come to use in some of the following proofs. Let

$$\iota'\big(\delta_1(v\zeta,\xi)\big) = v\iota'(\xi)\,, \quad \iota'\big(\delta_2(u\zeta,\xi)\big) = u^R\iota'(\xi)\,, \quad \iota'\big(A(u\zeta, v\zeta', \xi)\big) = u^R v\iota'(\xi)\,,$$

and $\iota'(\zeta) = \varepsilon$, for each $\xi \in \mathrm{T}(N_{\text{ex}} \cup \Delta)_0^1$, $u \in \{a,c\}^*$, $v \in \{b,d\}^*$, and ζ, $\zeta' \in \{\#\} \cup X_3$. There is the following connection between ι and ι'.

Observation 4.3. *Consider a tree $t \in \mathrm{T}(\Sigma)_0^1$ of the form*

$$t = \sigma\big(\#, u_1\#, x\big) \circ\!\!- \sigma\big(v_1\#, u_2\#, x\big) \circ\!\!- \cdots \circ\!\!- \sigma\big(v_{n-1}\#, u_n\#, x\big) \circ\!\!- \sigma\big(v_n\#, \#, \#\big),$$

where $n \geq 1$, $u_1, \ldots, u_n \in \{a,c\}^+$, and $v_1, \ldots, v_n \in \{b,d\}^+$. Further, consider $s \in \mathrm{T}(\Delta)_0^1$ with

$$s = \delta_1(\#, x) \circ\!\!- \delta_2(u_1'\#, x) \circ\!\!- \delta_1(v_1'\#, x) \circ\!\!- \cdots \circ\!\!- \delta_2(u_m'\#, x) \circ\!\!- \delta_1(v_m'\#, x) \circ\!\!- \delta_2(\#, \#),$$

for $m \geq 1$, $u_1', \ldots, u_m' \in \{a,c\}^+$, and $v_1', \ldots, v_m' \in \{b,d\}^+$. Then

$$\iota(t) = \iota'(s) \qquad \textit{if and only if} \qquad h(t) = s\,.$$

The preimage of the tree from Example 4.2 is shown in Figure 4.3. As we see there, there is a close correspondence between factors of the chains of this tree, which is indicated by gray shading. For example, the factors cac and dbd correspond to each other. This correspondence holds for every $t \in L$, and can be formalized as follows. We view Γ as a parenthesis alphabet, such that b acts as right inverse to a, and d to c. Then $\iota(t)$ is a Dyck word, for every $t \in L$.

Lemma 4.4. *For every $t \in L$, $\iota(t) \in D_\Gamma^*$.*

Proof. We show for every $\xi \in \mathrm{T}(N_{\text{ex}} \cup \Delta)_0^1$ that if $\xi_{\text{ex}} \Rightarrow_{G_{\text{ex}}}^* \xi$, then $\iota'(\xi) \in D_\Gamma^*$. The proof is by induction on the length of the derivation. For the induction base, $\xi = \xi_{\text{ex}}$, and therefore $\iota'(\xi) = cd \in D_\Gamma^*$. For the induction step, assume $n \in \mathbb{N}$ such that $\xi_{\text{ex}} \Rightarrow_{G_{\text{ex}}}^n \xi$ and $\iota'(\xi) \in D_\Gamma^*$. Moreover, assume that

$$\xi = \xi_1 \circ\!\!- A(u\#,\, v\#,\, x) \circ\!\!- \xi_2$$

for some $\xi_1 \in \widetilde{\mathrm{T}}(N_{\text{ex}} \cup \Delta)_1^1$, $\xi_2 \in \mathrm{T}(N_{\text{ex}} \cup \Delta)_0^1$, $u \in \{a,c\}^*$, and $v \in \{b,d\}^*$. Clearly, the parameters of A are necessarily of this form. By the definition of ι', it is easy to see that then there are w_1 and $w_2 \in \Gamma^*$ such that $\iota'(\xi) = w_1 u^R v w_2$. We proceed by a case analysis on the production of G_{ex} that is applied in the next step.

(I) If

$$\xi_1 \circ\!\!- A(u\#, v\#, x) \circ\!\!- \xi_2 \Rightarrow_{G_{\mathrm{ex}}} \underbrace{\xi_1 \circ\!\!- A(au\#, bv\#, x) \circ\!\!- \xi_2}_{\xi'},$$

then $\iota'(\xi') = w_1 u^R abvw_2$. By the definition of the Dyck congruence, $u^R abv \equiv u^R v$, therefore $w_1 u^R abvw_2 \equiv w_1 u^R vw_2$ and $\iota'(\xi') \in D_\Gamma^*$.

(II) If

$$\xi_1 \circ\!\!- A(u\#, v\#, x) \circ\!\!- \xi_2 \Rightarrow_{G_{\mathrm{ex}}} \underbrace{\xi_1 \circ\!\!- A(ccu\#, d\#, x) \circ\!\!- A(c\#, ddv\#, x) \circ\!\!- \xi_2}_{\xi'},$$

then $\iota'(\xi') = w_1 u^R ccdcddvw_2$. Again, it is easy to check that $w_1 u^R ccdcddvw_2 \equiv w_1 u^R vw_2$, and therefore $\iota'(\xi') \in D_\Gamma^*$.

(III) Finally, if

$$\xi_1 \circ\!\!- A(u\#, v\#, x) \circ\!\!- \xi_2 \Rightarrow_{G_{\mathrm{ex}}} \underbrace{\xi_1 \circ\!\!- \delta_2(cu\#, x) \circ\!\!- \delta_1(dv\#, x) \circ\!\!- \xi_2}_{\xi'},$$

then $\iota'(\xi') = w_1 u^R cdvw_2$, and $w_1 u^R cdvw_2 \equiv w_1 u^R vw_2$, hence $\iota(\xi') \in D_\Gamma^*$.

* * *

By this property, we obtain that $\iota'(t) \in D_\Gamma^*$ for every $t \in \mathcal{L}(G_{\mathrm{ex}})$. Moreover, Observation 4.3 implies that for every $t \in \mathrm{T}(\Sigma)_0^1$, $\iota'(h(t)) = \iota(t)$, and hence if $\iota'(h(t)) \in D_\Gamma^*$, then $\iota(t) \in D_\Gamma^*$. This proves the lemma. $\square$

There is the following relation between the numbers of symbol occurrences in $t \in L$.

Lemma 4.5. *For every $t \in L$, $|t|_c = |t|_d = 4 \cdot |t|_\sigma - 6$.*

Proof. Define the function $f : \mathrm{T}(N_{\mathrm{ex}} \cup \Delta)^1 \to \mathbb{N}$ such that

$$f(\xi) = 4 \cdot |\xi|_{\delta_1} + 4 \cdot |\xi|_{\delta_2} + 6 \cdot |\xi|_A - 12 - |\xi|_c - |\xi|_d$$

for every $\xi \in \mathrm{T}(N_{\mathrm{ex}} \cup \Delta)^1$.

We show for every $n \in \mathbb{N}$ and $\xi \in \mathrm{T}(N_{\mathrm{ex}} \cup \Delta)_0^1$ that if $\xi_{\mathrm{ex}} \Rightarrow_{G_{\mathrm{ex}}}^n \xi$, then $f(\xi) = 0$.

The proof is by induction on n. The property is clear for the induction base $n = 0$, since then $\xi = \xi_{\mathrm{ex}}$. For the induction step, it is sufficient to note that for every production $A \to \varrho$ of G_{ex}, we have $f(\varrho) - f(A) = 0$, and therefore $f(\xi) = f(\xi_{\mathrm{ex}}) = 0$.

By this property, we obtain for every $t \in \mathcal{L}(G_{\mathrm{ex}})$ that

$$|t|_c = |t|_d = 2 \cdot |t|_{\delta_1} + 2 \cdot |t|_{\delta_2} - 6\,.$$

Since $|h(t')|_{\delta_1} + |h(t')|_{\delta_2} = 2 \cdot |t'|_\sigma$ for every $t' \in L$, and h is a surjection onto $\mathcal{L}(G_{\mathrm{ex}})$, we conclude that

$$|t'|_c = |t'|_d = 4 \cdot |t'|_\sigma - 6\,. \qquad \square$$

As the following lemma shows, each chain of $t \in L$ is determined uniquely by the other chains of t.

Lemma 4.6. *Let $t \in L$, let $w \in \mathrm{pos}(t)$ with $t(w) \in \Gamma \cup \{\#\}$, and let $s = t[x_1]_w$. There is exactly one $u \in \mathrm{T}(\Gamma \cup \{\#\})_0^1$ such that $s \cdot u \in L$, namely $u = t|_w$.*

Proof. By assumption, the position w is situated in one of the chains of t. Let us assume that this chain is an a-chain – the proof is analogous for the case that it is a b-chain. Let v_1, $v_2 \in \{a, c\}^*$ such that $t|_w = v_2\#$ and the a-chain we are considering is of the form $v_1 v_2 \#$. By inspection of the function ι, we see that $\iota(t) = w_1 v_2^R v_1^R w_2$ for some w_1, $w_2 \in \Gamma^*$.

Assume there is some $u \in \mathrm{T}(\Gamma \cup \#)_0^1$ such that $s \cdot u \in L$. By the form of L, there is some $u' \in \{a, c\}^*$ such that $u = u'\#$. Then $\iota(s \cdot u) = w_1 u'^R v_1^R w_2$, and by Lemma 4.4,

$$w_1 u'^R v_1^R w_2 \equiv w_1 v_2^R v_1^R w_2 \,.$$

Since v_2 and $u' \in \{a, c\}^*$, Lemma 1.8 can be applied. We obtain $u' \equiv v_2$, and since both these words are made up only of symbols a and c, this implies that $u' = v_2$. So the only $u \in \mathrm{T}(\Gamma \cup \#)_0^1$ such that $s \cdot u \in L$ is $t|_w$. □

Example 4.7. The preimage under h of the tree from Example 4.2 (cf. Figure 4.2) is

$$t \;=\; \sigma(\#, cac^2a^2c\#, x) \multimapinv \sigma(dbd\#, ca^2c\#, x) \multimapinv \sigma(db^2d^2b^2d\#, \#, \#)\,,$$

depicted in Figure 4.3. Obviously, $\iota(t) = ca^2c^2acdbdca^2cdb^2d^2b^2d$, and it takes only a little patience to verify that $\iota(t) \in D_\Gamma^*$. ◁

In the following sections, we will prove that there is no cftg G with $\mathcal{L}(G) = L$. Therefore, the following theorem holds.

Theorem 4.8. *There are an l-cftg G_{ex} and a linear, nondeleting, strict, and injective tree homomorphism h such that $h^{-1}(\mathcal{L}(G_{\mathrm{ex}}))$ is not a context-free tree language.*

Corollary 4.9. *The class of linear context-free tree languages is not closed under inverse linear tree homomorphisms.*

The theorem might seem surprising, as L and $\mathcal{L}(G_{\mathrm{ex}})$ are nearly the same: their only difference is that σ is split up into δ_1 and δ_2. However, this separation gives G_{ex} the power to create the chains under δ_1 and δ_2 independently, while a cftg generating L would have to derive them simultaneously. As described in the introduction, and proved further on, this would require nonterminals of unbounded rank and is therefore impossible.

4.1.3 A Normal Form for G

Assume there is a cftg $G = (N, \Sigma, \xi_0, P)$ such that $\mathcal{L}(G) = L$. In this section, we show (in a sequence of intermediate normal forms) that if G exists, then it can be chosen to be of a very specific form: Let

$$t = \sigma(v_1\#, u_1\#, x) \multimapinv \cdots \multimapinv \sigma(v_n\#, u_n\#, \#) \in L\,.$$

If we consider the trees $\sigma(v_i\#, u_i\#, x)$ as *symbols* from an infinite alphabet Λ, then t can be understood as a word (and L as a word language) over Λ. In fact, in the course of the next lemmas, we will see that G can be assumed to be of a form that is quite close to a context-free word grammar. For example, in Lemma 4.13 it will be shown that the productions of G may be assumed to be of the forms

(i) $A \to B \cdot u$, with $u \in \mathrm{C}(\Gamma)_p$,

(ii) $A \to B \multimap C$, and

(iii) $A \to \sigma(x_i, x_j, x_{p+1})$,

which correspond to *(i)* chain productions $A \to B$, *(ii)* nonterminal productions $A \to BC$ and *(iii)* terminal productions $A \to \sigma$ of context-free grammars. In the next lemma, we start out with distinguishing nonterminals by whether they contribute to the spine of a tree or to its chains.

Lemma 4.10. *We may assume for G that there is $p \in \mathbb{N}_1$ such that $N = N_s \cup N_c$ for two disjoint sets N_s and N_c with $N_s = N_s^{(2p)}$ and $N_c = N_c^{(p)}$. Moreover, $\xi_0 = S(\#, \ldots, \#)$ for some $S \in N_s$, and every production in P is of one of the following forms:*

(A1) $A \to B\big(C_1, \ldots, C_p, D_1, \ldots, D_p\big)$,

(A2) $A \to x_{p+q}$,

(A3) $A \to \sigma(x_i, x_j, x_{p+q})$,

(A4) $E \to F\big(C_1, \ldots, C_p\big)$,

(A5) $E \to x_q$,

(A6) $E \to \gamma(x_q)$,

where A, B, D_1, ..., $D_p \in N_s$, E, F, C_1, ..., $C_p \in N_c$, i, j, $q \in [p]$, and $\gamma \in \Gamma$.

Proof. We begin by assuming that there is a number $p \in \mathbb{N}_1$ such that $N = N^{(p)}$, the productions in P are of the forms

(N1) $A(x_1, \ldots, x_p) \to B(C_1(x_1, \ldots, x_p), \ldots, C_p(x_1, \ldots, x_p))$,

(N2) $A(x_1, \ldots, x_p) \to x_i$ for some $i \in [p]$,

(N3) $A(x_1, \ldots, x_p) \to \gamma(x_i)$ for some $\gamma \in \Gamma$ and $i \in [p]$, or

(N4) $A(x_1, \ldots, x_p) \to \sigma(x_i, x_j, x_q)$ for some i, j, $q \in [p]$,

and that $\xi_0 = S(\#, \ldots, \#)$ for some $S \in N^{(p)}$. This assumption comes without loss of generality: we may demand that G is in normal form and then introduce dummy parameters to make every nonterminal of rank $p > 0$. One fixed parameter x_q can be used to store $\#$ through the course of every derivation, then it is possible to use the production $A \to x_q$ instead of $A \to \#$.

Let the regular tree grammar $H = (Q, \Sigma, s, R)$ be given by $Q = \{s, c\}$, where R contains the productions

$$s \to \sigma(c, c, s) \;+\; \# \qquad \text{and} \qquad c \to \gamma(c) \;+\; \#$$

for every $\gamma \in \Gamma$.

We use the construction from Theorem 2.32 to obtain a cftg $G' = (N', \Sigma, \xi'_0, P')$ such that $\mathcal{L}(G') = \mathcal{L}(G) \cap \mathcal{L}(H)$. Since $\mathcal{L}(G) \subseteq \mathcal{L}(H)$, it is clear that $\mathcal{L}(G') = \mathcal{L}(G)$. However, as a side-effect of the method, G' is of the desired form. We describe the method's application briefly, in our own notation. Let $N' = \{A_s^{(2p)} \mid A \in N^{(p)}\} \cup \{A_c^{(p)} \mid A \in N^{(p)}\}$.

Define two functions $\Phi_s \colon \mathrm{T}(N)_p^1 \to \mathrm{T}(N')_{2p}^1$ and $\Phi_c \colon \mathrm{T}(N)_p^1 \to \mathrm{T}(N')_p^1$ by simultaneous induction, such that

$$\Phi_c(x_i) = x_i\,, \quad \text{and} \quad \Phi_s(x_i) = x_{p+i}$$

for every $x_i \in X_p$, and

$$\Phi_c\big(A(\xi_1, \ldots, \xi_p)\big) = A_c\big(\Phi_c(\xi_1), \ldots, \Phi_c(\xi_p)\big)\,,$$
$$\Phi_s\big(A(\xi_1, \ldots, \xi_p)\big) = A_s\big(\Phi_c(\xi_1), \ldots, \Phi_c(\xi_p), \Phi_s(\xi_1), \ldots, \Phi_s(\xi_p)\big)$$

for every $A \in N$, and $\xi_1, \ldots, \xi_p \in \mathrm{T}(N)_p^1$.

For every production $A(x_1, \ldots, x_p) \to \varrho$ in P of form *(N1)* or *(N2)*, the set P' contains the productions

$$A_c(x_1, \ldots, x_p) \to \Phi_c(\varrho) \qquad \text{and} \qquad A_s(x_1, \ldots, x_{2p}) \to \Phi_s(\varrho)\,.$$

Moreover, for every production in P of form *(N3)*, resp. *(N4)*, P' contains the productions

$$A_c(x_1, \ldots, x_p) \to \gamma\big(\Phi_c(x_i)\big) = \gamma(x_i)\,,$$

and, resp.,

$$A_s(x_1, \ldots, x_{2p}) \to \sigma\big(\Phi_c(x_i), \Phi_c(x_j), \Phi_s(x_q)\big) = \sigma(x_i, x_j, x_{p+q})\,.$$

Let $N_s = \{A_s \mid A \in N\}$ and $N_c = \{A_c \mid A \in N\}$, and let $\xi'_0 = S_s(\#, \ldots, \#)$. Then it is easy to see that G' is of the form as demanded above. $\square$

In the next step we show that we may require that there are at most two spine-producing nonterminals on the right-hand side of a production of G.

Lemma 4.11. *We may assume for G that there is $p \in \mathbb{N}_1$ such that $N = N_c \cup N_s$ with $N_c = N^{(p)}$ and $N_s = N^{(p+1)}$. Moreover, $\xi_0 = S(\#, \ldots, \#)$ for some $S \in N_s$, and every production of G is of one of the following forms:*

(B1) $A \to B(C_1, \ldots, C_p, x_{p+1})$,

(B2) $A \to B \circ\!\!- D$,

(B3) $A \to x_{p+1}$,

(B4) $A \to \sigma(x_i, x_j, x_{p+1})$,

(B5) $E \to F(C_1, \ldots, C_p)$,

(B6) $E \to x_i$,

(B7) $E \to \gamma(x_i)$,

where A, B, $D \in N_s$, E, F, C_1, $\ldots$, $C_p \in N_c$, i, $j \in [p]$, and $\gamma \in \Gamma$.

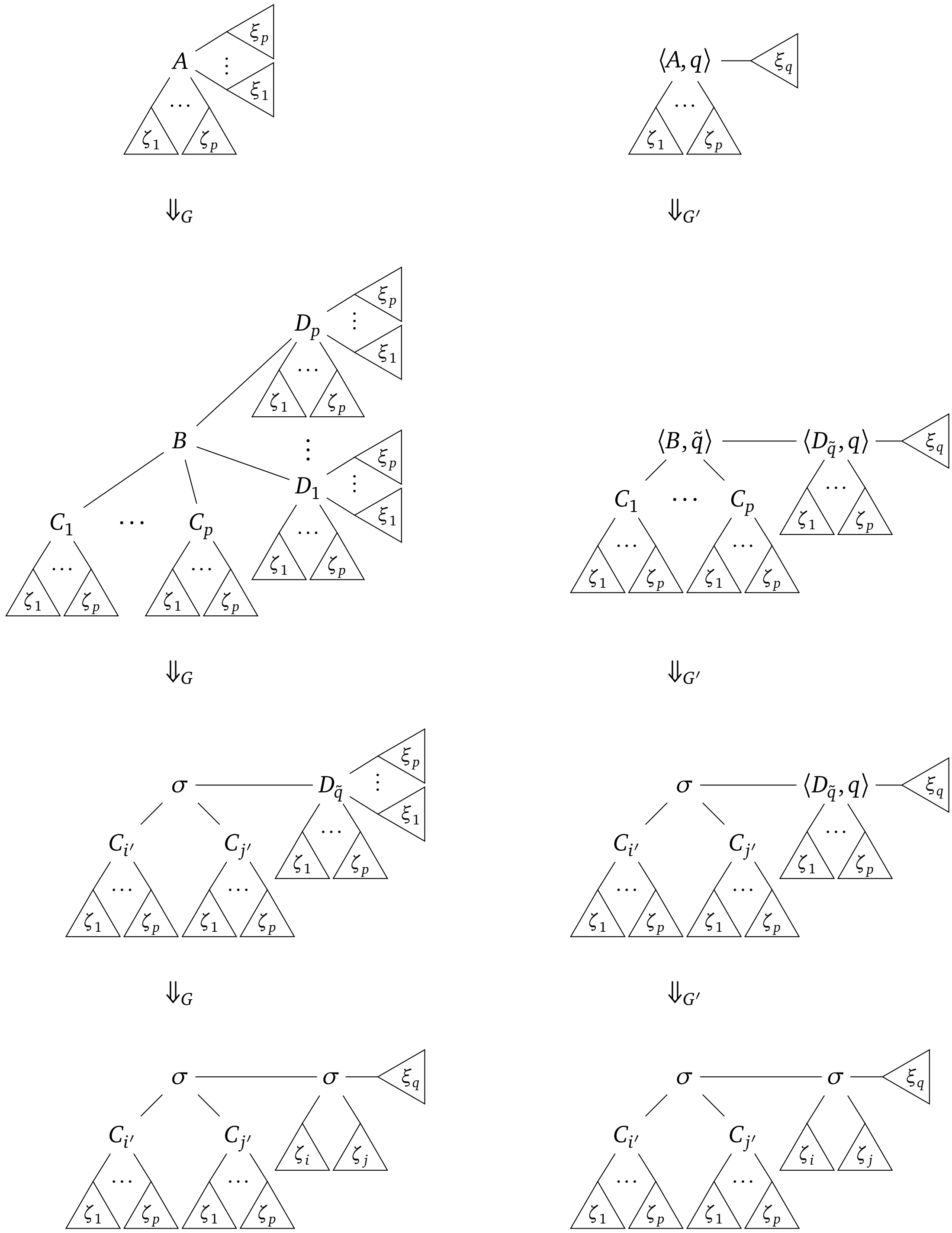


Figure 4.4: Corresponding derivations in G and G'

Proof. Assume that $G = (N, \Sigma, \xi_0, P)$ is of the form as given in Lemma 4.10. We will construct an equivalent cftg G'' of the form demanded above.

However, we construct first an intermediate cftg $G' = (N', \Sigma, \xi_0', P')$, where

$$N' = N_c \cup N_s' \cup \{S'\},$$

such that $S' \notin N_c \cup N_s'$ is a new nonterminal symbol of rank $p+1$, and

$$N_s' = \left\{ \langle A, q\rangle^{(p+1)} \mid A \in N_s,\, q \in [p] \right\}.$$

Moreover, $\xi_0' = S'(\#, \ldots, \#)$, and P' contains the productions

(i) $\langle A, q\rangle \to \langle B, \tilde{q}\rangle\big(C_1, \ldots, C_p, \langle D_{\tilde{q}}, q\rangle\big)$
for every production of form *(A1)*, and every q, $\tilde{q} \in [p]$;

(ii) $\langle A, q\rangle \to x_{p+1}$ for every production of form *(A2)*;[2]

(iii) $\langle A, q\rangle \to \sigma(x_i, x_j, x_{p+1})$ for every production of form *(A3)*;[2]

(iv) every production of form *(A4)*, *(A5)*, or *(A6)*,

(v) $S' \to \langle S, q\rangle$ for every $q \in [p]$.

Compare Figure 4.4 for the intuition behind this construction. On the left-hand side, a derivation in G is depicted (vertically). There, A is first rewritten using a production of type *(A1)*, then a type *(A3)* production is applied to the nonterminal B, discarding all spine-producing parameters except for the one with $D_{\tilde{q}}$ at its root. Finally, another type *(A3)* production is used to rewrite $D_{\tilde{q}}$, thereby choosing the parameter ξ_q and deleting all other ξ_j.

As a matter of fact, for every $A \in N_s$ and $t \in \mathcal{L}(G, A)$, there is precisely one occurrence of a variable from $\{x_{p+1}, \ldots, x_{2p}\}$ in t. The construction works by guessing beforehand which of these spine-producing parameters will eventually be chosen. Of course this guess must be propagated: it is encoded into the new nonterminals' second component. For example $\langle A, q\rangle$ means that our guess is that t will contain precisely x_{p+q}.

The right-hand side of Figure 4.4 shows the corresponding derivation in G'. As eventually ξ_q is chosen, the derivation begins with the nonterminal $\langle A, q\rangle$. This nonterminal is rewritten using the production $\langle A, q\rangle \to \langle B, \tilde{q}\rangle(C_1, \ldots, C_p, \langle D_{\tilde{q}}, q\rangle)$. This choice means that we guess that *before* the q-th parameter, the $\tilde{q}$-th parameter is selected. Note that these two guesses are checked afterwards, in the application of the two productions of type *(iii)*, according to their definition.

* * *

We now prove that $\mathcal{L}(G') = \mathcal{L}(G)$. To this end, it is necessary to consider only OI derivations, as otherwise counting derivation steps becomes bothersome. It is easy to prove by induction for every $n \in \mathbb{N}$, chain-producing nonterminal $C \in N_c$ and $t \in \mathrm{T}(\Sigma)_p^1$ that $C \overset{\mathrm{OI}}{\Rightarrow}{}_G^n\, t$ if and

[2]Note that here q is given by the respective productions *(A2)* and *(A3)*.

only if $C \overset{\text{OI}}{\Rightarrow}{}^n_{G'} t$. This is due to the fact that G and G' have exactly the same productions for nonterminals from N_c.

Next, we show for every $n \in \mathbb{N}$, $q \in [p]$, $A \in N_s$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$, that

$$A \overset{\text{OI}}{\Rightarrow}{}^n_G t \multimap x_{p+q} \qquad \text{if and only if} \qquad \langle A, q\rangle \overset{\text{OI}}{\Rightarrow}{}^n_{G'} t \multimap x_{p+1}\,.$$

Let us stress again that for every $A \in N_s$ and $t \in \mathcal{L}(G, A)$, there is precisely one occurrence of a variable from $\{x_{p+1}, \dots, x_{2p}\}$ in t; this is also the fundamental property behind the following proof. We proceed by complete induction on n (using Lemma 2.14 to decompose OI derivations). The base case $n = 0$ holds vacuously. Continue by a case analysis on the production applied first in the derivation. Let $n \in \mathbb{N}$, $A \in N_s$, $q \in [p]$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$.

(I) Assume that the production $A \to B(C_1, \dots, C_p, D_1, \dots, D_p)$ is in P. Then

$$A \overset{\text{OI}}{\Rightarrow}_G B(C_1, \dots, C_p, D_1, \dots, D_p) \overset{\text{OI}}{\Rightarrow}{}^n_G t \multimap x_{p+q}$$

iff $\exists m, n_1, n_2 \in \mathbb{N}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)^1_m$, $\vartheta \in \Theta^m_{2p}$, $v \in \mathrm{T}(\Sigma)^m_{2p}$:
$B \overset{\text{OI}}{\Rightarrow}{}^{n_1}_G \tilde{u} \cdot \vartheta$, $\quad \vartheta \cdot [C_1, \dots, C_p, D_1, \dots, D_p] \overset{\text{OI}}{\Rightarrow}{}^{n_2}_G v$, $\quad t \multimap x_{p+q} = \tilde{u} \cdot v$, $\quad n = n_1 + n_2$

iff $\exists m, n_1, n_2, n_3 \in \mathbb{N}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)^1_{m+1}$, $\vartheta \in \Theta^m_p$, $v \in \mathrm{T}(\Sigma)^m_p$, $\tilde{q} \in [p]$, $w \in \mathrm{T}(\Sigma)^1_{p+1}$: $\qquad (\dagger)$
$B \overset{\text{OI}}{\Rightarrow}{}^{n_1}_G \tilde{u} \cdot [\vartheta, x_{p+\tilde{q}}]$, $\quad \vartheta \cdot [C_1, \dots, C_p] \overset{\text{OI}}{\Rightarrow}{}^{n_2}_G v$, $\quad D_{\tilde{q}} \overset{\text{OI}}{\Rightarrow}{}^{n_3}_G w \multimap x_{p+q}$,
$t = \tilde{u} \cdot [v, w]$, $\quad n = n_1 + n_2 + n_3$

iff $\exists m, n_1, n_2, n_3 \in \mathbb{N}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)^1_{m+1}$, $\vartheta \in \Theta^m_p$, $v \in \mathrm{T}(\Sigma)^m_p$, $\tilde{q} \in [p]$, $w \in \mathrm{T}(\Sigma)^1_{p+1}$:
$\langle B, \tilde{q}\rangle \overset{\text{OI}}{\Rightarrow}{}^{n_1}_{G'} \tilde{u} \cdot [\vartheta, x_{p+1}]$, $\quad \vartheta \cdot [C_1, \dots, C_p] \overset{\text{OI}}{\Rightarrow}{}^{n_2}_{G'} v$, $\quad \langle D_{\tilde{q}}, q\rangle \overset{\text{OI}}{\Rightarrow}{}^{n_3}_{G'} w \multimap x_{p+1}$,
$t = \tilde{u} \cdot [v, w]$, $\quad n = n_1 + n_2 + n_3$

iff $\langle A, q\rangle \overset{\text{OI}}{\Rightarrow}_{G'} \langle B, \tilde{q}\rangle(C_1, \dots, C_p, \langle D_{\tilde{q}}, q\rangle) \overset{\text{OI}}{\Rightarrow}{}^n_{G'} t$.

To understand why direction "only if" holds at point (†) above, observe that at this point, $\pi_m \cdot v$ has the form $w \multimap x_{p+q}$, for some $w \in \mathrm{T}(\Sigma)^1_{p+1}$. Since $\pi_m \cdot v$ is generated by

$$\pi_{\vartheta(m)} \cdot [C_1, \dots, C_p, D_1, \dots, D_p]\,,$$

there is some $\tilde{q} \in [p]$ such that $D_{\tilde{q}} \overset{\text{OI}}{\Rightarrow}{}^*_G w \multimap x_{p+q}$.

(II) If the production $A \to x_{p+q}$ is in P, with $q \in [p]$, then

$$A \overset{\text{OI}}{\Rightarrow}_G x_{p+1} \multimap x_{p+q} \qquad \text{if and only if} \qquad \langle A, q\rangle \overset{\text{OI}}{\Rightarrow}_{G'} x_{p+1}$$

by construction.

(III) Finally, if $A \to \sigma(x_i, x_j, x_{p+q})$ is in P, then

$$A \overset{\text{OI}}{\Rightarrow}_G \sigma(x_i, x_j, x_{p+1}) \multimap x_{p+q} \qquad \text{if and only if} \qquad \langle A, q\rangle \overset{\text{OI}}{\Rightarrow}_{G'} \sigma(x_i, x_j, x_{p+1})\,.$$

* * *

So for every $t \in \mathrm{T}(\Sigma)^1_{2p}$, we have that $t \in \mathcal{L}(G,S)$ if and only if there is some $q \in [p]$ such that $t \multimap x_{p+1} \in \mathcal{L}(G', \langle S,q \rangle)$.

Let $s \in \mathrm{T}_\Sigma$. Then $s \in \mathcal{L}(G)$ if and only if there is $t \in \mathcal{L}(G,S)$ such that $s = t \cdot \langle \#, \dots, \# \rangle$, and by the above, this is equivalent to $t \multimap x_{p+1} \in \mathcal{L}(G', \langle S,q \rangle)$ for some $q \in [p]$. By use of the productions *(v)*, this holds precisely if $t \multimap x_{p+1} \in \mathcal{L}(G', S')$, i.e., $s \in \mathcal{L}(G')$. Therefore, $\mathcal{L}(G) = \mathcal{L}(G')$.

The cftg G'' results from G' by replacing every production of form $A \to B(C_1, \dots, C_p, D)$ in P' by the two productions

$$A \to B_{C_1 \cdots C_p} \multimap D \quad \text{and} \quad B_{C_1 \cdots C_p} \to B(C_1, \dots, C_p, x_{p+1})$$

for some new nonterminal $B_{C_1 \cdots C_p}$ of G''. It is easy to see that $\mathcal{L}(G'') = \mathcal{L}(G')$, so a formal proof is omitted. □

The next normal form follows from the property that the form of a chain of $t \in L$ is already determined after the spine of t has been derived (cf. Lemma 4.6). We can therefore omit chain-producing nonterminals.

Lemma 4.12. *We may assume that G is of the form $G = (N, \Sigma, \xi_0, P)$, such that $N = N^{(p+1)}$ for some $p \in \mathbb{N}_1$, $\xi_0 = S(\#, \dots, \#)$ for some $S \in N$ and the productions in P are of the forms*

(C1) $A \to B \cdot u$, *where* $u \in \mathrm{C}(\Gamma)_p$,

(C2) $A \to B \multimap C$,

(C3) $A \to x_{p+1}$,

(C4) $A \to \sigma(x_i, x_j, x_{p+1})$, *where* $i, j \in [p]$,

and where A, B, $C \in N$.

Proof. Assume that G is of the form given in Lemma 4.11.

The construction's idea is simply to replace in every production of G each occurrence of a nonterminal symbol $E \in N_c$ by an arbitrary tree that can be generated by E. The tricky part is to show that the choice of this tree (there may indeed be more than one such tree) does not matter.

Moreover, we may assume that $\mathcal{L}(G,E) \neq \emptyset$ for every $E \in N_c$, by Lemma 2.4, whose construction preserves our normal form.

Note that for every $E \in N_c$, we have $\mathcal{L}(G,E) \subseteq \mathrm{T}(\Gamma)^1_p$. Choose some fixed tree $u_E \in \mathcal{L}(G,E)$ for each $E \in N_c$, and let n, $m \in \mathbb{N}$. Moreover, let $N' = N_s$, where $\mathrm{rk}(A) = p+1$ for every $A \in N'$. Given $\xi \in \mathrm{T}(N \cup \{\#\})^n_m$, we define $\varphi(\xi) \in \mathrm{T}(N' \cup \Sigma)^n_m$ as follows. If $n \neq 1$, let

$$\varphi(\xi) = \big[\varphi(\pi_1 \cdot \xi), \dots, \varphi(\pi_n \cdot \xi)\big].$$

If $n = 1$, let

$$\begin{aligned}
\varphi(A \cdot \xi) &= A \cdot \varphi(\xi) && \text{for every } A \in N_s \text{ and } \xi \in \mathrm{T}(N)^{p+1}_m\,, \\
\varphi(E \cdot \xi) &= u_E \cdot \varphi(\xi) && \text{for every } E \in N_c \text{ and } \xi \in \mathrm{T}(N)^{p}_m\,, \\
\varphi(x_q) &= x_q && \text{for every } q \in [m]\text{, and} \\
\varphi(\#) &= \#\,.
\end{aligned}$$

We construct the cftg $G' = (N', \Sigma, \xi_0, P')$ where P' contains the productions *(i)* $A \to B(\varphi(C_1), \ldots, \varphi(C_p), x_{p+1})$ for every production of form *(B1)* in P, and

(ii) every production from P of form *(B2)*, *(B3)*, or *(B4)*.

Observe that in *(i)*, $\varphi(C_i) \in \mathrm{T}(\Gamma)^1_p$ for each $i \in [p]$.

(⊇) To prove that $\mathcal{L}(G') \subseteq \mathcal{L}(G)$, we show for every $n \in \mathbb{N}$, $A \in N'$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$ that

$$A \overset{\text{oi}}{\Rightarrow}{}^n_{G'} t \qquad \text{implies} \qquad A \Rightarrow^*_G t\,.$$

The induction base holds trivially. We continue with the following case analysis. Let $n \in \mathbb{N}$ and $t \in \mathrm{T}(\Sigma)^1_{p+1}$.

(I) Let

$$A \overset{\text{oi}}{\Rightarrow}_{G'} B(\varphi(C_1), \ldots, \varphi(C_p), x_{p+1}) \overset{\text{oi}}{\Rightarrow}{}^n_{G'} t \cdot [\varphi(C_1), \ldots, \varphi(C_p), x_{p+1}]$$

for some production $A \to B(C_1, \ldots, C_p, x_{p+1})$ in P. By the induction hypothesis, $B \Rightarrow^*_G t$, and clearly $C_i \Rightarrow^*_G \varphi(C_i)$ for each $i \in [p]$, therefore

$$A \Rightarrow_G B \cdot [C_1, \ldots, C_p, x_{p+1}] \Rightarrow^*_G t \cdot [\varphi(C_1), \ldots, \varphi(C_p), x_{p+1}]\,.$$

(II) Let $A \overset{\text{oi}}{\Rightarrow}_{G'} B \multimapdotinv D \overset{\text{oi}}{\Rightarrow}{}^n_{G'} t$ for some production $A \to B \multimapdotinv D$ in P. Then there are n_1, $n_2 \in \mathbb{N}$, u and $v \in \mathrm{T}(\Sigma)^1_{p+1}$ such that

$$B \overset{\text{oi}}{\Rightarrow}{}^{n_1}_{G'} u\,, \quad D \overset{\text{oi}}{\Rightarrow}{}^{n_2}_{G'} v\,, \quad n = n_1 + n_2\,, \quad \text{and} \quad t = u \multimapdotinv v\,.$$

By the induction hypothesis, we have that $B \Rightarrow^*_G u$ and $D \Rightarrow^*_G v$, and therefore

$$A \Rightarrow_G B \multimapdotinv D \Rightarrow^*_G u \multimapdotinv v\,.$$

(III) Let $A \overset{\text{oi}}{\Rightarrow}_{G'} x_{p+1}$. This means that also $A \Rightarrow_G x_{p+1}$.

(IV) Let $A \overset{\text{oi}}{\Rightarrow}_{G'} \sigma(x_i, x_j, x_{p+1})$. Then $A \Rightarrow_G \sigma(x_i, x_j, x_{p+1})$.

* * *

Let $s \in \mathcal{L}(G')$. Then there is some $t \in \mathrm{T}(\Sigma)^1_{p+1}$ such that $S \overset{\text{oi}}{\Rightarrow}{}^*_{G'} t$ and $s = t \cdot [\#, \ldots, \#]$. By the above, $t \in \mathcal{L}(G, S)$, and therefore $s \in \mathcal{L}(G)$. Thus, $\mathcal{L}(G') \subseteq \mathcal{L}(G)$.

