Expressing Context-Free Tree Languages by Regular Tree Grammars

Dissertation

zur Erlangung des akademischen Grades
Doktor rerum naturalium (Dr. rer. nat.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Inf. Markus Teichmann
geboren am 15. September 1989 in Halle (Saale)

eingereicht am 28. Oktober 2016
verteidigt am 12. April 2017

Betreuer:
Prof. Dr.-Ing. habil. Dr. h.c./Univ. Szeged Heiko Vogler
Technische Universität Dresden

Gutachter:
Prof. Dr.-Ing. habil. Dr. h.c./Univ. Szeged Heiko Vogler
Technische Universität Dresden

Prof. Dr. Frank Drewes
Umeå University

Abstract

In this thesis, three methods are investigated to express context-free tree languages by regular tree grammars. The first method is a characterization. We show restrictions to context-free tree grammars such that, for each restricted context-free tree grammar, a regular tree grammar can be constructed that induces the same tree language. The other two methods are approximations. An arbitrary context-free tree language can be approximated by a regular tree grammar with a restricted pushdown storage. Furthermore, we approximate weighted context-free tree languages, induced by weighted linear nondeleting context-free tree grammars, by showing how to approximate optimal weights for weighted regular tree grammars.

Contents

1 Introduction

The motivation for this thesis stems from the field of *natural language processing*. In particular, we consider the area of *machine translation*. The goal of this area is to translate sentences of a natural language as described in the following. An input sentence in one natural language, e.g., English, is given. This input sentence is transformed into a sentence of a second language, e.g., French. The machine translation process is split into three steps:

- *Modeling*: The data structure and algorithms used for the translation are specified.
- *Training*: The model is fitted to best suit given training data.
- *Decoding*: Given an input sentence, a translation is obtained.

Formal grammars can be used through all three steps. There are many ways to model the structure of one language and to model the transformation of sentences into their translation. There is the well-known *source-channel model* (cf., e.g. [7, Sec. 2]) which considers the target sentence as a transformation of the source sentence through a noisy channel. The goal is to retrieve the source sentence given the target sentence (effectively reversing source and target language). Furthermore, the translation in this model is split into (i) a *translation model* between the two natural languages and (ii) a *language model* of the source side.

The translation model describes how sentences are translated. It can be realized as a synchronous grammar which synchronously generates a pair of sentences; one in each language. A prominent example is a synchronous context-free string grammar (cf. *transduction grammar* in [42]). The language model determines the grammaticality of a given natural language sentence. A variety of grammar formalisms have been proposed as the basis for a language model.

In this work, we relate two specific formal grammars, viz. context-free tree grammars and regular tree grammars. The choice in formalisms is motivated by the following two contradicting objectives in natural language processing:

- Maximizing the expressive power of the applied formalisms to model complex linguistic features, and
- minimizing the computational complexity to allow for efficient training and decoding.

Context-free tree grammars can express some interesting linguistic phenomena, e.g. cross-serial dependencies or unbounded scrambling. Furthermore, they generalize existing formalisms. However, they cannot be used for efficient decoding. In contrast, regular tree grammars are less expressive but require less computational effort for decoding. In Section 1.1, the resulting tradeoff is explained by showing different grammar formalisms, their ability to model language phenomena and the required computational power for decoding. We will present methods how the language induced by an expressive grammar formalism can

be expressed by a computational easier one. One possibility is to (automatically) analyze whether a specific instance of a grammar formalism does indeed utilize its full expressive capabilities. If this is not the case, the modeled formal language can be transformed into one that can be handled more easily. This is outlined in more detail in Section 1.2. Furthermore, in Section 1.3 we describe that a computationally favorable grammar formalism can be used to approximate a more expressive one that is used to allow detailed modeling. Lastly, we outline how weights for approximations can be obtained in Section 1.4. All three methods have been investigated for string grammars. In this thesis, we lift the results to tree-generating formalisms, viz. context-free tree grammars and regular tree grammars.

1.1 Tradeoff between Expressiveness and Computational Complexity

We describe the tradeoff between expressiveness and computational complexity in the choice of grammar formalisms. For this, we outline how grammar formalisms evolved in research, show their limitations and their computational complexity. The resulting tradeoff motivates the choice in formalisms for this thesis and why it is worthwhile to express context-free tree languages by regular tree grammars.

Chomsky [11] defined a well-known hierarchy of grammar formalisms. *Regular string grammars (REGs)* are the simplest proposed form of a grammar. They generate regular string languages which can be parsed in linear time based on the size of the input (cf. e.g. [30, Sec. 4.3.3]). Regular string grammars have a very restricted expressive power and fail to describe certain languages with an unbounded number of long-distance relations between symbols, e.g., the language of all palindromes (cf., e.g., [30, Sec. 5.1.1]). It has been shown that English contains such non-local dependencies and thus, regular grammars are not well-suited to describe natural languages [9].

Non-local dependencies can partly be modeled by *context-free string grammars (CFGs)*. For example, we present the CFG M_1 with the rules

$$B_0 \to aB_0a \mid bB_0b \mid a \mid b \mid \varepsilon$$

where B_0 is a nonterminal, a and b are terminals, and ε is the empty word (cf. Section 2.1 for a formal definition of CFGs). The CFG M_1 generates all palindromes over the alphabet containing a and b. The parsing time for CFGs is cubic in the size of the input (cf. e.g. [30, Thm. 7.33]). There was a long-lasting discussion whether CFGs can be used to capture all linguistic phenomena [56]. A final negative answer was found by showing features in certain natural languages that cannot be captured by context-free string languages [64]. The example given in [64] is a Swiss-German sentence that has the structure $a^k b^\ell a^k b^\ell$ where a and b are terminals, and k and ℓ are natural numbers. This language is not context-free [30] and employs a linguistic phenomenon called *cross-serial dependency*. Thus, research focused on grammar formalisms that can express more details of a sentence. There are two orthogonal approaches. First, grammar formalisms are considered that induce string languages which are more expressive than context-free string languages. Second, tree-generating grammar formalisms that annotate a sentence with information about its structure are employed. We will present examples of both approaches and show how the computational complexity increases with improved modeling capabilities.

String Grammars Inducing Context-Sensitive String Languages

In the Hierarchy of Chomsky [9], *context-sensitive string languages* have an increased expressive power compared to context-free string languages. *Linear context-free rewriting systems* [69] and *multiple context-free grammars (MCFG)* [63] are two examples of string grammar formalism that can induce string languages which are context-sensitive but not context-free. However, their increased expressive power comes with higher computational cost. For example, MCFGs can be parsed in time $\mathcal{O}\big(n^{(r(M)+1)\cdot\varphi(M)}\big)$ [63, 27] where n denotes the size of the input, M is the grammar under consideration, $r(M)$ denotes the maximal rank of a nonterminal in M, and $\varphi(M)$ is the maximal *fan-out* of a nonterminal in M. We note that CFGs are special MCFGs with rank 2 and fan-out 1. This confirms that the parsing time of CFGs is in $\mathcal{O}(n^3)$. It has been shown that the higher the fan-out of a MCFG, the more expressive is the corresponding grammar [63, Thm. 3.4] while simultaneously the complexity of parsing grows.

Multiple context-free grammars and linear context-free rewriting systems are *mildly context-sensitive* formalisms [69, 62]. Such formalisms allow modeling cross-serial dependencies, have a constant growth, and can be parsed in polynomial time. This is an important restriction for decoding [34, Sec. 6.3.3].

We illustrate how to generate the context-sensitive language containing all words of the form $a^k b^\ell a^k b^\ell$ and simultaneously describe *outside-in macro grammars (MACs)* introduced in [20] as another grammar formalism that can describe context-sensitive string languages. We present the MAC M_2 and let it contain the rules

$$\begin{aligned} A_0 \to A(\varepsilon,\varepsilon)\ , \qquad & A(x_1,x_2) \to A(ax_1,ax_2) \ \mid\ B(x_1,x_2)\ , \\ \text{and} \qquad & B(x_1,x_2) \to B(x_1b,x_2b) \ \mid\ x_1x_2 \end{aligned}$$

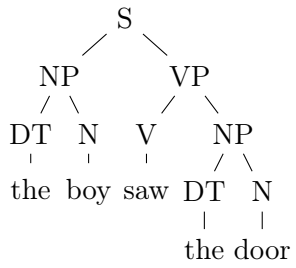
where A_0, A, and B are nonterminals, a and b are terminals, and x_1 and x_2 are variables (cf. Section 3.8 for a formal description of macro grammars). We note that in a MAC, nonterminals are considered as ranked symbols and may have arguments. A derivation step of M_2 applies to an outermost occurrence of $A'(s_1,\dots,s_k)$ of a nonterminal A' and its arguments $s_1,\dots,s_k$ as follows. We replace $A'(s_1,\dots,s_k)$ by the right-hand side of a rule with left-hand side $A(x_1,\dots,x_k)$ where each variable x_i is replaced by s_i. The MAC M_2 generates the word $aab\,aab$ via the derivation

$$A_0 \Rightarrow A(\varepsilon,\varepsilon) \Rightarrow A(a,a) \Rightarrow A(aa,aa) \Rightarrow B(aa,aa) \Rightarrow B(aab,aab) \Rightarrow aab\,aab\ .$$

We note that MACs are not mildly context-sensitive formalisms, since they can violate the constant growth property by duplicating argument values, e.g., using the rule $A(x_1) \to A(x_1x_1)$. Furthermore, it has been shown that parsing of MAC is NP-complete, i.e., there might be no polynomial parsing algorithm (cf., [57, Prop. 2] for indexed languages and [20, Thm. 4.2.8] for equivalence of indexed languages and outside-in macro languages).

Tree-Generating Grammars

We now describe grammar formalisms that augment a sentence with information about its structure by utilizing trees. As an example, we consider the tree (cf. [45, Fig. 3.1])



that represents the sentence 'the boy saw the door' together with its grammatical structure. The sequence of leaves of a tree read from left to right is called the *yield* of that tree. Thus, the sentence is the yield while each interior node is labeled by a syntactic category that is associated with the subtree below the node. For example, the part of the tree rooted in the node labeled VP is a verb phrase, and consists of a verb (V) and a noun phrase (NP). Such structural and grammatical information can improve the language model and help to resolve ambiguity within a sentence. In this thesis, we abstract from syntactic categories and use Greek symbols as nodes of a tree, since we are interested in the capabilities of formal grammars and not in specific details of one natural language.

A CFG can be used as a tree-generating formalism by considering the derivation trees instead of the derived strings. However, using this approach, a CFG can only describe local tree languages [26, p. 33]. Thus, it seems worthwhile to investigate tree-generating formalisms that overcome this limitation. *Regular tree grammars (RTGs)* [6] generate trees that can model non-local dependencies. The right-hand side of a rule in a RTG is a tree whose leaves might contain nonterminals. The tree generated by each nonterminal occurrence replaces the respective nonterminal occurrence. As an example, we present the RTG H_1 that uses nonterminals B_0 and B_1, terminals α, β, σ, and κ, and the rules

$$B_0 \to \overset{\kappa}{\underset{B_1\ B_1}{/\ \backslash}} \quad , \qquad B_1 \to \overset{\sigma}{\underset{\alpha\ B_2}{/\ \backslash}} \;\Big|\; \overset{\sigma}{\underset{B_2\ \beta}{/\ \backslash}} \;\Big|\; \alpha \;\Big|\; \beta \quad , \text{ and} \qquad B_2 \to \overset{\sigma}{\underset{\alpha\ B_1}{/\ \backslash}} \;\Big|\; \overset{\sigma}{\underset{B_1\ \beta}{/\ \backslash}} \quad .$$

We refer to Section 2.2 for a formal definition of RTG. Informally, an occurrence of a nonterminal B' is replaced with the right-hand side of a rule whose left-hand side is B'. A derivation starts from the initial nonterminal B_0 and derives until there are no more nonterminal occurrences. In each rule, nonterminals may only occur at the leaves of the right-hand side. Hence, nonterminals cannot occur nested, i.e., below one another in rules or in a derivation. By a close inspection of the rules of H_1, it can be seen that the yield of each tree in the induced tree language of H_1 has the form $\alpha^k\beta^\ell\alpha^m\beta^n$ where k, ℓ, m, and n are natural numbers such that $k+\ell+m+n \geq 2$ and $k+\ell+m+n$ is even. We note that an even number of leaves requires an even number of σ's in the tree. This is a non-local tree property which cannot be modeled by CFGs considered as a tree-generating formalism. However, it has been proved that the yield of the trees generated by a RTG can also be generated by a CFG [26, Prop. 14.3]. Hence, concerning only the language at the

yield, RTGs are as expressive as CFGs. It is thus not surprising that parsing in a RTG, i.e., finding the set of trees that yield a certain string, can be done in the same time as parsing of that string in a CFG, viz. $\mathcal{O}(n^3)$ [44, Sec. 5.1].

As discussed before, a natural language is not context-free [64] and thus, *tree-adjoining grammars (TAGs)* [35] were considered. They induce a tree language whose yield is a context-sensitive language [35, 33] and they can be parsed in time $\mathcal{O}(n^6)$ [68, Sec. 3.4]. Furthermore, TAGs are a mildly-context-sensitive formalism [34]. Tree adjoining grammars correspond to a special subclass of context-free tree grammars. We will first informally recall context-free tree grammars and then explain the connection to TAG. *Context-free tree grammars (CFTGs)* [58, 18, 26] generalize RTGs by allowing nonterminals to have arguments. Those arguments contain trees over nonterminals and terminals and thus, nonterminals may occur nested. This generalization corresponds to the generalization of CFGs by MACs. We present the CFTG G_1 that uses nonterminals A_0 and A, terminals α, β, σ, and κ, variables x_1, x_2, x_3, and x_4, as well as the four rules

$$A_0 \to A(\alpha, \beta, \alpha, \beta) \qquad A(x_1, x_2, x_3, x_4) \to \sigma(\sigma(\alpha, x_1), \sigma(x_3, \alpha)) \mid A(x_1, \delta(x_3), \delta(x_2, x_4)) \mid \kappa(x_1, x_2, x_3, x_4)$$

where A_0 is the initial nonterminal. We refer to Section 2.2 for a formal definition of a CFTG. Deriving a tree works similarly to the case for MACs. A rule with left-hand side $A'(x_1, \ldots, x_k)$ can be applied at an occurrence of $A'(\xi_1, \ldots, \xi_k)$ where A' is a nonterminal and $\xi_1, \ldots, \xi_k$ are the trees in the argument positions of A. Then this occurrence is replaced by the right-hand side of the rule and each x_i is replaced by ξ_i. For example, the derivation

$$A_0 \Rightarrow A(\alpha, \beta, \alpha, \beta) \Rightarrow \sigma(\sigma(\beta, \alpha), \sigma(\beta, \alpha)) \Rightarrow \sigma(\beta, \sigma(\beta, \sigma(\sigma(\alpha, \alpha), \sigma(\alpha, \alpha)))) \Rightarrow \kappa(\sigma(\beta, \sigma(\beta, \sigma(\sigma(\alpha, \alpha), \sigma(\alpha, \alpha)))))$$

generates a tree with the yield $\alpha^3\beta\alpha^3\beta$. By a close inspection of the rules of G_1, it can be seen that the yield of each tree in the tree language induced by G_1 has the form $\alpha^{k+1}\beta^{\ell+1}\alpha^{k+1}\beta^{\ell+1}$ where k and ℓ are natural numbers. This corresponds to the string $a^k b^\ell a^k b^\ell$ from the Swiss-German example mentioned at the beginning of this section. However, there is more information encoded in the tree. Each time two α's are produced by a rule application, they are linked due to their position in the tree, i.e., the tree encodes that the leftmost α and the rightmost α in the yield correspond. This connection cannot be represented at the level of the yield. Note that in G_1 no argument positions are copied or deleted, i.e., each variable from the left-hand side occurs exactly once in the right-hand side. Thus, we say that G_1 is a *linear nondeleting* CFTG (for short: lnCFTG) [36, 37]. We call a CFTG *linear monadic* if all nonterminals use at most one variable and each variable occurs at most once in the right-hand side of the rules. Monadic CFTGs have been investigated in [22, 23, 43]. Linear monadic CFTGs can express the same tree languages as TAGs [21, 38] and there is a direct construction between both formalisms [24].

It can be seen that CFTGs combine both extensions of CFGs discussed before: The yields of the tree languages induced by CFTGs go beyond context-free string languages and

CFTGs can describe the grammatical structure of a natural language sentence by means of a tree. From this viewpoint, they are well-suited as a formalism for a language model. Furthermore, synchronous CFTGs can be used as a translation model [52, 53]. However, since the yield of a context free tree language (using outside-in derivation mode) is an outside-in macro language (cf. [58, p. 113] and [15, Thm. 7.17]), the NP-hardness result for parsing transfers. This motivates to investigate how the tree languages induced by CFTGs can be expressed by means of RTGs, which are computationally more feasible.

1.2 Characterization in a Less Expressive Formalism

The tradeoff between expressive power of a formalism and the rise in computational complexity, explained in Section 1.1, motivates to investigate whether the expressive power enabled by a formalism is indeed used by a given instance. As an example, we consider the CFG M_3 that contains the rules

$$B_0 \to aB_0a \mid aB_0b \mid bB_0a \mid bB_0b \mid \varepsilon \quad .$$

It is clear that M_3 is not a REG since there is a rule such that in its right-hand side there is a terminal to the right of a nonterminal symbol. However, the language induced by M_3 is the set of all strings s over a and b such that the number of letters in s is even. Trivially, the REG M_4 with the rules

$$B_0 \to aB_1 \mid bB_1 \mid \varepsilon \qquad\qquad B_1 \to aB_0 \mid bB_0$$

induces the same string language. Hence, M_3 does not fully utilize the expressive power enabled CFGs, since there is a REG that induces the same string language. We say that we characterize the context-free language induced by M_3 by the REG M_4 and thus show that the language induced by M_3 is a regular string language.

In general, it is undecidable whether the language induced by a CFG can be recognized by a REG [30, Thm. 8.15]. Thus, it is worthwhile to find a sufficient and decidable criterion for CFGs such that CFGs fulfilling this criterion induce a regular string language. Such a decidable criterion was found by Chomsky in [11]. A CFG is called *self-embedding* if there is a derivation $A \Rightarrow^* sAt$ where A is a nonterminal, and s and t are arbitrary, nonempty strings over terminals and nonterminals. Any CFG that does not fulfill this property induces a regular language [11]. The undecidability result from [30] transfers directly to the tree case, i.e, it is undecidable whether a given CFTG entails a regular tree grammar. In this thesis, criteria for CFTG analogous to self-embedding in the string case are explored, i.e., we investigate under which conditions the tree language induced by a CFTG can be characterized by a RTG. A characterization is the strongest form of expressing a context-free tree language by a RTG, since no information is lost.

Main Contributions. The following list enumerates the main contributions of this thesis regarding the topic described in this section. The first four contributions are mainly based on the article "Non-Self-Embedding Linear Context-Free Tree Grammars Generate Regular Tree Languages" by Nederhof, Teichmann, and Vogler [51]. Thus, these contributions are also attributed to my coauthors.

- I define a decidable property for linear CFTGs called *self-embedding* (cf. Definition 3.1.1 and Corollary 3.1.8).
- I show that the tree language induced by a non-self-embedding linear CFTG can be characterized by a RTG (cf. Theorem 3.3.11 for the case of lnCFTGs and Theorem 3.6.2 for the case of linear CFTGs).
- I define a decidable property for CFTGs called *weakly-self-embedding* (cf. Definition 3.7.8).
- I show that the tree languages induced by a non-weakly-self-embedding CFTG can be characterized by a RTG (cf. Theorem 3.7.12).

The following three contributions are an extension to the work presented in [51].

- I present an alternative proof of regularity of the tree language induced by a non-self-embedding lnCFTG (cf. Theorem 3.5.23).
- I show that there are regular tree languages induced by non-self-embedding lnCFTGs such that, for each of those tree languages, the size of the inducing lnCFTG is exponentially smaller than the size of the smallest equivalent RTG (cf. Lemma 3.10.2).
- I define the property *self-embedding* for linear MAC (cf. Definition 3.8.6) and the property *weakly-self-embedding* for MAC (cf. Definition 3.8.10). Furthermore, I show that the language induced by a non-self-embedding linear MAC or a weakly-non-self-embedding MAC can be characterized by a CFG (cf. Corollary 3.8.9 and 3.8.13).

1.3 Approximation

For modeling a natural language it is beneficial to utilize an expressive grammar formalism. However, for decoding it might suffice to approximate the induced language in a computationally favorable grammar formalism. For decoding, it is not desired to reject grammatical sentences. Hence, this thesis concentrates on superset approximations. A superset approximation of a language L is a grammar G such that $\mathcal{L}(G) \supseteq L$ where $\mathcal{L}(G)$ denotes the language induced by G. The approximation can be used either (i) to perform the whole decoding on the approximation, or (ii) to narrow down the search space during decoding and to perform a costly search only on the remaining elements.

As an example, we recall the CFG M_1 from Section 1.1 that generates all palindromes using the terminals a and b. We present the REG M_5 with the rules

$$B_0 \to aB_1 \mid bB_2 \mid a \mid b \mid \varepsilon \ , \quad B_1 \to aB_1 \mid bB_1 \mid a \ , \quad \textit{and} \quad B_2 \to aB_2 \mid bB_2 \mid b \ .$$

It can be seen that each string in the language of M_5 has the same terminal at the start and at the end. For strings with at least two terminals, the first terminal is encoded into the nonterminals of M_5, i.e., B_1 represents that the string starts with an a and B_2 represents a b as the first terminal. Then, the nonterminal ensures that the string ends with the respective terminal. Thus, each string with a length smaller than three is indeed a palindrome. Furthermore, each string that can be generated by M_1, can also be generated

by M_5. However, the REG M_5 can generate strings that are not in the language of M_1, e.g., we have that *abaa* is a word in $\mathcal{L}(M_5)$, but it is clearly not a palindrome. Hence, we have $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_5)$.

In order to utilize the computational efficiency of REG, several approximations of context-free languages by REG have been investigated (cf., e.g., [3, 4, 55, 28, 47]). In this thesis, we lift the concept of approximation to tree-generating grammars and describe a method to approximate the tree languages induced by a CFTG by means of a RTG. In this way, we express a context-free tree language by a RTG. For this, we first employ the *characterization* of the language induced by a CFTG in terms of a RTG extended by a pushdown [29, 17]. Intuitively, the characterization is obtained by traversing the rules of the CFTG and storing information about the state of the traversal in the pushdown. The pushdown contains a history of *return addresses* that describe in which rule and at which position of its right-hand side the traversal concerning one nonterminal was started. Once the rules for a nonterminal are explored, the traversal continues at the rule and position indicated in the most recent return address.

Second, the characterization can be turned into an approximation by limiting the pushdown of return addresses. Old return addresses on the pushdown are forgotten if the pushdown exceeds a fixed height. Then, if a return address is needed while the pushdown is empty, a possible return address is nondeterministically guessed. It is intuitively clear that an increased bound on the pushdown improves the quality of the approximation. We will show this by presenting a hierarchy of approximations.

Main Contributions. The following are the main contributions of this thesis regarding approximation.

- I show that a superset approximation can be obtained from the already known characterization of a CFTG by a RTG with associated pushdown storage (cf. Lemma 4.2.8).
- I develop a hierarchy of superset approximations by restricting the set of possible pushdown configurations (cf. Theorem 4.2.16).

1.4 Training

We also contribute to the training step of machine translation. In this step, given a formal grammar and training data, weights for the grammar are found such that the trained grammar is suitable for translation or to measure the quality of a given sentence. The result of the training step is a weighted grammar, which assigns a weight to each rule in the grammar. Then a weighted grammar is used to assign a weight to each terminal object (e.g., a string or a tree) as follows. A derivation is weighted by multiplying the weights of all rule occurrences in the derivation. If there are multiple derivations for the same terminal object, then the weights of all derivations for this object are summed up. For example, recall the macro grammar M_2 from Section 1.1. We note that for each word $a^k b^\ell a^k b^\ell$, there is only one derivation in M_2. We assume a training set containing the words

$$aabaab\ , \qquad aaaabbaaaabb\ , \qquad \text{and} \qquad aaaaaabbbaaaaaabbb\ .$$

Since M_2 is a rather easy example, the best weights can be found by considering how often the rules of M_2 are used to derive the three training words and normalize this with the count of all rules with the same nonterminal at the left-hand side. We obtain the following weights, depicted in square brackets.

$$A_0 \to A(\varepsilon, \varepsilon) \quad [\frac{1}{1} = 1\,]$$

$$A(x_1, x_2) \to A(ax_1, ax_2) \quad [\frac{12}{15} = \frac{4}{5}] \qquad A(x_1, x_2) \to B(x_1, x_2) \quad [\frac{3}{15} = \frac{1}{5}]$$

$$B(x_1, x_2) \to B(x_1 b, x_2 b) \quad [\frac{6}{9} = \frac{2}{3}] \qquad B(x_1, x_2) \to x_1 x_2 \quad [\frac{3}{9} = \frac{1}{3}]$$

Since there is only one derivation for each word, M_2 assigns the weight $1 \cdot \frac{4}{5} \cdot \frac{4}{5} \cdot \frac{1}{5} \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{32}{1125} \approx 0.028$ to the word *aabaab* (cf. Section 1.1 for the corresponding derivation).

Usually, the training data is a finite set of samples [12]. However, it is also possible to train a model using infinitely many samples given in the form of a grammar. For instance, in [12] a weighted CFG is trained using a probability distribution over an infinite set of derivation trees. In this thesis, we show another instance of training using infinitely many samples. More precisely, we show how to obtain a weight assignment for the rules of a given RTG such that the Kullback-Leibler (KL) divergence to a given weighted lnCFTG is minimal. This serves two purposes as explained in the following. First, the weighted lnCFTG can be seen as training data, the RTG as a model, and thus, the obtained weighted RTG is a trained language model. Hence, a weighted context-free tree language is expressed by a weighted RTG. Second, the weight training allows for qualified statements on the quality of an approximation. Since an approximation, in general, generates an infinite amount of trees, statements about the quality of an approximation based on language inclusions are often not feasible. Several approximations might be incomparable based on their induced languages. Using the KL divergence allows obtaining the best possible weighted approximation given the fixed state behavior. The obtained minimal KL divergence can then also be used to compare different approximations.

In more detail, a weighted RTG can be trained as described in the following. The given weighted lnCFTG and unweighted RTG are combined into a weighted lnCFTG that induces the tree language obtained by intersecting the tree languages induced by both given grammars. During this process, the weights of the original lnCFTG are preserved. This construction is a combination of existing results to intersect a lnCFTG and a RTG (cf. [58, 59, 52, 53]). The intersection allows approximating expected rule frequencies for the rules of the RTG. From the expected rule frequencies, we then obtain a weight assignment and show that this is the optimal weight assignment regarding the KL divergence. This approach is similar to the string case [12].

Main Contributions. The following list contains the main contributions of this thesis regarding the training of weighted RTG. The work on this matter is mainly based on my conference publication "Regular Approximation of Weighted Linear Nondeleting Context-Free Tree Languages" [66].

- I present an adapted version of an existing construction to obtain a (weighted) lnCFTG that induces the tree language obtained by intersecting the tree languages induced

by, respectively, a (weighted) lnCFTG and a (weighted) RTG (cf. Theorem 5.4.2 for the unweighted variant and Theorem 5.4.4 for the weighted case).

- Given a weighted lnCFTG and an unambiguous RTG, I show how to approximate the best weights for the RTG according to the KL divergence (cf. Theorem 5.5.6).

1.5 The Structure of this Thesis

This thesis is structured as follows. Chapter 2 contains preliminaries and basic definitions which are needed throughout different chapters. The main results of this thesis are found in Chapters 3, 4, and 5. Chapter 3 describes in detail how to obtain tree properties similar to self-embedding in the string case. It thus formalizes the idea described in Section 1.2. Chapter 4 describes how a CFTG can be approximated by a RTG and thus implements the idea discussed in Section 1.3. As the third and last main result, we show in Chapter 5 how to obtain the weight assignment for a RTG such that the weighted RTG is as close as possible to a given weighted lnCFTG as described in Section 1.4.

Throughout this thesis, we try to use similar variable names or identifiers for similar concepts. A table containing common names can be found in the naming scheme (cf. Appendix 6). An index of important concepts can be found on page 118.

2 Preliminaries

In this chapter, we recall basic notions and introduce notation needed in this thesis. Section 2.1 contains basic concepts. In Section 2.2, we recall the notions of context-free tree grammars and regular tree grammars. Furthermore, we present known results centered around these formalisms. These notions are crucial throughout the whole work and are thus recalled in detail.

2.1 Foundations

A reader familiar with the basic notation, string grammars, graphs, and trees may skip this section and consult it on demand.

Mathematical Notions

The *set of natural numbers* $\{0, 1, 2, \ldots\}$ is denoted by $\mathbb{N}$ and $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. The set of *finite sequences over* $\mathbb{N}_+$ is denoted by $\mathbb{N}_+^*$ (including the empty sequence). We let $n \in \mathbb{N}$. Then $[n] = \{1, \ldots, n\}$ and hence $[0] = \emptyset$. A *permutation of* $[n]$ is a bijective function $\pi \colon [n] \to [n]$ and we denote π by its values as $\langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$. For a finite set A, we let $|A|$ denote the number of elements in A.

An alphabet is a non-empty finite set. The *set of words* over the alphabet Σ is denoted by Σ^* with ε being the *empty word*. Let $v \in \Sigma^*$, $n \in \mathbb{N}$, $i \in [n]$, and $a_1, \ldots, a_n \in \Sigma$ such that $v = a_1 \ldots a_n$. Then the *i-th symbol of v*, denoted by $v(i)$, is $v(i) = a_i$. The *length* of v, denoted by $|v|$, is n. Any set $L \subseteq \Sigma^*$ is called a *string language*. The *set of reals* is denoted by $\mathbb{R}$ and the *set of positive reals* by $\mathbb{R}_{\geq 0}$. The logarithm using base 2 is denoted by log, and the natural logarithm is denoted by ln. Let U be a set. Then $\mathcal{P}(U)$ denotes the *powerset of U*, i.e., the set of all subsets of U.

Let I and A be sets and $f \colon I \to A$ a function. We call f an *I-indexed family over A* (for short: family), denoted by $f = (f_i \mid i \in I)$ with $f_i = f(i) \in A$.

We fix an infinite list $x_1, x_2, \ldots$ of pairwise distinct *variables*, let $X = \{x_1, x_2, x_3, \ldots\}$, and let $X_k = \{x_1, \ldots, x_k\}$. Furthermore, we abbreviate $x_1, \ldots, x_k$ to $x_{1..k}$. We apply this abbreviation also to sequences of other objects. Sometimes we will also use symbols different from $x_1, x_2, \ldots$ to denote variables, such as $z, z_1, z_2, \ldots$.

Context-Free String Grammars and Regular String Grammars

We recall context-free string grammars and regular string grammars [11].

Definition 2.1.1 A *context free string grammar (CFG)* is a tuple $M = (N, \Sigma, B_0, R)$ where

- N is a finite set of *nonterminals*,

- Σ is an alphabet of *terminals* such that $N \cap \Sigma = \emptyset$,
- $B_0 \in N$ is the *initial nonterminal*, and
- R is a finite set of *rules* of the form $B \to s$ where $B \in N$ and $s \in (N \cup \Sigma)^*$. □

Definition 2.1.2 Let $M = (N, \Sigma, B_0, R)$ be a CFG. The *derivation relation* of M, denoted by $\Rightarrow$, is defined as follows. For strings $s, t \in (N \cup \Sigma)^*$, we have $s \Rightarrow t$ if there is a rule of the form $B \to t'$ such that $s = s_1 B s_2$ for some $s_1, s_2 \in (N \cup \Sigma)^*$ and $t = s_1 t' s_2$. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$.

The *string language of* M, denoted by $\mathcal{L}(M)$, is defined as $\mathcal{L}(M) = \{s \in \Sigma^* \mid B_0 \Rightarrow^* s\}$.□

Definition 2.1.3 A *regular string grammar (REG)* is a CFG (N, Σ, B_0, R) if, for each rule $B \to s$ in R, we have that either $s \in \Sigma^*$, or $s = tB$ for some $t \in \Sigma^*$ and $B \in N$. □

Graphs

We let Σ be an alphabet. A *Σ-labeled directed graph* (for short: *graph*) is a pair (V, E) where V is a finite set of *vertices* and E is a finite set of *edges* satisfying $E \subseteq V \times \mathcal{P}(\Sigma) \times V$. For each edge (v_1, U, v_2), we call U the label of (v_1, U, v_2). Let (V, E) be a graph. Sometimes we denote the set of vertices by $\mathrm{V}_{(V,E)}$ and the set of edges by $\mathrm{E}_{(V,E)}$ or, if no confusion arises, by $\to$. Then $(v_1, U, v_2) \in \mathrm{E}_{(V,E)}$ will also be abbreviated by $v_1 \overset{U}{\to} v_2$ or just by $v_1 \to v_2$. We denote the reflexive closure of $\to$ by $\to^*$. A *path in* (V, E) is a sequence $v_1, v_2, \ldots, v_n$ such that, for each $i \in [n-1]$, we have that $v_i \to v_{i+1}$ is an edge in E. We denote such a path by $v_1 \to v_2 \to \ldots \to v_n$.

We let (V, E) be a graph and $V' \subseteq V$. The *V'-fragment of* (V, E), denoted by $(V, E)|_{V'}$, is the graph $(V', E \cap (V' \times \mathcal{P}(\Sigma) \times V'))$.

For each $V' \subseteq V$ and $P = (V, E)|_{V'}$, we say that P is *strongly connected* if, for each $v_1, v_2 \in V'$, we have $v_1 \to^* v_2$ and $v_2 \to^* v_1$ are paths in P. For each strongly connected V'-fragment $P = (V', E')$, we say that P is a *maximal strongly connected component (SCC)* if there is no $V'' \subseteq V$ such that $V' \subset V''$ and $(V, E)|_{V''}$ is strongly connected. We denote the set of maximal strongly connected components of (V, E) by $\mathrm{scc}((V, E))$.

Trees

Definition 2.1.4 A *ranked alphabet* is a pair $(\Delta, \mathrm{rk}_\Delta)$ where Δ is an alphabet and $\mathrm{rk}_\Delta \colon \Delta \to \mathbb{N}$ is a function. For every $\delta \in \Delta$, we call $\mathrm{rk}_\Delta(\delta)$ the *rank* of δ.

Sometimes, we write $\delta^{(k)}$ to indicate that δ has rank k. We abbreviate the set $\mathrm{rk}_\Delta^{-1}(k)$ to $\Delta^{(k)}$ and $(\Delta, \mathrm{rk}_\Delta)$ to Δ, respectively, assuming that rk_Δ is the rank function. □

In this work, Δ denotes an arbitrary ranked alphabet. We assume that $\Delta \cap X = \emptyset$.

Definition 2.1.5 Let U be a set such that $\Delta \cap U = \emptyset$. The set of *trees over* Δ *and* U, denoted by $\mathrm{T}_\Delta(U)$, is defined inductively as the smallest set T such that the following conditions are satisfied.

- $U \subseteq T$.

- For each $k \in \mathbb{N}$, $\delta \in \Delta^{(k)}$, and $\xi_{1..k} \in T$, we have $\delta(\xi_{1..k}) \in T$.

We abbreviate $\mathrm{T}_\Delta(\emptyset)$ by T_Δ. □

Definition 2.1.6 Any subset $L \subseteq \mathrm{T}_\Delta(U)$ is called a *tree language*. □

Positions in trees are identified by Gorn addresses, represented by finite sequences over $\mathbb{N}_+$. They allow addressing specific nodes in a tree.

Definition 2.1.7 We let $\xi \in \mathrm{T}_\Delta(U)$. The *set of positions of* ξ, denoted by $\mathrm{pos}(\xi)$, is defined inductively as follows:

(i) If $\xi \in U$, then $\mathrm{pos}(\xi) = \{\varepsilon\}$, and

(ii) if $\xi = \delta(\xi_{1..k})$ for some $\delta \in \Delta^{(k)}$, $k \in \mathbb{N}$, and $\xi_{1..k} \in \mathrm{T}_\Delta(U)$, then

$$\mathrm{pos}(\xi) = \{\varepsilon\} \cup \{iv \mid 1 \leq i \leq k, v \in \mathrm{pos}(\xi_i)\} \ .$$

For $W \subseteq \mathrm{pos}(\xi)$ and $w \in W$, we say that w is *outermost in* W if there is no $w_1 \in W$ such that $w = w_1 w_2$ for some $w_2 \in \mathbb{N}_+^* \setminus \{\varepsilon\}$. Since positions are finite sequences over $\mathbb{N}_+$, they have a natural lexicographic ordering.