(⊆) We continue the proof of correctness with the direction $\mathcal{L}(G) \subseteq \mathcal{L}(G')$. It rests on the following property. For every $n \in \mathbb{N}$, $A \in N_s$, $\xi \in \mathrm{T}(N \cup \{\#\})^{p+1}_0$, $s \in \widetilde{\mathrm{T}}(\Sigma)^1_1$, and $t \in \mathrm{T}(\Sigma)^1_0$,

$$\text{if} \quad \xi_0 \overset{\text{oi}}{\Rightarrow}{}^*_G s \cdot A \cdot \xi \overset{\text{oi}}{\Rightarrow}{}^n_G s \cdot t\,, \qquad \text{then also} \quad A \cdot \varphi(\xi) \Rightarrow^*_{G'} t\,.$$

The proof is by induction on n. The induction base holds vacuously, so again we continue with a case analysis. Let $n \in \mathbb{N}$, $s \in \widetilde{\mathrm{T}}(\Sigma)^1_1$, and $t \in \mathrm{T}(\Sigma)^1_0$.

(I) Let $\xi_0 \overset{\text{oi}}{\Rightarrow}{}^*_G s \cdot A \cdot \xi \overset{\text{oi}}{\Rightarrow}_G s \cdot B(C_1, \ldots, C_p, x_{p+1}) \cdot \xi \overset{\text{oi}}{\Rightarrow}{}^n_G s \cdot t$ by a production of form *(B1)*. Then

$$A \cdot \varphi(\xi) \Rightarrow_{G'} B \cdot [\varphi(C_1), \ldots, \varphi(C_p), x_{p+1}] \cdot \varphi(\xi) = B \cdot \varphi([C_1, \ldots, C_p, x_{p+1}] \cdot \xi)\,,$$

and by the induction hypothesis, $B \cdot \varphi([C_1, \ldots, C_p, x_{p+1}] \cdot \xi) \Rightarrow^*_{G'} t$.

(II) Let $\xi_0 \overset{\text{OI}}{\Rightarrow}{}^*_G s \cdot A \cdot \xi \overset{\text{OI}}{\Rightarrow}_G s \cdot B(x_1, \dots, x_p, D) \cdot \xi \overset{\text{OI}}{\Rightarrow}{}^n_G s \cdot t$ by a production of form *(B2)*. Then

$$A \cdot \varphi(\xi) \Rightarrow_{G'} B(x_1, \dots, x_p, D) \cdot \varphi(\xi) = B \cdot \varphi([x_1, \dots, x_p, D] \cdot \xi),$$

and by the induction hypothesis, $B \cdot \varphi([x_1, \dots, x_p, D] \cdot \xi) \Rightarrow^*_{G'} t$.

(III) Let $\xi_0 \overset{\text{OI}}{\Rightarrow}{}^*_G s \cdot A \cdot \xi \overset{\text{OI}}{\Rightarrow}_G s \cdot \pi_{p+1} \cdot \xi \overset{\text{OI}}{\Rightarrow}{}^n_G s \cdot t$ by the production $A \to x_{p+1}$. Then

$$A \cdot \varphi(\xi) \Rightarrow_{G'} x_{p+1} \cdot \varphi(\xi) = \varphi(\pi_{p+1} \cdot \xi).$$

If $\pi_{p+1} \cdot \xi = \#$, then $\varphi(\pi_{p+1} \cdot \xi) = \# = t$. Otherwise, $\pi_{p+1} \cdot \xi = B \cdot \kappa$ for some $B \in N_s$ and $\kappa \in \mathrm{T}(N \cup \{\#\})_0^{p+1}$. By the induction hypothesis, $\varphi(B \cdot \kappa) = B \cdot \varphi(\kappa) \Rightarrow^*_{G'} t$.

(IV) Let $u, v \in \Gamma^*$ such that, by a production of form *(B4)*,

$$\xi_0 \overset{\text{OI}}{\Rightarrow}{}^*_G s \cdot A \cdot \xi \overset{\text{OI}}{\Rightarrow}_G s \cdot \sigma(\pi_i \cdot \xi, \pi_j \cdot \xi, \pi_{p+1} \cdot \xi) \overset{\text{OI}}{\Rightarrow}{}^n_G s \cdot \sigma(u\#, v\#, t).$$

As in case **(III)**, either $\pi_{p+1} \cdot \xi = \#$, and then $\varphi(\pi_{p+1} \cdot \xi) = t$, or otherwise $\pi_{p+1} \cdot \xi = B \cdot \kappa$ with $B \cdot \varphi(\kappa) \Rightarrow^*_{G'} t$.

Moreover, as $s \cdot \sigma(u\#, v\#, t) \in \mathcal{L}(G)$, Lemma 4.6 implies that $\mathcal{L}(G, \pi_i \cdot \xi) = \{u\#\}$ and $\mathcal{L}(G, \pi_j \cdot \xi) = \{v\#\}$. Further, by the definition of φ, it is easy to see that $\pi_i \cdot \xi \Rightarrow^*_G \varphi(\pi_i \cdot \xi)$ and $\pi_j \cdot \xi \Rightarrow^*_G \varphi(\pi_j \cdot \xi)$. We conclude that $\varphi(\pi_i \cdot \xi) = u\#$ and $\varphi(\pi_j \cdot \xi) = v\#$. So

$$A \cdot \varphi(\xi) \Rightarrow_{G'} \sigma\big(u\#, v\#, \varphi(\pi_{p+1} \cdot \xi)\big) \Rightarrow^*_{G'} \sigma(u\#, v\#, t).$$

* * *

Let $t \in \mathcal{L}(G)$. Then $\xi_0 = S \cdot [\#, \dots, \#] \overset{\text{OI}}{\Rightarrow}{}^*_G t$. The above property yields

$$S \cdot [\#, \dots, \#] = S \cdot \varphi([\#, \dots, \#]) \Rightarrow^*_{G'} t,$$

and hence $t \in \mathcal{L}(G')$. □

It turns out that, to derive the spine of $t \in L$, no projecting productions $A \to x_i$ are required: since G is close to a context-free word grammar with productions *(C1)* $A \to B$, *(C2)* $A \to BC$, *(C3)* $A \to \varepsilon$ and *(C4)* $A \to \sigma$, we can eliminate the productions of form *(C3)* by using the well-known method to remove ε-productions from context-free grammars.

Lemma 4.13. *Lemma 4.12 still holds if (C3) is removed from its statement.*

Proof. Let G be of the form as in Lemma 4.12 and let

$$Q = \left\{A \in N \;\middle|\; A(x_1, \dots, x_{p+1}) \Rightarrow^*_G x_{p+1}\right\}.$$

Construct the cftg $G' = (N, \Sigma, \xi_0, P')$, where P' contains all productions from P of forms *(C1)*, *(C2)* and *(C4)*. Moreover, for every production of form *(C2)*, P' contains the productions

$$A \to B \quad \text{if} \quad C \in Q, \qquad \text{and} \qquad A \to C \quad \text{if} \quad B \in Q.$$

Observe that both productions are of form *(C1)*. We will now prove that $\mathcal{L}(G') = \mathcal{L}(G)$.

(⊆) For the direction $\mathcal{L}(G') \subseteq \mathcal{L}(G)$, we show for every $n \in \mathbb{N}$, $A \in N$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$, that if $A \Rightarrow^n_{G'} t$, then also $A \Rightarrow^*_G t$. The proof is by complete induction on n. The induction base is trivial; we proceed by a case analysis on the form of the production applied first.

The case that the assumed derivation begins with a production of form *(C2)* or *(C4)* is straightforward: after all, these productions are from P by construction.

So let us assume that $A \Rightarrow_{G'} B \Rightarrow^n_{G'} t$. From the induction hypothesis, $B \Rightarrow^*_G t$. There are three subcases. Either, the production $A \to B$ is in P, in which case $A \Rightarrow^*_G t$. Otherwise, by construction, there is some production $A \to B \multimap C$ or $A \to C \multimap B$ in P such that $C \Rightarrow^*_G x_{p+1}$. If it is $A \to B \multimap C$ (the other case is analogous), we have

$$A \Rightarrow_G B \multimap C \Rightarrow^*_G B \Rightarrow^*_G t\,.$$

* * *

The above property allows us to reason as follows. For every $s \in \mathcal{L}(G')$, there is some $t \in \mathcal{L}(G', S)$ with $s = t \cdot [\#, \ldots, \#]$. By the above, $S \Rightarrow^*_G t$, and therefore $s \in \mathcal{L}(G)$.

(⊇) It remains to show the direction $\mathcal{L}(G) \subseteq \mathcal{L}(G')$. We show for every $n \in \mathbb{N}$, $A \in N$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$, that if $A \Rightarrow^n_G t$ and $t \neq x_{p+1}$, then also $A \Rightarrow^*_{G'} t$. The proof is by complete induction on n. The induction base is trivial, so again we continue by a case analysis on derivations of nonzero length. Let $n \in \mathbb{N}$, $A \in N$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$ with $t \neq x_{p+1}$.

If $A \Rightarrow_G B \multimap C \Rightarrow^n_G t$, then there are n_1, $n_2 \in \mathbb{N}$ and t_1, $t_2 \in \mathrm{T}(\Sigma)^1_{p+1}$ with $t = t_1 \multimap t_2$, $B \Rightarrow^{n_1}_G t_1$, $C \Rightarrow^{n_2}_G t_2$, and $n = n_1 + n_2$. If neither $t_1 = x_{p+1}$ nor $t_2 = x_{p+1}$, then by the induction hypothesis, also

$$A \Rightarrow_{G'} B \multimap C \Rightarrow^*_{G'} t_1 \multimap t_2 = t\,.$$

If precisely one of t_1 and t_2 is equal to x_{p+1} (say $t_1 = x_{p+1}$, the other case is analogous), then the production $A \to C$ is in P'. So, with the induction hypothesis,

$$A \Rightarrow_{G'} C \Rightarrow^*_{G'} t_2 = t\,.$$

The case $t_1 = t_2 = x_{p+1}$ is precluded by the assumption that $t \neq x_{p+1}$.

Similarly, the case that the first production is of form *(C3)* is precluded by the assumption on t. For any other production, the proof goes through without surprises.

* * *

Now let $s \in \mathcal{L}(G)$. Then there is $t \in \mathcal{L}(G, S)$ such that $s = t \cdot [\#, \ldots, \#]$. Note that $t \neq x_{p+1}$, because $\# \notin \mathcal{L}(G)$. Thus by the above property, $S \Rightarrow^*_{G'} t$, and therefore $s \in \mathcal{L}(G')$. □

Finally, it is convenient to remove the torsions from productions of the form *(C1)*. Then whenever $A \Rightarrow^*_G B \cdot u$, we know that u is torsion-free, because torsion-free tuples are closed under concatenation with $\cdot$. The construction works by guessing which torsion will be applied in the next derivation step, and pre-arranging this torsion in the tuple of the current production. Of course, this guess must be stored in the new grammar's nonterminals. Moreover, there is a price to pay: we must now allow for torsions in "branching" productions $A \to B \cdot \vartheta_1 \multimap C \cdot \vartheta_2$.

Lemma 4.14. *We may assume that G is of the form $G = (N, \Sigma, \xi_0, P)$, such that $N = N^{(p+1)}$ for some $p \in \mathbb{N}$, $\xi_0 = S(\#, \dots, \#)$ for some $S \in N$ and the productions in P are of the forms*

(D1) $A \to B \cdot u$, *where* $u \in \widetilde{\mathrm{C}}(\Gamma)_p$,

(D2) $A \to B \cdot \vartheta_1 \multimap C \cdot \vartheta_2$, *where* ϑ_1, $\vartheta_2 \in \widehat{\Theta}_p$,

(D3) $A \to \sigma(x_i, x_j, x_{p+1})$, *where* i, $j \in [p]$,

and where A, B, $C \in N$.

Proof. Assume that G is as in Lemma 4.13. Construct a new cftg $G' = (N', \Sigma, \xi'_0, P')$, where $N' = \{A^{\vartheta} \mid A \in N, \vartheta \in \widehat{\Theta}_p\} \cup \{S'\}$ for some distinct nonterminal S', $\xi'_0 = S'(\#, \dots, \#)$, and P' contains the productions

(i) $A^{\vartheta'} \to B^{\vartheta} \cdot s$ for every production of form *(C1)* and $\vartheta \in \widehat{\Theta}_p$, where $\mathrm{lin}(\vartheta \cdot u) = (s, \vartheta')$;

(ii) $A^{\mathrm{Id}_{p+1}} \to B^{\vartheta_1} \cdot \vartheta_1 \multimap C^{\vartheta_2} \cdot \vartheta_2$ for every production of form *(C2)*, and ϑ_1, $\vartheta_2 \in \widehat{\Theta}_p$;

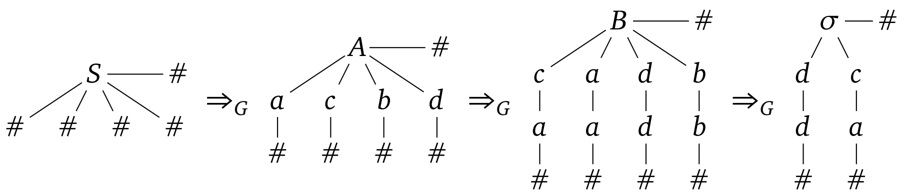
(iii) $A^{\mathrm{Id}_{p+1}} \to \sigma(x_i, x_j, x_{p+1})$ for every production of form *(C4)*;

(iv) $S' \to S^{\vartheta}$ for every $\vartheta \in \widehat{\Theta}_p$.

Observe that the productions generated in *(i)* are indeed of the form *(D1)*, because for every $\vartheta \in \widehat{\Theta}_p$ and $u \in \mathrm{C}(\Gamma)_p$, we have that $\mathrm{lin}(\vartheta \cdot u) = (s, \vartheta')$ for some $s \in \widetilde{\mathrm{C}}(\Gamma)_p$ and $\vartheta' \in \widehat{\Theta}_p$.

* * *

Before we prove the lemma, let us examine an example of the construction. Consider the hypothetic derivation[3]



that uses the productions

$$S \to A(ax_1, cx_2, bx_3, dx_4, x_5), \quad A \to B(cx_1, ax_1, dx_4, bx_3, x_5), \quad \text{and} \quad B \to \sigma(x_3, x_1, x_5).$$

We want to anticipate the torsion $\langle 5; x_1, x_1, x_4, x_3, x_5 \rangle$ in the right-hand side of the production for A. We do so by applying this torsion to the parameters in the right-hand side of the production for S. In concrete terms, we compute the tentative right-hand side

$$A \cdot \langle 5; x_1, x_1, x_4, x_3, x_5 \rangle \cdot \langle 5; ax_1, cx_2, bx_3, dx_4, x_5 \rangle = A(ax_1, ax_1, dx_4, bx_3, x_5).$$

[3] Clearly, this derivation does not lead to an element of L. However, a derivation of such an element would be quite a bit longer, but not more illuminating.

Since this right-hand side again has a non-unit torsion, its torsion must again be prepared earlier, by one of the special productions for the nonterminal S'. Altogether, we construct the productions

$$S' \to S^{\vartheta}\,, \qquad\qquad S^{\vartheta} \to A^{\vartheta}(ax_1, ax_2, dx_3, bx_4, x_5)\,,$$
$$A^{\vartheta} \to B^{\mathrm{Id}_5}(cx_1, ax_2, dx_3, bx_4, x_5)\,, \quad \text{and} \quad B^{\mathrm{Id}_5} \to \sigma(x_3, x_1, x_5)\,,$$

where $\vartheta = \langle 5; x_1, x_1, x_4, x_3, x_5\rangle$. Then the corresponding derivation in G' is of the form

$$\begin{array}{c} S'\text{---}\# \\ \diagup\ /\ \backslash\ \diagdown \\ \#\ \ \#\ \ \#\ \ \# \end{array} \Rightarrow_{G'} \begin{array}{c} S^{\vartheta}\text{---}\# \\ \diagup\ /\ \backslash\ \diagdown \\ \#\ \ \#\ \ \#\ \ \# \end{array} \Rightarrow_{G'} \begin{array}{cccc} \multicolumn{4}{c}{A^{\vartheta}\text{---}\#} \\ \multicolumn{4}{c}{\diagup\ /\ \backslash\ \diagdown} \\ a & a & d & b \\ | & | & | & | \\ \# & \# & \# & \# \end{array} \Rightarrow_{G'} \begin{array}{cccc} \multicolumn{4}{c}{B^{\mathrm{Id}_5}\text{---}\#} \\ \multicolumn{4}{c}{\diagup\ /\ \backslash\ \diagdown} \\ c & a & d & b \\ | & | & | & | \\ a & a & d & b \\ | & | & | & | \\ \# & \# & \# & \# \end{array} \Rightarrow_{G'} \begin{array}{cc} \multicolumn{2}{c}{\sigma\text{---}\#} \\ / & \backslash \\ d & c \\ | & | \\ d & a \\ | & | \\ \# & \# \end{array}\,.$$

* * *

After this short example, we now follow through with the announced proof of correctness. We demonstrate that for every $n \in \mathbb{N}$, $A \in N$, $v \in \mathrm{C}(\Gamma)_p$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$,

$$A \cdot v \Rightarrow^n_G t \qquad \text{if and only if} \qquad \exists \vartheta \in \widehat{\Theta}_p\colon\ A^{\vartheta} \cdot \vartheta \cdot v \Rightarrow^n_{G'} t\,.$$

The proof is by complete induction on n. The induction base holds trivially, hence we proceed by a case analysis of derivations with nonzero length. Assume therefore that $n \in \mathbb{N}$, $A \in N$, $v \in \mathrm{C}(\Gamma)_p$, and $t \in \mathrm{T}(\Sigma)^1_{p+1}$.

(I) By construction,

$$A \cdot v \Rightarrow_G \sigma(\pi_i \cdot v, \pi_j \cdot v, x_{p+1}) \qquad \text{if and only if} \qquad A^{\mathrm{Id}_{p+1}} \cdot v \Rightarrow_{G'} \sigma(\pi_i \cdot v, \pi_j \cdot v, x_{p+1})\,.$$

(II) Assume that $A \cdot v \Rightarrow_G B \cdot v \mathrel{\circ\!\!-} C \cdot v \Rightarrow^n_G t$. Then there are n_1, $n_2 \in \mathbb{N}$, and t_1, $t_2 \in \mathrm{T}(\Sigma)^{p+1}_{p+1}$ such that

$$B \cdot v \Rightarrow^{n_1}_G t_1\,, \quad C \cdot v \Rightarrow^{n_2}_G t_2\,, \quad t = t_1 \mathrel{\circ\!\!-} t_2\,, \quad \text{and} \quad n = n_1 + n_2\,.$$

By the induction hypothesis, there are ϑ_1, $\vartheta_2 \in \widehat{\Theta}_p$ such that $B^{\vartheta_1} \cdot \vartheta_1 \cdot v \Rightarrow^{n_1}_{G'} t_1$ and $C^{\vartheta_2} \cdot \vartheta_2 \cdot v \Rightarrow^{n_2}_{G'} t_2$. Thus,

$$A^{\mathrm{Id}_{p+1}} \cdot v \Rightarrow_{G'} B^{\vartheta_1} \cdot \vartheta_1 \cdot v \mathrel{\circ\!\!-} C^{\vartheta_2} \cdot \vartheta_2 \cdot v \Rightarrow^{n_1+n_2}_{G'} t_1 \mathrel{\circ\!\!-} t_2 = t\,.$$

Conversely, let ϑ_1, ϑ_2, $\vartheta_3 \in \widehat{\Theta}_p$, and $n \in \mathbb{N}$ such that

$$A^{\vartheta_1} \cdot \vartheta_1 \cdot v \Rightarrow_{G'} B^{\vartheta_2} \cdot \vartheta_2 \cdot \vartheta_1 \cdot v \mathrel{\circ\!\!-} C^{\vartheta_3} \cdot \vartheta_3 \cdot \vartheta_1 \cdot v \Rightarrow^n_{G'} t\,.$$

By construction, $\vartheta_1 = \mathrm{Id}_{p+1}$. Moreover, there are n_1, $n_2 \in \mathbb{N}$, t_1, and $t_2 \in \mathrm{T}(\Sigma)^{p+1}_{p+1}$ such that

$$B^{\vartheta_2} \cdot \vartheta_2 \cdot v \Rightarrow^{n_1}_{G'} t_1\,, \quad C^{\vartheta_3} \cdot \vartheta_3 \cdot v \Rightarrow^{n_2}_{G'} t_2\,, \quad t = t_1 \mathrel{\circ\!\!-} t_2\,, \quad \text{and} \quad n = n_1 + n_2\,.$$

By the induction hypothesis, $B \cdot v \Rightarrow^{n_1}_G t_1$ and $C \cdot v \Rightarrow^{n_2}_G t_2$, thus

$$A \cdot v \Rightarrow_G B \cdot v \mathrel{\circ\!\!-} C \cdot v \Rightarrow^{n_1+n_2}_G t_1 \mathrel{\circ\!\!-} t_2 = t\,.$$

(III) Assume finally that $A \cdot v \Rightarrow_G B \cdot u \cdot v \Rightarrow_G^n t$ for some $n \in \mathbb{N}$. By the induction hypothesis, there is some $\vartheta \in \widehat{\Theta}_p$ such that $B^\vartheta \cdot \vartheta \cdot u \cdot v \Rightarrow_{G'}^n t$. By construction, there is a production $A^{\vartheta'} \to B^\vartheta \cdot s$, where $s \in \widetilde{\mathrm{C}}(\Gamma)_p$ with $s \cdot \vartheta' = \vartheta \cdot u$. Thus we have

$$A^{\vartheta'} \cdot \vartheta' \cdot v \Rightarrow_{G'} B^\vartheta \cdot s \cdot \vartheta' \cdot v = B^\vartheta \cdot \vartheta \cdot u \cdot v \Rightarrow_{G'}^n t\,.$$

For the other direction, let $A^{\vartheta'} \cdot \vartheta' \cdot v \Rightarrow_{G'} B^\vartheta \cdot s \cdot \vartheta' \cdot v \Rightarrow_{G'}^n t$ for some $n \in \mathbb{N}$, ϑ and $\vartheta' \in \widehat{\Theta}_p$. By construction, there is some production $A \to B \cdot u$ in P, such that $s \cdot \vartheta' = \vartheta \cdot u$. Hence, $B^\vartheta \cdot s \cdot \vartheta' \cdot v = B^\vartheta \cdot \vartheta \cdot u \cdot v \Rightarrow_{G'}^n t$. By the induction hypothesis, $B \cdot u \cdot v \Rightarrow_G^n t$. Thus also $A \cdot v \Rightarrow_G B \cdot u \cdot v \Rightarrow_G^n t$.

* * *

Let $t \in \mathrm{T}(\Sigma)_0^1$. Then $t \in \mathcal{L}(G)$ if and only if $t \in \mathcal{L}(G, S \cdot [\#, \dots, \#])$. By the above property, this holds precisely if there is some $\vartheta \in \widehat{\Theta}_p$ such that $t \in \mathcal{L}(G', S^\vartheta \cdot \vartheta \cdot [\#, \dots, \#])$, and it is easy to see that $S^\vartheta \cdot \vartheta \cdot [\#, \dots, \#] = S^\vartheta \cdot [\#, \dots, \#]$. By construction of G', we obtain that

$$t \in \mathcal{L}(G) \quad \text{iff} \quad t \in \mathcal{L}(G, S \cdot [\#, \dots, \#]) \quad \text{iff} \quad t \in \mathcal{L}(G', S' \cdot [\#, \dots, \#]) \quad \text{iff} \quad t \in \mathcal{L}(G')\,,$$

and therefore $\mathcal{L}(G) = \mathcal{L}(G')$. □

By way of contradiction, assume that G is a cftg of the form stated in Lemma 4.14 such that $\mathcal{L}(G) = L$. Furthermore, let $\chi = [\#, \dots, \#]$. Note that then $\xi_0 = S \cdot \chi$.

4.1.4 Derivation Trees

A derivation of a tree $t \in \mathcal{L}(G)$ can be described faithfully by a full binary tree κ.[4] These *derivation trees* will help us analyze the structure of the derivations in G. Intuitively, each node of a derivation tree κ contains a subderivation of the form $A \Rightarrow^* B \cdot s$, for some A, $B \in N$ and $s \in \widetilde{\mathrm{C}}(\Gamma)_p$, while branching productions of the form $A \to B \cdot \vartheta_1 \multimap C \cdot \vartheta_2$ are associated with the forks of κ. Further, every leaf of κ corresponds to some terminal production $A \to \sigma(x_i, x_j, x_{p+1})$ of G.

Formally, let κ be a full binary tree such that every position $\delta \in \mathrm{pos}(\kappa)$ is equipped with two nonterminal symbols A_δ and $B_\delta \in N$, a torsion-free tuple $s_\delta \in \widetilde{\mathrm{C}}(\Gamma)_p$, and a torsion $\vartheta_\delta \in \widehat{\Theta}_p$. If δ is a leaf position, it is moreover equipped with two numbers i_δ and $j_\delta \in [p]$. Then κ is an $(A_\varepsilon, \vartheta_\varepsilon)$-*derivation tree* if for every $\delta \in \mathrm{pos}(\kappa)$,

(i) $A_\delta \Rightarrow_G^* B_\delta \cdot s_\delta$,

(ii) if δ is a leaf of κ, then the production $B_\delta \to \sigma(x_{i_\delta}, x_{j_\delta}, x_{p+1})$ is in P,

(iii) if δ is not a leaf, then $B_\delta \to A_{\delta 1} \cdot \vartheta_{\delta 1} \multimap A_{\delta 2} \cdot \vartheta_{\delta 2}$ is a production in P.

Let $t \in \mathrm{T}(\Sigma)_{p+1}^1$. We say that κ is an $(A_\varepsilon, \vartheta_\varepsilon)$-*derivation tree of t* (or: *κ derives t*) if

[4] Since we do not care about its labels, κ will be presented solely by its finite set of positions $\mathrm{pos}(\kappa) \subseteq \{1,2\}^*$. A binary tree κ is said to be full if for every $w \in \mathbb{N}_1^*$, we have $w1 \in \mathrm{pos}(\kappa)$ if and only if $w2 \in \mathrm{pos}(\kappa)$.

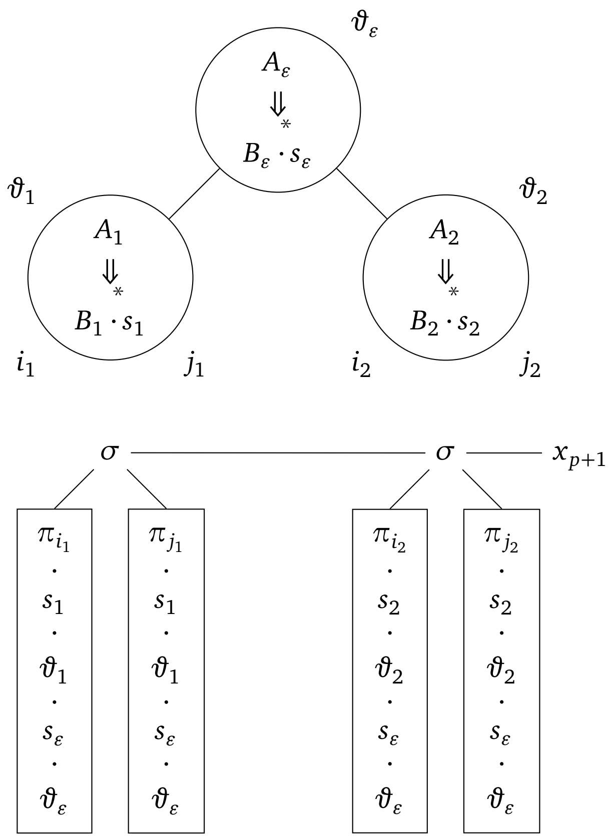


Figure 4.5: An example derivation tree and its derived tree

(i) either κ has only one node and $t = \sigma(x_{i_\varepsilon}, x_{j_\varepsilon}, x_{p+1}) \cdot s_\varepsilon \cdot \vartheta_\varepsilon$, or, otherwise,

(ii) there are t_1, $t_2 \in \mathrm{T}(\Sigma)^1_{p+1}$ such that $\kappa|_1$ is an (A_1, ϑ_1)-derivation tree of t_1, $\kappa|_2$ is an (A_2, ϑ_2)-derivation tree of t_2,[5] and such that $t = (t_1 \multimapinv t_2) \cdot s_\varepsilon \cdot \vartheta_\varepsilon$.

An (S, Id_{p+1})-derivation tree (of t) will simply be called a *derivation tree (of t)*. There is the following relation between derivations and derivation trees.

Lemma 4.15. *Let $t \in \mathrm{T}(\Sigma)^1_{p+1}$, let $A \in N$, and $\vartheta \in \widehat{\Theta}_p$. Then $A \cdot \vartheta \Rightarrow^*_G t$ if and only if there is an (A, ϑ)-derivation tree of t.*

Proof. The proof is by complete induction on $|t|_\sigma$. The case $|t|_\sigma = 0$ is vacuously true, as neither can such a tree t be the product of a derivation, nor of a derivation tree.

Assume that $|t|_\sigma = 1$. For the direction "only if", let $A \cdot \vartheta \Rightarrow^*_G t$. By our assumption on the shape of G, this derivation is of the form

$$A \cdot \vartheta \Rightarrow^*_G B \cdot s \cdot \vartheta \Rightarrow_G \sigma(x_i, x_j, x_{p+1}) \cdot s \cdot \vartheta = t\,,$$

[5]Of course, we demand here that each $\delta \in \mathrm{pos}(\kappa|_1)$ is equipped with the same nonterminals, torsions, and numbers as the position $1\delta \in \mathrm{pos}(\kappa)$, and analogously for $\kappa|_2$.

for some $B \in N$, $s \in \widetilde{\mathrm{C}}(\Gamma)_p$, and $i,\ j \in [p]$. Define the derivation tree $\kappa = \{\varepsilon\}$ such that $A_\varepsilon = A$, $B_\varepsilon = B$, $s_\varepsilon = s$, $i_\varepsilon = i$, and $j_\varepsilon = j$. By definition, κ is an (A, ϑ)-derivation tree of t.

For the other direction "if", assume an (A, ϑ)-derivation tree of t. Clearly, $|t|_\sigma = 1$ implies that $\mathrm{pos}(\kappa) = \{\varepsilon\}$. But then κ determines the derivation

$$A \cdot \vartheta = A_\varepsilon \cdot \vartheta_\varepsilon \Rightarrow_G^* B_\varepsilon \cdot s_\varepsilon \cdot \vartheta_\varepsilon \Rightarrow_G \sigma(x_{i_\varepsilon}, x_{j_\varepsilon}, x_{p+1}) \cdot s_\varepsilon \cdot \vartheta_\varepsilon = t\,.$$

For the induction step, let $|t|_\sigma > 1$. Assume for the direction "only if" that $A \cdot \vartheta \Rightarrow_G^* t$. This derivation is of form

$$A \cdot \vartheta \Rightarrow_G^* B \cdot s \cdot \vartheta \Rightarrow_G C_1 \cdot \widehat{\vartheta}_1 \cdot s \cdot \vartheta \multimap C_2 \cdot \widehat{\vartheta}_2 \cdot s \cdot \vartheta \Rightarrow_G^* t_1 \cdot s \cdot \vartheta \multimap t_2 \cdot s \cdot \vartheta = t\,,$$

for some B, C_1, $C_2 \in N$, $s \in \widetilde{\mathrm{C}}(\Gamma)_p$, $\widehat{\vartheta}_1$, $\widehat{\vartheta}_2 \in \widehat{\Theta}_p$, and t_1, $t_2 \in \mathrm{T}(\Sigma)^1_{p+1}$. By virtue of the induction hypothesis, there is a $(C_\ell, \widehat{\vartheta}_\ell)$-derivation tree κ_ℓ of t_ℓ for every $\ell \in [2]$. Let us denote its associated nonterminals, torsions, and numbers by $A^{(\ell)}_\delta$, $B^{(\ell)}_\delta$, $\vartheta^{(\ell)}_\delta$, $i^{(\ell)}_\delta$, and $j^{(\ell)}_\delta$, respectively, for each position $\delta \in \mathrm{pos}(\kappa_\ell)$. Then we construct the derivation tree κ, such that

$$\mathrm{pos}(\kappa) = \{\varepsilon\} \cup 1 \cdot \mathrm{pos}(\kappa_1) \cup 2 \cdot \mathrm{pos}(\kappa_2)$$

with $A_\varepsilon = A$, $B_\varepsilon = B$, $\vartheta_\varepsilon = \vartheta$, and for every $\ell \in [2]$ and $\delta \in \mathrm{pos}(\kappa_\ell)$, we have $A_{\ell\delta} = A^{(\ell)}_\delta$, $B_{\ell\delta} = B^{(\ell)}_\delta$, $\vartheta_{\ell\delta} = \vartheta^{(\ell)}_\delta$, and, if defined, $i_{\ell\delta} = i^{(\ell)}_\delta$ and $j_{\ell\delta} = j^{(\ell)}_\delta$. Going through the relevant definitions, it is easy to check that κ is an (A, ϑ)-derivation tree of t.

For the direction "if", let us point to the case $|t|_\sigma = 1$. In the same manner as there, a derivation tree κ of t determines a corresponding derivation of t. $\square$

Corollary 4.16. *For every $t \in \mathrm{T}(\Sigma)^1_{p+1}$, $t \cdot \chi \in L$ if and only if there is a derivation tree of t.*

Example 4.17. For an example of a derivation tree with three nodes ε, 1, and 2, and its derived tree, compare Figure 4.5. This derivation tree corresponds to the derivation

$$\begin{aligned}
A_\varepsilon \cdot \vartheta_\varepsilon &\Rightarrow_G^* B_\varepsilon \cdot s_\varepsilon \cdot \vartheta_\varepsilon \\
&\Rightarrow_G (A_1 \cdot \vartheta_1 \multimap A_2 \cdot \vartheta_2) \cdot s_\varepsilon \cdot \vartheta_\varepsilon \\
&\Rightarrow_G^* (B_1 \cdot s_1 \cdot \vartheta_1 \multimap B_2 \cdot s_2 \cdot \vartheta_2) \cdot s_\varepsilon \cdot \vartheta_\varepsilon \\
&\Rightarrow_G^* \sigma\big(\pi_{i_1} \cdot s_1 \cdot \vartheta_1 \cdot s_\varepsilon \cdot \vartheta_\varepsilon,\ \pi_{j_1} \cdot s_1 \cdot \vartheta_1 \cdot s_\varepsilon \cdot \vartheta_\varepsilon, x_{p+1}\big) \\
&\qquad \multimap \sigma\big(\pi_{i_2} \cdot s_2 \cdot \vartheta_2 \cdot s_\varepsilon \cdot \vartheta_\varepsilon,\ \pi_{j_2} \cdot s_2 \cdot \vartheta_2 \cdot s_\varepsilon \cdot \vartheta_\varepsilon, x_{p+1}\big)\,.
\end{aligned}$$ $\lhd$

In the derivation tree of an ε-free context-free word grammar in normal form, there is a one-to-one correspondence between the tree's leaf nodes and the terminal symbols occurring in the derived word. There is a similar correspondence for our notion of derivation trees. Consider a tree $t \in L$ of the form

$$t = \sigma(v_1\#, u_1\#, x) \multimap \cdots \multimap \sigma(v_n\#, u_n\#, x) \multimap \#$$

for some $n \in \mathbb{N}$, and v_1, u_1, …, v_n, $u_n \in \Gamma^*$. Further, consider a derivation tree κ of t. It is easy to see that κ has n leaf nodes. We will say for every $i \in [n]$ that the i-th leaf node of κ

(enumerated from left to right) *contributes the tree* $\sigma(u_i\#, v_i\#, x)$ *to* t. Figure 4.5 illustrates this notion.

We close our discussion of derivation trees with the following pumping lemma. It states that if there is some s_δ in κ which has a sufficiently large component, then an iterable pair of nonterminals occurs in the derivation of s_δ.

> *In the sequel, fix the pumping number* $H = |N| \cdot h_{\max}$*, where* $h_{\max}$ *is the maximal size of a component of* u *in a production of* G *of form (D1).*

Lemma 4.18. *Let* κ *be a derivation tree and* $\delta \in \mathrm{pos}(\kappa)$*. If there are* $i \in [p]$ *and* $w, w' \in \Gamma^*$ *such that* $\pi_i \cdot s_\delta = w'wx_i$ *and* $|w| > H$*, then there exist* $v, y, z \in \widetilde{\mathrm{C}}(\Gamma)_p$ *such that*

(i) $s_\delta = v \cdot y \cdot z$*,*

(ii) $\pi_i \cdot y \cdot z$ *is a suffix of* wx_i*,*

(iii) $|\pi_i \cdot y| > 0$*, and*

(iv) for each $j \in \mathbb{N}$*,* $A_\delta \Rightarrow_G^* B_\delta \cdot v \cdot y^j \cdot z$*.*

Proof. By definition of κ, $A_\delta \Rightarrow_G^* B_\delta \cdot s_\delta$. So there are

$$n \in \mathbb{N}\,, \qquad C_1, \ldots, C_n \in N\,, \qquad \text{and} \qquad e^{(1)}, \ldots, e^{(n)} \in \widetilde{\mathrm{C}}(\Gamma)_p$$

such that

$$C_1 \cdot e^{(1)} \Rightarrow_G C_2 \cdot e^{(2)} \cdot e^{(1)} \Rightarrow_G \cdots \Rightarrow_G C_n \cdot e^{(n)} \cdots e^{(1)}$$

where $C_1 = A_\delta$, $C_n = B_\delta$, $e^{(1)} = \mathrm{Id}_{p+1}$, and $e^{(n)} \cdots e^{(1)} = s_\delta$.