For a position $w \in \mathrm{pos}(\xi)$, the *label of* ξ *at* w and the *subtree of* ξ *at* w are denoted by $\xi(w)$ and $\xi|_w$, respectively. For every $U' \subseteq \Delta \cup U$, we denote the *set of positions of* ξ *labeled by an element of* U' by $\mathrm{pos}_{U'}(\xi)$; if U' is a singleton $\{u\}$, then we simply write $\mathrm{pos}_u(\xi)$. □

Definition 2.1.8 Let U be a finite set with $\Delta \cap U = \emptyset$. A *context over* Δ *and* U is a tree in $\mathrm{T}_\Delta(U)$ in which each element $u \in U$ occurs exactly once. The set of all such contexts is denoted by $\mathrm{C}_\Delta(U)$. □

Next, we introduce the property of a position to be variable dominating. Such a position is a prefix of a position that is labeled by a variable.

Definition 2.1.9 Let $\xi \in \mathrm{T}_\Delta(X)$, $i \in \mathbb{N}$, $x_i \in X$, and $w \in \mathrm{pos}(\xi)$. We say that w is *x_i-dominating* if $\xi|_w$ contains a position labeled x_i. We call w *variable dominating* if it is x_i-dominating for some $x_i \in X$. □

Tree concatenation. We recall the notion of tree concatenation that is similar to string concatenation. Instead of adding another string to the right of an already existing one, we replace nullary symbols in a tree and add trees in their place.

Definition 2.1.10 We let $k \in \mathbb{N}$, $u_{1..k} \in U \cup \Delta^{(0)}$ be pairwise distinct symbols, $\xi \in \mathrm{T}_\Delta(U)$, and $\xi_{1..k} \in \mathrm{T}_\Delta(U)$. We define the *tree concatenation of* ξ *with* $\xi_{1..k}$ *at* $u_{1..k}$, denoted by $\xi[u_1/\xi_1, \ldots, u_k/\xi_k]$, as the tree obtained inductively on the structure of ξ as follows:

(*i*) $u_i[u_1/\xi_1, \ldots, u_k/\xi_k] = \xi_i$ for each $i \in [k]$,

(*ii*) $u[u_1/\xi_1, \ldots, u_k/\xi_k] = u$ for each $u \in U \setminus \{u_1, \ldots, u_k\}$, and

(*iii*) $\delta(\zeta_1, \ldots, \zeta_\ell)[u_1/\xi_1, \ldots, u_k/\xi_k] = \delta(\zeta_1[u_1/\xi_1, \ldots, u_k/\xi_k], \ldots, \zeta_\ell[u_1/\xi_1, \ldots, U_k/\xi_k])$ for each $\ell \in \mathbb{N}$ and $\delta \in \Delta^{(\ell)} \setminus \{u_1, \ldots, u_k\}$.

Furthermore, we let $L_{1..k} \subseteq \mathrm{T}_\Delta(U)$. Then the *tree concatenation of ξ with tree languages $L_{1..k}$ at $u_{1..k}$* is the tree language inductively defined by

(i') $u_i[u_1/L_1, \ldots, u_k/L_k] = L_i$,

(ii') $u[u_1/L_1, \ldots, u_k/L_k] = \{u\}$ for each $u \setminus \{u_1, \ldots, u_k\}$, and

(iii') $\delta(\zeta_1, \ldots, \zeta_\ell)[u_1/L_1, \ldots, u_k/L_k] = \{\delta(\xi'_1, \ldots, \xi'_\ell) \mid \xi'_1 \in \zeta_1[u_1/\xi_1, \ldots, u_k/\xi_k], \ldots, \xi'_\ell \in \zeta_\ell[u_1/\xi_1, \ldots, u_k/\xi_k]\}$ for each $\ell \in \mathbb{N}$ and $\delta \in \Delta^{(\ell)} \setminus \{u_1, \ldots, u_k\}$.

Moreover, for each $L \subseteq T_\Delta(U)$, we define

$$L[u_1/L_1, \ldots, u_k/L_k] = \bigcup\nolimits_{\xi \in L} \xi[u_1/L_1, \ldots, u_k/L_k] \ . \qquad \square$$

Tree concatenation is associative [18, Cor 2.4.2] and distributive over set union [18, p. 341]. Note that tree concatenation with trees yields a single tree, while tree concatenation with tree languages yields a tree language. It is always clear from the context which tree concatenation is used. Furthermore, note that we define tree concatenation with tree languages in a way that multiple occurrences of the same u_i ($i \in [k]$) are in general replaced by different trees.

For convenience, we will use the following abbreviations. For every $\xi \in \mathrm{T}_\Delta(X_k)$ and $\xi_{1..k} \in \mathrm{T}_\Delta(X)$, we abbreviate $\xi[x_1/\xi_1, \ldots, x_k/\xi_k]$ by $\xi[\xi_1, \ldots, \xi_k]$ or $\xi[\xi_{1..k}]$. This abbreviation is also used for $\xi \in \mathrm{T}_\Delta(U)$ when U is a finite set other than X_k, provided elements in U are ordered explicitly, or if U is a singleton. Moreover, we may write $\xi[u_i/\xi_i \mid i \in [k]]$ instead of $\xi[u_1/\xi_1, \ldots, u_k/\xi_k]$. These abbreviations transfer directly to the case of tree concatenation with tree languages.

Substitution. In the sequel, we also wish to substitute tree languages at symbols which are not necessarily nullary. For this, we require the tree languages to be sets of contexts. A more general version of this substitution is described in [5, pp. 195f]; the case here corresponds to the special case using the Boolean semiring.

Definition 2.1.11 We let $k \in \mathbb{N}$, $\delta_1, \ldots, \delta_k \in \Delta$ be pairwise distinct, and $L_1, \ldots, L_k \subseteq \mathrm{T}_\Delta(X)$ such that, for each $i \in [k]$, we have $L_i \subseteq \mathrm{C}_\Delta(X_{\mathrm{rk}_\Delta(\delta_i)})$. We define the function $\mathrm{tr}\colon \mathrm{T}_\Delta(X) \to \mathcal{P}(\mathrm{T}_\Delta(X))$ inductively on its argument as follows. For every $\ell \in \mathbb{N}$, $\delta \in \Delta^{(\ell)}$, and $\xi_1, \ldots, \xi_\ell \in \mathrm{T}_\Delta(X)$ we have

$$\mathrm{tr}(\delta(\xi_1, \ldots, \xi_\ell)) = \begin{cases} L_i[\mathrm{tr}(\xi_1), \ldots, \mathrm{tr}(\xi_\ell)] & \text{if } \delta = \delta_i \text{ for some } i \in [k], \\ \delta(x_{1..\ell})[\mathrm{tr}(\xi_1), \ldots, \mathrm{tr}(\xi_\ell)] & \text{otherwise,} \end{cases}$$

and moreover, $\mathrm{tr}(x) = \{x\}$ for each $x \in X$. For each $\xi \in \mathrm{T}_\Delta(X)$, we denote $\mathrm{tr}(\xi)$ by $\xi \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k)$. We let $L \subseteq \mathrm{T}_\Delta(X)$ and define the *tree substitution of $L_1, \ldots, L_k$ into L at $\delta_1, \ldots, \delta_k$* as the tree language $L \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k) = \bigcup_{\xi \in L} \xi \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k)$. Note that if $\delta_1, \ldots, \delta_k \in \Delta^{(0)} \cup X$, then $L \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k) = L[\delta_1/L_1, \ldots, \delta_k/L_k]$. For $k = 1$, we abbreviate $L \leftarrow (\delta_1/L_1)$ by $L \leftarrow_{\delta_1} L_1$. We may write $L \leftarrow (\delta_i/L_i \mid i \in [k])$ instead of $L \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k)$. Tree substitution is associative (cf. [5, Prop. 7] for the Boolean semiring). $\square$

Strings as Unary Trees. We will compare some results for trees to corresponding results in the string case. For this, we show how to represent a string as a tree.

Definition 2.1.12 Let Σ be an alphabet. The *tree representation of* Σ, denoted by $\widetilde{\Sigma}$, is the ranked alphabet $\widetilde{\Sigma} = (\Sigma \cup \{\sharp\}, \mathrm{rk}_{\widetilde{\Sigma}})$ where $\mathrm{rk}_{\widetilde{\Sigma}}(\sigma) = 1$ for all $\sigma \in \Sigma$ and $\mathrm{rk}_{\widetilde{\Sigma}}(\sharp) = 0$.

Let $s \in \Sigma^*$ such that $s = a_1 \dots a_k$ for some $k \in \mathbb{N}$ and $a_1, \dots, a_k \in \Sigma$. The *tree representation of* s, denoted by $\widetilde{s}$, is the tree $\widetilde{s} = a_1(a_2(\dots a_{k-1}(a_k(\sharp))\dots))$ in $\mathrm{T}_{\widetilde{\Sigma}}$. If $s = \varepsilon$, then $\widetilde{s} = \sharp$. □

If γ is a unary symbol, then we may write γ^n instead of the context $\underbrace{\gamma(\gamma(\dots\gamma}_{n \text{ times}}(z)\dots))$. Furthermore, we write $\gamma^n(\xi)$ instead of $\gamma^n[\xi]$ for any tree ξ.

Yield and Path Language. We have shown how to represent a string as a tree. We note that the tree representation of a string can be uniquely transformed into a string by reading the symbols from the root to the single leaf $\sharp$. In an arbitrary tree, there is one path from the root to each leaf. These paths are the path language of that tree. In accordance with [25, Def. 2.11.1], at each interior node we annotate at which child we continue.

Definition 2.1.13 Let $L \subseteq \mathrm{T}_\Delta(U)$ be a tree language. The *path language of* L is the string language $L' \subseteq (\Delta' \cup U)^*$ obtained as follows where $\Delta' = \bigcup_{\delta \in \Delta} \{\delta\} \times [\mathrm{rk}_\Delta(\delta)]$ is considered as an unranked alphabet. We let $L' = \bigcup_{\xi \in L} \mathrm{f}(\xi)$ where, for each $\xi = \delta(\xi_1, \dots, \xi_k)$,

$$\mathrm{f}(\delta(\xi_1, \dots, \xi_k)) = \begin{cases} \{(\delta, i)w \mid i \in [k], w \in \mathrm{f}(\xi_i)\} & \text{if } \delta \in \Delta^{(k)} \text{ and } k > 0 \\ \{\delta\} & \text{if } \delta \in \Delta^{(0)} \cup U \end{cases}.$$ □

An alternative way of obtaining a string from a tree is taking the yield of a tree, i.e., reading the string of labels at its leaves (symbols in $\Delta^{(0)}$ and U) from left to right.

Definition 2.1.14 Let $\xi \in \mathrm{T}_\Delta(U)$. The *yield of* ξ, denoted by $\mathrm{yield}(\xi)$, is a string over $(\Delta^{(0)} \cup U)^*$ inductively defined over the structure of the tree $\xi = \delta(\xi_1, \dots, \xi_k)$ as

$$\mathrm{yield}(\delta(\xi_1, \dots, \xi_k)) = \begin{cases} \mathrm{yield}(\xi_1) \dots \mathrm{yield}(\xi_k) & \text{if } \delta \in \Delta^{(k)} \text{ and } k > 0 \\ \delta & \text{if } \delta \in (\Delta^{(0)} \cup U) \end{cases}$$

Furthermore, we allow yield to be applied to tree languages by defining, for each $L \subseteq \mathrm{T}_\Delta$, that $\mathrm{yield}(L) = \{\mathrm{yield}(\xi) \mid \xi \in L\}$. □

2.2 Context-Free Tree Grammars and Regular Tree Grammars

We first formally define context-free tree grammars and regular tree grammars and then recall closely connected results.

Definitions

Definition 2.2.1 A *context-free tree grammar* (CFTG) is a tuple $G = (N, \Delta, A_0, R)$ where

- N is a ranked alphabet of *nonterminals*,
- Δ is a ranked alphabet of *terminals* such that $N \cap \Delta = \emptyset$,
- $A_0 \in N^{(0)}$ is the *initial nonterminal*, and
- R is a finite set of *rules* of the form $A(x_{1..k}) \to \xi$ where $k \in \mathbb{N}$, $A \in N^{(k)}$, and $\xi \in \mathrm{T}_{N \cup \Delta}(X_k)$.

We let $r\colon A(x_{1..k}) \to \xi$ be a rule. The *left-hand side (LHS) of* r is $A(x_{1..k})$, the *LHS-nonterminal of* r, denoted by $\mathrm{lhn}(r)$, is A, and the *right-hand side (RHS) of* r, denoted by $\mathrm{rhs}(r)$, is ξ. For each $k \in \mathbb{N}$ and $A \in N^{(k)}$, we abbreviate $A(x_{1..k})$ by $A(\overline{x})$. □

For technical convenience, we allow rules to use any finite combination of distinct variables instead of a prefix of the sequence $x_1, x_2, x_3, \ldots$, e.g., $A(x_2, x_5) \to \sigma(x_2, x_5)$. Such a rule can be transformed into the syntactically correct form by consistently renaming variables.

Throughout this work, we use capital letters from the start of the alphabet to denote nonterminals (e.g., A or B), the initial nonterminal is indexed by a 0 (e.g., A_0 or B_0), and terminals from Δ are denoted by lowercase Greek letters (e.g. α, β, or γ). This convention allows us to represent a CFTG by only presenting its rules; nonterminals and terminals can be read off from the rules.

In the following six definitions, *we let* $G = (N, \Delta, A_0, R)$ *be an arbitrary CFTG.*

Definition 2.2.2 The *derivation relation of* G, denoted by $\Rightarrow$, is defined as follows. For trees $\xi, \xi' \in \mathrm{T}_{N \cup \Delta}(X)$, a rule $r\colon A(x_{1..k}) \to \zeta$ in R, and $w \in \mathrm{pos}(\xi)$, we have $\xi \Rightarrow_{r,w} \xi'$ if (i) $\xi(w) = A$ and (ii) ξ' is obtained from ξ by replacing the subtree at position w by $\zeta[\xi|_{w1}, \ldots, \xi|_{wk}]$.

We write $\xi \Rightarrow_r \xi'$ if there is a position $w \in \mathrm{pos}(\xi)$ such that $\xi \Rightarrow_{r,w} \xi'$. Similarly, we write $\xi \Rightarrow \xi'$ if there is an $r \in R$ such that $\xi \Rightarrow_r \xi'$.

Furthermore, we denote the reflexive, transitive closure of $\Rightarrow$ by $\Rightarrow^*$.

If there are $n \in \mathbb{N}$, $r_1, r_2, \ldots, r_n \in R$, $\xi_0, \xi_1, \ldots, \xi_n \in \mathrm{T}_{N \cup \Delta}(X)$, $w_1 \in \mathrm{pos}(\xi_0)$, $w_2 \in \mathrm{pos}(\xi_1)$ …, and $w_n \in \mathrm{pos}(\xi_{n-1})$ such that

$$\xi_0 \Rightarrow_{r_1,w_1} \xi_1 \Rightarrow_{r_2,w_2} \xi_2 \Rightarrow_{r_3,w_3} \cdots \Rightarrow_{r_{n-1},w_{n-1}} \xi_{n-1} \Rightarrow_{r_n,w_n} \xi_n \ , \qquad (*)$$

then we call $(*)$ a *derivation starting in* ξ_0 in G. If ξ_0 is clear from the context, we call $(*)$ just a *derivation*. In many cases we refrain from explicitly denoting the positions at which the rules are applied and assume that these are understood from the context. Then, we denote $(*)$ by $\xi_0 \Rightarrow_d \xi_n$ where $d \in R^*$ such that $d = r_1 r_2 \ldots r_n$.

The derivation $\xi_0 \Rightarrow_d \xi_n$ is called *complete*, if $\xi_n \in \mathrm{T}_\Delta$. □

In general, we do not impose any restriction on the order in which nonterminals are derived (unrestricted derivation [20]). However, for the tree language induced by a CFTG, we fix an order where, in each derivation step, one of the nonterminals at an outermost position is derived.

Definition 2.2.3 Let $\xi_0 \Rightarrow_{r_1,w_1} \xi_1 \Rightarrow_{r_2,w_2} \xi_2 \Rightarrow_{r_3,w_3} \ldots \Rightarrow_{r_{n-1},w_{n-1}} \xi_{n-1} \Rightarrow_{r_n,w_n} \xi_n$ be a derivation. We call it an *outside-in derivation* (cf. [20, p. 2-15]), if, for each $i \in [n]$, the position w_i is outermost in $\mathrm{pos}_N(\xi_{i-1})$. □

In an outside-in derivation there is some nondeterminism in the choice of the nonterminal that will be derived next, since any nonterminal at an outermost position can be chosen. To completely remove the nondeterminism, we may choose to consider leftmost outermost derivations where among all outermost nonterminals the leftmost one is chosen.

Definition 2.2.4 Let $\xi_0 \Rightarrow_{r_1,w_1} \xi_1 \Rightarrow_{r_2,w_2} \xi_2 \Rightarrow_{r_3,w_3} \ldots \Rightarrow_{r_{n-1},w_{n-1}} \xi_{n-1} \Rightarrow_{r_n,w_n} \xi_n$ be a derivation. We call it a *leftmost outermost derivation*, if, for each $i \in [n]$, the position w_i is outermost in $\mathrm{pos}_N(\xi_{i-1})$ and w_i is the smallest with respect to the lexicographic ordering of positions among all outermost positions in $\mathrm{pos}_N(\xi_{i-1})$. □

Definition 2.2.5 For $k \in \mathbb{N}$ and $\zeta \in \mathrm{T}_{N\cup\Delta}(X_k)$, the *(outside-in) tree language induced by ζ on G* is

$$\mathcal{L}(G,\zeta) = \{\xi \in \mathrm{T}_\Delta(X_k) \mid \zeta \Rightarrow_d \xi,\ d \text{ is outside-in}\} \ .$$

The *(outside-in) tree language induced by G*, denoted by $\mathcal{L}(G)$, is defined as $\mathcal{L}(G) = \mathcal{L}(G, A_0)$. Note that $\mathcal{L}(G) \subseteq \mathrm{T}_\Delta$. Two lnCFTGs G and G' are *equivalent*[1] if $\mathcal{L}(G) = \mathcal{L}(G')$.

Since we only consider the outside-in tree languages induced by G, we drop the prefix and call the outside-in tree language induced by G just the *tree language induced by G*.

A tree language $L \subseteq \mathrm{T}_\Delta$ is called *context-free* if there is a CFTG $G = (N, \Delta, A_0, R)$ such that $\mathcal{L}(G) = L$. □

Definition 2.2.6 We say that G is a *linear* CFTG (lCFTG) if, for each rule $A(\overline{x}) \to \xi$ in R and each $i \in [\mathrm{rk}_N(A)]$, we have that x_i appears at most once in ξ. We say that G is a *nondeleting* CFTG if, for each rule $A(\overline{x}) \to \xi$ in R and each $i \in [\mathrm{rk}_N(A)]$, we have that x_i occurs at least once in ξ. Accordingly, each rule $A(\overline{x}) \to \xi$ in R such that there is an $i \in [\mathrm{rk}_N(A)]$ and $\mathrm{pos}_{x_i}(\xi) = \emptyset$ is called a *deleting* rule. A CFTG that contains at least one deleting rule is called *deleting CFTG*.

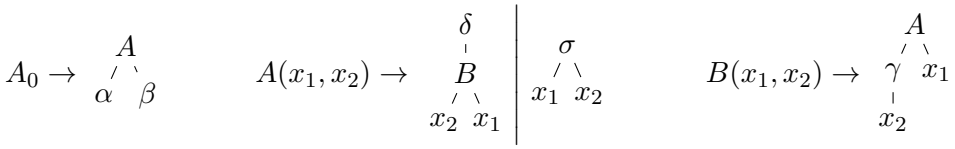
If G is a linear CFTG and a nondeleting CFTG, then we say that it is a *linear nondeleting* CFTG (lnCFTG). □

Note that in a lnCFTG, the RHS of each rule $A(\overline{x}) \to \xi$ is a context over $X_{\mathrm{rk}_N(A)}$.

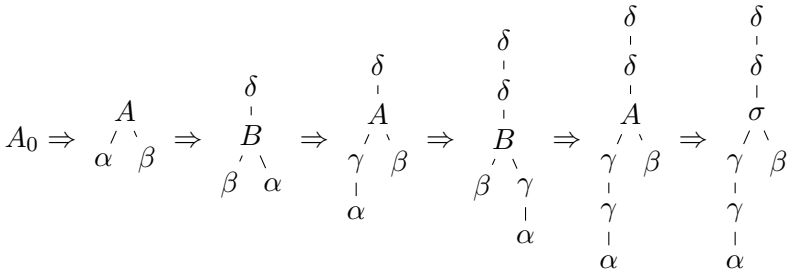
Example 2.2.7 Figure 2.1(a) presents the lnCFTG G_2 and Figure 2.1(b) one derivation of G_2. It can be seen that $\mathcal{L}(G_2) = \{\delta^n(\sigma(\gamma^n(\alpha), \beta)) \mid n \in \mathbb{N}\}$. ○

Later we wish to analyze derivations which, for some given subset $N' \subseteq N$, contain only rules whose LHS-nonterminals are in N'. This is achieved by considering all symbols from $N \setminus N'$ as terminal symbols.

[1] in the NLP-community this is called *strongly equivalent*



(a) Rules.



(b) Derivation.

Figure 2.1: The CFTG G_2 and one of its derivations.

Definition 2.2.8 We let $N' \subseteq N$. The *N'-fragment of R*, denoted by $R|_{N'}$, is defined to be the set $\{r \in R \mid \mathrm{lhn}(r) \in N'\}$. If $N' = \{A\}$ for some $A \in N$, then we also denote $R|_{\{A\}}$ by $R|_A$.

The *N'-fragment of G* is the CFTG

$$G|_{N'} = (N', \Delta \cup (N \setminus N'), \_, R|_{N'}) \ .$$

where the initial nonterminal of $G|_{N'}$ is irrelevant, and we only address its tree language using an explicitly given initial tree ζ via $\mathcal{L}(G|_{N'}, \zeta)$. Note that if N' does not contain a nullary nonterminal symbol, then we may add a new nonterminal as a placeholder without affecting the induced tree language. □

Example 2.2.9 As an example, we consider the lnCFTG G_3 using the set of nonterminals $\{A_0, A, B\}$, the set of terminals $\Delta = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}, \beta^{(0)}\}$, and the rules R contain $A_0 \to A(\alpha, \beta)$, $A(x_1, x_2) \to A(B(x_1), x_2) \mid \sigma(x_1, x_2)$, and $B(x_1) \to \gamma(B(x_1))$. The $\{A\}$-fragment of G_3 has only the two rules (namely both rules of R with LHS-nonterminal A) and considers A_0 and B as terminal symbols. The tree language induced by $A(x_1, x_2)$ on the fragment is $\mathcal{L}(G|_{\{A\}}, A(x_1, x_2)) = \{\sigma(B^n(x_1), x_2) \mid n \in \mathbb{N}\} \subseteq \mathrm{T}_{\Delta \cup \{B\}}(X_2)$. ○

A regular tree grammar can be obtained from a CFTG by requiring that each nonterminal has rank 0, i.e., nonterminals have no arguments and only appear as leaves of trees.

Definition 2.2.10 A *regular tree grammar* (RTG) is a CFTG in which each nonterminal has rank 0, i.e., $N = N^{(0)}$. A tree language $L \subseteq \mathrm{T}_\Delta$ is *regular* if there is a RTG $G = (N, \Delta, A_0, R)$ such that $\mathcal{L}(G) = L$. □

We note that each RTG is also a lCFTG and a lnCFTG.

Known Results for lnCFTGs and RTGs

In this section, we recall results for lnCFTGs and RTGs that will help us later on at multiple points throughout this work. First, we consider normal forms for lnCFTGs and RTGs and then we recall that the order of rule application in the derivation of a lnCFTG does not matter. Furthermore, we show that deleting rules can be removed from a lCFTG and useless rules can be pruned from a lnCFTG. Lastly, we recall closure properties for RTGs.

Normal Forms. In the following, we introduce two special forms; one for lnCFTGs and one for RTGs, respectively. In the existing literature, the term *normal form* is used for both special forms. Thus, we refrain from using the term normal form and rather use distinct names that also describe the required feature.

For lnCFTGs we use the *nonterminal form*, i.e., the RHS of each rule either consists of a tree over nonterminals and variables (and no terminals), or the RHS of a rule is a single terminal with variables according to its rank. For RTGs, we specify the property of being *producing*. A RTG fulfills this property if each RHS of rule created exactly one terminal and continues deriving with nonterminals at every argument position of this terminal.

Definition 2.2.11 We let $G = (N, \Delta, A_0, R)$ be a lnCFTG. We say that G is in *nonterminal form* if every rule is of one of the following two forms:

- Type I: $A(x_{1..k}) \to \xi$ with $\xi \in \mathrm{C}_N(X_k)$, or
- Type II: $A(x_{1..k}) \to \delta(x_{1..k})$ for some $\delta \in \Delta^{(k)}$. □

Lemma 2.2.12 For every lnCFTG G, there is a lnCFTG G' in nonterminal form such that $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof. G' consists of all rules from G which are of Type I or II. Furthermore, each rule r of G which is neither of Type I nor II can be transformed into a rule of Type I in G' by replacing each node labeled by $\delta \in \Delta^{(k)}$ in the RHS of r by a new nonterminal A_δ and adding the Type II rule $A_\delta(x_{1..k}) \to \delta(x_{1..k})$. It is easy to see that G' constructed in this way is indeed in nonterminal form and that $\mathcal{L}(G) = \mathcal{L}(G')$. ■

Definition 2.2.13 We let $H = (N_H, \Delta, B_0, R_H)$ be a RTG. We say that H is *producing* if each rule is of the form $B \to \delta(B_{1..k})$ for some $\delta \in \Delta$ and $B, B_1, \ldots, B_k \in N_H$. □

Lemma 2.2.14 [25, Ch. II, Lm. 3.4] For each RTG H, there is a producing RTG H' such that $\mathcal{L}(H) = \mathcal{L}(H')$.

Reordering of Derivations in a lnCFTG. The following lemma formalizes that the order of rule applications of a derivation in a lnCFTG can be changed without affecting the resulting tree. It is based on the structural theorems for macro grammars as presented in [20]. This property is also called *confluence* in the literature (cf., [14, Def. 2.10]).

Lemma 2.2.15 Let $G = (N, \Delta, A_0, R)$, $\xi \in \mathrm{T}_{N\cup\Delta}(X)$ be a lnCFTG and $r_1, r_2 \in R$ such that r_1 and r_2 can be applied to ξ at distinct positions. Then there is a $\xi' \in \mathrm{T}_{N\cup\Delta}(X)$ such that $\xi \Rightarrow_{r_1 r_2} \xi'$ and $\xi \Rightarrow_{r_2 r_1} \xi'$.

PROOF. The lemma is based on the structural theorems for macro grammars [20]. We let $w_1, w_2 \in \mathrm{pos}_N(\xi)$ be distinct positions at which r_1 and r_2, respectively, can be applied. If neither w_1 is a prefix of w_2 nor vice versa, then clearly r_1 and r_2 can be applied in any order at w_1 and w_2, respectively. Without loss of generality, we let w_1 be a prefix of w_2, i.e., $w_2 = w_1\, i\, u$ for some $i \in \mathbb{N}_+$ and $u \in \mathbb{N}_+^*$, then there is a $\xi'' \in \mathrm{T}_{N\cup\Delta}(X)$ such that $\xi \Rightarrow_{r_1} \xi''$ and $\xi|_{w_2} \neq \xi''|_{w_2}$, i.e., the application of r_1 changes the position at which r_2 has to be applied. We let $v \in \mathrm{pos}_{x_i}(\mathrm{rhs}(r_1))$ be the uniquely determined position of x_i in $\mathrm{rhs}(r_1)$. Then the application of r_1 at w_1 followed by the application of r_2 at $w_1\, v\, u$ leads to the same tree as the application of r_2 at w_2 followed by the application of r_1 at w_1. ∎

Observation 2.2.16 For a lnCFTG $G = (N, \Delta, A_0, R)$, we have that $\mathcal{L}(G) = \{\xi \in \mathrm{T}_\Delta \mid A_0 \Rightarrow^* \xi\}$, i.e., we may consider any derivation instead of just outside-in derivations.

Deleting Rules of a lnCFTG. It is known that for each linear CFTG we can find an equivalent linear nondeleting CFTG. The construction can be traced back to [20, Thm. 3.1.10] (also cf., e.g., [65, Lm. 3.1]).

Theorem 2.2.17 For each linear CFTG G, there is a linear nondeleting CFTG G' such that $\mathcal{L}(G) = \mathcal{L}(G')$.

PROOF. Let $G = (N, \Delta, A_0, R)$ be a lCFTG. We construct a lnCFTG $G' = (N', \Delta, A_0, R')$ as follows. We let $N' = \{A_\alpha \mid k \in \mathbb{N},\ A \in N^{(k)},\ \alpha \subseteq [k]\}$ where, for each $A_\alpha \in N'$, we define $\mathrm{rk}_{N'}(A_\alpha) = \mathrm{rk}_N(A) - |\alpha|$. Intuitively, the argument positions in subscript α are those that will be removed.

The set R' is defined as follows. Let $A(\overline{x}) \to \xi$ be in R where $k = \mathrm{rk}_N(A)$. We define the set $W = \{(w, \beta) \mid w \in \mathrm{pos}_N(\xi), \beta \subseteq [\mathrm{rk}_N(\xi(w))]\}$ of nonterminal occurrences in ξ together with a list of deleted argument positions. Then, for each $W' \subseteq W$ such that $|W'| = |\mathrm{pos}_N(\xi)|$ and the first components of W' are pairwise distinct, we define $\xi_{W'}$ to be the tree obtained from ξ as follows. We order the elements in W' descending by their first component (the elements with outermost position come last) and obtain $(w_1, \beta_1), \ldots, (w_n, \beta_n)$ where $n = |W'|$. We let $\xi_0 = \xi$ and, for each $i \in [n]$, we obtain ξ_i from ξ_{i-1} by replacing the subtree $B(\zeta_{1..\ell})$ at position w_i by the tree $B_{\beta_i}(\zeta_{\ell_1}, \ldots, \zeta_{\ell_j})$ where $\ell_1, \ldots, \ell_j$ are the elements in $[\ell] \setminus \beta_i$ ordered increasingly. Then, we let $\xi_{W'} = \xi_n$. Intuitively, each occurrence of a subtree $B(\xi_{1..\ell})$ in ξ is replaced by a subtree $B_\beta(\xi'_{1..\ell'})$ where β is some subset of $[\ell]$.

For each $W' \subseteq W$ as above, we let R' contain the rule $A_\alpha(x_{k_1}, \ldots, x_{k_q}) \to \xi_{W'}$ where α is the set of all variable indices of X_k that do not occur in $\xi_{W'}$ and $x_{k_1}, \ldots, x_{k_q}$ are the variables of X_k that occur in $\xi_{W'}$ in ascending order.

It can be shown that each tree which can be derived in G can also be derived in G' and vice versa [20, Thm. 3.1.10]. The lnCFTG G' merely predicts the deletion that will happen in G and refrains from creating the respective argument positions. ∎

Useless Rules of a lnCFTG. There may be nonterminals in a lnCFTG G that cannot contribute in a meaningful way to $\mathcal{L}(G)$. A nonterminal A may be unreachable from the initial nonterminal, or it can be such that there is no terminal tree derivable from it.

Definition 2.2.18 A nonterminal $A \in N$ is called *reachable*, if there is a $\xi \in \mathrm{T}_{N \cup \Delta}$ such that $A_0 \Rightarrow^* \xi$ and $\mathrm{pos}_A(\xi) \neq \emptyset$. We call A *productive*, if $\mathcal{L}(G, A(\overline{x})) \neq \emptyset$. If a nonterminal is not reachable or not productive, we call it *useless*.

If a rule r has a useless nonterminal as its LHS-nonterminal or a useless nonterminal occurs in its RHS, then the rule is said to be *useless* as well. Each nonterminal and, respectively, each rule which is not useless is considered *useful*. □

In a lnCFTG, all useless nonterminals and useless rules can be eliminated without changing the induced tree language. This is very similar to the case of CFG and is described in [31, Thm. 4.3].

Corollary 2.2.19 For each lnCFTG G, there is an equivalent lnCFTG G' that does not contain useless rules or useless nonterminals.

Note that the concept of useless nonterminals is only defined for the restricted class of lnCFTG rather than for the full class of CFTG. This is due to the fact that a CFTG could delete the trees generated by a nonterminal. Checking for such phenomena requires a more complicated algorithm. Since in this thesis, we utilize Corollary 2.2.19 only for lnCFTGs, an algorithm that applies to an arbitrary CFTG is not investigated.

Closures of Regular Tree Languages. We recall three theorems summarizing closure properties of regular tree languages.

Theorem 2.2.20 [26, Prop. 7.1 and 7.3] We let U be a set and $L_1, L_2 \subseteq \mathrm{T}_\Delta(U)$ be regular tree languages. Furthermore, we let $u \in \Delta^{(0)} \cup U$. Then, the union $L_1 \cup L_2$, the intersection $L_1 \cap L_2$, the set difference $L_1 \setminus L_2$, the complement $\mathrm{T}_\Delta(U) \setminus L_1$, and the tree concatenation $L_1[u/L_2]$ are regular tree languages.

The following definition of the Kleene star is similar to the string case. Instead of extending the string at its end, there is a designated nullary concatenation symbol u. Note that u may occur multiple times in one tree.

Definition 2.2.21 Let U be a set, $L \subseteq \mathrm{T}_\Delta(U)$ and $u \in \Delta^{(0)} \cup U$. We define the *Kleene star* of L, denoted by L^{*u}, to be the tree language $L^{*u} = \bigcup_{i \in \mathbb{N}} L^i$ where $L^0 = \{u\}$ and $L^i = L^{i-1}[u/L]$ for each $i \in \mathbb{N}_+$. □

Theorem 2.2.22 [26, Prop. 7.5] We let U be a set, $L \subseteq \mathrm{T}_\Delta(U)$, and $u \in \Delta^{(0)} \cup U$. If L is a regular tree language, then L^{*u} is a regular tree language.

The third and last theorem deals with tree substitution.

Theorem 2.2.23 We let $k \in \mathbb{N}_+$, $\delta_1, \ldots, \delta_k \in \Delta$ be pairwise different, $L \subseteq \mathrm{T}_\Delta(X)$, and $L_1, \ldots, L_k \subseteq \mathrm{T}_\Delta(X)$ such that, for each $i \in [k]$, we have $L_i \subseteq \mathrm{C}_\Delta(X_{\mathrm{rk}_\Delta(\delta_i)})$. If L and $L_1, \ldots, L_k$ are regular tree languages, then $L \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k)$ is a regular tree language.

Proof. This proof utilizes the notion of a tree transducer which is formally defined in, e.g., [39]. Since $L_1, \ldots, L_k$ are regular tree languages, there is a linear nondeleting

recognizable tree transducer $\mathcal{M}$ [39, p. 146] over the Boolean semiring $\mathbb{B}$ such that, for each input tree ξ, we have $\langle\langle\mathcal{M}\rangle\rangle(\xi) = \xi \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k)$ where $\langle\langle\mathcal{M}\rangle\rangle$ is the transduction computed by $\mathcal{M}$. We note that recognizable tree series over $\mathbb{B}$ coincide with regular tree languages. It is clear that $\langle\langle\mathcal{M}\rangle\rangle(L') = L' \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k)$ for any tree language L'.

In [39, Cor. 14] it is shown that, for any commutative continuous semiring S, the class of recognizable tree series over S is closed under linear nondeleting recognizable tree transductions over S. Hence, by applying this closure result to $\mathcal{M}$ and the regular tree language L, we obtain that $L \leftarrow (\delta_1/L_1, \ldots, \delta_k/L_k)$ is a regular tree language. ∎

Tree Representation of a CFG. Recall the tree representation $\widetilde{s}$ of a string $s \in \Sigma$ using the unary alphabet $\Sigma \cup \{\sharp\}$ (cf. Definition 2.1.12). We let $\widetilde{s}[\sharp/x_1]$ be the tree obtained from $\widetilde{s}$ by replacing the single occurrence of $\sharp$ by x_1. We show that the language induced by a CFG can be represented by a lnCFTG which uses only unary nonterminals (except for the nullary initial nonterminal) and unary terminals (except for the special nullary terminal $\sharp$). This result corresponds to [15, Thm. 7.13] for the special case of $n = 1$.

Definition 2.2.24 For each CFG $M = (N, \Sigma, B_0, R)$ we define the *tree representation of* M, denoted by $\widetilde{M}$, as the lnCFTG $\widetilde{M} = (N', \widetilde{\Sigma}, A_0, R')$ as follows. We let $N' = N \cup \{A_0\}$ where A_0 is a new nonterminal, $\mathrm{rk}_{N'}(B) = 1$ for each $B \in N$, and $\mathrm{rk}_{N'}(A_0) = 0$. For each $B \to s$ in R, we let $B(x_1) \to \widetilde{s}[\sharp/x_1]$ be a rule in R'. Furthermore, we add $A_0 \to B_0(\sharp)$ to R'. □

Observation 2.2.25 For each CFG M, we have that $\mathcal{L}(\widetilde{M}) = \{\widetilde{s} \mid s \in \mathcal{L}(M)\}$.