Assume that $\pi_i \cdot s_\delta = w'wx_i$ for some $i \in [p]$ and $w, w' \in \Gamma^*$ with $|w| > H$. If $C_1, \ldots, C_n$ are pairwise distinct, then $n \le |N|$ and the size of every component of s_δ is at most $|N| \cdot h_{\max} = H$, which contradicts the assumption that $|w| > H$. We can therefore choose two indices ℓ, $k \in [n]$ with $\ell < k$ such that $C_\ell = C_k$, the size of $\pi_i \cdot e^{(k)} \cdots e^{(\ell+1)}$ is nonzero, and ℓ and k are the two smallest numbers with these properties. Such indices ℓ and k do indeed exist, because if for every such pair the size of $\pi_i \cdot e^{(k)} \cdots e^{(\ell+1)}$ was zero, then the size of $\pi_i \cdot s_\delta$ would be bounded by H, which contradicts our assumption for w. Let

$$v = e^{(n)} \cdots e^{(k+1)}\,, \quad y = e^{(k)} \cdots e^{(\ell+1)}\,, \quad \text{and} \quad z = e^{(\ell)} \cdots e^{(1)}\,.$$

Then for every $j \in \mathbb{N}$,

$$A_\delta \cdot \mathrm{Id}_{p+1} \Rightarrow_G^* C_\ell \cdot z \Rightarrow_G^* C_k \cdot y^j \cdot z \Rightarrow_G^* B_\delta \cdot v \cdot y^j \cdot z\,.$$

Moreover, the size of $\pi_i \cdot y \cdot z$ is at most H, therefore $\pi_i \cdot y \cdot z$ is a suffix of wx_i. □

4.1.5 Dyck Words and Sequences of Chains

This section prepares some necessary notions for the upcoming counterexample. We introduce an infinite sequence $U_1, U_2, \ldots$ of Dyck words. Later, an element of this sequence will contribute the chains to the tree t used in the counterexample. As described in the introduction, the proof revolves around the factorization of t into trees t_1 and t_2 that is induced by a derivation of t. So we will analyze the corresponding factorizations of the Dyck words U_i.

Moreover, we will introduce here the notion of *defects*, which can be understood as the "unclosed parentheses" in t_1, resp. t_2. Finally, a lemma on *perturbations* is given, which will be used to show that if the defects in t_1 are modified (or: perturbed), then the word formed by the chains of the resulting tree lies in another Dyck congruence class. This implies that the resulting tree does not "fit together" with t_2 any longer.

First of all, let us fix the following constants. Let $q = 2p$, and let $m = 2^{q-1} + 1$. For every $i \in \mathbb{N}$, let $\alpha_i = ca^{imH}c$ and $\beta_i = db^{imH}d$. Note that $\alpha_i^R = \alpha_i$, $\beta_i^R = \beta_i$, and $\alpha_i\beta_i \in D_\Gamma^*$. Define the sequence $U_1, U_2, \ldots$ of words over Γ by

$$U_1 = \alpha_1\beta_1 \qquad \text{and} \qquad U_{i+1} = \alpha_{i+1}U_iU_i\beta_{i+1} \quad \text{for every } i \geq 1\,.$$

We begin with the following lemma on the form of the sequence's elements.

Lemma 4.19. *For every $i \in \mathbb{N}_1$,*

(i) $U_i \in D_\Gamma^*$*, and*

(ii) $U_i = u_1v_1 \cdots u_nv_n$*, where* $n = 2^{i-1}$, $u_j \in (ca^+c)^+$*, and* $v_j \in (db^+d)^+$*, for each* $j \in [n]$*. More precisely, for each* $j \in [n]$*, there are* ℓ, $\ell' \in \mathbb{N}_1$ *such that* u_j *is of the form* $\alpha_\ell \cdots \alpha_1$*, and* v_j *is of the form* $\beta_1 \cdots \beta_{\ell'}$*.*

Proof. We prove both facts by induction on i. For the induction base $i = 1$, clearly both statements hold. So assume that $i > 1$. By the induction hypothesis, we have $U_i \in D_\Gamma^*$, and therefore $\alpha_{i+1}U_iU_i\beta_{i+1} \equiv \alpha_{i+1}\beta_{i+1} \equiv \varepsilon$, so $U_{i+1} \in D_\Gamma^*$. Hence it remains to show item *(ii)*. Since $U_i = u_1v_1 \cdots u_nv_n$ for the number n and words $u_1, v_1, \ldots, u_n, v_n$ as stated above, we have

$$U_{i+1} = \underbrace{\alpha_{i+1}u_1}_{u'_1}\ \underbrace{v_1}_{v'_1}\ \cdots\ \underbrace{u_n}_{u'_n}\ \underbrace{v_n}_{v'_n}\ \underbrace{u_1}_{u'_{n+1}}\ \underbrace{v_1}_{v'_{n+1}}\ \cdots\ \underbrace{u_n}_{u'_{n'}}\ \underbrace{v_n\beta_{n+1}}_{v'_{n'}}$$

with $n' = 2 \cdot n = 2^i$. Since u_1 begins with α_i, and v_n ends with β_i, we obtain that for each $j \in [n']$, u'_j is of the form $\alpha_\ell \cdots \alpha_1$, and v'_j is of the form $\beta_1 \cdots \beta_{\ell'}$ for some $\ell, \ell' \in \mathbb{N}_1$. □

For each U_i of the form given in Lemma 4.19*(ii)*, let

$$Z_i = \langle u_1^R, v_1, \ldots, u_n^R, v_n \rangle\,.$$

The components u_ℓ^R and v_ℓ of Z_i will also be called *chains*, as later on they will end up as the chains of some tree $t \in L$. For every factorization of Z_i into

$$Z'_i = \langle u_1^R, v_1, u_2^R, v_2, \ldots, u_j^R \rangle \quad \text{and} \quad Z''_i = \langle v_j, u_{j+1}^R, v_{j+1}, \ldots, u_n^R, v_n \rangle\,, \quad j \in [n]\,,$$

consider the respective factors $P_{i,j} = u_1v_1u_2v_2 \cdots u_j$ and $S_{i,j} = v_ju_{j+1}v_{j+1} \cdots u_nv_n$ of U_i.

Lemma 4.20. *The factors $P_{i,j}$ and $S_{i,j}$ can be written as*

$$P_{i,j} = \alpha_iV_{i-1}\alpha_{i-1} \cdots V_1\alpha_1 \qquad \text{and} \qquad S_{i,j} = \beta_1W_1 \cdots \beta_{i-1}W_{i-1}\beta_i\,, \tag{4.2}$$

such that V_ℓ, $W_\ell \in \{\varepsilon, U_\ell\}$ and $V_\ell \neq W_\ell$ for every $\ell \in [i-1]$.

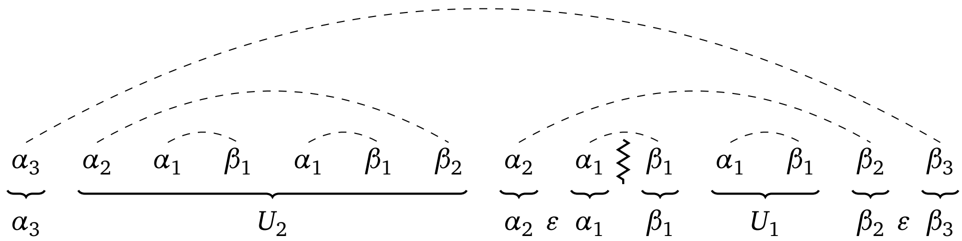


Figure 4.6: A factorization of U_3

Proof. Compare Figure 4.6 for intuition, which depicts a factorization of the word U_3 and the corresponding factors, written as in the lemma's statement.

The proof of the lemma is by induction on i. The base case $U_1 = \alpha_1\beta_1$ has only one factorization, $P_{1,1} = \alpha_1$ and $S_{1,1} = \beta_1$, which fulfills the property. Let $i \geq 1$ and consider $U_{i+1} = \alpha_{i+1}U_iU_i\beta_{i+1}$. A factorization $P_{i+1,j}S_{i+1,j}$ of U_{i+1} induces a factorization of either the first or the second occurrence of U_i into, say $P_{i,j'}$ and $S_{i,j'}$ for some $j' \in [2^{i-1}]$. Therefore,

$$U_{i+1} = \alpha_{i+1}V_iP_{i,j'}S_{i,j'}W_i\beta_{i+1}$$

for V_i, $W_i \in \{\varepsilon, U_i\}$ with $V_i \neq W_i$. By the induction hypothesis, $P_{i,j'} = \alpha_iV_{i-1}\alpha_{i-1}\cdots V_1\alpha_1$, and thus

$$P_{i+1,j} = \alpha_{i+1}V_i\alpha_iV_{i-1}\alpha_{i-1}\cdots V_1\alpha_1\,,$$

for $V_i, \ldots, V_1$ as given above. The same kind of argument works for $S_{i+1,j}$. □

Consider a factorization of U_i into $P_{i,j}$ and $S_{i,j}$ as given in (4.2). Then we define the word

$$D_{i,j} = \$\alpha_iV'_{i-1}\alpha_{i-1}\cdots V'_1\alpha_1\$\beta_1W'_1\cdots\beta_{i-1}W'_{i-1}\beta_i\$$$

over $\Gamma \cup \{\$\}$, where for every $\ell \in [i-1]$,

$$V'_\ell = \begin{cases} \$ & \text{if } V_\ell = U_\ell \\ \varepsilon & \text{if } V_\ell = \varepsilon, \end{cases} \qquad \text{and analogously} \qquad W'_\ell = \begin{cases} \$ & \text{if } W_\ell = U_\ell \\ \varepsilon & \text{if } W_\ell = \varepsilon. \end{cases}$$

Let ℓ, $k \in \mathbb{N}$ with $\ell \leq k$. We say that a word $\gamma = \alpha_\ell\cdots\alpha_k$ (resp. $\gamma = \beta_\ell\cdots\beta_k$) is an *a-defect* (resp. a *b-defect*) *in* $D_{i,j}$ if $\$\gamma^R\$$ (resp. $\$\gamma\$$) occurs in $D_{i,j}$. When the factorization is clear, the reference to $D_{i,j}$ is omitted. Both a-defects and b-defects will be called *defects*. A chain in Z_i whose suffix is a defect is called a *critical chain*.

Lemma 4.21. *Consider a factorization of U_i into $P_{i,j}$ and $S_{i,j}$.*

1. *There is no $\ell \in [i]$ such that α_ℓ (or β_ℓ) occurs in two distinct defects.*
2. *The number of defects in $D_{i,j}$ is $i+1$.*
3. *Each a-defect (resp. b-defect) is the suffix of some chain u_h (resp. v_h) in Z_i, with $h \in [2^{i-1}]$.*

Proof. For *(1)*, observe that the a-defects in $D_{i,j}$ are disjoint (non-overlapping) factors of the word $\alpha_1 \cdots \alpha_i$. A similar observation can be made for the b-defects in $D_{i,j}$. For *(2)*, it is easy to see from Lemma 4.20 that there are exactly $i+2$ occurrences of the symbol \$ in $D_{i,j}$. So there are $i+1$ factors of the form $\$\gamma\$$ in $D_{i,j}$, for $\gamma \in \Gamma^*$. By *(1)*, the defects are pairwise distinct, so $D_{i,j}$ contains precisely $i+1$ defects.

Regarding *(3)*, let $\gamma = \alpha_\ell \cdots \alpha_k$, $\ell \le k$, be an a-defect in $D_{i,j}$ and let

$$D_{i,j} = D'\$\underbrace{\alpha_k \cdots \alpha_\ell}_{\gamma^R}\$D'' \qquad \text{for some } D', D'' \in (\Gamma \cup \{\$\})^* .$$

By definition of $D_{i,j}$, $P_{i,j}$ is of the form

$$P_{i,j} = P'U_k\alpha_k \cdots \alpha_\ell P'' \qquad \text{for some } P', P'' \in \Gamma^*$$

if $k < i$, and $P_{i,j} = \alpha_k \cdots \alpha_\ell P''$ if $k = i$. As U_k ends with β_k, γ is the suffix of some chain u_h in Z_i. A similar argument can be made if γ is a b-defect. □

Let P, $P' \in \Gamma^*$. We say that P' is a *perturbation* of P if it results from P by modifying the exponents of a and b in P. More precisely, let P be of the form

$$P = w_0 v_1^{f_1} w_1 \cdots w_{\ell-1} v_\ell^{f_\ell} w_\ell ,$$

such that $\ell \in \mathbb{N}$, $w_0, \ldots, w_\ell \in \{c, d\}^*$, $v_1, \ldots, v_\ell \in \{a, b\}$, and for each $i \in [\ell]$, $f_i \in \mathbb{N}_1$. Then $P' \in \Gamma^*$ is called a *perturbation* of P if

$$P' = w_0 v_1^{f_1'} w_1 \cdots w_{\ell-1} v_\ell^{f_\ell'} w_\ell ,$$

for some $f_1', \ldots, f_\ell' \in \mathbb{N}$. The only perturbation of ε is ε itself.

Lemma 4.22. *Consider a factorization of U_i into $P_{i,j}$ and $S_{i,j}$, and let $P_{i,j}'$ be a perturbation of $P_{i,j}$, i.e.,*

$$P_{i,j} = \alpha_i V_{i-1} \alpha_{i-1} \cdots V_1 \alpha_1 \qquad \text{and} \qquad P_{i,j}' = \alpha_i' V_{i-1}' \alpha_{i-1}' \cdots V_1' \alpha_1' . \tag{4.3}$$

Then $P_{i,j}' \equiv P_{i,j}$ if and only if $V_\ell' \equiv \varepsilon$ for every $\ell \in [i-1]$ and $\alpha_\ell' = \alpha_\ell$ for every $\ell \in [i]$.

Proof. The direction "if" is trivial. For the other direction, we first prove for every $i > 0$ and every perturbation U_i' of U_i that either $U_i' \equiv \varepsilon$ or the reduct of U_i' is cXd for some $X \not\equiv \varepsilon$. The proof is by induction on i. For the base case, consider a perturbation $U_1' = ca^p cdb^q d$ of U_1, where $p, q \in \mathbb{N}$. Since $U_1' \equiv ca^p b^q d$, $U_1' \not\equiv \varepsilon$ implies that $p \ne q$. Therefore the reduct of U_1' is cXd for some $X \in \{a\}^+ \cup \{b\}^+$, and thus $X \not\equiv \varepsilon$. Consider now a perturbation

$$U_{i+1}' = ca^p cU_i'U_i''db^q d , \qquad p, q \in \mathbb{N} ,$$

of U_{i+1}, where U_i' and U_i'' are perturbations of U_i, and assume that $U_{i+1}' \not\equiv \varepsilon$. If $U_i'U_i'' \equiv \varepsilon$, then again $p \ne q$, and we make the same argument as above. Otherwise, the reduct of $U_i'U_i''$ is of the form cXd with $X \not\equiv \varepsilon$, as at least one of the reducts of U_i' and U_i'' has this shape. But then clearly the reduct of U_{i+1}' is also of this shape.

We can now prove the direction "only if" of the lemma. Let $P'_{i,j} \equiv P_{i,j}$. As $V_\ell \in \{U_\ell, \varepsilon\}$ for every $\ell \in [i-1]$, $P_{i,j}$ reduces to $\alpha_i \cdots \alpha_1$. Assume that there is some $\ell \in [i-1]$ with $V'_\ell \not\equiv \varepsilon$. Then the reduct of $P'_{i,j}$ would contain an occurrence of d, by the property shown above. But this is in contradiction to the assumption that $P'_{i,j} \equiv P_{i,j}$. Hence, $V'_1 \equiv \cdots \equiv V'_{i-1} \equiv \varepsilon$. Then clearly also $\alpha'_\ell = \alpha_\ell$ for every $\ell \in [i]$. □

Let us remark that an analogous lemma can be formulated for perturbations of $S_{i,j}$. However, we will only consider perturbations of $P_{i,j}$ afterwards.

4.1.6 A Witness for $\mathcal{L}(G) \neq L$

In this section, we choose a tree $t \in L$ whose chains form a sufficiently large word U_i. By viewing a derivation tree κ of t, which induces a factorization $t = t_1 \circ\!\!- t_2$, we will see that the pumping lemma from Section 4.1.4 can be applied, and this leads to a perturbation in the defects of t_1. By Lemma 4.22 right above, we receive the desired contradiction.

Let $Z_q = \langle u_1, v_1, \ldots, u_{m-1}, v_{m-1} \rangle$, recalling from Lemma 4.19 *(ii)* that $m = 2^{q-1} + 1$. Moreover, let

$$t = \sigma\big(\#, u_1\#, x\big) \circ\!\!- \sigma\big(v_1\#, u_2\#, x\big) \circ\!\!- \cdots \circ\!\!- \sigma\big(v_{m-2}\#, u_{m-1}\#, x\big) \circ\!\!- \sigma\big(v_{m-1}\#, \#, \#\big).$$

Observe that t contains m occurrences of σ, and that $\iota(t) = U_q$. By Lemma 4.19 *(ii)*, the chains of t are of the form $\alpha_1 \cdots \alpha_\ell$, resp. $\beta_1 \cdots \beta_\ell$, for some $\ell \in [q]$.

Lemma 4.23. $t \in L$.

Proof. The lemma's proof is based on the following property. For every $i \in \mathbb{N}_1$, every u, $v \in \Gamma^*$, and every ζ_1, $\zeta_2 \in X_2 \cup \{\#\}$, there is a tree

$$s \in \mathcal{L}\big(G_{\text{ex}}, A(cu\zeta_1,\, dv\zeta_2,\, x_3)\big) \qquad \text{such that} \qquad \iota'(s) = u^R U_i v\,.$$

We show this property by induction on i. For the induction base, let $i = 1$ and consider the derivation

$$\begin{aligned} A(cu\zeta_1, dv\zeta_2, x_3) &\Rightarrow^*_{G_{\text{ex}}} A(a^{mH}cu\zeta_1, b^{mH}dv\zeta_2, x_3) \\ &\Rightarrow_{G_{\text{ex}}} \delta_2(ca^{mH}cu\zeta_1, x_3) \circ\!\!- \delta_1(db^{mH}dv\zeta_2, x_3)\,. \end{aligned}$$

We let $s = \delta_2(ca^{mH}cu\zeta_1, x_3) \circ\!\!- \delta_1(db^{mH}dv\zeta_2, x_3)$ and obtain

$$\iota'(s) = u^R \underbrace{ca^{mH}c}_{\alpha_1}\underbrace{db^{mH}d}_{\beta_1}v = u^R U_1 v\,.$$

For the induction step, assume the property holds for some $i \in \mathbb{N}_1$. We will prove it for $i+1$. Consider the derivation

$$\begin{aligned} A(cu\zeta_1, dv\zeta_2, x_3) &\Rightarrow^*_{G_{\text{ex}}} A(a^{(i+1)mH}cu\zeta_1, b^{(i+1)mH}dv\zeta_2, x_3) \\ &\Rightarrow_{G_{\text{ex}}} A(cca^{(i+1)mH}cu\zeta_1, d\#, x_3) \circ\!\!- A(c\#, ddb^{(i+1)mH}dv\zeta_2, x_3) \\ &\Rightarrow^*_{G_{\text{ex}}} s' \circ\!\!- s''\,, \end{aligned}$$

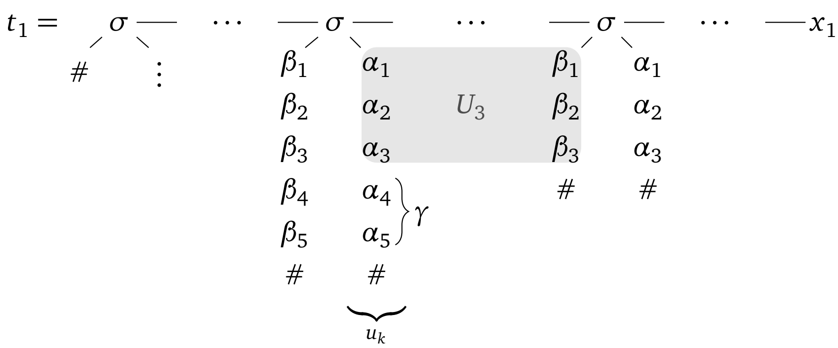


Figure 4.7: Occurrence of a defect γ in the critical chain u_k of t_1

where s' and $s'' \in \mathrm{T}(\Delta)_3^1$ are the trees guaranteed by the induction hypothesis. Let $s = s' \circ\!\!- s''$. Then

$$\iota'(s) = \iota'(s')\,\iota'(s'') = u^R \underbrace{c a^{(i+1)mH} c}_{\alpha_{i+1}} U_i U_i \underbrace{d b^{(i+1)mH} d}_{\beta_{i+1}} v = u^R U_{i+1} v\,,$$

which proves the property.

* * *

It remains to show that $t \in L$. The axiom of G_{ex} can be written

$$\xi_{\mathrm{ex}} = \delta_1(\#, x_1) \circ\!\!- A(cx_1, dx_2, x_3) \cdot [\#, \#, x_1] \circ\!\!- \delta_2(\#, \#)\,.$$

By the property above, there is a tree $s \in \mathcal{L}(G_{\mathrm{ex}}, A(cx_1, dx_2, x_3))$ such that $\iota'(s) = U_q$. Let

$$\hat{s} = \delta_1(\#, x_1) \circ\!\!- s \cdot [\#, \#, x_1] \circ\!\!- \delta_2(\#, \#)\,.$$

Then $\hat{s} \in \mathcal{L}(G_{\mathrm{ex}})$, and $\iota'(\hat{s}) = \iota'(s) = U_q = \iota(t)$. By Observation 4.3, $h(t) = \hat{s}$, hence $t \in L$. $\square$

By Lemma 4.15, there are a $\hat{t} \in \mathrm{T}(\Sigma)_{p+1}^1$ with $t = \hat{t} \cdot \chi$, and a derivation tree κ of $\hat{t}$. Moreover, as $m > 1$, there are t_1, $t_2 \in \mathrm{T}(\Sigma)_1^1$ such that

$$A_\varepsilon \cdot \chi \Rightarrow_G^* B_\varepsilon \cdot s_\varepsilon \cdot \chi \Rightarrow_G \left(A_1 \cdot \vartheta_1 \cdot s_\varepsilon \circ\!\!- A_2 \cdot \vartheta_2 \cdot s_\varepsilon\right) \cdot \chi \Rightarrow_G^* t_1 \circ\!\!- t_2 = t\,.$$

Since both t_1 and t_2 contain at least one occurrence of σ, there is a $j \in [m-1]$ such that

$$t_1 = \sigma(\#, u_1\#, x) \circ\!\!- \sigma(v_1\#, u_2\#, x) \circ\!\!- \cdots \circ\!\!- \sigma(v_{j-1}\#, u_j\#, x) \quad \text{and}$$
$$t_2 = \sigma(v_j\#, u_{j+1}\#, x) \circ\!\!- \cdots \circ\!\!- \sigma(v_{m-2}\#, u_{m-1}\#, x) \circ\!\!- \sigma(v_{m-1}\#, \#, \#)\,,$$

and this factorization of t induces an according factorization of Z_q into Z' and Z'' with

$$Z' = \langle u_1, v_1, \ldots, u_j \rangle \qquad \text{and} \qquad Z'' = \langle v_j, \ldots, u_{m-1}, v_{m-1} \rangle\,.$$

Example 4.24. Let us consider an example which relates the introduced concepts. Figure 4.7 displays the critical chain u_k in t_1, whose defect is $\gamma = \alpha_4\alpha_5$. In our intuition, γ is a sequence of opening parentheses which have no corresponding closing parenthesis in t_1. Therefore, t_2 must contain a suitable sequence of closing parentheses. Formally, γ^R occurs in $P_{q,j}$ as

$$P_{q,j} = P'U_5\alpha_5\alpha_4U_3P'' , \qquad \text{so} \qquad D_{q,j} = D'\$\gamma^R\$D'' ,$$

for some P', $P'' \in \Gamma^*$ and D', $D'' \in (\Gamma \cup \{\$\})^*$. Therefore, γ is indeed a defect by definition.

As u_k is critical, every a-chain $u_{k'}$ in t_1 to its right (i.e., with $k' > k$) is of the form $\alpha_1 \cdots \alpha_\ell$, for some $\ell \leq 3$. This can be seen from Lemma 4.20: to the right of the occurrence of α_4 in $P_{q,j}$, there may not occur any factors U_ℓ or α_ℓ with $\ell \geq 4$, and by definition U_ℓ has only factors $\alpha_{\ell'}$ with $\ell' \leq \ell$. ◁

By Lemma 4.21(2), the number of defects in $D_{q,j}$ is $q+1 = 2p+1$. Thus either t_1 contains at least $p+1$ critical chains, or t_2 does.

> *For the rest of Section 4.1, assume that t_1 contains at least $p+1$ critical chains. The proofs for the other case are obtained mainly by substituting b for a and β for α.*

Note that the height of the derivation tree κ of t is at most m. Therefore $|\delta| < m$ for every $\delta \in \mathrm{pos}(\kappa)$. If $\delta = i_1 \cdots i_d$, then we denote the prefix $i_1 \cdots i_{d-\ell}$ of δ by δ_ℓ, for every $\ell \in [0, d]$. In particular, $\delta_0 = \delta$ and $\delta_d = \varepsilon$.

Convention. *Let $s \in \mathrm{C}(\Gamma)_p$ and $w \in \Gamma^*$. If there is no possibility of confusion, we will briefly say that w is a* component *of s if s has a component of the form wx_i, for some $i \in [p]$.*

Lemma 4.25. *Let u_i be an a-chain of t_1, with $i \in [j]$. There is a leaf δ of κ such that*

$$u_i = w_0 \cdots w_d ,$$

where $d = |\delta|$, and w_ℓ is a component of s_{δ_ℓ}, for $\ell \in [0, d]$. Moreover, $\delta_{d-1} = 1$.

Proof. Recall that the chain u_i occurs in the tree $\sigma(v_i\#, u_i\#, x)$ in t. This tree is contributed to t by κ's i-th leaf node δ, when enumerated from left to right. Let $d = |\delta|$. By tracing the path from δ to the root of κ, we see that

$$u_i\# = \pi_{j_{\delta_0}} \cdot s_{\delta_0} \cdot \vartheta_{\delta_0} \cdots s_{\delta_{d-1}} \cdot \vartheta_{\delta_{d-1}} \cdot s_\varepsilon \cdot \chi .$$

Therefore $u_i = w_0 \cdots w_d$, where w_ℓ is a component of s_{δ_ℓ}, for each $\ell \in [0, d]$. □

In particular, w_d is a component of s_ε. The next lemma is a consequence of the fact that s_ε has only p components apart from x_{p+1}.

Lemma 4.26. *There is an a-defect γ whose critical chain is of the form $w'w$ for some w', $w \in \Gamma^*$ such that w is a component of s_ε, and $|\gamma| > |w| + mH$.*

Proof. Since t_1 contains more than p critical chains, by Lemmas 4.21 and 4.25 and the pigeonhole principle, there are two critical chains, say $u\gamma\alpha_i$ and $u'\gamma'\alpha_j$, where $\gamma\alpha_i$ and $\gamma'\alpha_j$ are distinct a-defects with $i < j$, such that

$$u\gamma\alpha_i = w'w \qquad \text{and} \qquad u'\gamma'\alpha_j = w''w \qquad \text{for some } w', w'' \in \Gamma^*,$$

and some component w of s_ε.

Observe that α_i is not a suffix of w, as otherwise α_i would be a suffix of α_j. Therefore $|w| < |\alpha_i|$, and hence

$$|w| + mH < |\alpha_i| + mH = |\alpha_{i+1}| \leq |\alpha_j| \leq |\gamma'\alpha_j|.$$

So the a-defect $\gamma'\alpha_j$ satisfies the properties in the lemma. □

Lemma 4.27. *There is some $t' \in \mathcal{L}(G) \setminus L$.*

Proof. Let γ be the a-defect from Lemma 4.26. Assume that γ's critical chain in t_1 is u_k, where $k \in [j]$. Then, by Lemma 4.25, $u_k = w_0 \cdots w_d$, where w_ℓ is a component of s_ℓ, for each $\ell \in [0, d]$. Moreover, $|\gamma| > |w_d| + mH$. Let f be the largest number such that $w_f \cdots w_d$ has γ as suffix. Then $f \in [0, \ldots, d-1]$, and there are w, $w' \in \Gamma^*$ such that $w_f = w'w$ and $\gamma = ww_{f+1} \cdots w_d$.

Since $d < m$ and $|ww_{f+1} \cdots w_{d-1}| > mH$, there is a $\tilde{w} \in \{w, w_{f+1}, \ldots, w_{d-1}\}$ such that $|\tilde{w}| > H$. In other words, there is an $\ell \in [f, d-1]$ such that $A_{\delta_\ell} \Rightarrow_G^* B_{\delta_\ell} \cdot s_{\delta_\ell}$, and there is some $i \in [p]$ such that either *(i)* $\ell = f$ and $\pi_i \cdot s_{\delta_\ell} = w'\tilde{w}x_i$, or *(ii)* $\ell \neq f$ and $\pi_i \cdot s_{\delta_\ell} = \tilde{w}x_i$. In both cases Lemma 4.18 can be applied, and we receive that $s_{\delta_\ell} = v \cdot y \cdot z$, and by pumping zero times, also $A_{\delta_\ell} \Rightarrow_G^* B_{\delta_\ell} \cdot v \cdot z$. Therefore a derivation tree κ' can be constructed from κ by replacing the tuple s_{δ_ℓ} by $v \cdot z$. As δ_ℓ begins with the symbol 1, this alteration does only concern t_1, thus κ' derives a tree $\hat{t}' \in \mathrm{T}(\Sigma)^1_{p+1}$ such that $\hat{t}' \cdot \chi = t'_1 \circ\!\!- t_2$, for some $t'_1 \in \mathrm{T}(\Sigma)^1_1$. Denote $\hat{t}' \cdot \chi$ by t'.

Let us compare the k-th a-chain u'_k of t'_1 to u_k. Assume that the i-th components of v, y, and z are, respectively, $v'x_i$, $y'x_i$ and $z'x_i$. Then in case *(i)*, there is a $w'' \in \Gamma^*$ such that $v' = w'w''$, as $y'z'$ is a suffix of w, by Lemma 4.18 *(ii)*. Therefore,

$$u_k = w_1 \cdots w' \underbrace{\overbrace{w''y'z'}^{\tilde{w}=w} w_{f+1} \cdots w_d}_{\gamma} \quad \text{and} \quad u'_k = w_1 \cdots w'w''z'w_{f+1} \cdots w_d\,.$$

In case *(ii)*,

$$u_k = w_1 \cdots \underbrace{ww_{f+1} \cdots w_{\ell-1} \overbrace{v'y'z'}^{\tilde{w}=w_\ell} w_{\ell+1} \cdots w_d}_{\gamma} \quad \text{and} \quad u'_k = w_1 \cdots w_{\ell-1} v'z' w_{\ell+1} \cdots w_d\,.$$

It is easy to see that $|t'|_\sigma = |t|_\sigma$, as the shape of κ was not modified. Thus Lemma 4.5 implies that if $t' \in L$, then also $|t'|_c = |t|_c$ and $|t'|_d = |t|_d$. In particular, $y' \in a^*$. Therefore, both in case *(i)* and *(ii)*, $P'_{q,j} = \iota(t'_1)$ is a perturbation of $P_{q,j}$. Say that $P_{q,j}$ and $P'_{q,j}$ are of the form as in (4.3). Since $|y'| > 0$ by Lemma 4.18, at least one a was removed from the occurrence of γ^R in $P_{q,j}$. Therefore, there is some $e \in [q]$ such that $\alpha_e \neq \alpha'_e$. Thus, by Lemma 4.22, $P'_{q,j} \not\equiv P_{q,j}$, i.e., $\iota(t'_1) \not\equiv \iota(t_1)$. Denote the reduct of $\iota(t_2)$ by R. Note that $R \in \{b, d\}^*$. Hence with Lemma 1.8, we may conclude

$$\iota(t') = \iota(t'_1)\,\iota(t_2) \equiv \iota(t'_1)\,R \not\equiv \iota(t_1)\,R \equiv \iota(t_1)\,\iota(t_2) \equiv \varepsilon\,.$$

So $\iota(t') \notin D^*_\Gamma$, and by Lemma 4.4, $t' \notin L$. □

Since Lemma 4.27 contradicts our previous assumption that $\mathcal{L}(G) = L$, there is no cftg G with $\mathcal{L}(G) = h^{-1}(\mathcal{L}(G_{\mathrm{ex}}))$, and we have proven Theorem 4.8, as restated below.

Theorem 4.8. *There are an l-cftg G_{ex} and a linear, nondeleting, strict, and injective tree homomorphism h such that $h^{-1}(\mathcal{L}(G_{\mathrm{ex}}))$ is not a context-free tree language.*

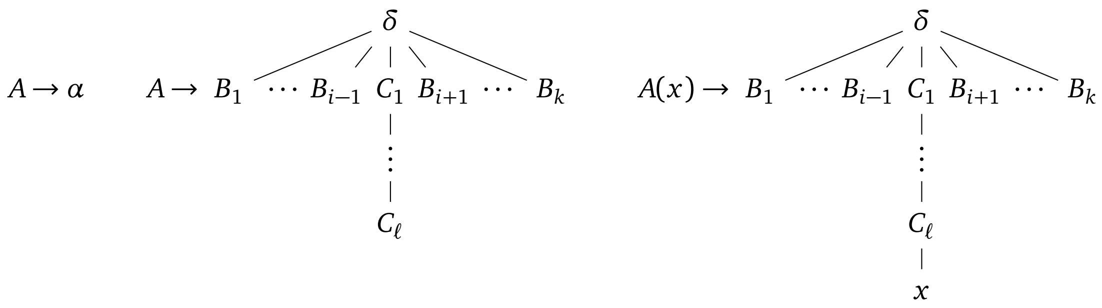


Figure 4.8: Types of productions of an lm-cftg in Greibach normal form

4.2 Linear Monadic Context-Free Tree Languages and Inverse Homomorphisms

In this section, we will prove the positive result announced in the chapter's introduction.

Theorem 4.28. *The class of linear monadic context-free tree languages is closed under inverse linear tree homomorphisms.*

We will prove this theorem in the remainder of this section. The following convention allows us to save some quantifications.

Convention. *In this section, Σ and Δ will denote arbitrary ranked alphabets, unless stated otherwise.*

Let us start by recalling a normal form for lm-cftg given by Fujiyoshi in [63]. Let $G = (N, \Delta, \xi_0, P)$ be an lm-cftg. We say that G is in *strong Greibach normal form*,[6] or *strongly Greibach*, if $\xi_0 = S$ for some $S \in N^{(0)}$ and each production in P is of one of the following forms:

(G1) $A \to \alpha$ for some $A \in N^{(0)}$, $\alpha \in \Delta^{(0)}$,

(G2) $A \to \delta(B_1, \ldots, B_{i-1}, \eta, B_{i+1}, \ldots, B_k)$ for some $A \in N^{(0)}$, and $\eta \in \mathrm{T}(N)^1_0$, or

(G3) $A \to \delta(B_1, \ldots, B_{i-1}, \eta, B_{i+1}, \ldots, B_k)$ for some $A \in N^{(1)}$ and $\eta \in \widetilde{\mathrm{T}}(N)^1_1$,

and $k \in \mathbb{N}_1$, $i \in [k]$, $\delta \in \Delta^{(k)}$, $B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_k \in N^{(0)}$. Compare also Figure 4.8 for an illustration of the productions' forms. Note that every strongly Greibach lm-cftg is nondeleting.

Lemma 4.29 (Fujiyoshi [63, Thm. 4.3]). *For every lm-cftg G there is an equivalent lm-cftg G' in strong Greibach normal form.*

[6] In [63], the normal form is simply called *Greibach*. However, Greibach cftg have already been defined more generally, as described in Section 2.6. Note that, indeed, every strongly Greibach cftg is Greibach.

By Lemma 1.34, every linear tree homomorphism can be decomposed into one that is linear and alphabetic, and a finite number of elementary tree homomorphisms. So in order to show that the linear monadic context-free tree languages are closed under inverse linear tree homomorphisms, it suffices to show closure under the inverses of these two restricted types of tree homomorphisms. This idea was already used to prove inverse homomorphic closure of the tree languages of (unrestricted) Greibach context-free tree grammars in [18]. The proofs of the following lemmas use similar techniques as in [18]. We stress, however, that our results are not direct consequences of the ones in [18]: there, the constructed cftg are nonlinear and non-monadic.

Lemma 4.30. *The class of linear monadic context-free tree languages is closed under inverse linear alphabetic tree homomorphisms.*

Proof. Consider an lm-cftg $G = (N, \Delta, S, P)$ in strong Greibach normal form, and let

$$h\colon \mathrm{T}(\Sigma) \to \mathrm{T}(\Delta)$$

be a linear alphabetic tree homomorphism. Let $H = (M, \Sigma, Z, R)$ be a regular tree grammar such that $\mathcal{L}(H) = \mathrm{T}_\Sigma$, and M is disjoint from N. We use the same idea as in [14, Thm. 4.1] to construct an lm-cftg $G' = (N', \Sigma, S, P')$ with $\mathcal{L}(G') = h^{-1}(\mathcal{L}(G))$. Let $N' = N \cup M \cup \{E^{(1)}\}$ for some distinct nonterminal symbol E, and let P' be given as follows.

(i) For every production of form *(G1)* in P, every $n \in \mathbb{N}$, and $\sigma \in \Sigma^{(n)}$, if $h(\sigma) = \alpha$, then P' contains $A \to E(\sigma(Z, \ldots, Z))$.

(ii) For every production of form *(G2)* or *(G3)* in P, every $n \in \mathbb{N}$, $\sigma \in \Sigma^{(n)}$, and $\vartheta \in \Theta_n^k$, if $h(\sigma) = \delta \cdot \vartheta$, then P' contains the production $A \to E \cdot \sigma \cdot \kappa$, with $\kappa \in \mathrm{T}(N')_1^n$ such that

$$\vartheta \cdot \kappa = [B_1, \ldots, B_{i-1}, \eta, B_{i+1}, \ldots, B_k] \quad \text{and} \quad \pi_j \cdot \kappa = Z \ \text{ for every } j \in [n] \setminus \vartheta([k]).$$

Note that κ is determined uniquely by these conditions.