It can be seen that $\widetilde{M}$ is of a special form. Each lnCFTG G of this form can be turned into a CFG recognizing the path language of $\mathcal{L}(G)$.

Definition 2.2.26 A lnCFTG $G = (N, \Delta, A_0, R)$ is called *strongly monadic*, if $\Delta = \widetilde{\Sigma}$ for some alphabet Σ, $N = N^{(1)} \cup \{{A_0}^{(0)}\}$, and the only rule with LHS-nonterminal A_0 is of the form $A_0 \to B_0(\sharp)$ for some $B_0 \in N^{(1)}$. □

The construction of the following observation is straightforward.

Observation 2.2.27 For each monadic lnCFTG G, there is a CFG M such that $\mathcal{L}(G) = \{\widetilde{s} \mid s \in \mathcal{L}(M)\}$.

We note that the tree representation of M is a strongly monadic lnCFTG. Hence, the following observation is intuitively clear.

Observation 2.2.28 There is a one-to-one correspondence between CFGs and strongly monadic lnCFTGs.

3 Non-Self-Embedding CFTGs

This chapter is mainly based on the journal article "Non-Self-Embedding Linear Context-Free Tree Grammars Generate Regular Tree Languages" which contains results I investigated in cooperation with my coauthors Mark-Jan Nederhof and Heiko Vogler [51]. Some parts are taken over verbatim, other parts are modified to fit the presentation of this thesis, and some results are added.

In [11] it was proved that each context-free string grammar (CFG) which is non-self-embedding generates a regular string language, where self-embedding means the existence of a derivation of the form $A \Rightarrow^* sAt$ with $s \neq \varepsilon$ and $t \neq \varepsilon$ for some strings s and t over terminals and nonterminals. In this chapter, we present a similar result for context-free tree grammars (CFTGs). We present a definition for self-embedding CFTGs and show that it is syntactically decidable. Note that we define the property of being self-embedding for all CFTGs. However, copying of argument positions is a source of non-regularity, since RTGs cannot copy trees. Hence, in Sections 3.2 to 3.5 we first consider restricted CFTGs, viz. non-self-embedding linear nondeleting CFTGs (lnCFTGs). Afterwards, in Sections 3.6 to 3.10, we consider more general CFTGs. As our main result of this chapter, we construct for each non-self-embedding lnCFTG an equivalent RTG, thus, in particular, we show that each non-self-embedding lnCFTG induces a regular tree language.

To motivate our criterion for regularity of the tree language induced by a lnCFTG, we illustrate potential sources of non-regularity using two examples. As a first example, we present the lnCFTG G_4 with the rules

$$A_0 \to A(\alpha, \alpha) \quad , \qquad A(x_1, x_2) \to \delta(B(x_1, x_2)) \mid \kappa(x_1, x_2) \quad , \text{ and} \qquad B(x_1, x_2) \to \gamma(A(x_1, x_2)) \quad .$$

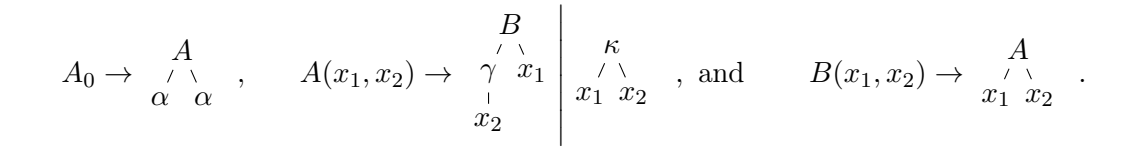
By inspecting the rules, it can be seen that $\mathcal{L}(G_4) = \{\delta^n(\kappa(\gamma^n(\alpha), \alpha)) \mid n \in \mathbb{N}\}$. By a pumping argument, it can be shown that $\mathcal{L}(G_4)$ is not regular. Furthermore, the derivation

$$A(x_1, x_2) \Rightarrow \delta(B(x_1, x_2)) \Rightarrow \delta(A(\gamma(x_1), x_2)) \Rightarrow \delta(\delta(B(\gamma(x_1), x_2))) \Rightarrow \delta(\delta(A(\gamma(\gamma(x_1)), x_2))) \Rightarrow \delta(\delta(\kappa(\gamma(\gamma(x_1)), x_2)))$$

of G_4 starting from $A(x_1, x_2)$ illustrates that δ's and γ's are synchronously generated above and below a repeated occurrence of the nonterminal A. Such non-regular generation

corresponds to the string case as follows. Instead of generating symbols to the left and to the right of repeated nonterminals, the symbols are generated above and below such nonterminal occurrences (also confer to the tree representation of a string in Definition 2.1.12). It is therefore tempting to try to define the notion of self-embedding for lnCFTGs in terms of the familiar notion of self-embedding for CFGs, applied to the path language of the trees derived from a lnCFTG. However, there is an additional source of non-regularity in lnCFTG that cannot be captured solely in terms of path languages as seen in the next example.

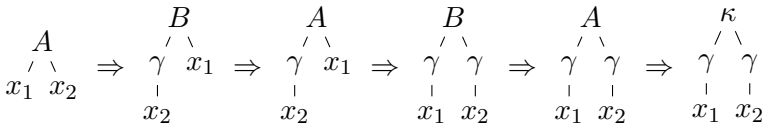
In the second example of this chapter, we present the lnCFTG G_5 which has the rules



By inspecting the rules it can be seen that

$$\mathcal{L}(G_5) = \{\kappa(\gamma^n(\alpha), \gamma^n(\alpha)) \mid n \in \mathbb{N}\} \cup \{\kappa(\gamma^{n+1}(\alpha), \gamma^n(\alpha)) \mid n \in \mathbb{N}\} \ .$$

By a pumping argument, it can be shown that $\mathcal{L}(G_5)$ is not regular. We consider the derivation

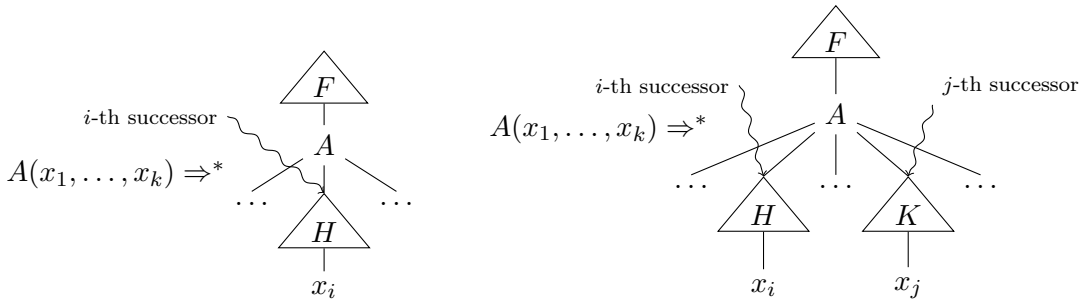


of G_5 starting from $A(x_1, x_2)$. We note that the numbers of γ's in the two argument positions of the repeated nonterminal A grow in a synchronized manner, viz. after four derivation steps, both argument positions contain the same number of γ's. Its noteworthy that the tree language generated from A_0 is not regular, even though the path language of the tree language induced by G_5 is a regular string language.

The lnCFTGs G_4 and G_5 are instances of the two types of potential non-regular behavior within CFTGs. We want to capture both types using our notion of self-embedding. We say that a CFTG is *self-embedding* if at least one of the two properties illustrated in Figure 3.1 is satisfied: Property (1) generalizes self-embedding from the string case (applied to paths), while Property (2) captures potential non-regularity due to different branches growing in a synchronized manner (cf. Section 3.1 for the formal definition).

Our main result of this chapter is the proof that a non-self-embedding lnCFTG generates a regular tree language. Our use of the term 'non-self-embedding' may already suggest this result, by analogy with the string case. However, due to the additional source of non-regularity (as described above), novel proof techniques are needed, which involve complications far beyond those of the string case. In the following, we describe the structure of this chapter.

First, in Section 3.1, we give the formal definition of self-embedding CFTG and show how this property can be decided. Then, we consider a restricted CFTG, viz. a non-self-embedding lnCFTG G. In Section 3.2, we prepare the proof that G induces a regular tree language. We transform G into an equivalent non-self-embedding lnCFTG G' that



(1): F and H are non-trivial trees.

(2): H and K are non-trivial trees.

Figure 3.1: Properties for self-embedding ($i, j \in \{1, \ldots, k\}$).

satisfies the novel property of being *unique in argument positions*. Roughly speaking, the effect of this transformation is that generation of symbols in distinct argument positions of one nonterminal of G is done through several newly introduced nonterminals in G'. This transformation is possible because the negation of Property (2) guarantees that the generation of symbols in distinct argument positions of one and the same nonterminal is independent.

Next, in Section 3.3, we analyze a non-self-embedding lnCFTG G which is unique in argument positions. We consider related nonterminals via the analysis of a certain graph. We can see that, due to Property (1) of self-embedding, unboundedly many symbols can never be created synchronously above and below a nonterminal. Hence, we can divide the generation into two classes, namely top-recursion, which deals with unbounded generation below a nonterminal, and bottom-recursion, which deals with unbounded generation above a nonterminal.

Relying on the properties of non-self-embedding and uniqueness in argument positions, top-recursion can be transformed to bottom-recursion. This is by a construction described in detail in Section 3.3.1. Subsequently, in Section 3.3.2, we show that a non-self-embedding lnCFTG which does not contain any top-recursion can be transformed into an equivalent RTG. This relies on the observation that the number of distinct values that may appear below a nonterminal is bounded. Summing up over Section 3.3, we prove our main theorem in Section 3.3.3.

In Section 3.4 we relate the definition of self-embedding for lnCFTG to the one used in the string case [47]. There is a close connection between the string approach and our result restricted to the tree representation of strings. Furthermore, in the string case (cf. [47, Sec. 3]), there is a conceptually different approach which is used to *approximate* a self-embedding CFG. A CFG M is first split into parts, these are approximated by individual REGs. The resulting family of REGs is then combined into one REG M' that approximates M. In Section 3.5, we use a similar concept to give an alternative proof of the regular *characterization* of non-self-embedding lnCFTG. This proof is an alternative to the one in Section 3.3.2. We split a lnCFTG G that is unique in argument positions and does not contain top-recursion into several parts and construct one *equivalent* RTG for each part. This procedure is explained in detail in Section 3.5.1. Afterwards, in Section 3.5.2, we describe how the obtained family of RTGs can be combined into one RTG H such that G

and H induce the same tree language.

Then we turn to more general CFTGs and further results centered around the property of self-embedding. In Section 3.6, we show that non-self-embedding linear CFTGs which may be deleting also induce regular tree languages. We recall another string generalization, viz. a self-embedding property for indexed grammars [54] in Section 3.7. We will generalize that property, called weakly-self-embedding, to CFTGs and show that each non-weakly-self-embedding CFTG induces a regular tree language. Next, we describe in Section 3.8 how our results can be transferred to macro grammars. In Section 3.9, we give an overview of the results obtained in the previous sections. Lastly, in Section 3.10, we show that non-self-embedding lnCFTGs can express tree languages more succinctly than RTGs, and we relate the self-embedding property to coregular languages.

3.1 Characterization of Self-Embedding CFTGs

We generalize Chomsky's notion of self-embedding [11] to the tree case. We first give the formal definition and then show a syntactic and decidable property such that a CFTG G satisfies the property iff G is self-embedding. We furthermore show that the syntactic property is decidable and thus our definition of self-embedding is decidable as well.

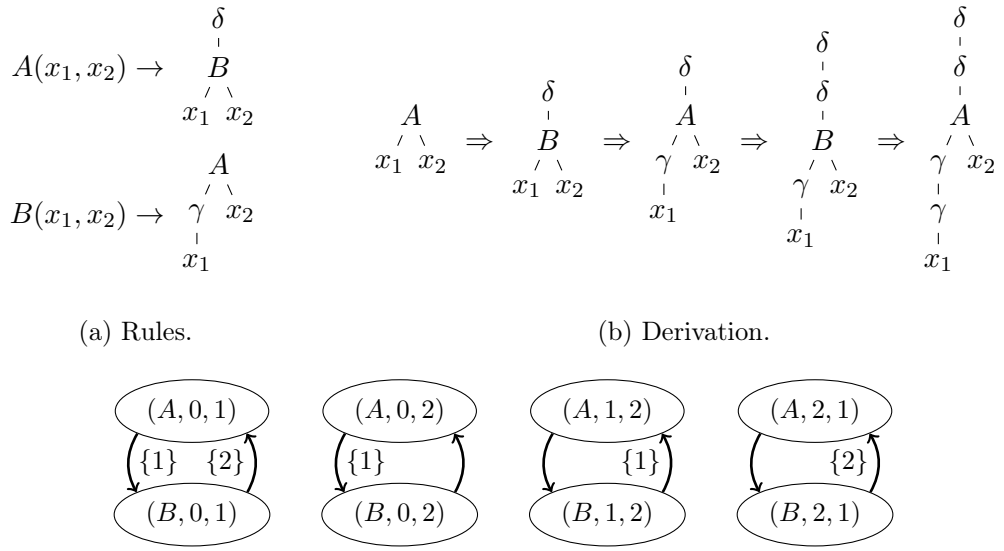
Definition 3.1.1 Let $G = (N, \Delta, A_0, R)$ be a CFTG. We say that G is *self-embedding* if there is a $k \in \mathbb{N}_+$ and an $A \in N^{(k)}$ such that at least one of the following two properties holds (viewing the variables in X_k as symbols with rank 0):

(1) There is an $i \in [k]$ and there are $F, A', H \in \mathrm{C}_{N \cup \Delta \cup X_k}(\{z\})$ such that
- $A(\overline{x}) \Rightarrow^* F[A'[H[x_i]]]$,
- $A'(\varepsilon) = A$, $A'(i) = z$, and
- $F \neq z$ and $H \neq z$.

(2) There are $i, j \in [k]$ with $i \neq j$ and there are $F, H, K \in \mathrm{C}_{N \cup \Delta \cup X_k}(\{z\})$ and $A' \in \mathrm{C}_{N \cup \Delta \cup X_k}(\{z_1, z_2\})$ such that
- $A(\overline{x}) \Rightarrow^* F[A'[H[x_i], K[x_j]]]$,
- $A'(\varepsilon) = A$, $A'(i) = z_1$, and $A'(j) = z_2$, and
- $H \neq z$ and $K \neq z$. □

The two properties of Definition 3.1.1 are depicted in Figure 3.1 and we illustrate them by three examples. Simultaneously, we will motivate the introduction of a particular finite graph which allows checking of these properties. Recall that we focus on lnCFTGs and thus, all examples will present lnCFTGs.

Example 3.1.2 As a first example, we recall the lnCFTG G_4 from the beginning of Chapter 3. To have all ingredients to the following arguments in one place, we repeat parts of the rules of G_4 in Figure 3.2(a). The derivation of G_4 depicted in Figure 3.2(b) shows that G_4 satisfies Property (1) of self-embedding: terminals are created above and below the repeated occurrence of the nonterminal A in a synchronized manner (the numbers of δ's and γ's are equal). To detect this phenomenon, it suffices to consider a finite directed



(a) Rules. (b) Derivation.

(c) Position pair graph.

Figure 3.2: Part of the lnCFTG G_4.

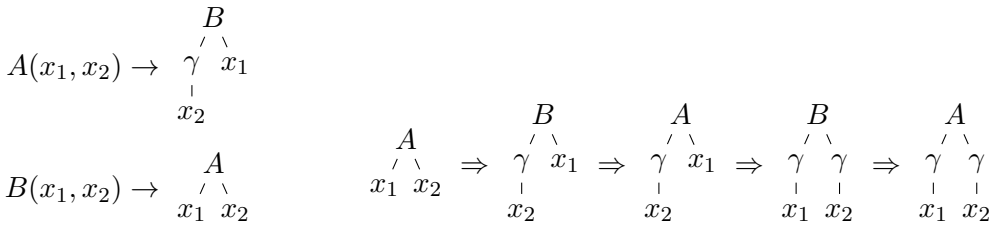
graph, called position pair graph, of which each vertex is a triple: a nonterminal and two of its argument positions. We include argument position 0, which represents the generation happening above the nonterminal; it may only occur in the first of the two argument positions. An edge from $(A, 0, j)$ to $(B, 0, m)$ indicates that there is a rule r with LHS-nonterminal A such that x_j appears in the argument position m of an occurrence of B in the RHS of r. An edge can be labeled by any subset of $\{1, 2\}$ (we drop the label $\emptyset$ in the figures). If there is a symbol above the occurrence of B, then the label contains a 1; if at least one symbol occurs between the occurrences of B and x_j, then the label of this edge contains a 2. The 1 pertains to the first of the two argument positions in $(A, 0, j)$ and $(B, 0, m)$, which are both 0, while 2 pertains to the second argument positions, which are j and m, respectively.

The leftmost two SCCs in Figure 3.2(c) show part of the position pair graph of G_4 dealing with this combination of generation above and below a nonterminal. (The rightmost two SCCs will be explained in Example 3.1.3.) The first two steps of the derivation in Figure 3.2(b) correspond to the path

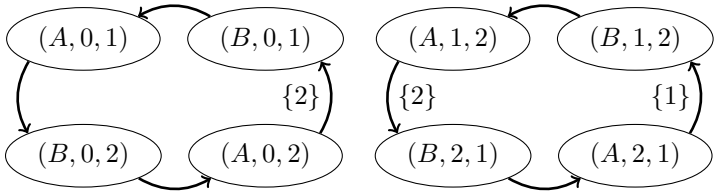
$$(A, 0, 1) \xrightarrow{\{1\}} (B, 0, 1) \xrightarrow{\{2\}} (A, 0, 1)$$

through the position pair graph. Since (i) this path is cyclic, (ii) the union of the edge labels contains 1 and 2, and (iii) the argument position 0 is involved, Property (1) of self-embedding is satisfied. ○

Example 3.1.3 As a second example, we recall two rules (cf. Figure 3.3(a)) of the lnCFTG G_5 from the beginning of Chapter 3. The derivation in Figure 3.3(b) shows



(a) Rules. (b) Derivation.



(c) Position pair graph.

Figure 3.3: Part of the lnCFTG G_5.

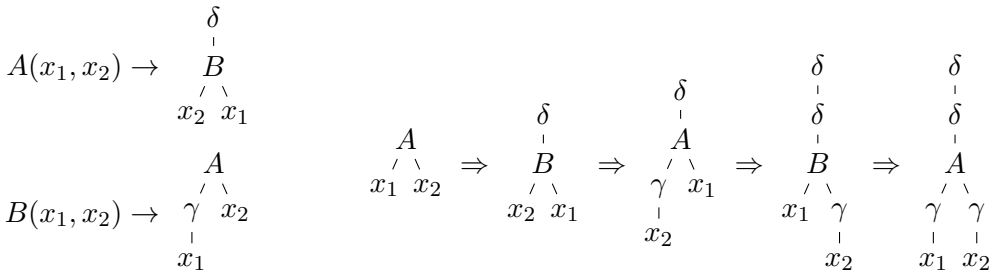
that G_5 generates terminals in a synchronized manner in two argument positions below a nonterminal: the numbers of γ's are equal after each fourth step. To capture this in the position pair graph, we employ vertices with two argument positions different from 0. An edge from (A, i, j) to (B, ℓ, m) (with $i \neq j$ and $\ell \neq m$) indicates that there is a rule r with LHS-nonterminal A such that x_i appears in the argument position ℓ of an occurrence of B in the RHS and x_j appears in the argument position m of the same occurrence of B. If at least one symbol occurs between the occurrences of B and x_i, then the edge label contains a 1. Likewise, if at least one symbol occurs between the occurrence of B and x_j, then the edge label contains a 2.

The position pair graph of G_5 is the one shown in Figure 3.3(c). The derivation in Figure 3.3(b) corresponds to the path

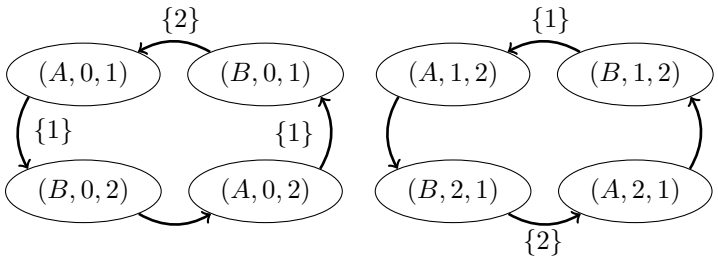
$$(A, 1, 2) \xrightarrow{\{2\}} (B, 2, 1) \longrightarrow (A, 2, 1) \xrightarrow{\{1\}} (B, 1, 2) \longrightarrow (A, 1, 2)$$

through the rightmost SCC of the position pair graph. Since (i) this path is cyclic, (ii) its union of edge labels contains 1 and 2, and (iii) the position 0 is not involved, Property (2) of self-embedding is satisfied. ○

Example 3.1.4 In the third and last example, we recall the lnCFTG G_2 from Example 2.2.7. In this example, we only consider the two rules of G_2 repeated in Figure 3.4(a). The lnCFTG G_2 simultaneously satisfies Properties (1) and (2) of self-embedding. It combines the non-regular behavior of G_4 and G_5 (cf. Examples 3.1.2 and 3.1.3). Considering the derivation of G_2 from Figure 3.4(b) it can be seen that symbols are synchronously produced above and below the repeated nonterminal A as well as in both arguments positions of A. In Figure 3.4(c) we depict the position pair graph of G_2. Note that there are two SCCs and in each SCC there is a cycle whose union of edge labels contains 1 and 2. ○



(a) Rules. (b) Derivation.



(c) Position pair graph.

Figure 3.4: Part of a lnCFTG G_2.

Next we formally define the notion of position pair graph.

Definition 3.1.5 Let $G = (N, \Delta, A_0, R)$ be a CFTG. The *position pair graph of* G is the $\{1, 2\}$-labeled directed graph $\mathrm{ppg}(G) = (V, E)$ where

$$V = \{(A, i, j) \mid A \in N^{(k)},\ i \in ([k] \cup \{0\}),\ j \in [k],\ i \neq j\}$$

and E is defined as follows. We let $(A, 0, j), (B, 0, m) \in V$ and $r \in R|_A$ such that there is a $w \in \mathrm{pos}_B(\mathrm{rhs}(r))$ for which wm is x_j-dominating. Then $((A, 0, j), U, (B, 0, m)) \in E$ where $U \subseteq \{1, 2\}$ is defined as follows:

- $w \neq \varepsilon$ iff $1 \in U$,
- $\mathrm{rhs}(r)(wm) \neq x_j$ iff $2 \in U$.

Furthermore, we let $(A, i, j), (B, \ell, m) \in V$ with $i \neq 0$, $\ell \neq 0$, and we let $r \in R|_A$ be such that there exists a $w \in \mathrm{pos}_B(\mathrm{rhs}(r))$ for which $w\ell$ is x_i-dominating and wm is x_j-dominating. Then $((A, i, j), U, (B, \ell, m)) \in E$ where $U \subseteq \{1, 2\}$ is defined as follows:

- $\mathrm{rhs}(r)(w\ell) \neq x_i$ iff $1 \in U$,
- $\mathrm{rhs}(r)(wm) \neq x_j$ iff $2 \in U$. □

Observation 3.1.6 Let $P \in \mathrm{scc}(\mathrm{ppg}(G))$. Then exactly one of the following two statements holds:

- For each vertex $(A, i, j) \in \mathrm{V}_P$, we have $i = 0$.
- For each vertex $(A, i, j) \in \mathrm{V}_P$, we have $i \neq 0$.

Now we formally characterize the property of the position pair graph of G that allows detecting whether a CFTG G is self-embedding or non-self-embedding.

Theorem 3.1.7 A CFTG G is self-embedding iff $\mathrm{ppg}(G)$ contains a vertex (A, i, j) and a path from (A, i, j) to (A, i, j) such that the union of all its labels contains 1 and 2.

Proof. [⇒]: If G is self-embedding, then Property (1) or (2) holds. If Property (1) holds, there is a derivation starting from $A(\overline{x})$ resulting in a tree with an x_i-dominating occurrence of A which is not at the root, and x_i occurs in its i-th argument position but not as its direct descendant. This derivation corresponds to a cycle in $\mathrm{ppg}(G)$ of the same length. Since symbols are generated both above A and between A and x_i, the union of the edge labels contains 1 and 2.

Similarly, if Property (2) holds, there is a derivation that corresponds to a cycle in $\mathrm{ppg}(G)$. The union of the edge labels of this path contains 1 and 2, because symbols are synchronously generated under two different argument positions.

[⇐]: Suppose that there is a cycle in the position pair graph and the union of its edge labels contains 1 and 2. Then we can construct a derivation in G which satisfies Property (1) or (2): For each edge in the cycle, we apply a rule that gave rise to this edge in the construction of $\mathrm{ppg}(G)$. ∎

Corollary 3.1.8 It is decidable in polynomial time whether a CFTG G is self-embedding.

Proof. The position graph of G can be constructed in polynomial time in the following parameters of G: number of nonterminals, the maximal rank of the nonterminals, the number of rules, and the maximal number of occurrences of nonterminals in the RHS of any rule. All SCCs of $\mathrm{ppg}(G)$ can be enumerated in linear time [13, p. 617]. For each SCC in $\mathrm{ppg}(G)$ it can be determined in linear time whether the union of all its edge labels is $\{1, 2\}$. By Theorem 3.1.7 this is all that is required to decide whether G is self-embedding. ∎

We note that by Definition 3.1.5, for each RTG H, we have $\mathrm{ppg}(H) = (\emptyset, \emptyset)$. Thus, the following observations follows from Theorem 3.1.7.

Observation 3.1.9 Each RTG H is non-self-embedding.

The reader might have realized that none of the examples presented so far contains nested nonterminals. This choice is reasonable, because each grammar remains self-embedding even if one replaces any occurrence of a terminal by a nonterminal (with arbitrary rules). However, applying this replacement to a non-self-embedding CFTG might lead to a non-self-embedding CFTG or a self-embedding CFTG. For instance, if we replace in the non-self-embedding lnCFTG with the rules

$$A(x) \to A(B(x)) \qquad\qquad B(x) \to \gamma(x)$$

the second rule by $B(x) \to A(x)$, then the resulting grammar is self-embedding. The formal investigation allows for nested nonterminals; however, to present examples in a compact and intuitive way, they will not contain nested nonterminals.

3.2 Movement of Values in Argument Positions

In this section, we introduce two concepts. First, we define the *position graph* of a CFTG in Section 3.2.1. This graph allows tracking of how the values in the argument positions of a nonterminal are moved during a derivation. In particular, we are interested whether the tree in an argument position of one nonterminal is moved into an argument position of another nonterminal.

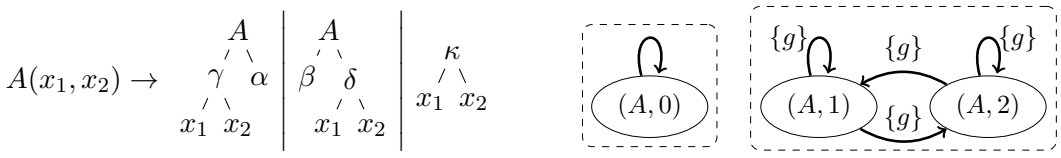
Second, in Section 3.2.2, we consider a property for *linear nondeleting* CFTGs called *unique in argument positions*. We prove that each non-self-embedding lnCFTG can be transformed into an equivalent non-self-embedding lnCFTG which is unique in argument positions. This syntactic restriction will turn out to be useful to construct an equivalent RTG to each non-self-embedding lnCFTG.

3.2.1 Position Graph

In this section, we let $G = (N, \Delta, A_0, R)$ be a CFTG.

The position graph contains one vertex for each pair of nonterminal and argument position (including the special argument position 0 as in the case of the position pair graph). Its edges represent the movement of values across argument positions of nonterminal occurrences. An edge is labeled with $\{g\}$ if new symbols are generated during movement, and with $\emptyset$ otherwise.

Example 3.2.1 As an example, we present some rules of the lnCFTG G_7 in Figure 3.5(a). The position graph of that part of G_7 is shown in Figure 3.5(b). We explain the two edges going out of $(A, 1)$. The rules of Figure 3.5(a) are denoted by r_1, r_2, and r_3 in order of appearance. Since we are interested in edges from $(A, 1)$, we inspect the variable x_1 in all rules with LHS-nonterminal A. In r_1, the variable x_1 occurs in the first argument position of the occurrence of A. Furthermore, the rule generates the symbol γ in between the occurrence of A and x_1. Hence, $\mathrm{pg}(G_7)$ contains the edge $(A, 1) \overset{\{g\}}{\to} (A, 1)$. Similarly, in r_2, the variable x_1 occurs in the second argument position of A and the symbol δ is produced in between. Thus, $(A, 1) \overset{\{g\}}{\to} (A, 2)$ is an edge in $\mathrm{pg}(G_7)$. The variable x_1 also occurs in r_3; however, it is not in the argument position of any nonterminal and thus, no edge is induced. ○



(a) Rules. (b) Position graph.

Figure 3.5: Part of the lnCFTG G_7.

Definition 3.2.2 The *position graph of* G is the $\{g\}$-labeled directed graph $\mathrm{pg}(G) = (V, E)$ where

$$V = \{(A, i) \mid A \in N^{(k)}, i \in ([k] \cup \{0\})\} \ .$$

In order to obtain E, we first define the auxiliary mapping $\mathrm{edg}\colon R \to \mathcal{P}(V \times \mathcal{P}(\{g\}) \times V)$ as follows. Let $A, B \in N$ and $r \in R|_A$ such that there exists a variable dominating position $w \in \mathrm{pos}_B(\mathrm{rhs}(r))$.

- If $w \neq \varepsilon$, then $\mathrm{edg}(r)$ contains the edge $((A, 0), \{g\}, (B, 0))$ and
- if $w = \varepsilon$, then $\mathrm{edg}(r)$ contains the edge $((A, 0), \emptyset, (B, 0))$.

Moreover, for each $i \in [\mathrm{rk}_N(A)]$ and $j \in [\mathrm{rk}_N(B)]$ such that wj is x_i-dominating we have that

- if $\mathrm{rhs}(r)(wj) \neq x_i$, then $\mathrm{edg}(r)$ contains the edge $((A, i), \{g\}, (B, j))$ and
- if $\mathrm{rhs}(r)(wj) = x_i$, then $\mathrm{edg}(r)$ contains the edge $((A, i), \emptyset, (B, j))$.

Furthermore, $\mathrm{edg}(r)$ does not contain any other elements. We define $E = \bigcup_{r \in R} \mathrm{edg}(r)$. □

If an edge is labeled by $\{g\}$, then we call it *generating*. We call $P \in \mathrm{scc}(\mathrm{pg}(G))$ *generating* if P contains a generating edge. Sometimes we will also be interested in the set of rules which have induced edges in P. Formally, for each $P \in \mathrm{scc}(\mathrm{pg}(G))$, we define the set of *rules of* P, denoted by $\mathrm{rules}(P)$, to be the set

$$\mathrm{rules}(P) = \{r \in R \mid \mathrm{E}_P \cap \mathrm{edg}(r) \neq \emptyset\} \ .$$

Let $n \in \mathbb{N}$ and $r_1, r_2, \ldots, r_n \in R$. We say that the sequence $r_1 r_2 \ldots r_n$ *induces a path*

$$(A_1, i_1) \to (A_2, i_2) \to \ldots \to (A_{n+1}, i_{n+1})$$

in $\mathrm{pg}(G)$ if, for each $k \in [n]$, the set $\mathrm{edg}(r_k)$ contains the edge $(A_k, i_k) \to (A_{k+1}, i_{k+1})$. For instance consider the rules in Figure 3.5(a), which we denote by r_1, r_2, and r_3, respectively. The sequence of rules $d = r_1 r_2 r_2$ induces the paths

$$\begin{aligned}
p_1\colon&\ (A,0) \overset{\emptyset}{\to} (A,0) \overset{\emptyset}{\to} (A,0) \overset{\emptyset}{\to} (A,0) \ , \\
p_2\colon&\ (A,1) \overset{\{g\}}{\to} (A,1) \overset{\{g\}}{\to} (A,2) \overset{\{g\}}{\to} (A,2) \ , \text{ and} \\
p_3\colon&\ (A,2) \overset{\{g\}}{\to} (A,1) \overset{\{g\}}{\to} (A,2) \overset{\{g\}}{\to} (A,2) \ .
\end{aligned}$$

We make three observations concerning the position graph, which will help us later.

Observation 3.2.3 Let $P \in \mathrm{scc}(\mathrm{pg}(G))$. Then exactly one of the following two statements holds:

- For each vertex $(A, i) \in \mathrm{V}_P$, we have $i = 0$.
- For each vertex $(A, i) \in \mathrm{V}_P$, we have $i \neq 0$.

Observation 3.2.4 Let $A, B \in N$ and $r \in R|_A$. Then $(A,i) \to (B,j)$ is in $\mathrm{edg}(r)$ for some $i \in [\mathrm{rk}_N(A)]$ and $j \in [\mathrm{rk}_N(B)]$ iff $(A,0) \to (B,0)$ is in $\mathrm{edg}(r)$.

Observation 3.2.5 Let $M \subseteq N$ and $M' = \{(A,i) \mid A \in M, i \in ([\mathrm{rk}_N(A)] \cup \{0\})\}$. By definition of the respective fragments, we have that $\mathrm{pg}(G|_M) = \mathrm{pg}(G)|_{M'}$.

For $P \in \mathrm{scc}(\mathrm{pg}(G))$, we denote *the set of all nonterminals occurring in* P by M_P, i.e., $M_P = \{A \mid k \in \mathbb{N}, (A,k) \in \mathrm{V}_P\}$.

If G is non-self-embedding, then the rules of a generating SCC of $\mathrm{pg}(G)$ that does not contain references to 0 have a particular form. This will be crucial while transforming the grammar.

Lemma 3.2.6 We let G be a non-self-embedding CFTG and $P \in \mathrm{scc}(\mathrm{pg}(G))$ be generating such that $(C,0) \notin \mathrm{V}_P$ for each $C \in N$. Then each rule in $\mathrm{rules}(P)$ has the form $A(\overline{x}) \to B(\zeta_{1..\ell})$ for some $A, B \in M_P$ and $\zeta_{1..\ell} \in \mathrm{T}_{N \cup \Delta}(X_k)$.

Proof. We prove this lemma by contradiction. Assume that there is a rule r in $\mathrm{rules}(P)$ such that $\mathrm{rhs}(r)(\varepsilon) \notin M_P$. Since $r \in \mathrm{rules}(P)$ and $(C,0) \notin \mathrm{V}_P$ for each $C \in N$, it follows that there are $A, B \in M_P$, $i \in [\mathrm{rk}_N(A)]$, $j \in [\mathrm{rk}_N(B)]$, and $U \subseteq \{g\}$ such that $((A,i), U, (B,j)) \in \mathrm{E}_P \cap \mathrm{edg}(r)$. From Definition 3.2.2, we get that there is an x_i-dominating position $w \in \mathrm{pos}_B(\mathrm{rhs}(r))$. By the assumption, we have that $w \neq \varepsilon$.

Since (A,i) and (B,j) are vertices in the same generating SCC P, there are $C, D \in M_P$, $m \in [\mathrm{rk}_N(C)]$, and $n \in [\mathrm{rk}_N(D)]$ such that $((C,m), \{g\}, (D,n)) \in \mathrm{E}_P$ and thus, the following path exists in P:

$$p : (A,i) \to (B,j) \to \ldots \to (C,m) \overset{\{g\}}{\to} (D,n) \to \ldots \to (A,i)$$

where the first edge is induced by r.

We will now use p and construct a cycle in $\mathrm{ppg}(G)$. For each edge $(A',i') \to (B',j')$ in p, there are $r' \in \mathrm{rules}(P)$ and $w' \in \mathrm{pos}_{B'}(\mathrm{rhs}(r'))$ such that $w'j'$ is $x_{i'}$-dominating. Then, by Definition 3.1.5, there is an edge $((A',0,i'), U', (B',0,j'))$ in $\mathrm{ppg}(G)$ for some $U' \subseteq \{1,2\}$. Hence, we obtain the cycle

$$p' : (A,0,i) \overset{U_1}{\to} (B,0,j) \to \ldots \to (C,0,m) \overset{U_2}{\to} (D,0,n) \to \ldots \to (A,0,i) \ .$$

Now we investigate U_1 and U_2. First consider the edge $((A,0,i), U_1, (B,0,j))$. This edge is induced using the rule r at position w in $\mathrm{rhs}(r)$. By Definition 3.1.5, we have that $1 \in U_1$, because $w \neq \varepsilon$. Second, consider the edge $((C,0,m), U_2, (D,0,n))$. Since $((C,m), \{g\}, (D,n))$ is in $\mathrm{pg}(G)$, there is a rule $r' \in \mathrm{rules}(P)$ that induced this edge and a position $w' \in \mathrm{rhs}(r')$ such that $w'n$ is x_m-dominating, and $\mathrm{rhs}(r')(w'n) \neq x_m$. Hence, by Definition 3.1.5, we have $2 \in U_2$.