(iii) For every $n \in \mathbb{N}$, $\sigma \in \Sigma^{(n)}$, and $j \in [n]$, if $h(\sigma) = x_j$, then P' contains the production $E(x) \to \sigma(\kappa_1, \ldots, \kappa_n)$, where for each $\ell \in [n]$,

$$\kappa_\ell = \begin{cases} x & \text{if } \ell = j, \\ Z & \text{otherwise.} \end{cases}$$

(iv) P' contains the productions $E(x) \to x$ and $E(x) \to E(E(x))$.

(v) P' contains all productions from R.

The construction's proof rests on the two following properties.

(A) $\mathcal{L}(G', Z) = \mathrm{T}_\Sigma$.

(B) $\mathcal{L}(G', E) = h^{-1}(x) \cap \widetilde{\mathrm{T}}(\Sigma)_1^1$.

The first property holds by construction, while the second one can be understood by a close look at rules *(iii)* and *(iv)* from above. We will prove that for every $\xi \in \mathrm{T}(N)_0^1$ and $t \in \mathrm{T}(\Sigma)_0^1$, we have

$$\xi \overset{\text{OI}}{\Rightarrow}{}^*_{G'} t \qquad \text{if and only if} \qquad \xi \overset{\text{OI}}{\Rightarrow}{}^*_{G} h(t)\,.$$

For both directions of the implication, the proof is by complete induction on the length of the derivation. The properties **(A)** and **(B)** will be used implicitly.

* * *

We begin with the direction “if”. Clearly, there is no derivation $\xi \overset{\text{OI}}{\Rightarrow}{}^0_G h(t)$, so the induction base holds vacuously. We proceed by a case distinction on the first production of a nonempty derivation in G.

(I) If the first production is of form *(G1)*, then $\xi = A$ and

$$A \overset{\text{OI}}{\Rightarrow}_G \alpha\,.$$

Assume a tree $t \in \mathrm{T}_\Sigma$ with $h(t) = \alpha$. Then there are $u \in \widetilde{\mathrm{T}}(\Sigma)_1^1$, $\sigma \in \Sigma$, and $v \in \mathrm{T}(\Sigma)_0$ with

$$t = u \cdot \sigma \cdot v\,, \qquad h(u) = x\,, \qquad \text{and} \qquad h(\sigma) = \alpha\,.$$

By construction of G',

$$A \overset{\text{OI}}{\Rightarrow}_{G'} E \cdot \sigma(Z, \ldots, Z) \overset{\text{OI}}{\Rightarrow}{}^*_{G'} u \cdot \sigma \cdot v = t\,.$$

(II) Otherwise, the derivation’s first production is of form *(G2)* or *(G3)*. As both cases are very similar, we will only show the proof for production type *(G3)*. For this purpose, let $\xi = A \cdot \zeta$ for some $A \in N^{(1)}$ and $\zeta \in \mathrm{T}(N)_0^1$, and let

$$A \cdot \zeta \overset{\text{OI}}{\Rightarrow}_G \delta(B_1, \ldots, B_{i-1}, \eta \cdot \zeta, B_{i+1}, \ldots, B_k) \overset{\text{OI}}{\Rightarrow}{}^m_G \delta \cdot s$$

for some production of form *(G3)*, some $m \in \mathbb{N}$, and some tuple $s \in \mathrm{T}(\Delta)_0^k$.

Assume a tree $t \in \mathrm{T}(\Sigma)_0^1$ such that $h(t) = \delta \cdot s$. Then there are $u \in \widetilde{\mathrm{T}}(\Sigma)_1^1$, $n \in \mathbb{N}$, $\sigma \in \Sigma^{(n)}$, a linear torsion $\vartheta \in \Theta_n^k$, and a tuple $v \in \mathrm{T}(\Sigma)_0^n$ such that

$$t = u \cdot \sigma \cdot v\,, \qquad h(u) = x\,, \qquad h(\sigma) = \delta \cdot \vartheta\,, \qquad \text{and} \qquad \vartheta \cdot h(v) = s\,.$$

By construction, P' contains a production

$$A \to E \cdot \sigma \cdot \kappa \qquad \text{such that} \qquad \vartheta \cdot \kappa = [B_1, \ldots, B_{i-1}, \eta, B_{i+1}, \ldots, B_k]\,,$$

and $\pi_j \cdot \kappa = Z$ for every $j \in [n] \setminus \vartheta([k])$. By applying the induction hypothesis, we see that

$$\vartheta \cdot \kappa = [B_1, \ldots, B_{i-1}, \eta \cdot \zeta, B_{i+1}, \ldots, B_k] \overset{\text{OI}}{\Rightarrow}{}^*_{G'} \vartheta \cdot v\,,$$

and since $\mathcal{L}(G', Z) = \mathrm{T}_\Sigma$, we conclude that $\kappa \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^*_{G'} v$. Moreover, as $h(u) = x$, we have that $E \overset{\text{OI}}{\Rightarrow}{}^*_{G'} u$. Altogether,

$$A \cdot \zeta \overset{\text{OI}}{\Rightarrow}_{G'} E \cdot \sigma \cdot \kappa \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^*_{G'} u \cdot \sigma \cdot v = t\,.$$

* * *

Now, let us argue for the direction “only if”. Again, the induction base holds vacuously, and we proceed by a case analysis on the production the derivation begins with.

(I) Assume that the first production has been introduced into P' by rule *(i)*. Then $\xi = A$ for some $A \in N^{(0)}$. Moreover, there are $m \in \mathbb{N}$ and some $\sigma \in \Sigma$ with $h(\sigma) = \alpha$, such that

$$A \overset{\text{OI}}{\Rightarrow}_{G'} E \cdot \sigma(Z, \dots, Z) \overset{\text{OI}}{\Rightarrow}{}^m_{G'} u \cdot \sigma \cdot v = t\,,$$

for some $u \in \mathcal{L}(G', E)$ and $v \in \mathcal{L}(G', [Z, \dots, Z])$. By construction, the production $A \to \alpha$ is in P, and we obtain

$$A \overset{\text{OI}}{\Rightarrow}_G \alpha = h(t)\,,$$

since $h(u) = x$.

(II) Otherwise, the first production has been introduced by rule *(ii)*. Again, we will only consider productions of form *(G3)*. So there are some m, $n \in \mathbb{N}$, $\sigma \in \Sigma^{(n)}$, and $\zeta \in \mathrm{T}(\Sigma)_0$, such that

$$\xi = A \cdot \zeta \overset{\text{OI}}{\Rightarrow}_{G'} E \cdot \sigma \cdot \kappa \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^m_{G'} u \cdot \sigma \cdot v = t\,,$$

for some $u \in \mathcal{L}(G', E)$ and $v \in \mathcal{L}(G', \kappa \cdot \zeta)$. By construction, P contains a production

$$A \to \delta(B_1, \dots, B_{i-1}, \eta, B_{i+1}, \dots, B_k)$$

of form *(G3)*, such that $h(\sigma) = \delta \cdot \vartheta$ for some linear torsion $\vartheta \in \Theta^k_n$, and

$$\vartheta \cdot \kappa = [B_1, \dots, B_{i-1}, \eta, B_{i+1}, \dots, B_k]\,.$$

Applying the induction hypothesis to each component of the tuple, we obtain $\vartheta \cdot \kappa \cdot \zeta \overset{\text{OI}}{\Rightarrow}{}^*_G \vartheta \cdot h(v)$. So

$$A \cdot \zeta \overset{\text{OI}}{\Rightarrow}_G \delta \cdot \underbrace{[B_1, \dots, B_{i-1}, \eta, B_{i+1}, \dots, B_k] \cdot \zeta}_{\vartheta \cdot \kappa \cdot \zeta} \overset{\text{OI}}{\Rightarrow}{}^*_G \delta \cdot \vartheta \cdot h(v) = h(u \cdot \sigma \cdot v) = h(t)\,.$$

The last but one equation holds because $h(u) = x$. □

Lemma 4.31. *The class of linear monadic context-free tree languages is closed under inverse elementary tree homomorphisms.*

Proof. For this purpose, let Ω be a ranked alphabet such that Ω and $\{\delta_1, \delta_2, \sigma\}$ are disjoint. Let $\Sigma = \Omega \cup \{\sigma^{(k)}\}$ and $\Delta = \Omega \cup \{\delta_1^{(n-k+1)}, \delta_2^{(k)}\}$ for some n, $k \in \mathbb{N}$. Let $h \colon \mathrm{T}(\Sigma) \to \mathrm{T}(\Delta)$ be the elementary tree homomorphism with

$$h\big(\sigma(x_1, \dots, x_n)\big) = \delta_1\big(x_1, \dots x_{\ell-1}, \delta_2(x_\ell, \dots, x_{\ell+k-1}), x_{\ell+k}, \dots, x_n\big)$$

for some $\ell \in [n+1]$, and h is the identity on Ω.

Consider an lm-cftg $G = (N, \Delta, S, P)$. By Lemma 4.29, G can be assumed to be in strong Greibach normal form. Moreover, we assume without loss of generality that

– $\mathcal{L}(G) \subseteq h(\mathrm{T}_\Sigma)$,[7]

[7] If it is not, one can apply the method from Theorem 2.35 to construct a cftg G' with $\mathcal{L}(G') = \mathcal{L}(G) \cap h(\mathrm{T}_\Sigma)$. Note that G' is again linear and monadic, and in strong Greibach normal form.

– G is total (by Lemma 2.4), and

– G has no unreachable nonterminal symbols, i.e., for every $A \in N$, there are $\xi \in \widetilde{\mathrm{T}}(N \cup \Delta)_1^1$ and $\zeta \in \mathrm{T}(N \cup \Delta)$ such that $S \Rightarrow_G^* \xi \cdot A \cdot \zeta$ (cf. [18, Prop. 14]).

Then, we can observe that the following property holds for G (cf. [14, Lem. 17]).

(A) *For all $A \in N$ and $t \in \mathrm{T}(\Delta \cup N)_1^1$ with $A \Rightarrow_G^* t$ we have that t contains no subtree of one of the following shapes:*

- $\gamma \cdot [u, \delta_2 \cdot v, w]$ *for some* $\gamma \in \Delta \setminus \{\delta_1\}$ *and* $u \in \mathrm{T}(\Delta \cup N)$,
- $\delta_1 \cdot [u, \gamma \cdot v, w]$ *for some* $\gamma \in \Delta \setminus \{\delta_2\}$ *and* $u \in \mathrm{T}(\Delta \cup N)_1^{\ell-1}$*, or*
- $\delta_1 \cdot [u, \delta_2 \cdot v, w]$ *for some* $u \in \mathrm{T}(\Delta \cup N)_1^m$ *and* $m \neq \ell - 1$,

and where v, $w \in \mathrm{T}(\Delta \cup N)$.

Let in the following

$$\tilde{N} = \left\{A \in N \;\middle|\; \exists u \in \mathrm{T}(\Sigma) \colon A \Rightarrow_G^* \delta_2 \cdot u\right\}.$$

As G is in strong Greibach normal form and total, and $\mathcal{L}(G) \subseteq h(\mathrm{T}_\Sigma)$, the following observation can be made.

(B) *Let $A \in \tilde{N}$. For every $t \in \mathcal{L}(G,A)$, we have $t(\varepsilon) = \delta_2$. In particular, for every production $A \to \varrho$ of G, we have $\varrho(\varepsilon) = \delta_2$.*

We will construct an lm-cftg G' such that $\mathcal{L}(G') = h^{-1}(\mathcal{L}(G))$. We proceed in two steps, constructing successively the lm-cftg G_1 and G_2 equivalent to G.

Recall that a production is *useless* if no terminal tree can be derived from its right-hand side; otherwise we call it *useful*. Our aim for G_2 is that in the right-hand side of each useful production of G_2, every occurrence of δ_1 has δ_2 as its ℓ-th child, and there are no other occurrences of δ_2. Formally, we demand for every production $A \to \varrho$ of G_2 with $\mathcal{L}(G, \varrho) \neq \emptyset$ that

$$\varrho(\varepsilon) \neq \delta_2\,, \qquad \text{and} \qquad \varrho(wj) = \delta_2 \quad \text{iff} \quad (j = \ell \text{ and } \varrho(w) = \delta_1) \tag{4.4}$$

for every $w \in \mathbb{N}_1^*$ and $j \in \mathbb{N}_1$ with $wj \in \mathrm{pos}(\varrho)$.

To establish this property, we first remove all productions of the form $A \to \varrho$ where $\varrho(\varepsilon) = \delta_2$ and A occurs in ϱ. For this, we construct the lm-cftg $G_1 = (N_1, \Sigma, S, P_1)$ with

$$N_1 = N \cup \{C_p \mid p \in P\}$$

and the following productions in P_1:

(i) Every production $A \to \varrho$ in P with $\varrho(\varepsilon) \neq \delta_2$ is also in P_1.

(ii) For every production $p = A \to \delta_2(B_1, \ldots, B_{i-1}, \eta, B_{i+1}, \ldots, B_k)$ in P, with $\eta \in \mathrm{T}(N)_0^1$, the productions $A \to \delta_2(B_1, \ldots, B_{i-1}, C_p, B_{i+1}, \ldots, B_k)$ and $C_p \to \eta$ are in P_1.

(iii) For every production $p = A(x) \to \delta_2(B_1, \dots, B_{i-1}, \eta, B_{i+1}, \dots, B_k)$ in P, with $\eta \in \widetilde{\mathrm{T}}(N)_1^1$, the productions $A(x) \to \delta_2(B_1, \dots, B_{i-1}, C_p(x), B_{i+1}, \dots, B_k)$ and $C_p(x) \to \eta$ are in P_1.

It is easy to see that $\mathcal{L}(G_1) = \mathcal{L}(G)$. Now consider a production

$$p' = A \to \delta_2(B_1, \dots, B_{i-1}, C_p, B_{i+1}, \dots, B_k)$$

in P_1. By properties **(A)** and **(B)**, $B_j \neq A$ for each $j \in [k] \setminus \{i\}$. For this reason and since C_p is a fresh nonterminal, A does not occur in the right-hand side of p'.

Thus, we can *eliminate* the production p' from G_1, as described in [117, Def. 11]. We construct an lm-cftg $\mathrm{Elim}(G_1, p')$ as follows: for each production $B \to \varrho$ in $P_1 \setminus \{p'\}$ and each subset $W \subseteq \{w \in \mathrm{pos}(\varrho) \mid \varrho(w) = A\}$, we construct a production $B \to \varrho'$ and insert it into P_1. Its right-hand side ϱ' is obtained by substituting the right-hand side of p' for A at each position in W. Then p' is removed from P_1. It is shown in [117, Lem. 12] that $\mathcal{L}(\mathrm{Elim}(G_1, p')) = \mathcal{L}(G_1)$. The same idea works for productions of the form $A(x) \to \delta_2(B_1, \dots, B_{i-1}, C_p(x), B_{i+1}, \dots, B_k)$ in P_1.

As an example, when we eliminate the production $p' = A(x) \to \delta_2(B, C(x))$ in G_1, we construct from the production $D(x) \to \delta_1\big(E, A(F(x)), B\big)$ in G_1 the *two* productions

$$p_1 = D(x) \to \delta_1\big(E, A(F(x)), B\big) \qquad \text{and} \qquad p_2 = D(x) \to \delta_1\big(E, \delta_2(B, C(F(x))), B\big),$$

and p' is discarded.

By applying this procedure successively for each production with a nonterminal from $\tilde{N}$ in its left-hand side, we obtain in finitely many steps an equivalent lm-cftg $G_2 = (N_2, \Sigma, S, P_2)$, where δ_2 only appears under δ_1 in its useful productions. The cftg G_2 may still contain productions which do not satisfy (4.4), but all of them are useless. In our example, if p' was the last production to be eliminated, then there is still the production p_1 left, where δ_2 does not occur under δ_1. However, it is easy to see that this production is useless: after all, by property **(B)**, there are no productions left for the nonterminal A. This observation applies to all productions $B \to \varrho$ which are not of the desired form.

* * *

We now proceed with an idea from [18, Lem. 18]. As δ_1 and δ_2 only appear right beneath each other in the useful productions of G_2, they can just be replaced by σ.

Formally, define a homomorphism

$$\varphi\colon \mathrm{T}(N_2 \cup \Sigma) \to \mathrm{T}(N_2 \cup \Delta)$$

such that $\varphi(A) = A$ for each $A \in N_2$ and $\varphi|_\Sigma = h$. We construct an lm-cftg $G' = (N_2, \Sigma, S, P')$ such that P' contains the production $A \to \varrho$ if and only if P_2 contains the production $A \to \varphi(\varrho)$. The formal proof that $\mathcal{L}(G') = h^{-1}(\mathcal{L}(G))$ is omitted, as it is essentially identical to [18, Lem. 18]. $\square$

By Lemmas 4.30, 4.31 and 1.34, we conclude that Theorem 4.28 holds, as restated below.

Theorem 4.28. *The class of linear monadic context-free tree languages is closed under inverse linear tree homomorphisms.*

4.3 Chapter Conclusion

In this chapter, we proved that the class of linear context-free tree languages is not closed under inverse linear tree homomorphisms: there is a linear context-free tree grammar, which is 3-adic, and whose preimage under a particular homomorphism is not context-free. However, the tree languages of linear monadic context-free tree grammars, which are employed in praxis under the pseudonym of tree-adjoining grammars, have been proved to be closed under this operation.

So there still remains a “gap” to close: the question whether the tree languages of 2-adic linear context-free tree grammars are closed under the examined operation.

We conjecture that also for these grammars, closure does not hold. For a potential witness, consider the 2-adic l-cftg $G = (N, \Delta, \xi_0, P)$ with

$$N = \{L^{(2)}, R^{(2)}\} \qquad \text{and} \qquad \Delta = \{\delta^{(2)}, a^{(1)}, b^{(1)}, c^{(1)}, d^{(1)}, \#^{(0)}\}\,,$$

axiom

$$\xi_0 = \begin{array}{lll} L & - & \# \\ | & & \\ \# & & \end{array}\,,$$

and the productions in P given by

$$\begin{array}{lll} L & - & x_2 \\ | & & \\ x_1 & & \end{array} \;\rightarrow\; \begin{array}{ccccc} L & - & R & - & x_2 \\ | & & | & & \\ a & & b & & \\ | & & | & & \\ x_1 & & \# & & \end{array} \;+\; \begin{array}{ccccc} L & - & R & - & x_2 \\ | & & | & & \\ c & & d & & \\ | & & | & & \\ x_1 & & \# & & \end{array} \;+\; \begin{array}{lll} \delta & - & x_2 \\ | & & \\ x_1 & & \end{array}$$

$$\begin{array}{lll} R & - & x_2 \\ | & & \\ x_1 & & \end{array} \;\rightarrow\; \begin{array}{ccccc} L & - & R & - & x_2 \\ | & & | & & \\ a & & b & & \\ | & & | & & \\ \# & & x_1 & & \end{array} \;+\; \begin{array}{ccccc} L & - & R & - & x_2 \\ | & & | & & \\ c & & d & & \\ | & & | & & \\ \# & & x_1 & & \end{array} \;+\; \begin{array}{lll} \delta & - & x_2 \\ | & & \\ x_1 & & \end{array}\,.$$

Let $\Sigma = \Delta \setminus \{\delta\} \cup \{\sigma^{(3)}\}$. The homomorphism $h \colon \mathrm{T}(\Sigma) \to \mathrm{T}(\Delta)$ is such that

$$h\big(\sigma(x_1, x_2, x_3)\big) = \delta\big(x_1, \delta(x_2, x_3)\big)$$

and h is the identity for all other symbols.

Our conjecture is that $h^{-1}(\mathcal{L}(G))$ is not context-free, since $\mathcal{L}(G)$ exhibits a relationship between chains that is similar to $\mathcal{L}(G_{\text{ex}})$. Therefore, it should be possible to employ comparable methods to those in this section. However, coming up with a pumping argument as in Lemma 4.18 is harder: In G_{ex}, it it possible to pump chains of a tree without modifying its spine. In contrast to this, the chains of trees in $\mathcal{L}(G)$ cannot be pumped independently from their spines. Thereby the number of chains is altered, which complicates the analysis tremendously. We leave it to other researchers to come up with a solution.

Chapter 5

Synchronous Context-Free Tree Transformations and Pushdown Tree Transducers

Translation is the art of failure.

(Umberto Eco)

In this chapter, we will concern ourselves with *synchronous* context-free tree grammars, which generate a tree transformation instead of a tree language.

Synchronous context-free *word* grammars (or *syntax directed translations*) are a venerable subject of theoretical computer science. They were discovered in the 1960s, due to the practical need for syntax-directed compilers for the nascent high-level programming languages. Indeed, they are such a natural concept that they were essentially discovered independently by several scholars [88, 35, 112, 135, 5].

Coarsely spoken, a synchronous cfg consists of two cfg, called the *input* and the *output* cfg. A production of a synchronous cfg is then a pair of an input and an output production – where in each pair there is a one-to-one correspondence between the occurrences of nonterminal symbols in the productions' right-hand sides. Therefore, a derivation tree of an input word determines a unique derivation tree of the output cfg; from this tree's yield, we obtain the (word) transformation generated by the synchronous cfg.

By this explanation, it is easy to see that synchronous cfg have a *bidirectional* semantics – the grammar's input and its ouput cfg are of the same form, and they derive input and output words simultaneously. In fact, one can construct from a synchronous cfg G a synchronous cfg G' that generates the inverse of the transformation of G, simply by swapping G's input and output cfg.

However, compiler construction necessitates a *unidirectional* translation formalism – i.e., a rule system which, given an input, derives an output by traversing the input from left to right. In [112], Lewis and Stearns give a partial answer to the problem of finding unidirectional devices that capture the transformations generated by synchronous cfg. The authors identify the subclass of *simple* synchronous cfg. A synchronous cfg G is simple if for each production of G, the i-th occurrence of a nonterminal symbol on the input side corresponds to the i-th occurrence of a nonterminal on the output side. Intuitively, simple synchronous cfg cannot permute parts of the input. Then the authors go on to show that the transformations of simple synchronous cfg are precisely those of *pushdown machines,* i.e., pushdown automata with output. A characterization of the full class of transformations of synchronous cfg has been given later, by endowing pushdown automata with registers [6].

Subsequent research extended the power of synchronous cfg, leading to devices such as the *generalized syntax directed translation* [7]. Then again, by uncoupling parsing and translation, these prompted the discovery of formalisms that define tree transformations, such as e.g. the *(generalized*[2]*) finite state transformation* [160], now known as the *top-down tree transducer* [49].

* * *

In the field of natural language translation, where tree transformations are used to make use of the grammatical structure of input sentences, many systems are based on bidirectional semantics. Hence there is an abundance of kinds of synchronous tree grammars, such as synchronous tree substitution grammars [48], synchronous tree insertion grammars [125], synchronous tree-adjoining grammars [154], and so on. In [124], Nederhof and Vogler introduce *synchronous context-free tree grammars,* which may allow modelling even more linguistic phenomena than former types of synchronous grammars.

Within this chapter, we will consider *weighted synchronous context-free tree grammars (wscftg)* – synchronous cftg augmented with weights from a semiring, thus allowing the model to define weighted tree transformations. We will define what it means for a wscftg to be *simple,* in analogy to the condition introduced by Lewis and Stearns for synchronous cfg. With the help of a normal form lemma, we will then show that the weighted tree transformations of simple wscftg are precisely those of *weighted pushdown extended (top-down) tree transducers (wpxtt)*. The latter model can be understood as a weighted extended top-down tree transducer whose state control is enhanced with tree pushdowns.

This characterization by a unidirectional formalism may serve as a starting point to define weighted tree transformations which are conditional probability distributions, as remarked in [28]. Moreover, it generalizes the classical result of Lewis and Stearns from formal languages to weighted tree languages.

The current chapter is organized as follows. In Section 5.1, we introduce wscftg and prove a production interchange lemma. Section 5.1.1 contains the definition of simple wscftg, and a normal form for these grammars. In Section 5.2, we present wpxtt, along with some technical lemmas and normal forms. Finally, Section 5.3 is dedicated to the announced characterization result.

Note: The results in this chapter were first reported in [129]. However, in this chapter we use a distinct and, hopefully, improved presentation of wscftg. The proofs underlying the normal form lemma for wscftg have been extended.

5.1 Synchronous Context-Free Tree Grammars

Let us begin by expressing formally the correspondence between nonterminal occurrences in a tuple of trees. Consider for this purpose ranked alphabets N, Σ, and Δ. We define, for every k_1 and $k_2 \in \mathbb{N}$, the set

$$\begin{aligned} \mathrm{S}(N,\Sigma,\Delta)_{k_1,k_2} = \big\{(\xi,\zeta,\lambda) \;\big|\; & \xi \in \mathrm{T}(N\cup\Sigma)^1_{k_1},\ \zeta \in \mathrm{T}(N\cup\Delta)^1_{k_2}, \\ & \xi \text{ and } \zeta \text{ are linear and nondeleting}, \\ & \lambda \text{ is a bijection between } \mathrm{pos}_N(\xi) \text{ and } \mathrm{pos}_N(\zeta)\big\}\,. \end{aligned}$$

So $\mathrm{S}(N,\Sigma,\Delta)_{k_1,k_2}$ contains tuples of linear and nondeleting trees, such that there is a bijective relation between the occurrences of symbols from N in both components. We will say that two occurrences are *linked* if they are related in this manner, and the elements of λ are called *links*. The first component ξ of a tuple $(\xi,\zeta,\lambda) \in \mathrm{S}(N,\Sigma,\Delta)_{k_1,k_2}$ will be called its *input side*, and the second one ζ its *output side*. Moreover, (ξ,ζ,λ) is called a *synchronized tree*. Similar to the notation for magmoids, we will write

$$\mathrm{S}(N,\Sigma,\Delta) = \bigcup_{k_1,k_2\in\mathbb{N}} \mathrm{S}(N,\Sigma,\Delta)_{k_1,k_2}\,.$$

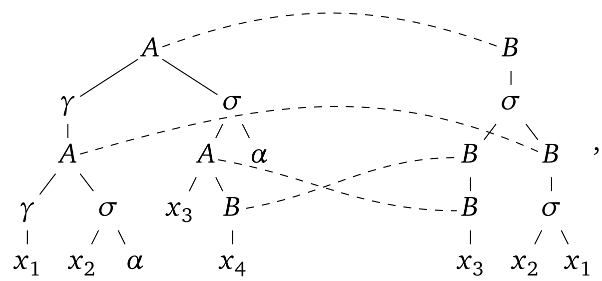
Example 5.1. Consider the ranked alphabets $N = \{A^{(2)}, B^{(1)}\}$ and $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$. Let

$$\begin{aligned} \xi &= A\Big(\gamma\big(A(\gamma(x_1),\ \sigma(x_2,\alpha))\big),\ \sigma\big(A(x_3, B(x_4)), \alpha\big)\Big)\,, \\ \zeta &= B\Big(\sigma\big(B(B(x_3)),\ B(\sigma(x_2,x_1))\big)\Big)\,, \end{aligned}$$

and

$$\lambda = \big\{(\varepsilon,\varepsilon),\ (11,12),\ (21,111),\ (212,11)\big\}\,.$$

Then $(\xi,\zeta,\lambda) \in \mathrm{S}(N,\Sigma,\Sigma)_{4,3}$. We can depict (ξ,ζ,λ) by



representing the links between the nonterminals by dashed arcs. $\lhd$

Now, a *weighted synchronous context-free tree grammar (wscftg)* is a tuple

$$G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$$

such that

- N is a ranked alphabet (of *nonterminal symbols*),
- Σ and Δ are ranked alphabets disjoint from N (its elements called *input*, resp. *output terminal symbols*),
- ξ_0 is an element of $\mathrm{S}(N,\Sigma,\Delta)_{0,0}$ (the *axiom*),
- P is a finite set (its elements called *productions*), where each production is of form

$$(A_1 \cdot \mathrm{Id}_{n_1}, A_2 \cdot \mathrm{Id}_{n_2}) \to \varrho$$

  for some $n_1,\, n_2 \in \mathbb{N}$, $A_1 \in N^{(n_1)}$, $A_2 \in N^{(n_2)}$, and $\varrho \in \mathrm{S}(N,\Sigma,\Delta)_{n_1,n_2}$,
- K is a complete semiring, and $wt\colon P \to K$ (the *weight mapping*).

Recall that a production of the form given above can be written briefly as $(A_1, A_2) \to \varrho$.

Given a wscftg G as above, the elements of $\mathrm{S}(N,\Sigma,\Delta)$ are called the *sentential forms* of G. Before we continue with presenting the rewrite relation of G, we must introduce an auxiliary function. Its purpose is to update the links of a sentential form during a rewrite step. For this end, let $n \in \mathbb{N}$, $w \in \mathbb{N}_1^*$ and $u_1, \ldots, u_n \in \mathbb{N}_1^*$. We define the function

$$\delta^{w}_{u_1,\ldots,u_n} \,:\, \mathbb{N}_1^* \setminus (w \cdot \mathbb{N}_1^*) \,\cup\, w \cdot [n] \cdot \mathbb{N}_1^* \;\to\; \mathbb{N}_1^*$$

such that, for every element v of the domain of $\delta^{w}_{u_1,\ldots,u_n}$,

$$\delta^{w}_{u_1,\ldots,u_n}(v) = \begin{cases} v & \text{if } w \not\sqsubseteq v \\ wu_iv' & \text{if } v = wiv' \text{ for some } i \in [n] \text{ and } v' \in \mathbb{N}_1^*. \end{cases}$$

Now we can define the rewrite relation, as follows. Assume a wscftg G as given above, and a production p of G of the form

$$(A_1 \cdot \mathrm{Id}_{n_1}, A_2 \cdot \mathrm{Id}_{n_2}) \to (\varrho_1, \varrho_2, \tilde{\lambda}).$$

For each $j \in [2]$, denote by p_j the cftg production $A_j \cdot \mathrm{Id}_{n_j} \to \varrho_j$. Then the *rewrite relation by p*, denoted by $\Rightarrow_p$, is defined to be the smallest relation on $\mathrm{S}(N,\Sigma,\Delta)$ that satisifes the following conditions. For every (ξ_1, ξ_2, λ) and $(\zeta_1, \zeta_2, \lambda') \in \mathrm{S}(N,\Sigma,\Delta)$, we have

$$(\xi_1, \xi_2, \lambda) \Rightarrow_p (\zeta_1, \zeta_2, \lambda')$$

if there is some link $(w_1, w_2) \in \lambda$ such that $\xi_j \overset{w_j}{\Rightarrow}_{p_j} \zeta_j$ for each $j \in [2]$; moreover λ' is the smallest set that satisfies the following.

(i) For every link $(v_1, v_2) \in \tilde{\lambda}$, λ' contains (w_1v_1, w_2v_2), and

(ii) For every $j \in [2]$ and $\ell \in [n_j]$, let u^j_ℓ be the unique position of x_ℓ in ϱ_j. Then, for every link $(v_1, v_2) \in \lambda \setminus \{(w_1, w_2)\}$, λ' contains the element

$$\Big(\delta^{w_1}_{u^1_1,\ldots,u^1_{n_1}}(v_1),\, \delta^{w_2}_{u^2_1,\ldots,u^2_{n_2}}(v_2)\Big).$$

In this situation, we will say that the production p of G *is applied at positions* w_1 *and* w_2. We will also write $\xRightarrow{(w_1,w_2)}_p$ to emphasize the positions where the production is applied.

Remark 5.2. Since in the above definition, ξ_1 is a tree over a ranked alphabet, the function $\delta^{w_1}_{u^1_1,\dots,u^1_{n_1}}$ is defined on v_1. The analogous holds for ξ_2 and v_2. Note that λ' is again a bijection. $\lhd$

The definition may seem technical – its intuition is as follows. Following the application of the production p, the links of the sentential form (ξ_1, ξ_2, λ) must be updated. Condition *(i)* inserts the links from the right-hand side of p at the correct positions. The function used in condition *(ii)* handles the displacement of the links from λ by the insertion of the right-hand side of p – positions which are in the j-th child tree of w_1 (or of w_2) are reinserted under the occurrence of the variable x_j in ϱ_1 (resp. in ϱ_2).

As always, we let $\Rightarrow_G = \bigcup_{p \in P} \Rightarrow_p$, and call $\Rightarrow_G$ the *rewrite relation of* G. A *leftmost derivation in* G is a sequence $p_1 \cdots p_n$ of productions $p_1, \dots, p_n$ of G, for some $n \in \mathbb{N}$, if there are $(\xi_0, \zeta_0, \lambda_0), \dots, (\xi_n, \zeta_n, \lambda_n) \in \mathrm{S}(N, \Sigma, \Delta)$ and $w_1, v_1, \dots, w_n, v_n \in \mathbb{N}^*_1$ such that

$$(\xi_0, \zeta_0, \lambda_0) \xRightarrow{(w_1,v_1)}_{p_1} (\xi_1, \zeta_1, \lambda_1) \xRightarrow{(w_2,v_2)}_{p_2} \cdots \xRightarrow{(w_n,v_n)}_{p_n} (\xi_n, \zeta_n, \lambda_n)\,,$$

and for every $i \in [n]$, w_i is the minimal position in $\mathrm{pos}_N(\xi_{i-1})$ with respect to $\leq_{\mathrm{lex}}$.

In this situation, we say that $p_1 \cdots p_n$ is a leftmost derivation *of* ξ_n *from* ξ_0. Let $\xi, \zeta \in \mathrm{S}(N, \Sigma, \Delta)$, and $m \in \mathbb{N}$. Then the set of all leftmost derivations of ζ from ξ is denoted by $\mathcal{D}_G(\xi, \zeta)$, and the set $\mathcal{D}_G(\xi, \zeta) \cap P^m$ of leftmost derivations of length m is denoted by $\mathcal{D}^{(m)}_G(\xi, \zeta)$.

The weight mapping wt of G is extended to sequences of productions as follows. For every $n \in \mathbb{N}$, and $p_1, \dots, p_n \in P$, let

$$wt(p_1 \cdots p_n) = wt(p_1) \cdots wt(p_n)\,.$$

We are now in a position to explain the semantics of the wscftg G. Define, for every $\xi \in \mathrm{S}(N, \Sigma, \Delta)$, the weighted tree transformation $[\![G, \xi]\!] \colon \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \to K$, such that for every $s \in \mathrm{T}_\Sigma$ and $t \in \mathrm{T}_\Delta$,

$$[\![G, \xi]\!](s, t) = \sum\Big(wt(d) \;\Big|\; d \in \mathcal{D}_G\big(\xi, (s, t, \emptyset)\big)\Big)\,.$$

Moreover, for every additional $m \in \mathbb{N}$, let

$$[\![G, \xi]\!]^{(m)}(s, t) = \sum\Big(wt(d) \;\Big|\; d \in \mathcal{D}^{(m)}_G\big(\xi, (s, t, \emptyset)\big)\Big)\,.$$

Clearly,

$$[\![G, \xi]\!](s, t) = \sum_{m \in \mathbb{N}} [\![G, \xi]\!]^{(m)}(s, t)\,.$$

The *weighted tree transformation generated by* G, denoted by $[\![G]\!] \colon \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \to K$, is defined by $[\![G]\!] = [\![G, \xi_0]\!]$. A weighted tree transformation is said to be *context-free* if it is generated by some wscftg.

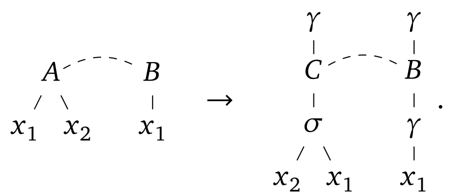
Remark 5.3. Let us compare our definition of wscftg to the definition of synchronous cftg given by Nederhof and Vogler in [124]. There, the links between nonterminal occurrences are not given explicitly by a relation, but implicitly by endowing each occurring nonterminal with a natural number, its *index*. An occurrence of a nonterminal is linked to another nonterminal occurrence in the component vis-à-vis if both occurrences are equipped with the same index. In the application of a production, a function f is applied to the indices, to make sure that the indices introduced from the production's right-hand side are distinct from the ones that were already present.

One can show by a straightforward analysis of definitions that the tree transformations of the synchronous cftg of Nederhof and Vogler coincide with the ones of our definition (over the semiring $\mathbb{B}$). Note, however, that the definition in [124] has the problem that the values of the indices in a sentential form depend on the order in which the grammar's productions are applied. This complicates giving a production interchange lemma (such as we will give in Lemma 5.9). The problem can be circumvented either by using the definition presented in this chapter, or by considering sentential forms only "up to renaming of indices," as developed in [129]. $\lhd$

Example 5.4. Let us show in an example how the functions from the family δ work. Consider the ranked alphabets $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and $N = \{A^{(2)}, B^{(1)}, C^{(1)}, D^{(0)}\}$. Further, consider the wscftg production p of the form

$$\big(A(x_1, x_2), B(x_1)\big) \to \Big(\gamma\big(C\big(\sigma(x_2, x_1)\big)\big), \gamma\big(B\big(\gamma(x_1)\big)\big), \big\{(1, 1)\big\}\Big).$$

Representing elements of $\mathrm{S}(N, \Sigma, \Delta)$ as in Example 5.1, we can depict p as

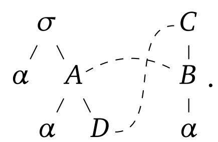


Here, we omit the parentheses from the production's left-hand side. Instead, we link the respective nonterminals A and B with a dashed arc, which symbolizes that A and B are to be rewritten in parallel.

Moreover, consider the sentential form $(\xi, \zeta, \lambda) \in \mathrm{S}(N, \Sigma, \Sigma)_{0,0}$ with

$$\xi = \sigma\big(\alpha, A(\alpha, D)\big), \qquad \zeta = C\big(B(\alpha)\big), \qquad \text{and} \qquad \lambda = \big\{(2, 1), (22, \varepsilon)\big\},$$

given by the picture

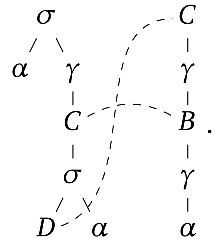


When we apply the production p at the linked positions 2 and 1, then this link is removed, and the link $(2 \cdot 1, 1 \cdot 1) = (21, 11)$ is introduced according to item *(i)* in the definition of the rewrite relation of wscftg.

Moreover, we must use the method from item *(ii)* to displace the link $(22, \varepsilon)$ of λ. We compute

$$\delta^2_{112,111}(22) = 2111 \qquad \text{and} \qquad \delta^1_{111}(\varepsilon) = \varepsilon\,.$$

Hence, the displaced link is $(2111, \varepsilon)$. The resulting sentential form is depicted as



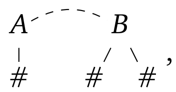
Therefore, the definition for the links appears to correspond to our intuition. $\triangleleft$

Example 5.5. Consider the wscftg $G = (N, \mathbb{B}, \Sigma, \Delta, \xi_0, P, wt)$, where

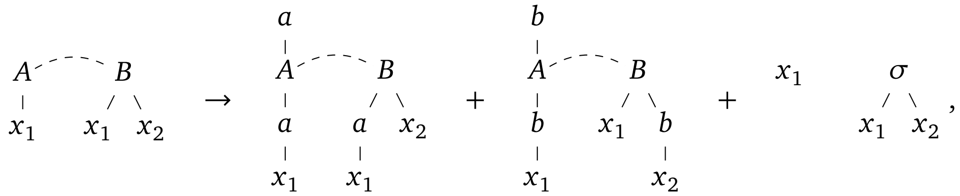
$$N = \{A^{(1)}, B^{(2)}\}\,, \qquad \Sigma = \{a^{(1)}, b^{(1)}, \#^{(0)}\}\,, \qquad \text{and} \qquad \Delta = \{a^{(1)}, b^{(1)}, \#^{(0)}, \sigma^{(2)}\}\,.$$

Moreover,

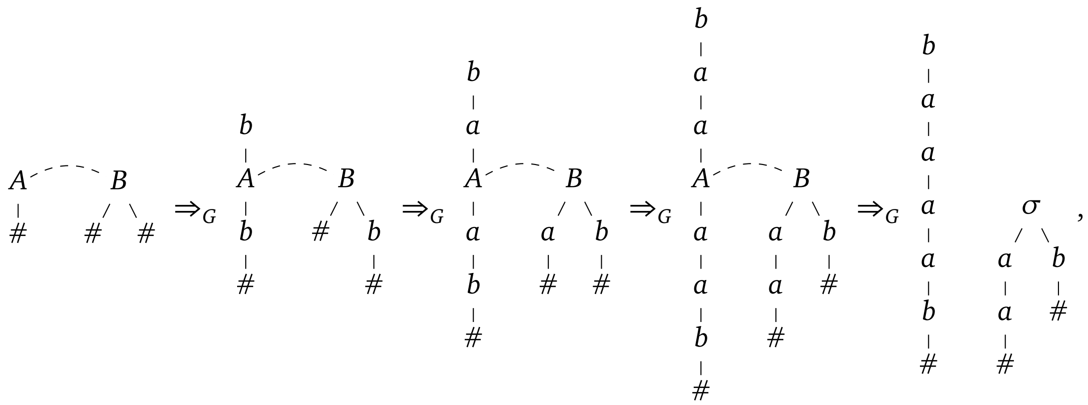
$\xi_0 =$



the productions in P are



and wt maps every production in P to 1. By a close look at the derivation



it is fairly easy to see that

$$\mathrm{supp}(\llbracket G \rrbracket) = \Big\{ \big(ww^R\#,\ \sigma\big(a^{|w|_a}\#, b^{|w|_b}\#\big)\big) \ \Big|\ w \in \{a,b\}^* \Big\}\,.$$

Intuitively, G checks if the input represents an even palindrome of form ww^R, and counts the symbols in w, using the two subtrees of σ. ◁

Let us introduce the following easy normal form. A wscftg $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$ has *initial nonterminals* if $\xi_0 = (S_1, S_2, \{(\varepsilon, \varepsilon)\})$ for some $S_1, S_2 \in N^{(0)}$.

Lemma 5.6. *For every wscftg G, there is a wscftg G' that has initial nonterminals and that satisfies $\llbracket G' \rrbracket = \llbracket G \rrbracket$.*

Proof. Analogous to Lemma 2.2. The new production has weight 1. □

Next, we will prove a production interchange lemma for wscftg. Before, however, there are two auxiliary lemmas which have to be established. Both lemmas describe how functions from the family δ commute under certain circumstances. Their purpose will become clear later in Lemma 5.9. The first lemma concerns the displacement of links already present in a sentential form.

Lemma 5.7. *Let n, $m \in \mathbb{N}$ and v, w, $u_1, \ldots, u_n$, $y_1, \ldots, y_m \in \mathbb{N}_1^*$. Assume that $u_1, \ldots, u_n$ are pairwise incomparable with respect to $\sqsubseteq$, and the same holds for $y_1, \ldots, y_m$. Then*

$$\delta_{y_1\cdots y_m}^{\delta_{u_1\cdots u_n}^{w}(v)} \circ \delta_{u_1\cdots u_n}^{w} = \delta_{u_1\cdots u_n}^{\delta_{y_1\cdots y_m}^{v}(w)} \circ \delta_{y_1\cdots y_m}^{v}$$

whenever both sides of the equation are defined.

Proof. The proof rests on the following extensive case analysis.

(I) Let $v \parallel w$. Then, since $\delta_{u_1\cdots u_n}^{w}(v) = v$ and $\delta_{y_1\cdots y_m}^{v}(w) = w$, the equation reduces to

$$\delta_{y_1\cdots y_m}^{v} \circ \delta_{u_1\cdots u_n}^{w} = \delta_{u_1\cdots u_n}^{w} \circ \delta_{y_1\cdots y_m}^{v}\,.$$

Let $z \in \mathbb{N}_1^*$. We distinguish the following cases:

(1) $v \not\sqsubseteq z$ and $w \not\sqsubseteq z$. Then

$$\delta_{y_1\cdots y_m}^{v}\big(\delta_{u_1\cdots u_n}^{w}(z)\big) = \delta_{y_1\cdots y_m}^{v}(z) = z = \delta_{u_1\cdots u_n}^{w}(z) = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(z)\big)\,.$$

(2) $z = viz'$ for some $i \in [m]$ and $z' \in \mathbb{N}_1^*$, and $w \not\sqsubseteq z$. Then we can show that $w \not\sqsubseteq vy_iz'$ as follows: if $w \sqsubseteq vy_iz'$, then this means either that $w \sqsubseteq v$, or that $v \sqsubseteq w$, and both alternatives contradict the assumption that $v \parallel w$.

Therefore,

$$\begin{aligned}\delta_{y_1\cdots y_m}^{v}\big(\delta_{u_1\cdots u_n}^{w}(z)\big) &= \delta_{y_1\cdots y_m}^{v}(viz') = vy_iz' \\ &= \delta_{u_1\cdots u_n}^{w}(vy_iz') = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(viz')\big) = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(z)\big)\,.\end{aligned}$$

(3) $z = wiz'$ for some $i \in [n]$ and $z' \in \mathbb{N}_1^*$, and $v \not\sqsubseteq z$. This case can be proven analogously to the one above, due to the symmetry of the equation.

(4) $z = wiz'$ for some $i \in [n]$ and $z' \in \mathbb{N}_1^*$, and $z = vjz''$ for some $j \in [m]$ and $z'' \in \mathbb{N}_1^*$. Clearly, then either $w \sqsubseteq v$ or $v \sqsubseteq w$, in contradiction to the premise that $v \parallel w$. The equation holds, as anything follows from falsehood.

(II) Let $v = wiv'$ for some $i \in [n]$ and $v' \in \mathbb{N}_1^*$. In this case, the equation reduces to

$$\delta_{y_1\cdots y_m}^{\delta_{u_1\cdots u_n}^{w}(v)} \circ \delta_{u_1\cdots u_n}^{w} = \delta_{u_1\cdots u_n}^{w} \circ \delta_{y_1\cdots y_m}^{v} .$$

Consider $z \in \mathbb{N}_1^*$. The following cases may arise:

(1) $w \not\sqsubseteq z$. Then clearly also $wu_iv' \not\sqsubseteq z$ and $v = wiv' \not\sqsubseteq z$. Thus

$$\delta_{y_1\cdots y_m}^{\delta_{u_1\cdots u_n}^{w}(v)}\big(\delta_{u_1\cdots u_n}^{w}(z)\big) = \delta_{y_1\cdots y_m}^{wu_iv'}(z) = z = \delta_{u_1\cdots u_n}^{w}(z) = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(z)\big).$$

(2) $z = wiz'$ for some $z' \in \mathbb{N}_1^*$, where i was fixed above. There are two subcases:

(a) $v' \not\sqsubseteq z'$. Then

$$\begin{aligned}\delta_{y_1\cdots y_m}^{\delta_{u_1\cdots u_n}^{w}(v)}\big(\delta_{u_1\cdots u_n}^{w}(z)\big) &= \delta_{y_1\cdots y_m}^{wu_iv'}(wu_iz') = wu_iz' \\ &= \delta_{u_1\cdots u_n}^{w}(wiz') = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(wiz')\big) = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(z)\big).\end{aligned}$$

The last but one equation holds because $v' \not\sqsubseteq z'$ implies that $v = wiv' \not\sqsubseteq wiz'$.

(b) $z' = v'\ell z''$ for some $\ell \in [m]$ and $z'' \in \mathbb{N}_1^*$. Then

$$\begin{aligned}\delta_{y_1\cdots y_m}^{\delta_{u_1\cdots u_n}^{w}(v)}\big(\delta_{u_1\cdots u_n}^{w}(z)\big) &= \delta_{y_1\cdots y_m}^{wu_iv'}(wu_iv'\ell z'') = wu_iv'y_\ell z'' \\ &= \delta_{u_1\cdots u_n}^{w}(wiv'y_\ell z'') = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(wiv'\ell z'')\big) = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(z)\big).\end{aligned}$$

(3) $z = wjz'$ for some $j \in [n]$ and $z' \in \mathbb{N}_1^*$, and $i \neq j$. This implies that $wu_iv' \parallel wu_jz'$, which can be shown by contradiction as follows. First, assume that $wu_iv' \sqsubseteq wu_jz'$. Then $u_iv' \sqsubseteq u_jz'$, and in particular, $u_i \sqsubseteq u_jz'$. Thus, either $u_i \sqsubseteq u_j$ or $u_j \sqsubseteq u_i$, both in contradiction to the assumption that $u_1, \ldots, u_n$ are pairwise incomparable. So $wu_iv' \not\sqsubseteq wu_jz'$. One can show in the same manner that the assumption $wu_jz' \sqsubseteq wu_iz'$ leads to absurdity, too. Therefore $wu_iv' \parallel wu_jz'$.

We conclude

$$\begin{aligned}\delta_{y_1\cdots y_m}^{\delta_{u_1\cdots u_n}^{w}(v)}\big(\delta_{u_1\cdots u_n}^{w}(z)\big) &= \delta_{y_1\cdots y_m}^{wu_iv'}(wu_jz') = wu_jz' \\ &= \delta_{u_1\cdots u_n}^{w}(wjz') = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(wjz')\big) = \delta_{u_1\cdots u_n}^{w}\big(\delta_{y_1\cdots y_m}^{v}(z)\big).\end{aligned}$$

The last but one equation holds because $i \neq j$, and therefore $v = wiv' \not\sqsubseteq wjz'$.

(III) Let $w = viw'$ for some $i \in [m]$ and $w' \in \mathbb{N}_1^*$. The proof for this case is analogous to the one for **(II)**, due to the symmetry of the examined equation.

* * *

We have covered all instances where both sides of the equation are defined. Therefore the lemma is proven. □

The second auxiliary lemma deals with links introduced from the right-hand side of a production.

Lemma 5.8. *Let $n, m \in \mathbb{N}$ and $v, w, u_1, \ldots, u_n, y_1, \ldots, y_m \in \mathbb{N}_1^*$. Let moreover $z \in \mathbb{N}_1^*$ such that there is no $j \in [n]$ with $u_j \sqsubseteq z$. Then*

$$\delta_{y_1 \cdots y_m}^{\delta_{u_1 \cdots u_n}^{w}(v)}(wz) = \delta_{y_1 \cdots y_m}^{v}(w) \cdot z$$

whenever both sides of the equation are defined.

Proof. We proceed with a case analysis, according to the definition of the family of functions δ.

(I) Let $w \not\sqsubseteq v$ and $v \not\sqsubseteq wz$. Then

$$\delta_{y_1 \cdots y_m}^{\delta_{u_1 \cdots u_n}^{w}(v)}(wz) = \delta_{y_1 \cdots y_m}^{v}(wz) = wz = \delta_{y_1 \cdots y_m}^{v}(w) \cdot z\,.$$

The last equation holds since $v \not\sqsubseteq wz$ implies that $v \not\sqsubseteq w$.

(II) Let $w \not\sqsubseteq v$ and $wz = viw'$ for some $i \in [m]$ and $w' \in \mathbb{N}_1^*$. Then there is some $\tilde{w} \in \mathbb{N}_1^*$ such that $w = vi\tilde{w}$. Hence

$$\delta_{y_1 \cdots y_m}^{\delta_{u_1 \cdots u_n}^{w}(v)}(wz) = \delta_{y_1 \cdots y_m}^{v}(vi\tilde{w}z) = vy_i\tilde{w}z = \delta_{y_1 \cdots y_m}^{v}(vi\tilde{w}) \cdot z = \delta_{y_1 \cdots y_m}^{v}(w) \cdot z\,.$$

(III) Let $v = wiv'$ for some $v' \in \mathbb{N}_1^*$ and $i \in [n]$, and $wu_iv' \not\sqsubseteq wz$. Then

$$\delta_{y_1 \cdots y_m}^{\delta_{u_1 \cdots u_n}^{w}(v)}(wz) = \delta_{y_1 \cdots y_m}^{wu_iv'}(wz) = wz = \delta_{y_1 \cdots y_m}^{v}(w) \cdot z\,,$$

where the last equation holds since the assumption $v = wiv'$ implies that $v \not\sqsubseteq w$.

(IV) Let $v = wiv'$ for some $v' \in \mathbb{N}_1^*$ and $i \in [n]$, and let $wz = wu_iv'jv''$ for some $v'' \in \mathbb{N}_1^*$ and $j \in [m]$. As $wz = wu_iv'jv''$ implies that $z = u_iv'jv''$, we obtain that $u_i \sqsubseteq z$, a contradiction to the lemma's premises. From falsehood, anything follows – the equation holds also in this case.

* * *

As the above cases encompass all situations where both sides of the equation are defined, the case analysis implies that the lemma is correct. □

With these two auxiliary lemmas, we can prove the following production interchange lemma for wscftg.

Lemma 5.9. *Let $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$ be a wscftg, consider productions p_1 and p_2 of G, let n_1, $n_2 \in \mathbb{N}$, and let (ξ, ζ, λ), $(\xi_1, \zeta_1, \lambda_1)$, and $(\xi_2, \zeta_2, \lambda_2) \in \mathrm{S}(N, \Sigma, \Delta)_{n_1, n_2}$. Assume that p_1 and p_2 are of the respective forms*

$$(A_1 \cdot \mathrm{Id}_{k_1}, A_2 \cdot \mathrm{Id}_{k_2}) \to (\varrho_1, \varrho_2, \tilde{\lambda}) \qquad \textit{and} \qquad (B_1 \cdot \mathrm{Id}_{\ell_1}, B_2 \cdot \mathrm{Id}_{\ell_2}) \to (\varrho'_1, \varrho'_2, \tilde{\lambda}').$$

Let $w_1 \in \mathrm{pos}(\xi)$, $w_2 \in \mathrm{pos}(\zeta)$, $v'_1 \in \mathrm{pos}(\xi_1)$, and $v'_2 \in \mathrm{pos}(\zeta_1)$ such that

$$v'_1 \notin w_1 \cdot \mathrm{pos}_{N \cup \Sigma}(\varrho_1), \qquad v'_2 \notin w_2 \cdot \mathrm{pos}_{N \cup \Sigma}(\varrho_2), \tag{5.1}$$

and

$$\big(\xi, \zeta, \lambda\big) \xRightarrow{(w_1, w_2)}_{p_1} \big(\xi_1, \zeta_1, \lambda_1\big) \xRightarrow{(v'_1, v'_2)}_{p_2} \big(\xi_2, \zeta_2, \lambda_2\big).$$

Then there is some $(\xi'_1, \zeta'_1, \lambda'_1) \in \mathrm{S}(N, \Sigma, \Delta)_{n_1, n_2}$ such that

$$(\xi, \zeta, \lambda) \Rightarrow_{p_2} (\xi'_1, \zeta'_1, \lambda'_1) \Rightarrow_{p_1} (\xi_2, \zeta_2, \lambda_2).$$

Proof. The proof idea can be summarized as follows. Since in the assumed derivation, p_2 is applied after p_1, the positions $(v'_1, v'_2) \in \lambda_1$ where p_2 is applied may have been displaced during the application of p_1 (by a function from the family δ). We reconstruct the values of these positions before the application of p_1 – let us call them (v_1, v_2). Next, we apply the production p_2 at (v_1, v_2). Of course, during this, the link (w_1, w_2) might be displaced – therefore, we determine its value (w'_1, w'_2) after displacement. It remains to apply p_1 at (w'_1, w'_2), and to show that the result is equal to $(\xi_2, \zeta_2, \lambda_2)$. For this purpose, we make use of the properties from Lemmas 5.7 and 5.8.

Formally, we proceed in the following way. Let, for each $j \in [2]$,

$$u^j_i = \mathrm{pos}_{x_i}(\varrho_j) \quad \text{for each } i \in [k_j] \qquad \text{and} \qquad y^j_i = \mathrm{pos}_{x_i}(\varrho'_j) \quad \text{for each } i \in [\ell_j],$$

and define

$$v_j = \begin{cases} v'_j & \text{if } w_j \not\sqsubseteq v'_j \\ w_j i z & \text{if there are } i \in \mathbb{N}_1,\, z \in \mathbb{N}_1^* \text{ with } v'_j = w_j u^j_i z. \end{cases}$$

Note that v_j is well-defined, due to the assumptions in (5.1). Moreover, it is easy to check that

$$(v'_1, v'_2) = \Big(\delta^{w_1}_{u^1_1 \cdots u^1_{k_1}}(v_1), \delta^{w_2}_{u^2_1 \cdots u^2_{k_2}}(v_2)\Big). \tag{5.2}$$

Let $(\xi'_1, \zeta'_1, \lambda'_1) \in \mathrm{S}(N, \Sigma, \Delta)_{n_1, n_2}$ such that

$$\big(\xi, \zeta, \lambda\big) \xRightarrow{(v_1, v_2)}_{p_2} \big(\xi'_1, \zeta'_1, \lambda'_1\big).$$

Next, define the positions w'_1, $w'_2 \in \mathbb{N}_1^*$ by

$$(w'_1, w'_2) = \Big(\delta^{v_1}_{y^1_1 \cdots y^1_{\ell_1}}(w_1), \delta^{v_2}_{y^2_1 \cdots y^2_{\ell_2}}(w_2)\Big). \tag{5.3}$$

Let $(\xi_2', \zeta_2', \lambda_2') \in \mathrm{S}(N, \Sigma, \Delta)_{n_1,n_2}$ such that

$$\left(\xi_1', \zeta_1', \lambda_1'\right) \xRightarrow{(w_1',w_2')}_{p_1} \left(\xi_2', \zeta_2', \lambda_2'\right).$$

Using the positions w_1, w_2, v_1, and v_2 to decompose the sentential forms as in the proof of Lemma 2.12, it is not hard to see that $\xi_2 = \xi_2'$ and $\zeta_2 = \zeta_2'$. We must still show that $\lambda_2 = \lambda_2'$. In fact, we will only show that $\lambda_2 \subseteq \lambda_2'$, as the other direction is analogous due to the inherent symmetry of the property we are showing.

So let us consider some element $(z_1, z_2) \in \lambda_2$. We must cover the following three cases.

(I) The link (z_1, z_2) was introduced by the production p_2. Formally, $(z_1, z_2) = (v_1' z_1', v_2' z_2')$ for some link $(z_1', z_2') \in \tilde{\lambda}'$. Then $(v_1 z_1', v_2 z_2') \in \lambda_1'$, and therefore

$$\left(\delta^{w_1'}_{u_1^1 \cdots u_{k_1}^1}(v_1 z_1'), \delta^{w_2'}_{u_2^1 \cdots u_{k_2}^2}(v_2 z_2')\right) \in \lambda_2'.$$

Since $z_1' \in \mathrm{pos}_N(\varrho_1')$ and analogously for z_2', we can apply Lemma 5.8. With the additional help of equations (5.2) and (5.3), we obtain

$$\left(\delta^{w_1'}_{u_1^1 \cdots u_{k_1}^1}(v_1 z_1'), \delta^{w_2'}_{u_2^1 \cdots u_{k_2}^2}(v_2 z_2')\right) = \left(\delta^{w_1}_{u_1^1 \cdots u_{k_1}^1}(v_1) \cdot z_1', \delta^{w_2}_{u_1^2 \cdots u_{k_2}^2}(v_2) \cdot z_2'\right) = \left(v_1' z_1', v_2' z_2'\right).$$

The latter tuple equals (z_1, z_2), and therefore $(z_1, z_2) \in \lambda_2'$.

(II) The link (z_1, z_2) was introduced by the production p_1. Formally, there is $(z_1', z_2') \in \tilde{\lambda}$ such that $(w_1 z_1', w_2 z_2') \in \lambda_1$ and

$$(z_1, z_2) = \left(\delta^{v_1'}_{y_1^1 \cdots y_{\ell_1}^1}(w_1 z_1'), \delta^{v_2'}_{y_1^2 \cdots y_{\ell_2}^2}(w_2 z_2')\right).$$

Then $(w_1' z_1', w_2' z_2') \in \lambda_2'$. Because $z_1' \in \mathrm{pos}_N(\varrho_1)$, and analogously for z_2', we may apply Lemma 5.8, and because of (5.2) and (5.3),

$$(w_1' z_1', w_2' z_2') = \left(\delta^{v_1}_{y_1^1 \cdots y_{\ell_1}^1}(w_1) \cdot z_1', \delta^{v_2}_{y_1^2 \cdots y_{\ell_2}^2}(w_2) \cdot z_2'\right) = \left(\delta^{v_1'}_{y_1^1 \cdots y_{\ell_1}^1}(w_1 z_1'), \delta^{v_2'}_{y_1^2 \cdots y_{\ell_2}^2}(w_2 z_2')\right),$$

hence $(z_1, z_2) \in \lambda_2'$.

(III) The link (z_1, z_2) originates in λ. Formally, there is some link $(z_1', z_2') \in \lambda$ such that

$$(z_1, z_2) = \left(\delta^{v_1'}_{y_1^1 \cdots y_{\ell_1}^1}\left(\delta^{w_1}_{u_1^1 \cdots u_{k_1}^1}(z_1')\right), \delta^{v_2'}_{y_1^2 \cdots y_{\ell_2}^2}\left(\delta^{w_2}_{u_1^2 \cdots u_{k_2}^2}(z_2')\right).\right)$$

Consider the link $(\hat{z}_1, \hat{z}_2) \in \lambda_2'$ given by

$$(\hat{z}_1, \hat{z}_2) = \left(\delta^{w_1'}_{u_1^1 \cdots u_{k_1}^1}\left(\delta^{v_1}_{y_1^1 \cdots y_{\ell_1}^1}(z_1')\right), \delta^{w_2'}_{u_1^2 \cdots u_{k_2}^2}\left(\delta^{v_2}_{y_1^2 \cdots y_{\ell_2}^2}(z_2')\right)\right).$$

Observe that Lemma 5.7 is applicable here, because the positions of the leaves of a tree are always pairwise incomparable with respect to the prefix order. By (5.2) and (5.3), together with Lemma 5.7, we obtain $(z_1, z_2) = (\hat{z}_1, \hat{z}_2)$, and therefore $(z_1, z_2) \in \lambda_2'$. This concludes the lemma's proof. □

5.1.1 Simple Synchronous Context-Free Tree Grammars

As mentioned in the chapter's introduction, there is a restriction for synchronous context-free word grammars, called *simple,* such that the transformations of simple synchronous cfg are precisely those of pushdown automata with output. In our generalization of this characterization, we will start out with defining what it means for a wscftg to be simple. Afterwards, we will introduce the concept of *characterizing tree homomorphisms,* which will later enable us to obtain a normal form for simple wscftg.

Before we define simple wscftg, we introduce the following abbreviation. Let $n \in \mathbb{N}$ and consider the synchronized trees $(\xi, \zeta, \lambda) \in \mathrm{S}(N, \Sigma, \Delta)_{n,n}$. We define

$$\hat{\lambda} = \lambda \cup \left\{ \left(\mathrm{pos}_{x_j}(\xi), \mathrm{pos}_{x_j}(\zeta)\right) \,\middle|\, j \in [n] \right\}.$$

Intuitively, $\hat{\lambda}$ also encompasses links between variables: we assume that an occurrence of x_j in ξ is linked to an occurrence of x_j in ζ. Observe that $\hat{\lambda}$ is a bijection between $\mathrm{pos}_{N \cup X}(\xi)$ and $\mathrm{pos}_{N \cup X}(\zeta)$, as both ξ and ζ are linear and nondeleting.

This abbreviation allows us to define simple wscftg. Let $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$ be a wscftg. Then G is called *simple* if it has initial nonterminals and for every production of G, say of form

$$(A_1 \cdot \mathrm{Id}_{n_1}, A_2 \cdot \mathrm{Id}_{n_2}) \to (\varrho_1, \varrho_2, \lambda),$$

the following conditions are fulfilled.

(i) We have $n_1 = n_2$, and for every $(v, w) \in \lambda$, $\mathrm{rk}(\varrho_1(v)) = \mathrm{rk}(\varrho_2(w))$.

(ii) For every (v_1, w_1), $(v_2, w_2) \in \hat{\lambda}$, and $i \in \mathbb{N}_1$,

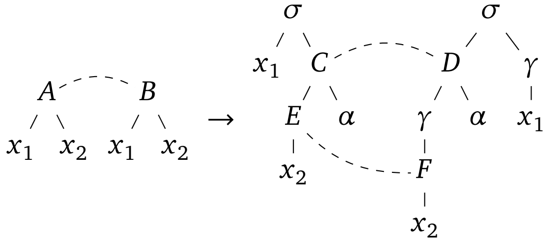
$$v_1 i \sqsubseteq v_2 \qquad \text{if and only if} \qquad w_1 i \sqsubseteq w_2\,.$$

Intuitively, condition *(ii)* demands that the ancestor relations of occurrences of nonterminals or variables in ϱ_1 and ϱ_2 must be compatible with the links in $\hat{\lambda}$: if a nonterminal (or variable) U in ϱ_1 occurs in the i-th subtree of another nonterminal A, then the same must hold for the nonterminals (or variables) A' and U' in ϱ_2 that are linked to A and U, and vice versa. Of course, a nonterminal will never occur as the descendant of a variable.

Example 5.10. Consider the ranked alphabets

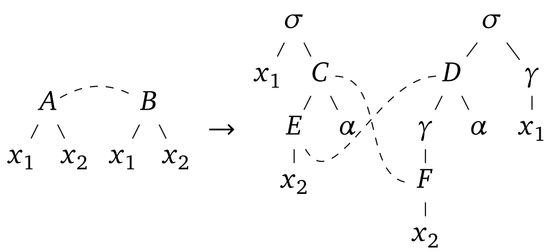
$$N = \left\{A^{(2)}, B^{(2)}, C^{(2)}, D^{(2)}, E^{(1)}, F^{(1)}\right\} \qquad \text{and} \qquad \Sigma = \Delta = \left\{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\right\}.$$

The wscftg production

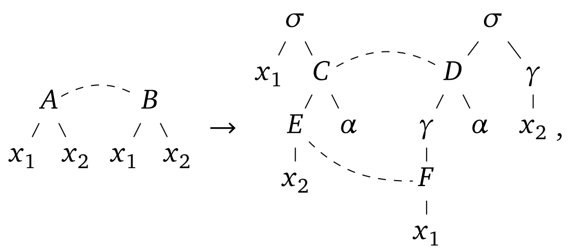


is a production of a simple wscftg with nonterminal alphabet N and terminal alphabets Σ and Δ, since the nonterminal E occurs in the first subtree of C on the input side, just as F occurs in the first subtree of D on the output side. Moreover, the variable x_2 occurs in both sides as the first child of E, resp. of F.

But the wscftg production



is not a production of a simple wscftg, as E occurs below C, but D occurs above F. Nor is the wscftg production



as x_1 occurs beneath no nonterminal in the input, but under F (and D) in the output side. $\lhd$

As the following lemma shows, simpleness is a proper restriction on the power of wscftg.

Lemma 5.11. *There is a context-free weighted tree transformation τ for which there is no simple wscftg G with $[\![G]\!] = \tau$.*

Proof. It is easy to see that wscftg with monadic nonterminal and terminal alphabets (and over the Boolean semiring) correspond precisely to synchronous cfg (resp. to syntax directed translations, as defined in [5]), by the isomorphism between monadic trees and words. Moreover, simple wscftg correspond to simple synchronous cfg.

As proven in [5, Thm. 2], the word transformation

$$\big\{(wcv, vcw) \;\big|\; v, w \in \{a, b\}^*\big\} \subseteq \Sigma^* \times \Sigma^*,$$

where $\Sigma = \{a, b, c\}$, can be generated by a synchronous cfg, but not by any simple one. By the correspondence that was just outlined, we can transfer this result to wscftg, proving the lemma. $\square$

Remark 5.12. It is also quite easy to see that the tree transformation τ given in Example 5.5 cannot be computed by a simple wscftg. The reason is that in simple wscftg, linked nonterminals must be of identical ranks, and moreover they can only generate linear and nondeleting trees. Thus we cannot translate the monadic tree $ww^R\#$ into the non-monadic tree $\sigma(a^{|w|_a}\#, b^{|w|_b}\#)$.

Curiously enough, there is a simple wscftg over the Boolean semiring that generates a weighted tree transformation whose support is

$$\tau' = \left\{ \Big(w\sigma(\#,\, w^R\#),\, \sigma(a^{|w|_a}\#,\, b^{|w|_b}\#) \Big) \,\Big|\, w \in \{a,b\}^* \right\}.$$

These two tree transformations are closely related – we have $\tau = h^{-1}\,;\tau'$ for the alphabetic tree homomorphism h that maps $\sigma(x_1,x_2)$ to x_2 and is the identity everywhere else. $\triangleleft$

If a nonterminal or variable in a production of a simple wscftg is not dominated by any other nonterminal, then the same holds for the symbol that is linked to it. This observation is expressed formally in the following lemma.

Lemma 5.13. *Consider a production* $(A_1 \cdot \mathrm{Id}_n, A_2 \cdot \mathrm{Id}_n) \to (\varrho_1, \varrho_2, \lambda)$ *of a simple wscftg. Denote the set of minimal elements of* $\mathrm{pos}_{N \cup X}(\varrho_1)$ *with respect to* $\sqsubseteq$ *by* M_1*, and analogously for* ϱ_2 *and* M_2*. Then* $\hat{\lambda} \cap M_1 \times M_2$ *is a bijection.*

Proof. Assume to the contrary that $\hat{\lambda} \cap (M_1 \times M_2)$ is not a bijection. As $\hat{\lambda}$ is a bijection on $\mathrm{pos}_{N\cup X}(\varrho_1) \times \mathrm{pos}_{N\cup X}(\varrho_2)$, this means that there is some tuple $(v,w) \in \hat{\lambda}$ such that one of its components is minimal, while the other one is not. Let us suppose without loss of generality that there are $v \in \mathrm{pos}_{N\cup X}(\varrho_1) \setminus M_1$ and $w \in M_2$ such that $(v,w) \in \hat{\lambda}$. As v is not minimal, there is another element $v' \in M_1$ with $v' \sqsubset v$. Hence, there is an $i \in \mathbb{N}_1$ such that $v'i \sqsubseteq v$. But by condition *(ii)* from above, the position $w' \in \mathrm{pos}_{N\cup X}(\varrho_2)$ with $(v',w') \in \hat{\lambda}$ satisfies $w'i \sqsubseteq w$ and therefore $w' \sqsubset w$. This stands in contradiction to the minimality of w and proves the lemma. $\square$

We continue with defining a concept called characterizing homomorphisms. This concept captures the notion that in a production of a simple wscftg, the ancestor relations between nonterminals and variables on the input and the output side are identical. These ancestor relations are encoded into a tree ζ, which is mapped by linear and nondeleting tree homomorphisms h_1 and h_2 to the production's input, resp. output side. The following definition generalizes the one of characterizing homomorphisms for the word case, introduced in [5].[1]

Consider a wscftg $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$, two disjoint ranked alphabets M and Γ, and linear and nondeleting tree homomorphisms

$$h_1 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Sigma) \qquad \text{and} \qquad h_2 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Delta)\,.$$

We say that G is *characterized* by h_1 and h_2 if

$$h_1(M) \subseteq N\,, \quad h_2(M) \subseteq N\,, \quad h_1(\Gamma) \subseteq \widetilde{\mathrm{T}}(\Sigma)\,, \quad \text{and} \quad h_2(\Gamma) \subseteq \mathrm{T}(\Delta)\,, \tag{5.4}$$

and for every production $(A_1, A_2) \to (\varrho_1, \varrho_2, \lambda)$ in P, there is a linear and nondeleting tree $\varrho \in \mathrm{T}(M \cup \Gamma)$[1] such that

$$|\varrho| > 0\,, \qquad h_1(\varrho) = \varrho_1\,, \qquad \text{and} \qquad h_2(\varrho) = \varrho_2\,.$$

As the following lemma shows, simple wscftg can be characterized by homomorphisms.

[1]There, the authors use the terminology "characterized by a context-free grammar" instead.

Lemma 5.14. *Consider a wscftg $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$. If G is simple, then there are ranked alphabets M and Γ, as well as linear and nondeleting tree homomorphisms*

$$h_1 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Sigma) \qquad \text{and} \qquad h_2 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Delta)$$

that characterize G. Moreover, for every $n \in \mathbb{N}$, the function

$$\langle h_1, h_2 \rangle \colon M^{(n)} \to N^{(n)} \times N^{(n)}, \qquad A \mapsto \big(h_1(A), h_2(A)\big)$$

is a bijection.

Proof. Define the ranked alphabet M such that for every $n \in \mathbb{N}$, $M^{(n)} = N^{(n)} \times N^{(n)}$. We will show for every $n \in \mathbb{N}$ and $(\xi_1, \xi_2, \lambda) \in \mathrm{S}(N, \Sigma, \Delta)_{n,n}$ that if (ξ_1, ξ_2, λ) may occur on the right-hand side of a production of some simple wscftg,[2] then there are a ranked alphabet Γ, linear and nondeleting tree homomorphisms

$$h_1 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Sigma) \qquad \text{and} \qquad h_2 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Delta)$$

that satisfy (5.4), and a linear and nondeleting tree $\zeta \in \mathrm{T}(M \cup \Gamma)^1_n$ such that $|\zeta| > 0$, $h_1(\zeta) = \xi_1$, and $h_2(\zeta) = \xi_2$.

The proof is by complete induction on $|\xi_1| + |\xi_2|$. For the induction base, assume that $\xi_1 = \xi_2 = x_1$. We let $\Gamma = \{\gamma^{(1)}\}$ and $\zeta = \gamma(x_1)$, for some symbol γ. Further, we define

$$h_1 \colon \gamma \mapsto x_1\,, \qquad h_2 \colon \gamma \mapsto x_1\,,$$

and

$$h_1 \colon (A_1, A_2) \mapsto A_1\,, \qquad h_2 \colon (A_1, A_2) \mapsto A_2 \tag{5.5}$$

for every $(A_1, A_2) \in M$. It is easy to see then that $\langle h_1, h_2 \rangle$ is a bijection on $M^{(n)}$ for each $n \in \mathbb{N}$. Moreover, clearly $h_1(\zeta) = \xi_1$ and $h_2(\zeta) = \xi_2$.

For the induction step, there are two cases.

(I) For the first case, assume that $\xi_1(\varepsilon)$ and $\xi_2(\varepsilon) \in N$ – say $\xi_1(\varepsilon) = A_1$ and $\xi_2(\varepsilon) = A_2$ for some $k \in \mathbb{N}$, and $A_1, A_2 \in N^{(k)}$. Since the variables in the subtrees of A_1 and A_2 must obey condition *(ii)* from the definition of simple wscftg, we can write

$$\xi_1 = A_1 \cdot (\xi^1_1 \otimes \cdots \otimes \xi^1_k) \cdot \vartheta \qquad \text{and} \qquad \xi_2 = A_2 \cdot (\xi^2_1 \otimes \cdots \otimes \xi^2_k) \cdot \vartheta$$

for some linear and nondeleting trees $\xi^1_1, \ldots, \xi^1_k \in \mathrm{T}(N \cup \Sigma)^1$, $\xi^2_1, \ldots, \xi^2_k \in \mathrm{T}(N \cup \Delta)^1$, and a linear and nondeleting torsion $\vartheta \in \Theta^n_n$. In particular, for every $j \in [k]$,

$$\big(\xi^1_j, \xi^2_j, \lambda_j\big) \in \mathrm{S}(N, \Sigma, \Delta)\,, \qquad \text{where} \qquad \lambda_j = \big\{(w_1, w_2) \in \mathbb{N}^*_1 \times \mathbb{N}^*_1 \;\big|\; (jw_1, jw_2) \in \lambda\big\}\,.$$

By the induction hypothesis, there are alphabets $\Gamma_1, \ldots, \Gamma_k$, tree homomorphisms $h^1_1, \ldots, h^k_1$, $h^2_1, \ldots, h^2_k$, and linear and nondeleting trees $\zeta_1, \ldots, \zeta_k \in \mathrm{T}(M \cup \Gamma)^1$ such that $h^1_j(\zeta_j) = \xi^1_j$ and $h^2_j(\zeta_j) = \xi^2_j$ for every $j \in [k]$. Observe that all these homomorphisms are defined identically

[2] That is, if (ξ_1, ξ_2, λ) fulfills the conditions *(i)* and *(ii)* from 181.

on M, as given in (5.5). Moreover, we can assume $\Gamma_1, \ldots, \Gamma_k$ to be pairwise disjoint without loss of generality. Therefore,

$$h_1 = h_1^1 \cup \cdots \cup h_k^1 \qquad \text{and} \qquad h_2 = h_1^2 \cup \cdots \cup h_k^2$$

are again tree homomorphisms. We let

$$\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_k \qquad \text{and} \qquad \zeta = (A_1, A_2) \cdot (\zeta_1 \otimes \cdots \otimes \zeta_k) \cdot \vartheta\,,$$

then clearly $h_1(\zeta) = \xi_1$ and $h_2(\zeta) = \xi_2$.