Since p' is a cycle in $\mathrm{ppg}(G)$ such that the union of its labels contains 1 and 2, the CFTG G is self-embedding by Theorem 3.1.7. This contradicts the assumption on G. ∎

3.2.2 Uniqueness in Argument Positions

Throughout this section, *we let* $G = (N, \Delta, A_0, R)$ *be a non-self-embedding lnCFTG.*

Using the position graph and the related results, we now introduce the novel property *uniqueness in argument positions*. Note that this section only applies to linear nondeleting CFTG. We start by presenting an example and give a formal definition afterwards.

Example 3.2.7 Recall the rules of the lnCFTG G_7 from Figure 3.5(a). It can be seen that G_7 can generate arbitrarily large trees, involving both the first and the second argument position of A. At first sight, this seems difficult to rhyme with the fact that G_7 is non-self-embedding, which we can prove using Theorem 3.1.7. Upon closer inspection however, we see that generation of γ's and δ's in different argument positions is not synchronized. Using the first rule repeatedly, an unbounded number of γ's can be generated in the first argument position. Likewise, using the second rule repeatedly, an unbounded number of δ's can be generated in the second argument position. However, by switching from the generation in argument position i (with $i \in \{1, 2\}$) to the generation in the other argument position, the value of argument position i is reset to a constant tree (β if $i = 1$; α if $i = 2$). Hence, there is no *synchronized* generation of symbols in different argument positions of one nonterminal. ○

In the following, we transform a non-self-embedding lnCFTG in such a way that the unsynchronized generation below *one* nonterminal explained in Example 3.2.7 is distributed over *distinct* nonterminals. After applying the transformation, each nonterminal of the transformed grammar generates unbounded material in at most one argument position. Before we present our method of transformation, we first give a formal characterization of the desired property.

Definition 3.2.8 Let $P \in \mathrm{scc}(\mathrm{pg}(G))$ be generating. We call P *unique in argument positions* if $(A, j), (A, j') \in \mathrm{V}_P$ implies $j = j'$. We call G *unique in argument positions* if each generating $P \in \mathrm{scc}(\mathrm{pg}(G))$ is unique in argument positions. □

The transformation involves the notion of *relative age* of an argument position. For an occurrence of a nonterminal A with rank k, the relative ages of the argument positions are expressed by a permutation of $[k]$. For example, if $k = 2$, then the sequence $\langle 2, 1\rangle$, which is a permutation of [2], states that the first position has relative age 2 and the second one has relative age 1. The intuition is that the value in the second position was 'born' after the value in the first position.

Upon applying a rule, the variables determine how relative ages are transferred from the LHS to positions of the root nonterminal in the RHS, say an occurrence of B with rank ℓ. This is subject to the following two constraints. First, if a new value is 'born' in argument position j of B, which is when there are no variables in that position, it receives a unique relative age that is smaller than any position that does have variables; we assign 0 as a preliminary value. This is exemplified by 0 in $\langle 2, 0\rangle$ in the preliminary RHS in Figure 3.6(a) and 0 in $\langle 0, 2\rangle$ in Figure 3.6(c). Second, if an argument position of B contains several variables, we take the maximum of the relative ages of the corresponding LHS positions as preliminary value. This is exemplified by 2 in $\langle 2, 0\rangle$ in the preliminary RHS of Figure 3.6(a), where $2 = \max\{2, 1\}$, and similarly 2 in $\langle 0, 2\rangle$ in Figure 3.6(c). We then assign the relative ages represented as a permutation according to the following rules. The higher the preliminary value of an argument position is, the higher its final relative age will be. The argument positions with preliminary value 0 are assigned decreasing values from left to right. Thus, the newly born argument positions are the youngest while the argument positions that contain older subtrees are older. Hence, $\langle 0, 2\rangle$ is turned into $\langle 1, 2\rangle$. This transformation yields the RHS of the newly constructed rule, as depicted in Figure 3.6.

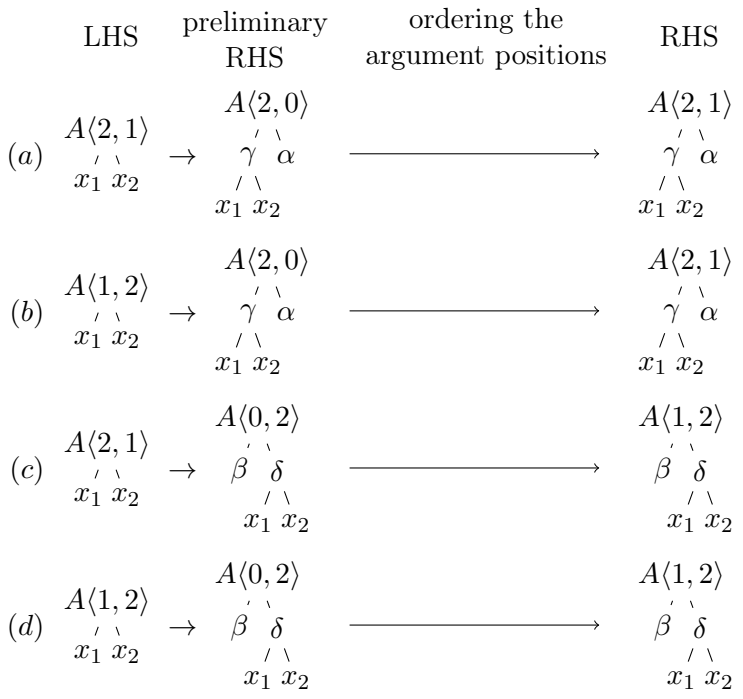


Figure 3.6: Rules from G_7 with annotated relative ages.

Lemma 3.2.9 For each non-self-embedding lnCFTG G, there is an equivalent lnCFTG G' that is non-self-embedding and unique in argument positions.

Proof. We assume that G is not unique in argument positions, and thus, there is a generating $P \in \mathrm{scc}(\mathrm{pg}(G))$ such that P is not unique in argument positions. We note that, for each $A \in N$, the vertex $(A, 0)$ is not in P (cf. Observation 3.2.3).

The following construction splits each nonterminal involved in P into new nonterminals of the form $A\langle\pi\rangle$ where π is a permutation of the argument positions of A. Each number in π represents the relative age of the corresponding argument with respect to the other arguments. The lower the number of an argument, the more recently its corresponding value was introduced, as explained above Lemma 3.2.9.

Formally, we let $r\colon A(\overline{x}) \to B(\zeta_{1..\ell})$ be a rule in $\mathrm{rules}(P)$ for some $A, B \in M_P$ (cf. Lemma 3.2.6). Furthermore, we let $k = \mathrm{rk}_N(A)$ and π be a permutation of $[k]$. These ingredients determine a permutation π_r of the argument positions of B. For this, we define the auxiliary mapping $\rho\colon [\ell] \to \mathbb{N}$ as follows. For each $j \in [\ell]$, let U_j denote the set of all $i \in [k]$ such that x_i occurs in ζ_j and let $\rho(j) = \max(\{\pi(i) \mid i \in U_j\})$ where $\max(\emptyset) = 0$. Then, we define the permutation π_r of $[\ell]$ as the unique permutation such that $\pi_r(j) < \pi_r(j')$ if

(i) $\rho(j) < \rho(j')$, or

(ii) $\rho(j) = \rho(j')$ and $j > j'$.

Note that, in case (ii), we have that $\rho(j) = \rho(j') = 0$ because of linearity.

We construct the lnCFTG $G' = (N', \Delta, A_0, R')$ where

- $N' = (N \setminus M_P) \cup \tilde{N}$ and $\tilde{N} = \{A\langle\pi\rangle^{(k)} \mid A \in M_P^{(k)}, \pi \text{ is a permutation of } [k]\}$,
- $R' = \mathrm{enr}((R \setminus R|_{M_P}) \cup \tilde{R}_1 \cup \tilde{R}_2)$, and $\tilde{R}_1$, $\tilde{R}_2$, and enr are defined as follows. For each rule $r\colon A(x_{1..k}) \to B(\zeta_{1..\ell})$ in $\mathrm{rules}(P)$ and for each permutation π of $[k]$ and π_r as constructed above, we let $A\langle\pi\rangle(x_{1..k}) \to B\langle\pi_r\rangle(\zeta_{1..\ell})$ be in $\tilde{R}_1$. Furthermore, for each rule $r \in (R|_{M_P} \setminus \mathrm{rules}(P))$ with $\mathrm{lhn}(r) = A^{(k)}$, let $A\langle\pi\rangle(x_{1..k}) \to \mathrm{rhs}(r)$ be in $\tilde{R}_2$ for each permutation π.

  The set of rules $(R \setminus R|_{M_P}) \cup \tilde{R}_1 \cup \tilde{R}_2$ is 'enriched' by the function enr, which replaces each occurrence of a nonterminal $A \in M_P^{(k)}$ in the RHS of each rule by $A\langle\tilde{\pi}\rangle$ where $\tilde{\pi}$ is the reversal of $[k]$, i.e., for each $i \in [k]$ we have $\pi(i) = k - i + 1$.

Due to the use of the maximum in the definition of the permutations, we have that if there is a path from $(B\langle\pi\rangle^{(k)}, i)$ to $(B'\langle\pi'\rangle^{(\ell)}, i')$ in $\mathrm{pg}(G')$, then $k - \pi(i) \geq \ell - \pi'(i')$ holds.

Claim 1: G and G' are equivalent. Moreover, G' is non-self-embedding.
Proof of Claim 1: For each derivation of G, there is precisely one way to add permutations to turn it into a derivation of G', since for each rule, the permutations at the LHS-nonterminal uniquely determine the permutations at the root of the RHS, an initial occurrence of a nonterminal from M_P is annotated by the fixed permutation $\tilde{\pi}$, and all permutations are incorporated, i.e., no annotation blocks the derivation process. Thus, G and G' are equivalent.

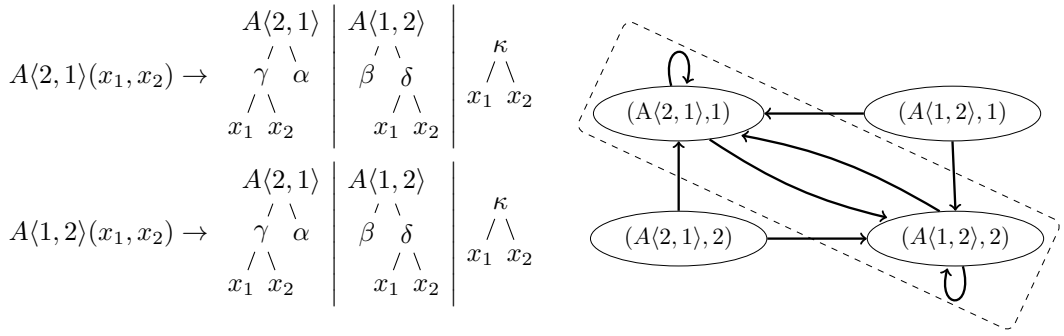
Furthermore, since each derivation in G' can be mapped onto a derivation in G, the lnCFTG G' is non-self-embedding. □

Claim 2: $G'|_{\tilde{N}}$ is unique in argument positions.
Proof of Claim 2: We already stated that if there is a path from $(B\langle\pi\rangle^{(k)}, i)$ to $(B\langle\pi'\rangle^{(\ell)}, i')$, then we have that $k - \pi(i) \geq \ell - \pi'(i')$. Recall that this property holds due to the use of the maximum in the construction of π'.

Consider any generating $P' \in \mathrm{scc}(\mathrm{pg}(G'|_{\tilde{N}}))$, $A^{(k)} \in M_P$, a permutation π of $[k]$, and $i, j \in [k]$ such that $(A\langle\pi\rangle, i) \in \mathrm{V}_{P'}$ and $(A\langle\pi\rangle, j) \in \mathrm{V}_{P'}$. Since $(A\langle\pi\rangle, i)$ and $(A\langle\pi\rangle, j)$ are in the same SCC P' we can deduce that (i) there is a path from $(A\langle\pi\rangle, i)$ to $(A\langle\pi\rangle, j)$ and (ii) there is a path in the other direction. We can thus conclude that $k - \pi(i) \geq k - \pi(j)$ and $k - \pi(j) \geq k - \pi(i)$. Since π is a permutation, we have $i = j$. Therefore, $G'|_{\tilde{N}}$ is unique in argument positions. □

The above process can be repeated until the resulting grammar is unique in argument positions. Termination is guaranteed, because in the transformation we only introduce SCCs which are unique in argument positions and thus, the total number of SCCs which are not unique in argument positions decreases in each step. ■

Example 3.2.10 As an example, we apply the construction of Lemma 3.2.9 to the rules of G_7 in Figure 3.5(a). We obtain the lnCFTG depicted in Figure 3.7(a). Figure 3.7(b) shows the relevant part of the corresponding position graph where each edge is generating (edge labels were omitted). The non-trivial SCC of Figure 3.7(b) is marked by a dashed box. It can be seen that each generating SCC contains, for each involved nonterminal, a unique argument position. ○



(a) Rules. (b) Part of the position graph.

Figure 3.7: The lnCFTG obtained by applying the construction of Lemma 3.2.9 to G_7.

3.3 Proving Regularity of Non-Self-Embedding lnCFTGs

In this section, we outline how to obtain a RTG H that is equivalent to a given non-self-embedding lnCFTG G. For this, we investigate two different ways in which a lnCFTG can repeatedly produce symbols. As discussed at the beginning of Chapter 3, symbols may be either produced above repeated nonterminals or below. To show regularity of G, we first transform all generation below nonterminals into generation above nonterminals (cf. Section 3.3.1) and afterwards, we can apply a saturation approach to obtain H (cf. Section 3.3.2). Our main result can be found in Section 3.3.3.

Throughout this section, *we let $G = (N, \Delta, A_0, R)$ be a non-self-embedding lnCFTG which is unique in argument positions.*

Consider the position graph of G. We classify a generating SCC $P \in \mathrm{scc}(\mathrm{pg}(G))$ according to whether nonterminals involved in P generate unbounded material below them, or above them. Formally, for each SCC $P \in \mathrm{scc}(\mathrm{pg}(G))$ we say that P is

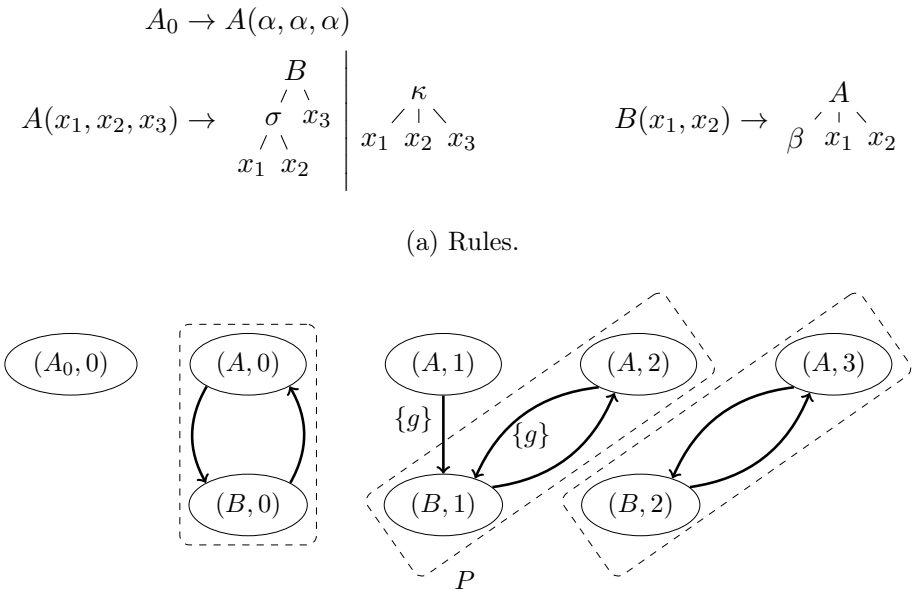
- *bottom-recursive* if P is generating and contains $(A, 0)$ for some $A \in N$,
- *top-recursive* if P is generating and does not contain $(A, 0)$ for each $A \in N$.

Example 3.3.1 As a running example, we consider the non-self-embedding lnCFTG

$$G_8 = (\{{A_0}^{(0)}, A^{(3)}, B^{(2)}\}, \{\alpha^{(0)}, \beta^{(0)}, \sigma^{(2)}, \kappa^{(2)}\}, A_0, R)$$

where R contains the rules depicted in Figure 3.8(a).

Figure 3.8(b) depicts the position graph of G_8, which shows that G_8 is unique in argument positions. We note that $\mathrm{pg}(G_8)$ contains five SCCs, from which three are non-trivial. The non-trivial SCCs are marked by dashed boxes. Furthermore, the SCC P is top-recursive, since it is generating and does not contain $(A, 0)$, $(B, 0)$, or $(A_0, 0)$. All other SCCs are not generating and thus neither top-recursive nor bottom-recursive. ○



(a) Rules.

(b) Position graph.

Figure 3.8: The lnCFTG G_8.

3.3.1 Transforming a Top-Recursive SCC into Bottom-Recursive SCCs

We present a construction which transforms a top-recursive SCC into at least one bottom-recursive SCC and a number (possibly 0) of non-generating SCCs. We repeat this process, until no more top-recursive SCCs remain in the position graph of the grammar. To be able to reason about termination of the process, we count the number of vertices in top-recursive SCCs.

Definition 3.3.2 The *top-recursion rank* of G, denoted by $\mathrm{topRank}(G)$, is the number of vertices in top-recursive SCCs in $\mathrm{pg}(G)$, or formally

$$\mathrm{topRank}(G) = \sum_{\substack{P \in \mathrm{scc}(\mathrm{pg}(G)) \\ P \text{ is top-recursive}}} |\mathrm{V}_P| \ . \qquad \square$$

Example 3.3.3 Recall the running example, i.e., the lnCFTG G_8 and its position graph (cf. Figure 3.8(b)). We have $\mathrm{topRank}(G_8) = 2$, since $(A, 2)$ and $(B, 1)$ are in the only top-recursive SCC P. $\bigcirc$

We recall Lemma 3.2.6 stating that, for each top-recursive SCC P, each rule $r \in \mathrm{rules}(P)$ has the form $A(\overline{x}) \to B(\xi_{1..\ell})$ where $A, B \in M_P$. The following two observations will be needed later.