(II) In the other case, at least one of ξ_1 and ξ_2 has a terminal symbol at its root. We will cut away the maximal subtrees that contain only terminal symbols from the tops of ξ_1 and ξ_2, and represent them by a new symbol in Γ.

For this purpose, let, for each $j \in [2]$, M_j be the set of positions in $\mathrm{pos}_{N \cup X}(\xi_j)$ that are minimal with respect to $\sqsubseteq$. By Lemma 5.13, we know that there is some $k \in \mathbb{N}$ such that

$$\hat{\lambda} \cap (M_1 \times M_2) = \big\{(v_1, w_1), \ldots, (v_k, w_k)\big\}$$

is a bijection between M_1 and M_2. We can assume without loss of generality that the positions v_j are ordered left-to-right, i.e., $v_1 <_{\mathrm{lex}} \cdots <_{\mathrm{lex}} v_k$. We set

$$\tilde{s} = \xi_1[x_1]_{v_1} \cdots [x_k]_{v_k} \qquad \text{and} \qquad t = \xi_2[x_1]_{w_1} \cdots [x_k]_{w_k}\,.$$

In particular, if $k = 0$, then $\tilde{s} = \xi_1$ and $t = \xi_2$. Clearly, $\tilde{s} \in \widetilde{\mathrm{T}}(\Sigma)_k^1$, and $t \in \mathrm{T}(\Delta)_k^1$ is linear and nondeleting. By condition *(ii)* from the definition of simple wscftg, there are linear and nondeleting trees $\xi_1^1, \ldots, \xi_k^1 \in \mathrm{T}(N \cup \Sigma)^1$, $\xi_1^2, \ldots, \xi_k^2 \in \mathrm{T}(N \cup \Delta)^1$, and a linear and nondeleting torsion $\vartheta \in \Theta_n^n$ such that

$$\xi_1 = \tilde{s} \cdot (\xi_1^1 \otimes \cdots \otimes \xi_k^1) \cdot \vartheta \qquad \text{and} \qquad \xi_2 = t \cdot (\xi_1^2 \otimes \cdots \otimes \xi_k^2) \cdot \vartheta\,,$$

and moreover, for every $j \in [k]$,

$$\Big(\xi_j^1, \xi_j^2, \lambda_j\Big) \in \mathrm{S}(N, \Sigma, \Delta)\,, \qquad \text{where} \qquad \lambda_j = \Big\{(v'_j, w'_j) \in \mathbb{N}_1^* \times \mathbb{N}_1^* \;\Big|\; (v_j v'_j, w_j w'_j) \in \lambda\Big\}\,.$$

Since $|\tilde{s}| + |t| > 0$, we can apply the induction hypothesis. By the argument from above, there are disjoint alphabets $\Gamma_1, \ldots, \Gamma_k$, homomorphisms $h_1^1, \ldots, h_1^k$, $h_1^2, \ldots, h_k^2$, and linear and nondeleting trees $\zeta_1, \ldots, \zeta_k \in \mathrm{T}(M \cup \Gamma)^1$ such that

$$h_j^1(\zeta_j) = \xi_j^1 \qquad \text{and} \qquad h_j^2(\zeta_j) = \xi_j^2 \qquad \text{for every} \quad j \in [k]\,.$$

Let $\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_k \cup \{\gamma^{(k)}\}$, where γ is a distinct symbol. Moreover, let

$$h_1 = h_1^1 \cup \cdots \cup h_k^1 \cup \big\{(\gamma, \tilde{s})\big\} \qquad \text{and} \qquad h_2 = h_1^2 \cup \cdots \cup h_k^2 \cup \big\{(\gamma, t)\big\}\,.$$

Again, $h_1 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Sigma)$ and $h_2 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Delta)$ are tree homomorphisms, and fulfill the conditions from (5.5). We set

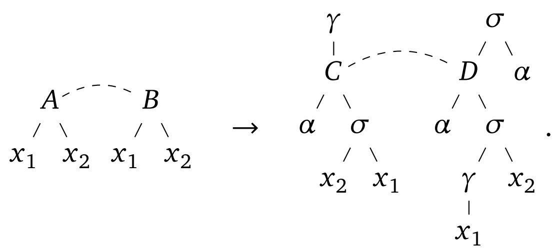
$$\zeta = \gamma \cdot (\zeta_1 \otimes \cdots \otimes \zeta_k) \cdot \vartheta\,,$$

and it is easy to check that then $h_1(\zeta) = \xi_1$ and $h_2(\zeta) = \xi_2$.

* * *

In order to finish the proof, we apply the property we just established to every production of G. Then we obtain $|P|$ tree homomorphisms. We may assume without loss of generality that their domains only intersect on M. On this intersection, they are defined identically, as seen in (5.5). The union of all these tree homomorphisms is the tree homomorphism that characterizes G. $\square$

Example 5.15. Consider ranked alphabets $N = \{A^{(2)}, B^{(2)}, C^{(2)}, D^{(2)}\}$, $\Sigma = \Delta = \{\sigma^{(2)}, \gamma^{(1)}\}$, and a production $(A,B) \to (\varrho_1, \varrho_2, \lambda)$ of a simple wscftg with nonterminals from N and terminals from Σ and Δ, of the form



Let $M = \big\{(A,B)^{(2)}, (C,D)^{(2)}\big\}$, and let

$$\Gamma = \big\{U^{(2)},\ V^{(1)},\ W^{(0)}, Y^{(1)}\big\}\,.$$

For simplicity's sake, let us assume that the tree homomorphisms h_1 and h_2 which characterize G only depend on the production $(A,B) \to (\varrho_1, \varrho_2, \lambda)$. This is the case when it is the only production of G. Then

$$h_1 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Sigma) \qquad \text{and} \qquad h_2 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Delta)$$

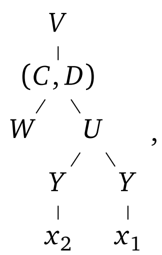
are given by

$$h_1 \colon (A,B) \mapsto A\,, \quad (C,D) \mapsto C\,, \quad U \mapsto \sigma(x_1, x_2)\,, \qquad V \mapsto \gamma(x_1)\,, \qquad W \mapsto \alpha\,, \quad Y \mapsto x_1\,,$$

and

$$h_2 \colon (A,B) \mapsto B\,, \quad (C,D) \mapsto D\,, \quad U \mapsto \sigma(\gamma(x_2), x_1)\,, \quad V \mapsto \sigma(x_1, \alpha)\,, \quad W \mapsto \alpha\,, \quad Y \mapsto x_1\,.$$

When we consider the tree ζ of the form



then it is easy to check that $h_1(\zeta) = \varrho_1$ and $h_2(\zeta) = \varrho_2$. $\triangleleft$

5.1.2 Simple Synchronous Context-Free Tree Grammars in Normal Form

Characterization by homomorphisms leads naturally to the following normal form for simple wscftg. We say that a simple wscftg $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$ is in *normal form* if

$$\xi_0 = \big(S, S, \{\varepsilon, \varepsilon\}\big) \qquad \text{for some} \qquad S \in N^{(0)}$$

and for every production of the form, say,

$$(A \cdot \mathrm{Id}_n,\, B \cdot \mathrm{Id}_n) \to (\varrho_1, \varrho_2, \lambda)$$

in P, we have $A = B$ and either

(i) $\varrho_1 \in \widetilde{\mathrm{T}}(\Sigma)_n^1$ and $\varrho_2 \in \mathrm{T}(\Delta)_n^1$, or

(ii) $\varrho_1 = \varrho_2 = \varrho$ for some linear and nondeleting tree $\varrho \in \mathrm{T}(N)_n^1$ with $|\varrho| > 0$.

We will call productions of form *(i) terminal productions*, and those of form *(ii)* will be called *nonterminal productions*. Observe that case *(i)* implies that $\lambda = \emptyset$, while in case *(ii)* the fact that G is simple implies that $\lambda = \mathrm{id}_{\mathrm{pos}_N(\varrho)}$.

Lemma 5.16. *For every simple wscftg G, there is a simple wscftg G' in normal form such that $[\![G]\!] = [\![G']\!]$.*

Proof. Consider a simple wscftg $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$, with $\xi_0 = (S_1, S_2, \{(\varepsilon, \varepsilon)\})$ for some S_1, $S_2 \in N$. By Lemma 5.14, there are disjoint ranked alphabets M and Γ, and tree homomorphisms $h_1 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Sigma)$ and $h_2 \colon \mathrm{T}(M \cup \Gamma) \to \mathrm{T}(N \cup \Delta)$ that characterize G. Moreover, $\langle h_1, h_2 \rangle \colon M^{(n)} \to N^{(n)} \times N^{(n)}$ is a bijection for every $n \in \mathbb{N}$.

We construct the simple wscftg $G' = (N', K, \Sigma, \Delta, \xi_0', P' \cup P'', wt')$ as follows. Let $N' = M \cup \Gamma$ and $\xi_0' = (S', S', \{\varepsilon\})$, where $S' = \langle h_1, h_2 \rangle^{-1}(S_1, S_2)$. Observe that S' is well-defined due to the bijectivity of $\langle h_1, h_2 \rangle$.

Define P' to be the smallest set that satisfies the following. For every production $p \in P$ of the form

$$(A_1 \cdot \mathrm{Id}_n,\, A_2 \cdot \mathrm{Id}_n) \to (\varrho_1, \varrho_2, \lambda)\,,$$

we know that there is a tree $\zeta \in \mathrm{T}(M \cup \Gamma)_n^1$ such that $h_1(\zeta) = \varrho_1$ and $h_2(\zeta) = \varrho_2$. Then P' contains the production p' of form

$$(A \cdot \mathrm{Id}_n,\, A \cdot \mathrm{Id}_n) \to (\zeta, \zeta, \lambda')\,,$$

where $A = \langle h_1, h_2 \rangle^{-1}(A_1, A_2)$, and $\lambda' = \mathrm{id}_{\mathrm{pos}_M(\zeta)}$. Again, A is well-defined. We set $wt(p') = wt(p)$.

Furthermore, P'' is the smallest set that contains, for every $k \in \mathbb{N}$ and $\gamma \in \Gamma^{(k)}$, the production p of the form

$$(\gamma \cdot \mathrm{Id}_k, \gamma \cdot \mathrm{Id}_k) \to \big(h_1(\gamma), h_2(\gamma), \emptyset\big)\,,$$

with $wt'(p) = 1$.

* * *

The proof that $[\![G]\!] = [\![G']\!]$ is both technical and unsurprising. Therefore, we will content ourselves with giving a rough overview.

Observe that every production of G can be simulated by a derivation which consists of one production from P' and a finite number of productions from P'', which replace all the symbols from Γ by trees from $\widetilde{\mathrm{T}}(\Sigma)^1$ and $\mathrm{T}(\Delta)^1$, respectively. Moreover, whenever there is such a sequence of productions, there is a corresponding production in P.

To formalize this observation, consider the tree homomorphisms

$$h'_1 \colon \mathrm{T}(M \cup \Sigma) \to \mathrm{T}(N \cup \Sigma) \qquad \text{and} \qquad h'_2 \colon \mathrm{T}(M \cup \Delta) \to \mathrm{T}(N \cup \Delta)$$

such that $h'_1|_M = h_1|_M$, $h'_2|_M = h_2|_M$, $h'_1|_\Sigma = \mathrm{id}_\Sigma$, and $h'_2|_\Delta = \mathrm{id}_\Delta$. Moreover, define the relations

$$\Rightarrow_{P'} = \bigcup_{p \in P'} \Rightarrow_p \qquad \text{and} \qquad \Rightarrow_{P''} = \bigcup_{p \in P''} \Rightarrow_p \ .$$

Then, for every $A \in M$, $\varrho_1 \in \mathrm{T}(M \cup \Sigma)$, $\varrho_2 \in \mathrm{T}(M \cup \Delta)$, and $\lambda \subseteq \mathrm{pos}_N(\varrho_1) \times \mathrm{pos}_N(\varrho_2)$,

$$\big(h'_1(A), h'_2(A), \{(\varepsilon, \varepsilon)\}\big) \Rightarrow_G \big(h'_1(\varrho_1), h'_2(\varrho_2), \lambda\big)$$

if and only if there is some $\xi \in \mathrm{T}(M \cup \Gamma)^1$ such that

$$\big(A, A, \{(\varepsilon, \varepsilon)\}\big) \Rightarrow_{P'} \big(\xi, \xi, \mathrm{id}_{\mathrm{pos}_N(\xi)}\big) \Rightarrow^*_{P''} (\varrho_1, \varrho_2, \lambda)\,.$$

As a consequence, for every $s \in \mathrm{T}_\Sigma$ and $t \in \mathrm{T}_\Delta$, and every derivation

$$\big(S_1, S_2, \{(\varepsilon, \varepsilon)\}\big) \Rightarrow^*_G (s, t, \emptyset)\,,$$

there is a corresponding derivation

$$\big(S, S, \{(\varepsilon, \varepsilon)\}\big) \Rightarrow^*_{G'} (s, t, \emptyset)\,,$$

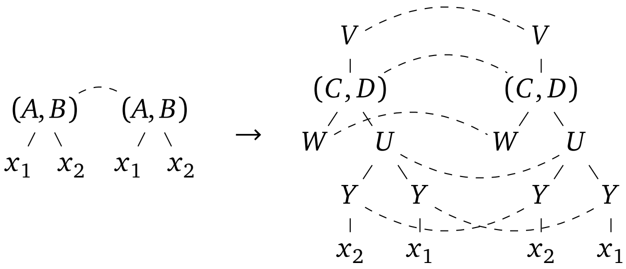
and vice versa. Note that the latter derivation need not necessarily be leftmost. But we can use Lemma 5.9 to permute the order of productions, and find a bijection b between

$$\bigcup_{\substack{s \in \mathrm{T}_\Sigma \\ t \in \mathrm{T}_\Delta}} \mathcal{D}_G\Big(\big(S_1, S_2, \{(\varepsilon, \varepsilon)\}\big), (s, t, \emptyset)\Big) \qquad \text{and} \qquad \bigcup_{\substack{s \in \mathrm{T}_\Sigma \\ t \in \mathrm{T}_\Delta}} \mathcal{D}_{G'}\Big(\big(S, S, \{(\varepsilon, \varepsilon)\}\big), (s, t, \emptyset)\Big).$$

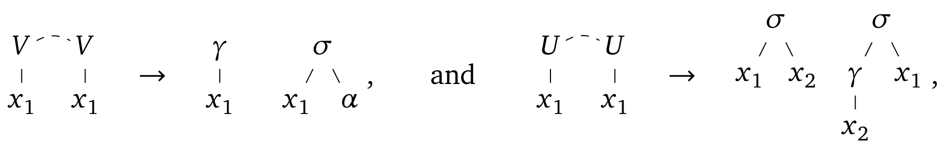
Because the weights of all productions from P'' are equal to 1, one can check that $wt(d) = wt'(b(d))$ for every derivation d in the domain of b. Therefore, for every $s \in \mathrm{T}_\Sigma$ and $t \in \mathrm{T}_\Delta$,

$$\begin{aligned} [\![G]\!](s,t) &= \sum\Big(wt(d) \;\Big|\; d \in \mathcal{D}_G(\xi_0, (s, t, \emptyset))\Big) \\ &= \sum\Big(wt'(d) \;\Big|\; d \in \mathcal{D}_{G'}(\xi'_0, (s, t, \emptyset))\Big) \\ &= [\![G']\!](s,t)\,. \end{aligned}$$ □

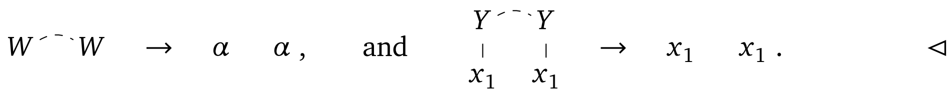
Example 5.17. Let us continue Example 5.15. To bring the wscftg G from there into normal form, we would construct the production



and add the productions



as well as



5.2 Pushdown Extended Tree Transducers

In this section, we will introduce weighted pushdown extended tree transducers, the model which we will later prove to characterize simple wscftg. In preparation for the equivalence proof, we show that these transducers can be assumed to have only one state (Lemma 5.22), and we prove that there is a particular normal form for them (Lemma 5.24).

Let us start out with defining the model. A *weighted pushdown extended (top-down) tree transducer (wpxtt)* is a tuple

$$M = (Q, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt)$$

such that

- Q is a ranked alphabet (its elements called *states*) such that $Q = Q^{(2)}$,
- Σ, Δ, and Γ are ranked alphabets (of *input*, *output*, and *pushdown symbols*),
- $q_0 \in Q$ and $\gamma_0 \in \Gamma$ (the *initial state* and *initial pushdown symbol*),
- K is a complete semiring and $wt\colon R \to K$ (the *weight mapping*), and
- R is a finite set (of *rules*), where each rule is of the form

$$q \cdot (\tilde{u} \otimes \gamma) \to v \cdot [q_1 \cdot (\pi_1^n \otimes \kappa_1), \dots, q_n \cdot (\pi_n^n \otimes \kappa_n)] \tag{5.6}$$

  for

  - some n, $k \in \mathbb{N}$, some $q, q_1, \dots, q_n \in Q$,
  - some $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)_n^1$, and some $v \in \mathrm{T}(\Delta)_n^1$ which is linear and nondeleting, and
  - some $\gamma \in \Gamma^{(k)}$, $\kappa_1, \dots, \kappa_n \in \mathrm{T}(\Gamma)_k^1$ such that $[\kappa_1, \dots, \kappa_n]$ is linear and nondeleting.

Remark 5.18. A rule of form (5.6) can be written equivalently

$$q\big(\tilde{u}, \gamma(x_{n+1}, \dots, x_{n+k})\big) \to v \cdot \Big[q_1\big(x_1, \kappa_1[x_{n+1}, \dots, x_{n+k}]\big), \dots, q_n\big(x_n, \kappa_n[x_{n+1}, \dots, x_{n+k}]\big)\Big].$$

For the sake of brevity, we will mainly stick with the form given in (5.6). ◁

Let M be a wpxtt as defined above, and consider a rule $r \in R$ of form (5.6). The *rewrite relation by* r, denoted by $\Rightarrow_r$, is the smallest relation on $\mathrm{T}_\Delta(Q(\mathrm{T}_\Sigma, \mathrm{T}_\Gamma))$ such that for every $\xi \in \mathrm{T}_\Delta(Q(\mathrm{T}_\Sigma, \mathrm{T}_\Gamma) \cup X_1)$ that contains x_1 exactly once, for every $\eta \in \mathrm{T}(\Gamma)_0^k$, and every $t \in \mathrm{T}(\Sigma)_0^n$, we have

$$\xi \cdot \big(q(\tilde{u} \cdot t, \gamma \cdot \eta)\big) \Rightarrow_r \xi \cdot \big(v \cdot \big[q_1(\pi_1 \cdot t, \kappa_1 \cdot \eta), \dots, q_n(\pi_n \cdot t, \kappa_n \cdot \eta)\big]\big).$$

In this situation, we say that r is applied at position $\mathrm{pos}_{x_1}(\xi)$. We write $\overset{w}{\Rightarrow}_r$ to emphasize that the rule r is applied at position w. The *rewrite relation of* M is given by $\Rightarrow_M = \bigcup_{r \in R} \Rightarrow_r$.

A sequence $r_1 \cdots r_m$ of rules $r_1, \ldots, r_m$ of M, $m \in \mathbb{N}$, is a *leftmost derivation in M* if there are $\xi_0, \ldots, \xi_m \in \mathrm{T}_\Delta(Q(\mathrm{T}_\Sigma, \mathrm{T}_\Gamma))$ such that

$$\xi_0 \overset{w_1}{\Rightarrow}_{r_1} \xi_1 \overset{w_2}{\Rightarrow}_{r_2} \cdots \overset{w_m}{\Rightarrow}_{r_m} \xi_m ,$$

and for every $j \in [m-1]$, w_j is the leftmost position in ξ_j that is labeled by an element of Q; i.e., w_j is the minimal position in $\mathrm{pos}_Q(\xi_j)$ with respect to $\leq_{\mathrm{lex}}$.

In this situation, we say that $r_1 \cdots r_m$ is a leftmost derivation *of ξ_m from ξ_0*. For every ξ, $\zeta \in \mathrm{T}_\Delta(Q(\mathrm{T}_\Sigma, \mathrm{T}_\Gamma))$, the set of leftmost derivations of ζ from ξ in M is denoted by $\mathcal{D}_M(\xi, \zeta)$, and for every $m \in \mathbb{N}$, we let $\mathcal{D}_M^{(m)}(\xi, \zeta) = \mathcal{D}_M(\xi, \zeta) \cap R^m$.

We extend the function wt to a mapping of type $R^* \to K$ in the natural way: for every $m \in \mathbb{N}$, and $r_1, \ldots, r_m \in R$, we let

$$wt(r_1 \cdots r_m) = wt(r_1) \cdots wt(r_m) .$$

Given a wpxtt $M = (Q, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt)$, we define for every $q \in Q$, $\eta \in \mathrm{T}_\Gamma$, and $m \in \mathbb{N}$ the mapping $[\![M, q, \eta]\!]^{(m)} \colon \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \to K$ such that

$$[\![M, q, \eta]\!]^{(m)}(s, t) = \sum \Big(wt(d) \;\Big|\; d \in \mathcal{D}_M^{(m)}(q(s, \eta), t) \Big)$$

for every $s \in \mathrm{T}_\Sigma$, $t \in \mathrm{T}_\Delta$. Then the *weighted tree transformation computed by M* is the mapping $[\![M]\!] \colon \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \to K$ such that for every $s \in \mathrm{T}_\Sigma$ and $t \in \mathrm{T}_\Delta$,

$$[\![M]\!](s, t) = \sum_{m \in \mathbb{N}} [\![M, q_0, \gamma_0]\!]^{(m)}(s, t) .$$

Remark 5.19. Weighted pushdown extended top-down tree transducers are related to the following formalisms. Let $M = (Q, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt)$ be a wpxtt.

- If $|\tilde{u}| \leq 1$ for every rule in R of form (5.6), and if $K = \mathbb{B}$, then M is a *top-down pushdown tree transducer* [165]. If additionally $\tilde{u} = v$, then M is a *pushdown tree automaton* as defined in [79].[3]

- If Γ is a singleton, then M is a *weighted extended top-down tree transducer* [17, 66]. Moreover, if $|\tilde{u}| = 1$ for every rule in R of form (5.6), and if $K = \mathbb{B}$, then M is a *top-down tree transducer* [49].

- Using the nomenclature of [59], wpxtt are (weighted) REG($\mathrm{TR}_{\mathrm{fin}} \times \mathrm{TP}$)-transducers, i.e., regular tree grammars equipped with a variant $\mathrm{TR}_{\mathrm{fin}}$ of the tree storage type TR that allows finite lookahead and decomposition, and with a tree pushdown storage type TP. ◁

Example 5.20. Consider the wpxtt $M = (Q, K, \Sigma, \Delta, \Gamma, q, \gamma_0, R, wt)$, where

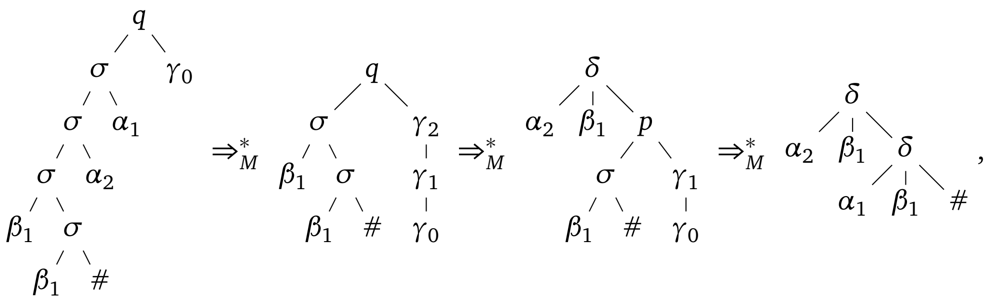
[3] Indeed, in [79] pushdown tree automata are defined as transducers which compute a partial identity. As mentioned in Remark 2.23, the pushdown tree automata defined in Section 2.2 are a special case of the general model of [79] (and therefore of wpxtt).

- $Q = \{q, p\}$ and $\Gamma = \{\gamma_1^{(1)}, \gamma_2^{(1)}, \gamma_0^{(0)}\}$,
- $K = (\mathbb{N}, +, \cdot, 0, 1)$ is the semiring of natural numbers,
- $\Sigma = \{\sigma^{(2)}, \alpha_1^{(0)}, \alpha_2^{(0)}, \beta_1^{(0)}, \beta_2^{(0)}, \#^{(0)}\}$ and $\Delta = \Sigma \setminus \{\sigma\} \cup \{\delta^{(3)}\}$,

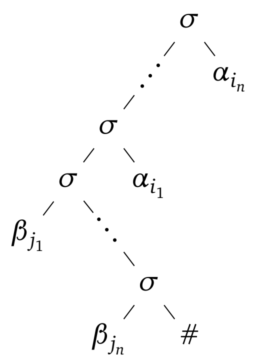
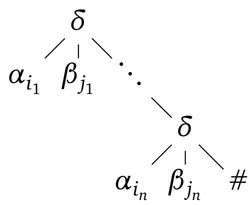
and R contains for every i and $j \in [2]$ the rules

$$\begin{aligned}
q\big(\sigma(x_1, \alpha_i), \gamma_0\big) &\to q\big(x_1, \gamma_i(\gamma_0)\big), \\
q\big(\sigma(x_1, \alpha_i), \gamma_j(x_2)\big) &\to q\big(x_1, \gamma_i\gamma_j(x_2)\big), \\
q\big(x_1, \gamma_j(x_2)\big) &\to p\big(x_1, \gamma_j(x_2)\big) \\
p\big(\sigma(\beta_i, x_1), \gamma_j(x_2)\big) &\to \delta\big(\alpha_j, \beta_i, p(x_1, x_2)\big), \quad \text{and} \qquad (5.7)\\
p(\#, \gamma_0) &\to \#.
\end{aligned}$$

Moreover, wt maps the rules in line (5.7) to the value 2, and all other rules to 1. Considering the derivation



we see that $\mathrm{supp}(\llbracket M \rrbracket)$ contains all tuples $(s, t) \in \mathrm{T}_\Sigma \times \mathrm{T}_\Delta$ with

$s =$  and $t =$ 

for some $n \in \mathbb{N}$ and $i_1, j_1, \ldots, i_n, j_n \in [2]$. As for every such tuple (s, t), there is precisely one derivation in M, (s, t) obtains the weight $\llbracket M \rrbracket(s, t) = 2^n$. ◁

The next lemma describes how to decompose leftmost derivations of a wpxtt, and is therefore helpful in induction arguments. Before that, however, we establish the following convention.

Convention. *In the following, we will encounter many sums which range over* $m_1, \ldots, m_n \in \mathbb{N}$ *such that* $n \in \mathbb{N}$ *and* $m_1 + \cdots + m_n = m$. *For brevity's sake, we will withhold the quantification "*$m_1, \ldots, m_n \in \mathbb{N}$*" in such cases, and write just "*$m_1 + \cdots + m_n = m$*".*

Moreover, we will follow the mathematical custom that sums are only taken over summands which are defined. If, for example, the quotient $\tilde{u} \backslash s$ *appears in a sum, and there are values for the trees* $\tilde{u}$ *and* s *such that there is no tuple* s' *with* $s = \tilde{u} \cdot s'$*, then the corresponding summand is taken to be* 0.

Lemma 5.21. *Let* $k, m \in \mathbb{N}$, $q \in Q$, $s \in \mathrm{T}_\Sigma$, $t \in \mathrm{T}_\Delta$, $\gamma \in \Gamma^{(k)}$, *and* $\eta \in \mathrm{T}(\Gamma)_0^k$. *Then*

$$[\![M, q, \gamma \cdot \eta]\!]^{(m+1)}(s,t) = \sum_{\substack{r \in R \text{ of form } (5.6),\\ m_1+\cdots+m_n=m}} wt(r) \cdot \prod_{j=1}^{n} [\![M, q_j, \kappa_j \cdot \eta]\!]^{(m_j)}\big(\pi_j^n \cdot (\tilde{u}\backslash s), \pi_j^n \cdot (v \backslash t)\big).$$

Proof. Define a function $b \colon \mathcal{D}_M^{(m+1)}\big(q(s, \gamma \cdot \eta), t\big) \to B$, where

$$\begin{aligned} B = \Big\{(r, d_1, \ldots, d_n) \,\Big|\, & r \in R \text{ of form as in (5.6)},\\ & m_1, \ldots, m_n \in \mathbb{N}, m_1 + \cdots + m_n = m,\\ & \text{for every } j \in [n],\, d_j \in \mathcal{D}_M^{(m_j)}\Big(q_j\big(\pi_j^n \cdot (\tilde{u}\backslash s), \kappa_j \cdot \eta\big),\, \pi_j^n \cdot (v \backslash t)\Big)\Big\}. \end{aligned}$$

The function b is defined as follows. Let $d \in \mathcal{D}_M^{(m+1)}\big(q(s, \gamma \cdot \eta), t\big)$, and assume that $d = rd'$ for some rule $r \in R$ – say r is of the form given in (5.6). As only leftmost derivations are considered, d' is of the form $d_1 \cdots d_n$ such that, for every $j \in [n]$, there is some $m_j \in \mathbb{N}$ with

$$d_j \in \mathcal{D}_M^{(m_j)}\Big(q_j\big(\pi_j^n \cdot (\tilde{u}\backslash s), \kappa_j \cdot \eta\big),\, \pi_j^n \cdot (v \backslash t)\Big).$$

Moreover, $m_1 + \cdots + m_n = m$. We set

$$b(d) = (r, d_1, \ldots, d_n)\,. \tag{5.8}$$

As the right-hand side is just a decomposition of the argument d into subderivations, it is easy to see that b is injective. It is also surjective, because for every tuple $(r, d_1, \ldots, d_l) \in B$, we have $b(rd_1 \ldots d_l) = (r, d_1, \ldots, d_l)$. Moreover, let us note that in (5.8), we have

$$wt(d) = wt(r) \cdot wt(d_1) \cdots wt(d_l)\,.$$

The identity that was stated in the lemma can now be proven as follows.

$$\begin{aligned} & [\![M, q, \gamma \cdot \eta]\!]^{(m+1)}(s,t)\\ = & \sum\Big(wt(d) \,\Big|\, d \in \mathcal{D}_M^{(m+1)}(q(s, \gamma \cdot \eta), t)\Big)\\ = & \sum\Big(wt(r) \cdot \prod_{j=1}^{n} wt(d_j) \,\Big|\, d \in \mathcal{D}_M^{(m+1)}(q(s, \gamma \cdot \eta), t),\, b(d) = (r, d_1, \ldots, r_n)\Big) \end{aligned}$$

$$= \sum\Big(wt(r) \cdot \prod_{j=1}^{n} wt(d_j) \;\Big|\; r \in R \text{ of form as in (5.6)}, m_1 + \cdots + m_n = m,$$
$$d_j \in \mathcal{D}_M^{(m_j)}\Big(q_j\big(\pi_j^n \cdot (\tilde{u}\backslash s), \kappa_j \cdot \eta\big), \pi_j^n \cdot (v \backslash t)\Big) \text{ for } j \in [n] \Big)$$

$$= \sum\Big(wt(r) \cdot \prod_{j=1}^{n} \sum\Big(wt(d_j) \;\Big|\; d_j \in \mathcal{D}_M^{(m_j)}\Big(q_j\big(\pi_j^n \cdot (\tilde{u}\backslash s), \kappa_j \cdot \eta\big), \pi_j^n \cdot (v \backslash t)\Big)\Big) \tag{5.9}$$
$$\Big|\; r \in R \text{ of form as in (5.6)}, m_1 + \cdots + m_n = m \Big)$$

$$= \sum\Big(wt(r) \cdot \prod_{j=1}^{n} [\![M, q_j, \kappa_j \cdot \eta]\!]^{(m_j)}\big(\pi_j^n \cdot (\tilde{u}\backslash s), \pi_j^n \cdot (v \backslash t)\big)$$
$$\Big|\; r \in R \text{ of form as in (5.6)}, m_1 + \cdots + m_n = m \Big).$$

Here, the equation (5.9) is obtained using the semiring's distributive law. □

5.2.1 One-State Transducers

We say that a wpxtt M is *one-state* if it has exactly one state. As we see in the lemma below, being one-state is no restriction to the power of wpxtt.

Lemma 5.22. *For every wpxtt M, there is a one-state wpxtt M' such that $[\![M]\!] = [\![M']\!]$.*

Proof. The proof idea is to encode the wpxtt's state behaviour into its pushdown symbols, similar to the analogous theorem for pushdown word automata (as shown, e.g., in [106, Lem. 25.1]).

Let, for this purpose, $M = (Q, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt)$ be a wpxtt. Construct the ranked alphabet Ω such that for every $k \in \mathbb{N}$,

$$\Omega^{(k)} = \big\{ (q, \gamma, q_1 \cdots q_k) \;\big|\; \gamma \in \Gamma^{(k)}, q, q_1, \ldots, q_k \in Q \big\}.$$

We define, for every $k \in \mathbb{N}$, and $q, q_1, \ldots, q_k \in Q$, the auxiliary function

$$\varphi^{q}_{q_1 \cdots q_n} \colon \mathrm{T}(\Gamma)^1_n \to \mathcal{P}\big(\mathrm{T}(\Omega)^1_n\big)$$

as follows. For every $i \in [n]$, let

$$\varphi^{q}_{q_1 \cdots q_n}(x_i) = \begin{cases} \{x_i\} & \text{if } q = q_i \\ \emptyset & \text{otherwise.} \end{cases}$$

Moreover, for every $k \in \mathbb{N}$, $\gamma \in \Gamma^{(k)}$, and $\kappa_1, \ldots, \kappa_k \in \mathrm{T}(\Gamma)^1_n$, let

$$\varphi^{q}_{q_1 \cdots q_n}\big(\gamma(\kappa_1, \ldots, \kappa_k)\big) = \big\{ (q, \gamma, p_1 \cdots p_k)\big(\kappa'_1, \ldots, \kappa'_k\big) \;\big|\; p_j \in Q, \kappa'_j \in \varphi^{p_j}_{q_1 \cdots q_n}(\kappa_j), j \in [k] \big\}.$$

Finally, for every $q_1, \ldots, q_m, p_1, \ldots, p_n \in Q$, where $m, n \in \mathbb{N}$, we define the function $\varphi^{q_1 \cdots q_m}_{p_1 \cdots p_n} \colon \mathrm{T}(\Gamma)^m_n \to \mathcal{P}(\mathrm{T}(\Omega)^m_n)$ such that, for every $\kappa \in \mathrm{T}(\Gamma)^m_n$,

$$\varphi^{q_1 \cdots q_m}_{p_1 \cdots p_n}(\kappa) = \big\{ \langle n; \kappa'_1, \ldots, \kappa'_m \rangle \;\big|\; \kappa'_j \in \varphi^{q_j}_{p_1 \cdots p_n}(\pi_j \cdot \kappa), j \in [m] \big\}. \tag{5.10}$$

The family of functions φ has the following two properties.

(A) We can restrict the subscripts to states that are necessary: in (5.10) above, let $\mathrm{lin}(\kappa) = (\tilde{\kappa}, \vartheta)$. Then

$$\varphi^{q_1\cdots q_m}_{p_1\cdots p_n}(\kappa) = \left\{\tilde{\kappa}' \cdot \vartheta \;\middle|\; \tilde{\kappa}' \in \varphi^{q_1\cdots q_m}_{p_{\vartheta(1)}\cdots p_{\vartheta(n)}}(\tilde{\kappa})\right\}.$$

This equation can be proven by structural induction on κ.

(B) Moreover, it is easy to show by structural induction on η that for every m, k, $n \in \mathbb{N}$, every $\eta \in \mathrm{T}(\Gamma)^m_k$ and $\kappa \in \mathrm{T}(\Gamma)^k_n$ that are linear and nondeleting, and every state $q_1, \ldots, q_m$, $p_1, \ldots, p_n \in Q$, the identity

$$\varphi^{q_1\cdots q_m}_{p_1\cdots p_n}(\eta \cdot \kappa) = \left\{\eta' \cdot \kappa' \;\middle|\; z_1, \ldots, z_k \in Q,\ \eta' \in \varphi^{q_1\cdots q_m}_{z_1\cdots z_k}(\eta),\ \kappa' \in \varphi^{z_1\cdots z_k}_{p_1\cdots p_n}(\kappa)\right\}$$

is satisfied.

* * *

We continue with constructing the wpxtt $M' = (\{*\}, K, \Sigma, \Delta, \Omega, *, (q_0, \gamma_0, \varepsilon), R', wt')$, where, for every rule $r \in R$ of the form given in (5.6), every $p_1, \ldots, p_k \in Q$, and every

$$\kappa'_1 \in \varphi^{q_1}_{p_1\cdots p_k}(\kappa_1), \qquad \ldots, \qquad \kappa'_n \in \varphi^{q_n}_{p_1\cdots p_k}(\kappa_n),$$

R' contains the rule r' of the form

$$* \cdot \big(\tilde{u} \otimes (q, \gamma, p_1 \cdots p_k)\big) \to v \cdot \big[* \cdot (\pi^n_1 \otimes \kappa'_1), \ldots, * \cdot (\pi^n_n \otimes \kappa'_n)\big] \tag{5.11}$$

with $wt'(r') = wt(r)$. Clearly, the rules of M' are linear and nondeleting, and M' is one-state.

It remains to show that $[\![M']\!] = [\![M]\!]$. For this purpose, we will prove the following property: for every $m \in \mathbb{N}$, $q \in Q$, $\eta \in \mathrm{T}_\Gamma$, $s \in \mathrm{T}_\Sigma$, and $t \in \mathrm{T}_\Delta$, we have

$$\sum\Big([\![M', *, \eta']\!]^{(m)}(s,t) \;\Big|\; \eta' \in \varphi^q_\varepsilon(\eta)\Big) = [\![M, q, \eta]\!]^{(m)}(s,t).$$

The property is proven by complete induction on m. Since both sides of the equation are zero for the base case $m = 0$, we proceed with the induction step. Let m, $k \in \mathbb{N}$, $\gamma \in \Gamma^{(k)}$, and $\eta \in \mathrm{T}(\Gamma)^k_0$. Assume that the property holds for every $m' \leq m$. Then

$$\begin{aligned}
&\sum\Big([\![M', *, \gamma' \cdot \eta']\!]^{(m+1)}(s,t) \;\Big|\; (\gamma' \cdot \eta') \in \varphi^q_\varepsilon(\gamma \cdot \eta), \gamma' \in \Gamma'\Big) \\
&= \sum\Big([\![M', *, (q, \gamma, p_1 \cdots p_k) \cdot \eta']\!]^{(m+1)}(s,t) \;\Big|\; p_1, \ldots, p_k \in Q,\ \eta' \in \varphi^{p_1\cdots p_k}_\varepsilon(\eta)\Big) \\
&= \sum\Big(wt'(r') \cdot \prod_{j=1}^n [\![M', *, \kappa'_j \cdot \eta']\!]^{(m_j)}\big(\pi_j \cdot (\tilde{u}\backslash s), \pi_j \cdot (v \backslash t)\big) \\
&\qquad\qquad \Big|\; m_1 + \cdots + m_n = m,\ r' \in R' \text{ as in (5.11)},\ p_1, \ldots, p_k \in Q, \\
&\qquad\qquad \eta' \in \varphi^{p_1\cdots p_k}_\varepsilon(\eta),\ \kappa'_j \in \varphi^{q_j}_{p_1\cdots p_k}(\kappa_j) \text{ for } j \in [n]\Big) \\
&= \sum\Big(wt'(r') \cdot \prod_{j=1}^n [\![M', *, \tilde{\kappa}'_j \cdot \vartheta_j \cdot \eta']\!]^{(m_j)}\big(\pi_j \cdot (\tilde{u}\backslash s), \pi_j \cdot (v \backslash t)\big) \qquad (5.12) \\
&\qquad\qquad \Big|\; m_1 + \cdots m_n = m,\ r' \in R' \text{ as in (5.11)},\ p_1, \ldots, p_k \in Q,\ \eta' \in \varphi^{p_1\cdots p_k}_\varepsilon(\eta), \\
&\qquad\qquad \mathrm{lin}(\kappa_j) = (\tilde{\kappa}_j, \vartheta_j),\ \mathrm{rk\,inf}(\tilde{\kappa}_j) = \ell_j,\ \tilde{\kappa}'_j \in \varphi^{q_j}_{p_{\vartheta_j(1)}\cdots p_{\vartheta_j(\ell_j)}}(\tilde{\kappa}_j) \text{ for } j \in [n]\Big)
\end{aligned}$$

$$= \sum\Big(wt(r) \cdot \prod_{j=1}^{n} \sum\Big([\![M', *, \tilde{\kappa}'_j \cdot \eta'_j]\!]^{(m_j)} \big(\pi_j \cdot (\tilde{u} \backslash s), \pi_j \cdot (v \backslash t)\big) \tag{5.13}$$
$$\Big|\ \mathrm{lin}(\kappa_j) = (\tilde{\kappa}_j, \vartheta_j),\ \mathrm{rk}\,\mathrm{inf}(\tilde{\kappa}_j) = \ell_j,\ p_1, \dots, p_{\ell_j} \in Q,$$
$$\tilde{\kappa}'_j \in \varphi^{q_j}_{p_1 \cdots p_{\ell_j}}(\tilde{\kappa}_j),\ \eta'_j \in \varphi^{p_1 \cdots p_{\ell_j}}_{\varepsilon}(\vartheta_j \cdot \eta) \text{ for } j \in [n] \Big)$$
$$\Big|\ m_1 + \dots + m_n = m,\ r \in R \text{ as in (5.6)} \Big)$$

$$= \sum\Big(wt(r) \cdot \prod_{j=1}^{n} \sum\Big([\![M', *, \chi]\!]^{(m_j)} \big(\pi_j \cdot (\tilde{u} \backslash s), \pi_j \cdot (v \backslash t)\big) \ \Big|\ \chi \in \varphi^{q_j}_{\varepsilon}(\kappa_j \cdot \eta) \Big) \tag{5.14}$$
$$\Big|\ m_1 + \dots + m_n = m,\ r \in R \text{ as in (5.6)} \Big)$$

$$= \sum\Big(wt(r) \cdot \prod_{j=1}^{n} [\![M, q_j, \kappa_j \cdot \eta]\!]^{(m_j)} \big(\pi_j \cdot (\tilde{u} \backslash s), \pi_j \cdot (v \backslash t)\big)$$
$$\Big|\ m_1 + \dots + m_n = m,\ r \in R \text{ as in (5.6)} \Big)$$

$$= [\![M, q, \gamma \cdot \eta]\!]^{(m+1)}(s, t)\,.$$

Let us explain the above equations. In (5.12), we linearize the tree pushdowns κ_j, obtaining torsion-free $\tilde{\kappa}_j \in \widetilde{\mathrm{T}}(\Gamma)^1_{\ell_j}$ for some $\ell_j \in \mathbb{N}$, and torsions ϑ_j such that $\kappa_j = \tilde{\kappa}_j \cdot \vartheta_j$. This allows us to determine the states $p_{\vartheta_j(1)}, \dots, p_{\vartheta_j(\ell_j)}$ necessary for the functions $\varphi^{q_j}_{p_{\vartheta_j(1)} \cdots p_{\vartheta_j(\ell_j)}}$, according to property **(A)**.

In the next equation, labeled (5.13), we can therefore move the summation over these states, and over the involved tree pushdowns, into the product. This transformation is valid due to the distributivity of the semiring. Finally, in (5.14), we use property **(B)** to abridge the summation over $p_1, \dots, p_{\ell_j}$. After that, all that remains is to apply the induction hypothesis.

* * *

We are still obliged to show how this property implies the construction's correctness. Consider $s \in \mathrm{T}_\Sigma$ and $t \in \mathrm{T}_\Delta$. Then

$$\sum_{m \in \mathbb{N}} [\![M, q_0, \gamma_0]\!]^{(m)}(s, t) = \sum_{\substack{m \in \mathbb{N},\\ \gamma'_0 \in \varphi^{q_0}_{\varepsilon}(\gamma_0)}} [\![M', *, \gamma'_0]\!]^{(m)}(s, t) = \sum_{m \in \mathbb{N}} [\![M', *, (q_0, \gamma_0, \varepsilon)]\!]^{(m)}(s, t)\,,$$

and therefore $[\![M]\!](s, t) = [\![M']\!](s, t)$. □

5.2.2 Transducers in Normal Form

In addition to being one-state, we can further restrict the form of wpxtt, without detriment to their power. In this manner, we obtain the following normal form for wpxtt.

A wpxtt $M = (Q, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt)$ is in *normal form* if each of its rules is either of form

$$q \cdot (\pi_1^1 \otimes \gamma) \to p \cdot (\pi_1^1 \otimes \kappa),$$

for some q, $p \in Q$, $k \in \mathbb{N}$, $\gamma \in \Gamma^{(k)}$ and linear and nondeleting tree $\kappa \in \mathrm{T}(\Gamma)_k^1$ with $|\kappa| > 0$, or of form

$$q \cdot (\tilde{u} \otimes \gamma) \to v \cdot \left[q_1 \cdot (\pi_1^n \otimes \pi_1^n), \cdots, q_n \cdot (\pi_n^n \otimes \pi_n^n) \right]$$

for some $n \in \mathbb{N}$, $\gamma \in \Gamma^{(n)}$, $\tilde{u} \in \widetilde{\mathrm{T}}(\Sigma)_n^1$ and some $v \in \mathrm{T}(\Delta)_n^1$ that is linear and nondeleting, as well as some states $q_1, \ldots, q_n \in Q$. Rules of the first form will be called *push rules*, while those of the second form are *pop rules*.

Remark 5.23. A wpxtt in normal form is very close to the *creative dendrogrammar* introduced by Rounds in [140]. The push rules of our wpxtt correspond to *index-creating* productions of creative dendrogrammars, while pop rules are essentially *index-erasing* productions. ◁

Lemma 5.24. *For every wpxtt M, there is a wpxtt M' in normal form such that $[\![M]\!] = [\![M']\!]$, and M' has the same number of states as M.*

Proof. Assume we are given the wpxtt $M = (Q, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt)$. We construct the wpxtt $M' = (Q, K, \Sigma, \Delta, \Gamma', q_0, \gamma_0, R', wt')$ as follows. Construe R as a ranked alphabet disjoint from Γ, such that for every rule $r \in R$ of form (5.6), we have $\mathrm{rk}(r) = n$. We set $\Gamma' = \Gamma \cup R$.

Further, R' is the smallest set that satisfies the following. Let $r \in R$ be a rule of form (5.6). Then R' contains the two rules r', of form

$$q(\pi_1^1 \otimes \gamma) \to q\big(\pi_1^1 \otimes r \cdot [\kappa_1, \cdots, \kappa_n]\big),$$

and r'', of form

$$q(\tilde{u} \otimes r) \to v \cdot \left[q_1(\pi_1^n \otimes \pi_1^n), \ldots, q_n(\pi_n^n \otimes \pi_n^n) \right].$$

We set $wt'(r') = wt(r)$ and $wt'(r'') = 1$.

* * *

In order to show the construction's correctness, assume that $m \in \mathbb{N}$, $s \in \mathrm{T}_\Sigma$, and $t \in \mathrm{T}_\Delta$. For every derivation $r_1 \cdots r_m \in \mathcal{D}_M(q(s, \gamma_0), t)$, let

$$b_m(r_1 \cdots r_m) = r_1' r_1'' \cdots r_m' r_m'',$$

where for each $j \in [m]$, r_j' and $r_j'' \in R'$ are the rules that were built from r_j following the construction above. As each rule r_j' has only one state on its right-hand side, the positions where the rules r_1', r_1'', …, r_m', r_m'' are applied are ordered lexicographically. Hence b_m is a function of type

$$\mathcal{D}_M^{(m)}(q(s, \gamma_0), t) \to \mathcal{D}_{M'}^{(2m)}(q(s, \gamma_0), t).$$

In fact, it is easy to see from the form of the constructed rules that b_m is a bijection. Finally, $wt(d) = wt'(b_m(d))$ for every derivation $d \in \mathcal{D}_M^{(m)}(q(s, \gamma_0), t)$, as the weight of every other rule in $b_m(d)$ is equal to 1. We obtain

$$\begin{aligned}
[\![M]\!](s,t) &= \sum\Big(wt(d) \;\Big|\; m \in \mathbb{N},\, d \in \mathcal{D}_M^{(m)}(q(s,\eta),t)\Big) \\
&= \sum\Big(wt'(d) \;\Big|\; m \in \mathbb{N},\, d \in \mathcal{D}_{M'}^{(2m)}(q(s,\eta),t)\Big) \\
&= [\![M']\!](s,t)\,.
\end{aligned}$$

The last equation is valid because, clearly, every derivation in M' is of even length. $\square$

5.3 Characterization of Simple Weighted Context-Free Tree Transformations

At this point, the close resemblance between simple wscftg in normal form and one-state wpxtt in normal form should be apparent. We exploit this similarity to obtain the announced characterization result. For this purpose, we define a notion of relatedness between the two models, and prove that relatedness implies equality of semantics (Lemma 5.25). Since given one model, a related model of the other type is straightforward to construct, the proof for the main result of this chapter (Theorem 5.26) is an easy consequence.

Let Σ and Δ be ranked alphabets. Consider a simple wscftg $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$ in normal form, and a one-state wpxtt $M = (\{*\}, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt')$ in normal form. We say that G and M are *related* if

(i) $N = \Gamma$,

(ii) $\xi_0 = \big(\gamma_0, \gamma_0, \{(\varepsilon, \varepsilon)\}\big)$,

(iii) P contains a nonterminal production p of the form

$$(A \cdot \mathrm{Id}_n, A \cdot \mathrm{Id}_n) \to (\varrho, \varrho, \mathrm{id}_{\mathrm{pos}_N(\varrho)}) \tag{5.15}$$

if and only if R contains a push rule r of the form

$$* \cdot (\pi_1^1 \otimes A \cdot \mathrm{Id}_n) \to * \cdot (\pi_1^1 \otimes \varrho) \tag{5.16}$$

with $wt(p) = wt'(r)$, and

(iv) P contains a terminal production p of the form

$$(A \cdot \mathrm{Id}_n, A \cdot \mathrm{Id}_n) \to (\tilde{u}, v, \emptyset) \tag{5.17}$$

if and only if R contains a pop rule r of the form

$$* \cdot (\tilde{u} \otimes A \cdot \mathrm{Id}_n) \to v \cdot \big[* \cdot (\pi_1^n \otimes \pi_1^n), \ldots, * \cdot (\pi_n^n \otimes \pi_n^n)\big] \tag{5.18}$$

with $wt(p) = wt'(r)$.

Lemma 5.25. *Let G be a simple wscftg in normal form, and let M be a one-state wpxtt in normal form. If G and M are related, then $[\![G]\!] = [\![M]\!]$.*

Proof. Assume that $G = (N, K, \Sigma, \Delta, \xi_0, P, wt)$ and $M = (\{*\}, K, \Sigma, \Delta, \Gamma, q_0, \gamma_0, R, wt')$ are related. We will show for every $m \in \mathbb{N}$, $\xi \in \mathrm{T}(N)_0^1$, $s \in \mathrm{T}_\Sigma$, and $t \in \mathrm{T}_\Delta$ that

$$[\![G, (\xi, \xi, \mathrm{id}_{\mathrm{pos}_N(\xi)})]\!]^{(m)}(s, t) = [\![M, *, \xi]\!]^{(m)}(s, t)\,.$$

The proof is by complete induction on m, and the base case $m = 0$ is trivial. So consider some $m, n \in \mathbb{N}$, $A \in N^{(n)}$ and $\xi \in \mathrm{T}(N)_0^n$. Then

$$
\begin{aligned}
&\Big[\!\!\Big[G, (A \cdot \xi, A \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(A \cdot \xi)}) \Big]\!\!\Big]^{(m+1)}(s,t) \\
&\quad = \sum \Big(wt(p) \cdot \Big[\!\!\Big[G, (\varrho \cdot \xi, \varrho \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(\varrho \cdot \xi)}) \Big]\!\!\Big]^{(m)}(s,t) \;\Big|\; p \in P \text{ of form (5.15)} \Big) \qquad (5.19) \\
&\qquad + \sum \Big(wt(p) \cdot \prod_{j=1}^{n} \Big[\!\!\Big[G, \big(\pi_j^n \cdot \xi, \pi_j^n \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(\pi_j^n \cdot \xi)}\big) \Big]\!\!\Big]^{(m_j)} \big(\pi_j^n \cdot (\tilde{u} \backslash s), \pi_j^n \cdot (v \backslash t) \big) \\
&\qquad\qquad \;\Big|\; p \in P \text{ of form (5.17)}, m_1 + \cdots + m_n = m \Big) \\
&\quad = \sum \Big(wt'(r) \cdot [\![M, \varrho \cdot \xi]\!]^{(m)}(s,t) \;\Big|\; r \in R \text{ of form (5.16)} \Big) \qquad (5.20) \\
&\qquad + \sum \Big(wt'(r) \cdot \prod_{j=1}^{n} \Big[\!\!\Big[M, *, \pi_j^n \cdot \xi \Big]\!\!\Big]^{(m_j)} \big(\pi_j^n \cdot (\tilde{u} \backslash s), \pi_j^n \cdot (v \backslash t) \big) \\
&\qquad\qquad \;\Big|\; r \in R \text{ of form (5.18)}, m_1 + \cdots + m_n = m \Big) \\
&\quad = \Big[\!\!\Big[M, *, A \cdot \xi \Big]\!\!\Big]^{(m+1)}(s,t) \,.
\end{aligned}
$$

The decomposition of $\Big[\!\!\Big[G, (A \cdot \xi, A \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(A \cdot \xi)}) \Big]\!\!\Big]^{(m+1)}(s,t)$ into the two sums in (5.19) is valid because of the special form of productions of G: every derivation of $(s,t,\emptyset)$ from $(A \cdot \xi, A \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(A \cdot \xi)})$ is either of the form

$$\big(A \cdot \xi, A \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(A \cdot \xi)} \big) \Rightarrow_p \big(\varrho \cdot \xi, \varrho \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(\varrho \cdot \xi)} \big) \Rightarrow_G^* (s,t,\emptyset)$$

for some production p of form (5.15), or it reads

$$\big(A \cdot \xi, A \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(A \cdot \xi)} \big) \Rightarrow_p \big(\tilde{u} \cdot \xi, v \cdot \xi, \lambda \big) \Rightarrow_G^* \big(\tilde{u} \cdot (\tilde{u} \backslash s), v \cdot (v \backslash t), \emptyset \big) = (s,t,\emptyset)\,,$$

for some production p of form (5.17), and where

$$\lambda = \big\{ \big(\mathrm{pos}_{x_j}(\tilde{u})\, w_1, \mathrm{pos}_{x_j}(v)\, w_2 \big) \;\big|\; j \in [n], (jw_1, jw_2) \in \mathrm{id}_{\mathrm{pos}_N(A \cdot \xi)} \big\}\,.$$

Observe that in the situation above, $(jw_1, jw_2) \in \mathrm{id}_{\mathrm{pos}_N(A \cdot \xi)}$ if and only if $w_1 = w_2 = w$ for some $w \in \mathrm{pos}_N(\pi_j \cdot \xi)$. Therefore the remaining derivation after the application of p consists of derivations

$$\big(\pi_j \cdot \xi, \pi_j \cdot \xi, \mathrm{id}_{\mathrm{pos}_N(\pi_j \cdot \xi)} \big) \Rightarrow_G^* \big(\pi_j \cdot (\tilde{u} \backslash s), \pi_j \cdot (v \backslash t), \emptyset \big)\,,$$

for each $j \in [n]$. A similar argument can be given to justify the sums in (5.20).

* * *

Now, for every $s \in \mathrm{T}_\Sigma$ and $t \in \mathrm{T}_\Delta$,

$$[\![G]\!](s,t) = \sum_{m \in \mathbb{N}} \Big[\!\!\Big[G, \big(\gamma_0, \gamma_0, \{(\varepsilon, \varepsilon)\} \big) \Big]\!\!\Big]^{(m)}(s,t) = \sum_{m \in \mathbb{N}} [\![M, *, \gamma_0]\!]^{(m)}(s,t) = [\![M]\!](s,t)\,,$$

and the proof is concluded. □

As an easy corollary, we obtain the main theorem.

Theorem 5.26. *Consider ranked alphabets Σ and Δ, and a complete semiring K. Let*

$$\tau \colon \mathrm{T}_\Sigma \times \mathrm{T}_\Delta \to K$$

be a weighted tree transformation. Then τ is generated by a simple wscftg if and only if it is computed by a wpxtt.

Proof. Let G be a simple wscftg such that $\tau = [\![G]\!]$. By Lemma 5.16, G can be assumed to be in normal form. By reading the definition of relatedness as a construction, it is straightforward to come up with a wpxtt M that is related to G. But then $[\![M]\!] = [\![G]\!]$ by Lemma 5.25.

For the other direction, let M be a wpxtt such that $\tau = [\![M]\!]$. We can assume that M is one-state by Lemma 5.22, and in normal form by Lemma 5.24. Again, it is easy to find a simple wscftg G that is related to M. By Lemma 5.25, $[\![G]\!] = [\![M]\!]$. □

5.4 Chapter Conclusion

In this chapter, we have recalled Nederhof and Vogler's synchronous context-free tree grammars, and presented a semiring-weighted version of them. We identified a syntactic restriction of this model and developed a machine characterization of the respective subclass. The involved machines can be understood as the common generalization of extended top-down tree transducers and pushdown tree transducers. Synchronous context-free tree grammars can be generalized further to *multiple context-free tree grammars*, cf. [53].

The proof for the normal form of simple wscftg has the following interesting implication. Using Lemma 5.14, one can give a bimorphism characterization[4] for simple wscftg: the weighted tree transformations of wscftg are precisely those of bimorphisms with weighted linear and nondeleting cftg generating the center language, while the morphisms are both linear and nondeleting tree homomorphisms. The reader is invited to contrast this characterization to the bimorphism characterization for unrestricted synchronous cftg given in [124, Thm. 1], where the center language is recognizable, and the morphisms are particular macro tree transductions. We omit the formal proof of this claim, as it requires some definitions on weighted tree languages which have not been introduced in this work.

Moreover, we pose the following open problem: is there a machine characterization for the whole class of (weighted) tree transformations of wscft?

[4]In the setting of unweighted tree languages, a *bimorphism* is a triple (f_1, L, f_2), where $L \subseteq \mathrm{T}_\Gamma$ is a tree language, and $f_1 \colon \mathrm{T}_\Gamma \to \mathrm{T}_\Sigma$ and $f_2 \colon \mathrm{T}_\Gamma \to \mathrm{T}_\Delta$ are deterministic tree transformations, for some ranked alphabets Σ, Δ, and Γ [17, 153]. The tree transformation induced by the bimorphism (f_1, L, f_2) is $f_1^{-1} ; \mathrm{id}_L ; f_2$. In the weighted setting, the definitions are analogous, using composition of weighted tree transformations.

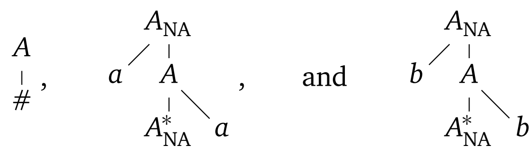
Chapter 6

Footed and Linear Monadic Context-Free Tree Grammars

Que sert, hélas! d'arroser le feuillage quand l'arbre est coupé par le pied?

(Jean-Jacques Rousseau)

Tree-adjoining grammars are a well-established grammar formalism in the field of computational linguistics [91, 90, 92]. They have been introduced to capture some *mildly context-sensitive* phenomena which occur in natural languages (cf. the introduction of Chapter 2). One prominent example of the use of tree-adjoining grammars for natural language processing is the XTAG project, with the aim to develop a tree-adjoining grammar for English [43]. Tree-adjoining grammars are tree-generating grammars consisting of a finite number of *elementary trees*. The basic operation which underlies a derivation step of tree-adjoining grammars is called *adjoining.*[1] For an example, consider the three elementary trees

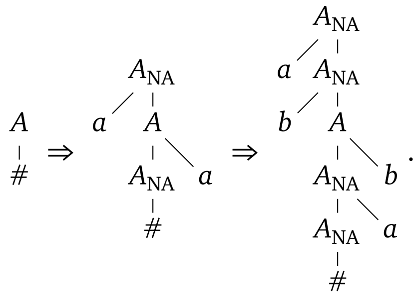


The trees' nodes are labeled by symbols from the alphabet $\{\#, a, b, A\}$. Every node may be equipped with the *negative adjoining* constraint NA, which prohibits applying the adjoining operation to this node. Moreover, at most one leaf node of an elementary tree can be labeled with a star $*$. This means the node is the *foot node* of the tree. The foot node of an elementary tree is often required to carry the same label as the tree's root.

Let us now describe the adjoining operation. We can adjoin an elementary tree e with a foot node into another tree t at the node w of t if w does not carry the negative adjoining constraint, and w and the root of t have the same label. The outcome is the tree which is the result of inserting e into t at node w; the subtrees of w become subtrees of the foot node of t.

[1]In the original definition, there is also another operation, called *substitution*. However, substitution can be reduced to adjoining, and we will not dwell on it further.

For an example derivation using the above elementary trees, consider



When we regard the set L of all trees which are derivable in this manner from $A(\#)$, we see that

$$\mathrm{yd}(L) = \left\{ w\#w \;\middle|\; w \in \{a, b\}^* \right\} .$$

As proven independently by Mönnich [122] and Fujiyoshi and Kasai [65], the yield languages of tree-adjoining grammars are precisely those of linear monadic cftg. The relation between the tree languages of the two formalisms remained open over one decade. In [100], Kepser and Rogers showed that the tree languages of a variant of tree-adjoining grammar, called *non-strict (ns-tag),* are exactly the linear monadic context-free tree languages. The proof is by a number of intermediate constructions. First of all, the authors identify a counterpart of ns-tag in the world of cftg, called *footed cftg.* The right-hand sides of productions of footed cftg look essentially like elementary trees of tree-adjoining grammars, but the foot node of each tree may have the variables $x_1, \ldots, x_k$ as children, for some $k \in \mathbb{N}$, and this is the only place where variables may appear.

Kepser and Rogers prove that the tree languages of footed cftg are the linear monadic context-free tree languages. Then they show that every footed cftg is equivalent to a *spinal-formed* cftg. Spinal-formed cftg have been discovered by Fujiyoshi and Kasai [65] and been proven to generate linear monadic context-free tree languages. This result establishes one direction of the equivalence of footed cftg and lm-cftg. For the other direction, the authors obtain from an lm-cftg an equivalent nondeleting lm-cftg. Finally, they remove collapsing productions from this grammar, and the resulting equivalent cftg can be shown to be footed.

In [70], together with Kilian Gebhardt, we have given a direct construction of an equivalent lm-cftg from an ns-tag. This construction has the additional benefit that the shape of the elementary trees is essentially preserved in the right-hand sides of the constructed lm-cftg's productions. In this chapter, we will describe the construction.

As ns-tag are based on unranked trees (i.e., the label at a node does not uniquely determine the number of the node's subtrees), they do not fit neatly into the framework of this thesis. Therefore, we show that the counterparts to ns-tag in the world of cftg, i.e., the footed cftg, are expressively equivalent to lm-cftg. The idea behind the construction is essentially the one from our paper. For the relationship between ns-tag and footed cftg, consult [100, Thm. 2].

Note: The proof of this chapter's main theorem is essentially from the conference article [70] in collaboration with Kilian Gebhardt. It has been rewritten and adapted to the case of footed cftg. As noted, the original proof of the theorem is by Kepser and Rogers [100].

6.1 Footed and Linear Monadic Context-Free Tree Grammars

Let $G = (N, \Sigma, S, P)$ be a cftg with inital nonterminal. We say that G is *footed* if every production of G is of the form

$$A \cdot \mathrm{Id}_k \to \tilde{\varrho} \cdot F \cdot \mathrm{Id}_k \tag{6.1}$$

for some $k \in \mathbb{N}$, $A \in N^{(k)}$, $F \in N^{(k)} \cup \Sigma^{(k)}$, and $\tilde{\varrho} \in \widetilde{\mathrm{T}}(N \cup \Sigma)_1^1$. Informally, the variables x_1, $\ldots$, x_k in the production's right-hand side may occur only under the symbol F, precisely in this order. In a production as above, F is called the production's *foot*.

Convention. *When we consider a production of a footed cftg, say of the form given in* (6.1)*, then we will often omit the quantifications for A, $\tilde{\varrho}$, and F. They should be understood from the production's form.*

Example 6.1. Consider a cftg $G = (N, \Sigma, S, P)$ with

$$N = \{S^{(0)}, A^{(1)}, B^{(2)}\} \qquad \text{and} \qquad \Sigma = \{a^{(0)}, b^{(0)}, \#^{(0)}, \gamma^{(1)}, \sigma^{(2)}\},$$

where P contains the following productions.

$$S \to A(\#)$$

$$A(x_1) \to \sigma(a, B(\gamma(x_1), a)) + \sigma(b, B(\gamma(x_1), b)) + \gamma(x_1)$$

$$B(x_1, x_2) \to \sigma(a, B(\sigma(x_1, x_2), a)) + \sigma(b, B(\sigma(x_1, x_2), b)) + \sigma(x_1, x_2)$$

The cftg G is footed, and closely related to the tree-adjoining grammar given in the introduction. In fact, it is (up to a renaming of symbols) the footed cftg that is constructed from the tree-adjoining grammar according to [100, Thm. 2]. $\lhd$

Lemma 6.2 (Kepser and Rogers [100])**.** *For every linear monadic cftg G, there is a footed cftg G' with $\mathcal{L}(G') = \mathcal{L}(G)$.*

Proof. We recall the proof idea presented in [100]. Using Theorem 2.16, we can construct from G an equivalent ln-cftg G'. Then we eliminate from G' collapsing productions of form $A(x_1) \to x_1$, by [109, Thm. III.7], resulting in the equivalent monadic ln-cftg G''. Clearly, G'' is footed, as the right-hand side of every production is nonempty, and the only variable x_1 occurs at most once in a right-hand side. $\square$

Lemma 6.3. *For every footed cftg G, there is a linear monadic cftg G' such that $\mathcal{L}(G') = \mathcal{L}(G)$.*

Proof. Consider a footed cftg $G = (N, \Sigma, S, P)$. Let

$$N' = \{A_\sigma^{(1)} \mid k \in \mathbb{N}_1, A \in N^{(k)}, \sigma \in N^{(k)}\} \cup N^{(0)}.$$

We define the function $\varphi\colon \mathrm{T}(N \cup \Sigma) \to \mathcal{P}(\mathrm{T}(N' \cup \Sigma))$ as follows. For every $n \in \mathbb{N}$ with $n \neq 1$, and every $\xi \in \mathrm{T}(N \cup \Sigma)^n$, let

$$\varphi(\xi) = \{[\xi'_1, \ldots, \xi'_n] \mid \xi'_1 \in \varphi(\pi_1 \cdot \xi), \ldots, \xi'_n \in \varphi(\pi_n \cdot \xi)\}.$$

For every $i \in \mathbb{N}$, let $\varphi(x_i) = \{x_i\}$. Consider some $k \in \mathbb{N}$ and $\xi_1, \ldots, \xi_k \in \mathrm{T}(N \cup \Sigma)^1$. For every $\sigma \in \Sigma$, let

$$\varphi\big(\sigma(\xi_1, \ldots, \xi_k)\big) = \{\sigma(\xi'_1, \ldots, \xi'_k) \mid \xi'_1 \in \varphi(\xi_1), \ldots, \xi'_k \in \varphi(\xi_k)\}.$$

Let $A \in N^{(k)}$. If $k = 0$, then $\varphi(A) = \{A\}$. Otherwise,

$$\varphi\big(A(\xi_1, \ldots, \xi_k)\big) = \{A_\sigma\big(\sigma(\xi'_1, \ldots, \xi'_k)\big) \mid \sigma \in \Sigma^{(k)}, \xi'_1 \in \varphi(\xi_1), \ldots, \xi'_k \in \varphi(\xi_k)\}.$$

The following property of φ is easy to show.

(A) *For every $\tilde{\xi}$ and $\tilde{\zeta} \in \widetilde{\mathrm{T}}(N \cup \Sigma)$,*

$$\varphi(\tilde{\xi} \cdot \tilde{\zeta}) = \{\tilde{\xi}' \cdot \tilde{\zeta}' \mid \tilde{\xi}' \in \varphi(\tilde{\xi}), \tilde{\zeta}' \in \varphi(\tilde{\zeta})\}.$$

Now we construct the cftg $G' = (N', \Sigma, S, P')$, where the set of productions P' is given as follows.

(i) For every production in P of form

$$A \cdot \mathrm{Id}_0 \to \varrho \qquad \text{and every} \qquad \varrho' \in \varphi(\varrho),$$

insert the production $A \to \varrho'$ into P'.

(ii) For every production in P of form

$$A \cdot \mathrm{Id}_k \to \tilde{\varrho} \cdot \sigma \cdot \mathrm{Id}_k,$$

and every $\tilde{\varrho}' \in \varphi(\tilde{\varrho})$, insert into P' the production $A_\sigma(x) \to \tilde{\varrho}'$.

(iii) For every production in P of form

$$A \cdot \mathrm{Id}_k \to \tilde{\varrho} \cdot B \cdot \mathrm{Id}_k \qquad \text{for some } B \in N^{(k)},$$

every $\tilde{\varrho}' \in \varphi(\tilde{\varrho})$ and every $\sigma \in \Sigma^{(k)}$, insert into P' the production $A_\sigma(x) \to \tilde{\varrho}' \cdot B_\sigma \cdot \mathrm{Id}_1$.

* * *

The idea behind the construction is as follows. Consider a k-ary nonterminal A of G, for some $k \in \mathbb{N}_1$, and a derivation

$$A \Rightarrow_G \tilde{\xi}_1 \cdot B_1 \Rightarrow_G \tilde{\xi}_1 \cdot \tilde{\xi}_2 \cdot B_2 \Rightarrow_G \cdots \Rightarrow_G \tilde{\xi}_1 \cdots \tilde{\xi}_n \cdot B_n \Rightarrow_G \tilde{\xi}_1 \cdots \tilde{\xi}_n \cdot \tilde{\xi}_{n+1} \cdot \sigma$$

for some $n \in \mathbb{N}$, $B_1, \ldots, B_n \in N^{(k)}$, $\tilde{\xi}_1, \ldots, \tilde{\xi}_n \in \widetilde{\mathrm{T}}(N \cup \Sigma)_1^1$, and $\sigma \in \Sigma^{(k)}$. We see that the derivation results eventually in the foot node σ. The construction tries to anticipate the production of σ: using the tree transformation φ, we guess for every (non-foot) occurrence of the nonterminal symbol A the terminal foot node it will produce eventually. For example, when we guess that σ is produced, then A is replaced with $A_\sigma \cdot \sigma$. Of course, the guess of σ must be propagated along $B_1, \ldots, B_n$. A corresponding derivation in G' is therefore of the form

$$A_\sigma \Rightarrow_{G'} \tilde{\xi}'_1 \cdot (B_1)_\sigma \Rightarrow_{G'} \tilde{\xi}'_1 \cdot \tilde{\xi}'_2 \cdot (B_2)_\sigma \Rightarrow_{G'} \cdots \Rightarrow_{G'} \tilde{\xi}'_1 \cdots \tilde{\xi}'_n \cdot (B_n)_\sigma \Rightarrow_{G'} \tilde{\xi}'_1 \cdots \tilde{\xi}'_n \cdot \tilde{\xi}'_{n+1}\,,$$

for some $\tilde{\xi}'_1 \in \varphi(\tilde{\xi}_1), \ldots, \tilde{\xi}'_{n+1} \in \varphi(\tilde{\xi}_{n+1})$.

* * *

We now turn to the proof of $\mathcal{L}(G) = \mathcal{L}(G')$, which consists of two parts.

(⊆) For the inclusion $\mathcal{L}(G) \subseteq \mathcal{L}(G')$, we prove for every $n \in \mathbb{N}$, $\xi \in \mathrm{T}(N \cup \Sigma)_0^1$, and $\xi' \in \varphi(\xi)$ that

$$S \Rightarrow_G^n \xi \qquad \text{implies} \qquad S \Rightarrow_{G'}^n \xi'\,.$$

The proof is by mathematical induction on n, and the base case $n = 0$ is trivial. We proceed by a case analysis on the last production of the derivation. Let $n \in \mathbb{N}$.

(I) Let $\tilde{\xi} \in \widetilde{\mathrm{T}}(N \cup \Sigma)_1^1$ such that

$$S \Rightarrow_G^n \tilde{\xi} \cdot A \Rightarrow_G \tilde{\xi} \cdot \varrho$$

by some production of form $A \cdot \mathrm{Id}_0 \to \varrho$. Let $\xi' \in \varphi(\tilde{\xi} \cdot \varrho)$. By property **(A)**, there are $\tilde{\xi}' \in \varphi(\tilde{\xi})$ and $\varrho' \in \varphi(\varrho)$ such that $\xi' = \tilde{\xi}' \cdot \varrho'$. By construction, the production $A \cdot \mathrm{Id}_0 \to \varrho'$ is contained in P'. Moreover, $\tilde{\xi}' \cdot A \in \varphi(\tilde{\xi} \cdot A)$, so we can apply the induction hypothesis. Thus

$$S \Rightarrow_{G'}^n \tilde{\xi}' \cdot A \Rightarrow_{G'} \tilde{\xi}' \cdot \varrho' = \xi'\,.$$

(II) Consider $k \in \mathbb{N}_1$, $\tilde{\xi} \in \widetilde{\mathrm{T}}(N \cup \Sigma)_1^1$ and $\zeta \in \mathrm{T}(N \cup \Sigma)_0^k$ such that

$$S \Rightarrow_G^n \tilde{\xi} \cdot A \cdot \zeta \Rightarrow_G \tilde{\xi} \cdot \tilde{\varrho} \cdot \sigma \cdot \zeta$$

by some production of form $A \cdot \mathrm{Id}_k \to \tilde{\varrho} \cdot \sigma \cdot \mathrm{Id}_k$, where $\sigma \in \Sigma^{(k)}$. Every $\xi' \in \varphi(\tilde{\xi} \cdot \tilde{\varrho} \cdot \sigma \cdot \zeta)$ is of the form

$$\xi' = \tilde{\xi}' \cdot \tilde{\varrho}' \cdot \sigma \cdot \zeta' \qquad \text{for some} \qquad \tilde{\xi}' \in \varphi(\tilde{\xi})\,,\ \tilde{\varrho}' \in \varphi(\tilde{\varrho})\,,\ \text{and } \zeta' \in \varphi(\zeta)\,.$$

The production $A_\sigma(x) \to \tilde{\varrho}'$ is contained in P'. Furthermore, $\tilde{\xi}' \cdot A_\sigma \cdot \sigma \cdot \zeta' \in \varphi(\tilde{\xi} \cdot A \cdot \zeta)$, so the induction hypothesis can be applied. Hence

$$S \Rightarrow_{G'}^n \tilde{\xi}' \cdot A_\sigma \cdot \sigma \cdot \zeta' \Rightarrow_{G'} \tilde{\xi}' \cdot \tilde{\varrho}' \cdot \sigma \cdot \zeta' = \xi'\,.$$

(III) Let $k \in \mathbb{N}_1$, $\tilde{\xi} \in \widetilde{\mathrm{T}}(N \cup \Sigma)_1^1$ and $\zeta \in \mathrm{T}(N \cup \Sigma)_0^k$ such that

$$S \Rightarrow_G^n \tilde{\xi} \cdot A \cdot \zeta \Rightarrow_G \tilde{\xi} \cdot \tilde{\varrho} \cdot B \cdot \zeta$$

by some production of form $A \cdot \mathrm{Id}_k \to \tilde{\varrho} \cdot B \cdot \mathrm{Id}_k$, where $B \in N^{(k)}$. For every $\xi' \in \varphi(\tilde{\xi} \cdot \tilde{\varrho} \cdot B \cdot \zeta)$, there are

$$\tilde{\xi}' \in \varphi(\tilde{\xi}), \qquad \tilde{\varrho}' \in \varphi(\tilde{\varrho}), \qquad \sigma \in \Sigma^{(k)}, \qquad \text{and} \qquad \zeta' \in \varphi(\zeta)$$

such that

$$\xi' = \tilde{\xi}' \cdot \tilde{\varrho}' \cdot B_\sigma \cdot \sigma \cdot \zeta' .$$

By construction, the production $A_\sigma(x) \to \tilde{\varrho}' \cdot B_\sigma \cdot \mathrm{Id}_1$ is in P'. Moreover, since $\tilde{\xi}' \cdot A_\sigma \cdot \sigma \cdot \zeta' \in \varphi(\tilde{\xi} \cdot A \cdot \zeta)$, the induction hypothesis is applicable, and we obtain that

$$S \Rightarrow_{G'}^n \tilde{\xi}' \cdot A_\sigma \cdot \sigma \cdot \zeta' \Rightarrow_{G'} \tilde{\xi}' \cdot \tilde{\varrho}' \cdot B_\sigma \cdot \sigma \cdot \zeta' = \xi' .$$

* * *

Consider some $t \in \mathcal{L}(G)$. Since $t \in \varphi(t)$, we obtain that $t \in \mathcal{L}(G')$, and therefore $\mathcal{L}(G) \subseteq \mathcal{L}(G')$.

($\supseteq$) The direction $\mathcal{L}(G) \supseteq \mathcal{L}(G')$ rests upon the following property. For every $n \in \mathbb{N}$, and $\xi' \in \mathrm{T}(N' \cup \Sigma)_0^1$,

$$\text{if} \qquad S \Rightarrow_{G'}^n \xi' , \qquad \text{then} \qquad \exists \xi \in \mathrm{T}(N \cup \Sigma)_0^1 \colon S \Rightarrow_G^n \xi \;\wedge\; \xi' \in \varphi(\xi) .$$

The proof is by weak induction on n, and as the case $n = 0$ is trivial, we head right to the induction step. Let $n \in \mathbb{N}$.

(I) Assume that

$$S \Rightarrow_{G'}^n \tilde{\xi}' \cdot A \Rightarrow_{G'} \tilde{\xi}' \cdot \varrho'$$

for some $A \in N^{(0)}$, $\tilde{\xi}' \in \widetilde{\mathrm{T}}(N' \cup \Sigma)_1^1$, and some production $A \to \varrho'$ in P'. By construction of G', there is some production $A \to \varrho$ in P such that $\varrho' \in \varphi(\varrho)$.

By the induction hypothesis, there is some $\xi \in \mathrm{T}(N \cup \Sigma)_0^1$ such that $\tilde{\xi}' \cdot A \in \varphi(\xi)$. The form of φ implies the existence of some $\tilde{\xi} \in \widetilde{\mathrm{T}}(N \cup \Sigma)_1^1$ such that $\xi = \tilde{\xi} \cdot A$ and $\tilde{\xi}' \in \varphi(\tilde{\xi})$. Moreover,

$$S \Rightarrow_G^n \tilde{\xi} \cdot A \Rightarrow_G \tilde{\xi} \cdot \varrho ,$$

and clearly, $\tilde{\xi}' \cdot \varrho' \in \varphi(\tilde{\xi} \cdot \varrho)$.

(II) Let $k \in \mathbb{N}_1$, $\tilde{\xi}' \in \widetilde{\mathrm{T}}(N' \cup \Sigma)_1^1$, $A \in N^{(k)}$, $\sigma \in \Sigma^{(k)}$, and $\zeta' \in \mathrm{T}(N' \cup \Sigma)_0^k$ such that

$$S \Rightarrow_{G'}^n \tilde{\xi}' \cdot A_\sigma \cdot \zeta' \Rightarrow_{G'} \tilde{\xi}' \cdot \tilde{\varrho}' \cdot \zeta'$$

by some production of form $A_\sigma \cdot \mathrm{Id}_1 \to \tilde{\varrho}' \cdot \mathrm{Id}_1$ created according to rule *(ii)* from above. Then there is some production $A \cdot \mathrm{Id}_k \to \tilde{\varrho} \cdot \sigma \cdot \mathrm{Id}_k$ in P with $\tilde{\varrho}' \in \varphi(\tilde{\varrho})$.

By the induction hypothesis, there is some $\xi \in \mathrm{T}(N \cup \Sigma)_0^1$ with $\tilde{\xi}' \cdot A_\sigma \cdot \zeta' \in \varphi(\xi)$. From the definition of φ, we see that there are $\zeta'' \in \mathrm{T}(N' \cup \Sigma)_0^k$ such that $\zeta' = \sigma \cdot \zeta''$, as well as $\tilde{\xi}$, $\zeta \in \mathrm{T}(N \cup \Sigma)$ with $\tilde{\xi}' \in \varphi(\tilde{\xi})$ and $\zeta'' \in \varphi(\zeta)$. Moreover,

$$S \Rightarrow_G^n \tilde{\xi} \cdot A \cdot \zeta \Rightarrow_G \tilde{\xi} \cdot \tilde{\varrho} \cdot \sigma \cdot \zeta\,.$$

Then, using property **(A)**,

$$\tilde{\xi}' \cdot \tilde{\varrho}' \cdot \zeta' = \tilde{\xi}' \cdot \tilde{\varrho}' \cdot \sigma \cdot \zeta'' \in \varphi(\tilde{\xi} \cdot \tilde{\varrho} \cdot \sigma \cdot \zeta)\,.$$

(III) Finally, let $k \in \mathbb{N}_1$, $\tilde{\xi}' \in \widetilde{\mathrm{T}}(N' \cup \Sigma)_1^1$, $A, B \in N^{(k)}$, $\sigma \in \Sigma^{(k)}$, and $\zeta' \in \mathrm{T}(N' \cup \Sigma)_0^k$ such that

$$S \Rightarrow_{G'}^n \tilde{\xi}' \cdot A_\sigma \cdot \zeta' \Rightarrow_{G'} \tilde{\xi}' \cdot \tilde{\varrho}' \cdot B_\sigma \cdot \zeta'$$

by a production of form $A_\sigma \cdot \mathrm{Id}_1 \to \tilde{\varrho}' \cdot B_\sigma \cdot \mathrm{Id}_1$, created according to rule *(iii)*.

Following the same reasoning as above in case **(II)**, there are $\zeta'' \in \mathrm{T}(N' \cup \Sigma)_0^k$, as well as $\tilde{\xi}$, $\zeta \in \mathrm{T}(N \cup \Sigma)$ such that

$$\tilde{\xi}' \in \varphi(\tilde{\xi})\,, \qquad \zeta'' \in \varphi(\zeta)\,, \qquad \text{and} \qquad \zeta' = \sigma \cdot \zeta''\,.$$

Moreover,

$$S \Rightarrow_G^n \tilde{\xi} \cdot A \cdot \zeta \Rightarrow_G \tilde{\xi} \cdot \tilde{\varrho} \cdot B \cdot \zeta\,,$$

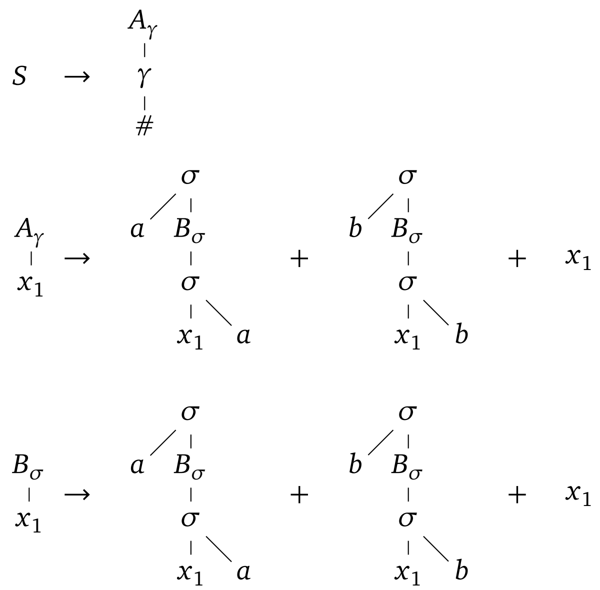
and

$$\tilde{\xi}' \cdot \tilde{\varrho}' \cdot B_\sigma \cdot \zeta' = \tilde{\xi}' \cdot \tilde{\varrho}' \cdot B_\sigma \cdot \sigma \cdot \zeta'' \in \varphi(\tilde{\xi} \cdot \tilde{\varrho} \cdot B \cdot \zeta)\,.$$

* * *

Consider some $t \in \mathcal{L}(G')$, and observe that whenever $t \in \varphi(\xi)$ for some $\xi \in \mathrm{T}(N \cup \Sigma)$, then $\xi = t$. Thus, the property proven above implies that $t \in \mathcal{L}(G)$, and hence $\mathcal{L}(G') \subseteq \mathcal{L}(G)$. □

Example 6.4. Recall the cftg G from Example 6.1. When we apply the construction from Lemma 6.2, we obtain the lm-cftg $G' = (N', \Sigma, S, P')$ with $N' = \{S, A_\gamma^{(1)}, B_\sigma^{(1)}\}$ and the productions in P' given as follows.



It is easy to check that $\mathcal{L}(G') = \mathcal{L}(G)$. The example substantiates our above claim that the shape of productions is largely preserved. The construction merely "cuts" away the foot nodes of every right-hand side, and replaces nonterminal nodes A with subtrees of the form $A_\sigma \cdot \sigma$. $\lhd$

The main theorem is a direct consequence of Lemmas 6.2 and 6.3.

Theorem 6.5. *Let L be a tree language. The following are equivalent:*

1. *L is a linear monadic context-free tree language.*
2. *L is generated by some footed context-free tree grammar.*

6.2 Chapter Conclusion

In this section, we have reproven the equivalence of footed cftg and linear monadic cftg. The construction of an equivalent lm-cftg from a footed cftg is direct, in contrast to the original proof by Kepser and Rogers.

The same construction as in our proof has been used in [53, Thm. 61] to show that multiple context-free tree grammars have the same tree-generating power as monadic multiple context-free tree grammars.

Conclusion

Des is wia bei jeda Wissenschaft,
am Schluss stellt sich dann heraus,
dass ois ganz anders war.

(Karl Valentin)

With this work, we have tried to add some new entries to the list of established results on context-free tree languages. Let us give a brief overview on what has been done. After recalling some preliminaries from mathematics and theoretical computer science in Chapter 1, we called to mind the definition of context-free tree grammars in Chapter 2. The definition has been framed using notation established in the context of algebraic structures called magmoids. Moreover, we re-stated various results on cftg, as e.g. a production interchange lemma, a parallel derivation lemma, theorems on closure properties, on decision procedures, and the theorem of equivalence with pushdown tree automata. To keep the thesis self-contained, many of these properties have been reproven, or the constructions have at least been displayed without proof. A whole subsection of Chapter 2 has been devoted to linear cftg, and their relationship to nonlinear cftg. We have reproved that the latter generate a strictly larger class of tree languages, using an asymptotic growth argument based on the combinatorics of sentential forms of linear cftg. The chapter concludes with an overview of literature on cftg, which might be interesting in itself.

Chapter 3 is on the computational complexity of decision problems of cftg. The main effort in this chapter lies in finding a decision procedure for the uniform membership problem of cftg. While the idea of the construction goes back to Aho's proof that the indexed languages are context-sensitive [3], we frame the construction in terms of pushdown tree automata, and prove its correctness formally. Further, the chapter contains new results on the complexity of decision problems of linear (and nondeleting) cftg.

In Chapter 4, we have strengthened a classical non-closure result on cftg. We show that there is a linear context-free tree language whose preimage under a particular linear tree homomorphism is not context-free. Therefore, the class CFT_ℓ is not closed under inverse linear tree homomorphisms, and neither under extended top-down tree transductions. However, when one considers only linear monadic cftg, then closure under inverse linear tree homomorphisms can be established. The chapter ends with a conjectured witness for the non-closure of the 2-adic linear context-free tree languages under inverse linear tree homomorphisms.

Chapter 5 is concerned with weighted synchronous context-free tree grammars. These grammars define weighted tree transformations, and are a generalization to arbitrary complete semirings of the model introduced by Nederhof and Vogler in [124]. We have identified a syntactic restriction of wscftg, and have characterized it by means of a novel type of pushdown transducer, called weighted pushdown extended top-down tree transducer.

Finally, Chapter 6 covers the relationship between tree-adjoining grammars, their cftg counterparts, called footed cftg, and linear monadic cftg. We have given a direct proof of the fact that for every footed cftg, there is an equivalent lm-cftg. The construction behind the proof leaves the shape of the elementary trees largely intact.

Index

Bibliography

[1] S. Aaronson. $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$. `http://www.scottaaronson.com/papers/pnp.pdf`, 2017.

[2] A. Aho and J. Ullman. *Foundations of Computer Science: C Edition*. Principles of Computer Science. W. H. Freeman, 1994.

[3] A. V. Aho. Indexed Grammars—An Extension of Context-Free Grammars. *Journal of the ACM*, 15(4):647–671, 1968.

[4] A. V. Aho. Nested Stack Automata. *Journal of the ACM*, 16(3):383–406, 1969.

[5] A. V. Aho and J. D. Ullman. Properties of Syntax Directed Translations. *Journal of Computer and System Sciences*, 3(3):319–334, 1969.

[6] A. V. Aho and J. D. Ullman. Syntax Directed Translations and the Pushdown Assembler. *Journal of Computer and System Sciences*, 3(1):37–56, 1969.

[7] A. V. Aho and J. D. Ullman. Translations on a Context Free Grammar. *Information and Control*, 19(5):439–475, 1971.

[8] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, 1972.

[9] A. Arnold. *Systèmes d'Equations dans le Magmoide – Ensembles Rationnels et Algébriques d'Arbres*. PhD thesis, Université de Lille, 1977.

[10] A. Arnold and M. Dauchet. Translations de Forêts Reconaissables Monadiques; Forêts Corégulières. *RAIRO*, 10:5–21, 1976.

[11] A. Arnold and M. Dauchet. Un Théorème de Duplication Pour Les Forêts Algébriques. *Journal of Computer and System Sciences*, 13(2):223–244, 1976.

[12] A. Arnold and M. Dauchet. Theorie des Magmoïdes. Technical report, Université de Lille, 1977. Common PhD Thesis Preliminaries.

[13] A. Arnold and M. Dauchet. Un Théorème de Chomsky-Schützenberger pour les Forêts Algébriques. *Calcolo*, 14(2):161–184, 1977.

[14] A. Arnold and M. Dauchet. Forêts Algébriques et Homomorphismes Inverses. *Information and Control*, 37(2):182–196, 1978.

[15] A. Arnold and M. Dauchet. Théorie des Magmoïdes (I). *RAIRO – Theoretical Informatics and Applications - Informatique Théorique et Applications*, 12(3):235–257, 1978.

[16] A. Arnold and M. Dauchet. Théorie des magmoïdes (II). *RAIRO – Theoretical Informatics and Applications - Informatique Théorique et Applications*, 13(2):135–154, 1979.

[17] A. Arnold and M. Dauchet. Morphismes et Bimorphismes d'Arbres. *Theoretical Computer Science*, 20:33–93, 1982.

[18] A. Arnold and B. Leguy. Une Propriété des Forêts Algébriques «de Greibach». *Information and Control*, 46(2):108–134, 1980.

[19] A. Arnold and M. Nivat. Formal Computations of Non Deterministic Recursive Program Schemes. *Mathematical Systems Theory*, 236:219–236, 1979.

[20] P. R. Asveld. Time and Space Complexity of Inside-Out Macro Languages. *International Journal of Computer Mathematics*, 10(1):3–14, 1981.

[21] J.-M. Autebert, J. Berstel, and L. Boasson. Context-Free Languages and Pushdown Automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, chapter 3, page 111–174. Springer, 1997.

[22] Y. Bar-Hillel, M. A. Perles, and E. Shamir. On Formal Properties of Simple Phrase Structure Grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172, 1961.

[23] F. Bossut. *Rationalité et Reconnaissabilité dans des Graphes*. PhD thesis, Université de Lille, 1986.

[24] S. Bozapalidis. Context-Free Series on Trees. *Information and Computation*, 169(2):186–229, 2001.

[25] S. Bozapalidis and A. Kalampakas. Pattern Language Recognition and Generation. *Pure Mathematics and Applications*, 22, 2011.

[26] W. S. Brainerd. Tree Generating Regular Systems. *Information and Control*, 14(2):217–231, 1969.

[27] J. Bresnan, R. M. Kaplan, S. Peters, and A. Zaenen. Cross-Serial Dependencies in Dutch. In W. J. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, editors, *The Formal Complexity of Natural Language*, page 286–319. Springer, 1982.

[28] M. Büchse, A. Maletti, and H. Vogler. Unidirectional Derivation Semantics for Synchronous Tree-Adjoining Grammars. In H. Yen and O. H. Ibarra, editors, *Proceedings of the 16th International Conference on Developments in Language Theory*, pages 1067–1076, 2012.

[29] J. R. Büchi. Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92, 1960.

[30] G. Cantor. *Contributions to the Founding of the Theory of Transfinite Numbers*. Dover, 1915.

[31] N. Chomsky. Three Models for the Description of Language. *IRE Transactions on Information Theory*, 2:113–124, 1956.

[32] N. Chomsky. On Certain Formal Properties of Grammars. *Information and Control*, 2(2):137–167, 1959.

[33] N. Chomsky and M.-P. Schützenberger. The Algebraic Theory of Context-Free Languages. *Studies in Logic and the Foundations of Mathematics*, page 118–161, 1963.

[34] S. A. Cook. The Complexity of Theorem-Proving Procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[35] K. Čulík. Semantics and Translation of Grammars and ALGOL-like Languages. *Kybernetika*, 1(1):47–49, 1965.

[36] C. Culy. The Complexity of the Vocabulary of Bambara. In W. J. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, editors, *The Formal Complexity of Natural Language*, page 349–357. Springer, 1985.

[37] W. Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20(2):95–207, 1982.

[38] W. Damm and A. Goerdt. An Automata-Theoretical Characterization of the OI-Hierarchy. *Information and Control*, 71(1/2):1–32, 1986.

[39] M. Dauchet and S. Tison. Structural Complexity of Classes of Tree Languages. In M. Nivat and A. Podelski, editors, *Tree Automata and Languages*, pages 327–353. Elsevier, 1992.

[40] O. Deiser. *Einführung in die Mengenlehre*. Springer, 2010.

[41] J. Doner. Tree Acceptors and some of their Applications. *Journal of Computer and System Sciences*, 4(5):406–451, 1970.

[42] J. E. Doner. Decidability of the Weak Second-Order Theory of Two Successors. *Notices of the American Mathematical Society*, 12:365–468, 1965.

[43] C. Doran, D. Egedi, B. A. Hockey, B. Srinivas, and M. Zaidel. XTAG System – A Wide Coverage Grammar for English. In *Proceedings of the 15th Conference on Computational Linguistics*, pages 922–928, 1994.

[44] P. J. Downey. Formal Languages and Recursion Schemes. Technical Report TR-16-74, Harvard University, 1974.

[45] M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2009.

[46] J. Duske, R. Parchmann, and J. Specht. A Homomorphic Characterization of Indexed Languages. *Elektronische Informationsverarbeitung und Kybernetik*, 15(4):187–195, 1979.

[47] S. Eilenberg. *Automata, Languages, and Machines Vol. A*. Pure and Applied Mathematics. Elsevier, 1974.

[48] J. Eisner. Learning Non-Isomorphic Tree Mappings for Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 205–208, 2003.

[49] J. Engelfriet. Bottom-Up and Top-Down Tree Transformations—A Comparison. *Mathematical Systems Theory*, 9(2):198–231, 1975.

[50] J. Engelfriet. Top-Down Tree Transducers with Regular Look-Ahead. *Mathematical Systems Theory*, 10:289–303, 1976.

[51] J. Engelfriet. Context-Free Grammars with Storage. Technical report, Rijksuniversiteit Leiden, 1986.

[52] J. Engelfriet. Tree Automata and Tree Grammars. arXiv:1510.02036 [cs.FL], 2015.

[53] J. Engelfriet, A. Maletti, and S. Maneth. Multiple Context-Free Tree Grammars: Lexicalization and Characterization. arXiv:1707.03457v1 [cs.FL], 2017.

[54] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree Transducers, L Systems, and Two-Way Machines. *Journal of Computer and System Sciences*, 20(2):150–202, 1980.

[55] J. Engelfriet and E. M. Schmidt. IO and OI. I. *Journal of Computer and System Sciences*, 15(3):328–353, 1977.

[56] J. Engelfriet and E. M. Schmidt. IO and OI. II. *Journal of Computer and System Sciences*, 16(1):67–99, 1978.

[57] J. Engelfriet and S. Skyum. Copying Theorems. *Information Processing Letters*, 4(6):157–161, 1976.

[58] J. Engelfriet and H. Vogler. Macro Tree Transducers. *Journal of Computer and System Sciences*, 31(1):71–146, 1985.

[59] J. Engelfriet and H. Vogler. Pushdown Machines for the Macro Tree Transducer. *Theoretical Computer Science*, 42(3):251–368, 1986.

[60] M. J. Fischer. *Grammars with Macro-Like Productions*. Phd thesis, Harvard University, 1968.

[61] M. J. Fischer. Grammars with Macro-Like Productions. In *IEEE Conference Record of 9th Annual Symposium on Switching and Automata Theory*, pages 131–142. IEEE, 1968.

[62] S. Fratani and E. M. Voundy. Homomorphic Characterizations of Indexed Languages. In A.-H. Dediu, J. Janoušek, C. Martín-Vide, and B. Truthe, editors, *Proceedings of the 10th International Conference on Language and Automata Theory and Applications*, pages 359–370. Springer, 2016.

[63] A. Fujiyoshi. Analogical Conception of Chomsky Normal Form and Greibach Normal Form for Linear, Monadic Context-Free Tree Grammars. *IEICE Transactions on Information and Systems*, E89-D(12):2933–2938, 2006.

[64] A. Fujiyoshi. Linear-Time Recognizable Classes of Tree Languages by Deterministic Linear Pushdown Tree Automata. *IEICE Transactions*, 92-D(2):248–254, 2009.

[65] A. Fujiyoshi and T. Kasai. Spinal-Formed Context-Free Tree Grammars. *Theory of Computing Systems*, 33(1):59–83, 2000.

[66] Z. Fülöp, A. Maletti, and H. Vogler. Weighted Extended Tree Transducers. *Fundamenta Informaticae*, 111:163–202, 2011.

[67] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[68] G. Gazdar. Applicability of Indexed Grammars to Natural Languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. Springer, 1988.

[69] G. Gazdar and G. K. Pullum. Computationally Relevant Properties of Natural Languages and Their Grammars. In W. J. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, editors, *The Formal Complexity of Natural Language*, page 387–437. Springer, 1985.

[70] K. Gebhardt and J. Osterholzer. A Direct Link between Tree-Adjoining and Context-Free Tree Grammars. In T. Hanneforth and C. Wurm, editors, *Proceedings of the 12th International Conference on Finite-State Methods and Natural Language Processing*. ACL, 2015.

[71] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[72] F. Gécseg and M. Steinby. Tree Languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer, 1997.

[73] S. Ginsburg, S. A. Greibach, and M. A. Harrison. One-Way Stack Automata. *Journal of the ACM*, 14(2):389–418, 1967.

[74] S. Ginsburg and H. G. Rice. Two Families of Languages Related to ALGOL. *Journal of the ACM*, 9(3):350–371, 1962.

[75] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. Initial Algebra Semantics and Continuous Algebras. *Journal of the ACM*, 24(1):68–95, 1977.

[76] G. Grätzer. *Universal Algebra*. Springer, 2008.

[77] S. A. Greibach. A New Normal-Form Theorem for Context-Free Phrase Structure Grammars. *Journal of the ACM*, 12(1):42–52, 1965.

[78] I. Guessarian. *Algebraic Semantics*, volume 99 of *Lecture Notes in Computer Science*. Springer, 1981.

[79] I. Guessarian. Pushdown Tree Automata. *Mathematical Systems Theory*, 16(1):237–263, 1983.

[80] M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

[81] U. Hebisch and H. J. Weinert. *Semirings*. World Scientific, 1998.

[82] L. Herrmann and H. Vogler. A Chomsky-Schützenberger Theorem for Weighted Automata with Storage. In A. Maletti, editor, *Proceedings of the 6th International Conference on Algebraic Informatics*, pages 115–127. Springer, 2015.

[83] J. Higginbotham. English is Not a Context-Free Language. In W. J. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, editors, *The Formal Complexity of Natural Language*, page 335–348. Springer, 1984.

[84] D. Hofbauer, M. Huber, and G. Kucherov. Some Results on Top-Context-Free Tree Languages. In S. Tison, editor, *Proceedings of the 19th International Colloquium on Trees in Algebra and Programming*, pages 157–171. Springer, 1994.

[85] D. Hofbauer, M. Huber, and G. Kucherov. How to Get Rid of Projection Rules in Context-Free Tree Grammars. In J. Ginzburg, Z. Khasidashvili, C. Vogel, and J.-J. Levy, editors, *Studies in Logic, Language and Information*, pages 235–247. Center for the Study of Language and Information and The European Association for Logic, Language and Information, 1998.

[86] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, first edition, 1979.

[87] K. Inaba and S. Maneth. The Complexity of Tree Transducer Output Languages. In *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 244–255, 2008.

[88] E. T. Irons. A Syntax Directed Compiler for ALGOL 60. *Communications of the ACM*, 4(1):51–55, 1961.

[89] A. Jeż and M. Lohrey. Approximation of Smallest Linear Tree Grammar. *Information and Computation*, 251:215–251, 2016.

[90] A. K. Joshi. Tree Adjoining Grammars: How Much Context-Sensitivity is Required to Provide Reasonable Structural Descriptions? In D. R. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, 1985.

[91] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975.

[92] A. K. Joshi and Y. Schabes. Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 2, pages 69–123. Springer, 1997.

[93] A. K. Joshi, K. Vijay-Shanker, and D. Weir. The Convergence Of Mildly Context-Sensitive Grammar Formalisms. Technical report, University of Pennsylvania, 1990.

[94] L. Kallmeyer. On the Mild Context-Sensitivity of k-Tree Wrapping Grammar. In A. Foret, G. Morrill, R. Muskens, R. Osswald, and S. Pogodalla, editors, *Proceedings of the 20th and 21st International Conferences on Formal Grammar*, pages 77–93. Springer, 2016.

[95] M. Kanazawa. Context-Free Tree Grammars. Lecture notes. `http://research.nii.ac.jp/~kanazawa/Courses/2011/Kyoto/cft.pdf`, 2012.

[96] M. Kanazawa. A Generalization of Linear Indexed Grammars Equivalent to Simple Context-Free Tree Grammars. In G. Morrill, R. Muskens, R. Osswald, and F. Richter, editors, *Proceedings of the 19th International Conference on Formal Grammar*, pages 86–103, 2014.

[97] M. Kanazawa. Multidimensional Trees and a Chomsky-Schützenberger-Weir Representation Theorem for Simple Context-Free Tree Grammars. *Journal of Logic and Computation*, 26(5):1469–1516, 2014.

[98] L. Kari, G. Rozenberg, and A. Salomaa. L Systems. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, chapter 5, pages 253–328. Springer, 1997.

[99] S. Kepser and U. Mönnich. Closure Properties of Linear Context-Free Tree Languages with an Application to Optimality Theory. *Theoretical Computer Science*, 354(1):82–97, 2006.

[100] S. Kepser and J. Rogers. The Equivalence of Tree Adjoining Grammars and Monadic Linear Context-free Tree Grammars. *Journal of Logic, Language and Information*, 20(3):361–384, 2011.

[101] S. C. Kleene. Representation of Events in Nerve Nets and Finite Automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[102] T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-Order Pushdown Trees Are Easy. In M. Nielsen and U. Engberg, editors, *Proceedings of the 5th International Conference on Foundations of Software Science and Computation*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002.

[103] G. M. Kobele and S. Salvati. The IO and OI Hierarchies Revisited. *Information and Computation*, 243:205–221, 2015.

[104] D. Kozen. Lower Bounds for Natural Proof Systems. In *Proceedings of the 18th Symposium on the Foundations of Computer Science*, pages 254–266, 1977.

[105] D. Kozen. On the Myhill-Nerode Theorem for Trees. *Bulletin of the EATCS*, 47:170–173, 1992.

[106] D. Kozen. *Automata and Computability*. Springer, New York, 1997.

[107] W. Kuich. Semirings and Formal Power Series: Their Relevance to Formal Languages and Automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, chapter 9, pages 609–677. Springer, 1997.

[108] C. Lautemann, T. Schwentick, and D. Thérien. Logics for Context-Free Languages. In L. Pacholski and J. Tiuryn, editors, *Selected Papers from the 8th Workshop on Computer Science Logic*. Springer, 1995.

[109] B. Leguy. *Reductions, Transformations et Classification des Grammaires Algebriques D'Arbres*. PhD thesis, Université de Lille, 1980.

[110] B. Leguy. Grammars Without Erasing Rules. The OI Case. In E. Astesiano and C. Böhm, editors, *Proceedings of the 6th Colloquium on Trees and Algebras in Programming*, pages 268–279. Springer, 1981.

[111] B. Leguy. Reducing Algebraic Tree Grammars. In F. Gécseg, editor, *Proceedings of the International Conference on Fundamentals of Computation Theory*. Springer, 1981.

[112] P. M. Lewis and R. E. Stearns. Syntax-Directed Transduction. *Journal of the ACM*, 18(3):465–488, 1968.

[113] M. Lohrey. On the Parallel Complexity of Tree Automata. In *Proceedings of the 12th International Conference on Rewriting Techniques and Applications*, pages 201–215, 2001.

[114] T. Maibaum. A Generalized Approach to Formal Languages. *Journal of Computer and System Sciences*, 439, 1974.

[115] T. Maibaum. Pumping Lemmas for Term Languages. *Journal of Computer and System Sciences*, pages 319–330, 1978.

[116] A. Maletti. Survey: Weighted Extended Top-down Tree Transducers Part II - Application in Machine Translation. *Fundamenta Informaticae*, 112(2-3):239–261, 2011.

[117] A. Maletti and J. Engelfriet. Strong Lexicalization of Tree Adjoining Grammars. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 506–515, 2012.

[118] A. Maletti, J. Graehl, M. Hopkins, and K. Knight. The Power of Extended Top-Down Tree Transducers. *SIAM Journal on Computing*, 39(2):410–430, 2009.

[119] A. N. Maslov. The Hierarchy of Indexed Languages of an Arbitrary Level. *Soviet Mathematics – Doklady Akademii Nauk SSSR*, 15(5):1170–1174, 1974.

[120] K. Mehlhorn. Parsing Macro Grammars Top Down. *Information and Control*, 143:123–143, 1979.

[121] J. Mezei and J. Wright. Algebraic Automata and Context-Free Sets. *Information and Control*, 11(1-2):3–29, 1967.

[122] U. Mönnich. Adjunction as Substitution: An Algebraic Formulation of Regular, Context-Free and Tree Adjoining Languages. In *Proceedings of the 3rd Conference on Formal Grammar*, 1997.

[123] M.-J. Nederhof, M. Teichmann, and H. Vogler. Non-Self-Embedding Linear Context-Free Tree Grammars Generate Regular Tree Languages. *Journal of Automata, Languages and Combinatorics*, 21(3):203–246, 2016.

[124] M.-J. Nederhof and H. Vogler. Synchronous Context-Free Tree Grammars. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 55–63, 2012.

[125] R. Nesson, S. M. Shieber, and A. Rush. Induction of Probabilistic Synchronous Tree-Insertion Grammars for Machine Translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 128–137, 2006.

[126] M. Nivat. Transductions des Langages de Chomsky. *Annales de l'Institut Fourier*, 18(1):339–455, 1968.

[127] M. Nivat. On the Interpretation of Recursive Program Schemes. Technical report, Fachbereich Angewandte Mathematik und Informatik, Universität des Saarlandes, 1974.

[128] L. Ong. Higher-Order Model Checking: An Overview. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–15. IEEE Computer Society, 2015.

[129] J. Osterholzer. Pushdown Machines for Weighted Context-Free Tree Translation. In M. Holzer and M. Kutrib, editors, *Proceedings of the 19th International Conference on Implementation and Application of Automata*, pages 290–303. Springer, 2014.

[130] J. Osterholzer. Complexity of Uniform Membership of Context-Free Tree Grammars. In A. Maletti, editor, *Proceedings of the 6th International Conference on Algebraic Informatics*, pages 176–188, 2015.

[131] J. Osterholzer, T. Dietze, and L. Herrmann. Linear Context-Free Tree Languages and Inverse Homomorphisms. arXiv:1510.04881 [cs.FL], 2015.

[132] J. Osterholzer, T. Dietze, and L. Herrmann. Linear Context-Free Tree Languages and Inverse Homomorphisms. In A.-H. Dediu, J. Janoušek, C. Martín-Vide, and B. Truthe, editors, *Proceedings of the 10th International Conference on Language and Automata Theory and Applications*, pages 478–489, 2016.

[133] J. Osterholzer, T. Dietze, and L. Herrmann. Linear Context-Free Tree Languages and Inverse Homomorphisms. *Information and Computation*, 2017. Accepted for publication.

[134] C. H. Papadimitriou. *Computational Complexity*. John Wiley and Sons Ltd., 2003.

[135] L. Petrone. Syntax Directed Mappings of Context-Free Languages. In *IEEE Conference Record of 9th Annual Symposium on Switching and Automata Theory*, pages 160–175, 1968.

[136] G. K. Pullum and G. Gazdar. Natural Languages and Context-Free Languages. In W. J. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, editors, *The Formal Complexity of Natural Language*, page 138–182. Springer, 1987.

[137] M. O. Rabin and D. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[138] H. G. Rice. Classes of Recursively Enumerable Sets and their Decision Problems. *Transactions of the American Mathematical Society*, 74(2):358–358, Feb 1953.

[139] W. C. Rounds. Context-Free Grammars on Trees. In *Proceedings of the First Annual ACM Symposium on Theory of Computing*, pages 143–148, 1969.

[140] W. C. Rounds. Mappings and Grammars on Trees. *Theory of Computing Systems*, 4(3):257–287, 1970.

[141] W. C. Rounds. Tree-Oriented Proofs of Some Theorems on Context-Free and Indexed Languages. In *Proceedings of the Second Annual ACM Symposium on Theory of Computing*, pages 109–116, 1970.

[142] W. C. Rounds. Complexity of Recognition in Intermediate Level Languages. *IEEE Conference Record of 14th Annual Symposium on Switching and Automata Theory*, 1973.

[143] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[144] W. J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[145] W. J. Savitch. Context-Sensitive Grammar and Natural Language Syntax. In W. J. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, editors, *The Formal Complexity of Natural Language*, page 358–368. Springer, 1987.

[146] K. M. Schimpf and J. H. Gallier. Tree Pushdown Automata. *Journal of Computer and System Sciences*, 30(1):25–40, 1985.

[147] M.-P. Schützenberger. On the Definition of a Family of Automata. *Information and Control*, 4:245–270, 1961.

[148] M.-P. Schützenberger. On Context-Free Languages and Push-Down Automata. *Information and Control*, 6(3):246–264, 1963.

[149] H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On Multiple Context-Free Grammars. *Theoretical Computer Science*, 88(2):191–229, 1991.

[150] J. Shallit. Open Problems in Automata Theory: An Idiosyncratic View. LMS Keynote Address in Discrete Mathematics, British Colloquium for Theoretical Computer Science, 2014.

[151] E. Shamir. A Representation Theorem for Algebraic and Context-Free Power Series in Noncommuting Variables. *Information and Control*, 11(1-2):239–254, 1967.

[152] S. M. Shieber. Evidence Against the Context-Freeness of Natural Language. In W. J. Savitch, E. Bach, W. Marsh, and G. Safran-Naveh, editors, *The Formal Complexity of Natural Language*, page 320–334. Springer, 1985.

[153] S. M. Shieber. Unifying Synchronous Tree-Adjoining Grammars and Tree Transducers via Bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 377–384, 2006.

[154] S. M. Shieber and Y. Schabes. Synchronous Tree-Adjoining Grammars. *Proceedings of the 13th Conference on Computational Linguistics*, 3:253–258, 1990.

[155] T. Smith. A New Pumping Lemma for Indexed Languages, with an Application to Infinite Words. *Information and Computation*, 252:176–186, 2017.

[156] H. Stamer. *Restarting Tree Automata: Formal Properties and Possible Variations*. PhD thesis, Universität Kassel, 2009.

[157] S. Tanaka and T. Kasai. The Emptiness Problem for Indexed Language is Exponential-Time Complete. *Systems and Computers in Japan*, 17(9):29–37, 1986.

[158] M. Teichmann. *Expressing Context-Free Tree Languages by Regular Tree Grammars*. PhD thesis, Technische Universität Dresden, 2017.

[159] J. W. Thatcher. Characterizing Derivation Trees of Context-Free Grammars Through a Generalization of Finite Automata Theory. *Journal of Computer and System Sciences*, 1(4):317–322, 1967.

[160] J. W. Thatcher. Generalized^2 Sequential Machine Maps. *Journal of Computer and System Sciences*, 4(4):339–367, 1970.

[161] J. W. Thatcher and J. B. Wright. Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.

[162] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(1):230–265, 1937.

[163] I. Walukiewicz. Automata Theory and Higher-Order Model-Checking. *SIGLOG News*, 3(4):13–31, 2016.

[164] W. Wechler. *Universal Algebra for Computer Scientists*. Springer, 1992.

[165] K. Yamasaki. Fundamental Properties of Pushdown Tree Transducers. *IEICE Transactions on Information and Systems*, E76-D(10):1234 – 1242, 1993.