Observation 3.3.4 We let $r \in R$ be of the form $A(x_{1..k}) \to B(\xi_{1..\ell})$. Then, for each $i \in [k]$, there is a unique $j_i \in [\ell]$ such that x_i occurs in ξ_{j_i}. Thus, there is an edge $(A, i) \to (B, j_i)$ in $\mathrm{edg}(r)$.

Given an outside-in derivation d consisting exclusively of rules from a top-recursive SCC P and given a vertex $(A, i) \in \mathrm{V}_{\mathrm{pg}(G)}$ where $A \in M_P$ and $A(\overline{x}) \Rightarrow_d \xi$ for some $\xi \in \mathrm{T}_{N \cup \Delta}(X_{\mathrm{rk}_N(A)})$, there are uniquely determined $B \in M_P$, $j \in [\mathrm{rk}_N(B)]$, and $p\colon (A, i) \to^* (B, j)$ such that each edge along the path p is determined by d according to Observation 3.3.4. In this case, we say that d *top-induces* the path p.

Observation 3.3.5 Let $r \in R$ be of the form $A(\overline{x}) \to B(\xi_{1..\ell})$. If there are $i, j \in [\mathrm{rk}_N(A)]$ with $i \neq j$ and $k \in [\ell]$ such that x_i and x_j both occur in ξ_k, then $\mathrm{edg}(r)$ contains the edges $((A, i), \{g\}, (B, k))$ and $((A, j), \{g\}, (B, k))$.

The proof of the following lemma incorporates two constructions and is rather lengthy. A full example can be found after the proof and may be consulted alongside.

Lemma 3.3.6 Let G be a non-self-embedding lnCFTG which is unique in argument positions and $\mathrm{topRank}(G) \geq 1$. Then we can construct a non-self-embedding lnCFTG G' which is unique in argument positions such that $\mathcal{L}(G') = \mathcal{L}(G)$ and $\mathrm{topRank}(G') < \mathrm{topRank}(G)$.

Proof. Since $\mathrm{topRank}(G) \geq 1$, there is a $P \in \mathrm{scc}(\mathrm{pg}(G))$ which is top-recursive and not reachable from any other top-recursive SCC. Let P now be fixed. For each $B \in M_P$, we denote the unique index $j \in [\mathrm{rk}_N(B)]$ such that $(B, j) \in \mathrm{V}_P$ by j_B.

We will construct a set K of items of the form $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle$. Each item represents the fact that there is a ξ_{j_B} such that $A(\overline{x}) \Rightarrow^* B(\xi_{1..\ell})$. Intuitively, an item in K captures a context in which trees ξ_{j_B} can be generated. We use $\mathbf{d}$ as a placeholder for the *dynamic position*. We will show that K is finite and use the elements of K as nonterminals for new rules that will replace the rules from $\mathrm{rules}(P)$ and thereby decrease the top-recursion rank.

Formally, we define K through a family $(K_i \mid i \in \mathbb{N})$ as follows.

- $K_0 = \{\langle A, A, x_{1..(j_A-1)}, \mathbf{d}, x_{(j_A+1)..k}\rangle \mid k \in \mathbb{N}, A \in M_P^{(k)}\}$.
- We let $i \in \mathbb{N}$. Then K_{i+1} is the smallest set K' satisfying the following condition. If there are $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K_i$, $r \in \mathrm{rules}(P)|_B$, $m \in \mathbb{N}$, $C \in M_P^{(m)}$ and $\xi'_{1..m} \in \mathrm{T}_{N\cup\Delta}(X_{\mathrm{rk}_N(A)})$ such that $B(\xi_{1..(j_B-1)}, x_{j_B}, \xi_{(j_B+1)..\ell}) \Rightarrow_r C(\xi'_{1..m})$ is an outside-in derivation, then $\langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle$ is in K'.
- $K = \bigcup_{i\in\mathbb{N}} K_i$.

Claim 1: Let $n \in \mathbb{N}$. Furthermore, let $A \in N^{(k)}$ and $B \in N^{(\ell)}$ with $k, \ell \in \mathbb{N}_+$, and $\xi_{1..(j_B-1)}, \xi_{(j_B+1)..\ell} \in \mathrm{T}_{N\cup\Delta}(X_k)$. The following are equivalent.

(i) There are a $\xi_{j_B} \in \mathrm{T}_{N\cup\Delta}(X_k)$ and an outside-in derivation d such that $|d| = n$ and $A(\overline{x}) \Rightarrow_d B(\xi_{1..\ell})$.

(ii) $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K_n$.

Proof of Claim 1: (i) ⇒ (ii): We can show Claim 1 by well-founded induction on n.

Clearly, for each $A \in M_P$, we have that $A(\overline{x})$ derives to $A(\overline{x})$ within zero rule application steps and, by definition, $\langle A, A, x_{1..(j_A-1)}, \mathbf{d}, x_{(j_A+1)..k}\rangle \in K_0$.

Now assume that Claim 1 holds for derivations of length n for some $n \in \mathbb{N}$. Assume a derivation d of length n and a rule $r \in R$ such that $A(\overline{x}) \Rightarrow_d B(\xi_{1..\ell}) \Rightarrow_r C(\xi'_{1..m})$ is an outside-in derivation. By the induction hypothesis, we have $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K_n$ and thus, by definition of K_{n+1}, it follows that $\langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle \in K_{n+1}$.

(ii) ⇒ (i): For each $A \in M_P$ we have $\langle A, A, x_{1..(j_A-1)}, \mathbf{d}, x_{(j_A+1)..k}\rangle \in K_0$ and it holds that $A(\overline{x})$ derives to $A(x)$ within zero rule application steps. Now let $n \in \mathbb{N}$ and assume that Claim 1 holds for each element in K_n. Furthermore, assume $\langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle \in K_{n+1}$. By definition of K_{n+1}, there are

- $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K_n$,
- a rule $r\colon B(\overline{x}) \to C(\zeta_{1..m})$, and
- some $\xi'_{j_C} \in \mathrm{T}_{N\cup\Delta}(X_k)$

such that $B(\xi_{1..(j_B-1)}, x_{j_B}, \xi_{(j_B+1)..\ell}) \Rightarrow_r C(\xi'_{1..m})$ is an outside-in derivation. By the induction hypothesis, there are $\xi_{j_B} \in \mathrm{T}_{N\cup\Delta}(X_k)$ and a derivation $A(\overline{x}) \Rightarrow_d B(\xi_{1..\ell})$ of length n. Then, we extend d with r and obtain

$$A(x) \Rightarrow_d B(\xi_{1..\ell}) \Rightarrow_r C(\xi'_{1..(j_C-1)}, \xi'_{j_C}[x_{j_B}/\xi_{j_B}], \xi'_{(j_C+1)..m}) \ .$$

□

Claim 2: The set K is finite.
Proof of Claim 2: In this proof let $n_1 = \max\{|\mathrm{pos}_{\Delta\cup(N\setminus M_P)}(\mathrm{rhs}(r))| \mid r \in \mathrm{rules}(P)\}$, $n_2 = |M_P|$, and $n_3 = \max\{\mathrm{rk}_N(C) \mid C \in M_P\}$. We prove Claim 2 by contradiction.

Assume that K is an infinite set. Then there are $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$, $i, m \in [\ell] \setminus \{j_B\}$, and rule sequences d, d_1, and d_2 of rules in $\mathrm{rules}(P)$ such that

(1) $|\mathrm{pos}(\xi_i)| > n_1 \cdot n_2 \cdot n_3$,
(since Δ and N are finite sets, there is such a tree ξ_i)

(2) d is an outside-in derivation and there is a ξ_{j_B} with $A(\overline{x}) \Rightarrow_d B(\xi_{1..\ell})$,
(cf. Claim 1)

(3) d_1 is a subsequence of d such that d_1 top-induces a generating cycle $p_1\colon (B, m) \to^* (B, m)$ in a SCC of $\mathrm{pg}(G)$ different from P
(since ξ_i is sufficiently large, there must be a generating cycle)

(4) d_2 is an outside-in derivation and it top-induces a generating cycle $p_2\colon (B, j_B) \to^* (B, j_B)$ in P.
(since P is a top-recursive SCC, there must be such a generating cycle)

We show the following statement by induction.

Statement (†): For each $n \in \mathbb{N}$, there is an $m_n \in [\mathrm{rk}_N(B)]$ such that

(i) for each $i \in [n]$, the sequence $d_1 d_2$ top-induces a path $(B, m_{i-1}) \to^* (B, m_i)$, and

(ii) $m_n \notin \{j_B, m_0, \ldots, m_{n-1}\}$.

For the induction base ($n = 0$), we let $m_0 = m$ and recall that $m \neq j_B$. For the induction step, we assume that (†) holds for $n \in \mathbb{N}$. A consequence of Observation 3.3.4 is that there is a unique m' such that d_1d_2 top-induces the path $(B, m_n) \to^* (B, m')$. We let $m_{n+1} = m'$. Then, (i) holds for m_{n+1}. Now, we show that m_{n+1} satisfies (ii). If $m_{n+1} = j_B$, then $(B, m) \to^* (B, j_B)$, but because (B, m) is in a generating SCC, this would contradict the assumption that P is not reachable from any other generating SCC. It remains to prove $m_{n+1} \notin \{m_0, \ldots, m_n\}$.

Assume that $m_{n+1} = m_j$ for some $j \in \{0, 1, \ldots, n\}$. The sequence $(d_1d_2)^{n-j+1}$ top-induces

$$p\colon (B, m_j) \to_{(d_1d_2)^{n-j}} (B, m_n) \to_{d_1d_2} (B, m_j)$$
$$p'\colon (B, j_B) \to_{(d_1d_2)^{n-j}} (B, j_B) \to_{d_1d_2} (B, j_B) \ .$$

We show that p and p' are generating. If $j = 0$, then the cycle p is generating, because it contains p_1. If $j \neq 0$, then d_1d_2 top-induces $p''\colon (B, m_n) \to_{d_1d_2} (B, m_j)$ and $p'''\colon (B, m_{j-1}) \to_{d_1d_2} (B, m_j)$, and, by Observation 3.3.5, p'' is generating and therefore p is generating. Thus, p is generating regardless of the choice of j. The path p' is generating, because it contains p_2.

We will now combine p and p' into one cycle in $\mathrm{ppg}(G)$. For this, we consider each step simultaneously in both paths. We let $k \in [|d_1d_2| \cdot (n - j + 1)]$ and consider the k-th step. We let $((B_1, i_1), U_1, (B_2, i_2))$ be the k-th edge in p and $((B_1, j_1), U_2, (B_2, j_2))$ be the k-th edge in p'. Both edges are top-induced by the same rule r. Hence, we have that x_{i_1} and x_{j_1} occur in the subtrees $\mathrm{rhs}(r)|_{i_2}$ and $\mathrm{rhs}(r)|_{j_2}$, respectively. By Observation 3.3.4 and since $m_j \neq j_B$, we have that $i_1 \neq j_1$ and $i_2 \neq j_2$. By Definition 3.1.5, there is an edge $((B_1, i_1, j_1), U', (B_2, i_2, j_2))$ in $\mathrm{ppg}(G)$. Furthermore, we have that $1 \in U'$ if $U_1 = \{g\}$ and $2 \in U'$ if $U_2 = \{g\}$.

Hence, from p and p', we obtain the following path $\tilde{p}$ in $\mathrm{ppg}(G)$:

$$\tilde{p}\colon (B, m_j, j_B) \to_{(d_1d_2)^{n-j}} (B, m_n, j_B) \to_{d_1d_2} (B, m_j, j_B) \ .$$

Since p and p' are both generating, $\tilde{p}$ is a cycle such that the union of all its path labels contains 1 and 2. By Theorem 3.1.7, this contradicts G being non-self-embedding and thus, (ii) holds for m_{n+1}. This proves (†).

However, (†) conflicts with the finiteness of $\mathrm{rk}_N(B)$ and thus, K is a finite set. □

We modify G with the help of K to construct a lnCFTG G'. We let G' contain all original rules, except the ones of $\mathrm{rules}(P)$. We further transform the rules from $\mathrm{rules}(P)$ into bottom-recursive rules and add the transformed rules to G'. This transformation is achieved by reversing the rules, i.e., if G applies r_1 and afterwards r_2 ($r_1, r_2 \in \mathrm{rules}(P)$), then G' applies first r_2 and then r_1.

Reversing a rule is achieved by considering the rule in the context of a derivation. This context is represented by using the elements from K as nonterminals for G'.

As an example, consider the rule $r\colon A(x_1, x_2, x_3) \to B(\sigma(x_1, x_2), x_3)$ from G_8 (cf. Example 3.3.1). We have $j_A = 2$ and $j_B = 1$ and we consider the context $k_1 = \langle A, A, \beta, \mathbf{d}, x_3\rangle$ in K. If we consider r in the context of k_1 we obtain

$$A(\beta, x_2, x_3) \to B(\sigma(\beta, x_2), x_3) \ .$$

In the RHS, we obtain the context $k_2 = \langle A, B, \mathbf{d}, x_3\rangle$. We reverse the rule and construct a new rule with LHS $k_2(x_1, x_2)$ where x_1 and x_2 are those variables of $X_{\mathrm{rk}_N(A)}$ not present in k_2. The RHS of the new rule is obtained from the subtree of $\mathrm{rhs}(r)$ at position j_B as follows. We replace x_{j_A} by $k_1(x_1, x_2)$ where again, x_1 and x_2 are the variables of $X_{\mathrm{rk}_N(A)}$ not present in k_1. Hence, r is turned into the rule

$$\langle A, B, \mathbf{d}, x_3\rangle(x_1, x_2) \to \sigma(\beta, \langle A, A, \beta, \mathbf{d}, x_3\rangle(x_1, x_2), x_3) \ .$$

Furthermore, we add some rules which handle the connection to rules outside of $\mathrm{rules}(P)$. For each $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$ and each rule $r\colon B(\overline{x}) \to \zeta$ in $R|_{M_P} \setminus \mathrm{rules}(P)$, we create the rule

$$A(\overline{x}) \to \zeta[\xi_{1..(j_B-1)}, \langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}), \xi_{(j_B+1)..\ell}] \ .$$

Intuitively, the nonterminal from K generates all symbols that would have been generated by an iteration of rules in $\mathrm{rules}(P)$ below the dynamic position of B. By the substitution into ζ, we ensure that the result of the recursion is placed outside of the nonterminal and argument position participating in P.

Formally, we construct $G' = (N', \Delta, A_0, R')$ as follows. We let $N' = N \cup K$ where, for each $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$, we define its rank to be the number of variables from $X_{\mathrm{rk}_N(A)}$ not present in $\xi_{1..(j_B-1)}, \xi_{(j_B+1)..\ell}$. We define R' using the following rules.

(1) $R \setminus R|_{M_P} \subseteq R'$;

(2) for each $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$ and each $B(\overline{x}) \to \zeta$ in $R|_{M_P} \setminus \mathrm{rules}(P)$, we let

$$A(\overline{x}) \to \zeta[\xi_{1..(j_B-1)}, \langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}), \xi_{(j_B+1)..\ell}]$$

be in R' where $x_{\ell_1}, \ldots, x_{\ell_q}$ are those variables of $X_{\mathrm{rk}_N(A)}$ that do not occur in $\xi_{1..(j_B-1)}, \xi_{(j_B+1)..\ell}$ in ascending order;

(3) for each $A \in M_P$, we let

$$\langle A, A, x_{1..(j_A-1)}, \mathbf{d}, x_{(j_A+1)..\mathrm{rk}_N(A)}\rangle(x_{j_A}) \to x_{j_A}$$

be a rule in R';

(4) for each $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$ and $r \in \mathrm{rules}(P)|_B$ of the form $B(\overline{x}) \to C(\zeta_{1..m})$ such that $B(\xi_{1..(j_B-1)}, x_{j_B}, \xi_{(j_B+1)..\ell}) \Rightarrow_r C(\xi'_{1..m})$ is an outside-in derivation, we let

$$\begin{aligned}&\langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle(x_{m_1}, \ldots, x_{m_{q'}})\\ &\quad\to \zeta_{j_C}[\xi_{1..(j_B-1)}, \langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}), \xi_{(j_B+1)..\ell}]\end{aligned}$$

be in R' where $x_{m_1}, \ldots, x_{m_{q'}}$ and $x_{\ell_1}, \ldots, x_{\ell_q}$ are those variables of $X_{\mathrm{rk}_N(A)}$ which are not present in $\xi'_{1..(j_C-1)}, \xi_{(j_C+1)..m}$ and $\xi_{1..(j_B-1)}, \xi_{(j_B+1)..\ell}$, respectively, in ascending order of their indices;

(5) no other rules are in R'.

By inspecting all newly introduced rules, it can be verified that G' is non-self-embedding.

Claim 3: Let $A, B \in M_P$, $\ell = \mathrm{rk}_N(B)$, and $\xi_{1..\ell} \in \mathrm{T}_{N\cup\Delta}(X_{\mathrm{rk}_N(A)})$. Then the following are equivalent.

(i) There is an outside-in derivation d consisting exclusively of rules in rules(P) such that $A(\overline{x}) \Rightarrow_d B(\xi_{1..\ell})$.

(ii) There is a derivation d' of rules in G' created due to (4) such that

$$\begin{aligned}&\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q})\\ &\quad\Rightarrow_{d'} \xi_{j_B}[x_{j_A}/\langle A, A, x_{1..(j_A-1)}, \mathbf{d}, x_{(j_A+1)..\mathrm{rk}_N(A)}\rangle(x_{j_A})]\end{aligned}$$

where $x_{\ell_1}, \ldots, x_{\ell_q}$ are those variables of $X_{\mathrm{rk}_N(A)}$ that do not occur in the sequence of trees $\xi_{1..(j_B-1)}, \xi_{(j_B+1)..\ell}$.

Proof of Claim 3: We abbreviate $x_{1..(j_A-1)}, \mathbf{d}, x_{(j_A+1)..\mathrm{rk}_N(A)}$ by $\overline{x_{\mathbf{d}}}$.

(i)⇒(ii): We prove Claim 3 by induction on the length of d. For $|d| = 0$, we trivially get $|d'| = 0$. Now we assume that $d = d_1 r$ for some sequence d_1 of rules from rules(P) and $r\colon B(\overline{x}) \to C(\zeta_{1..m})$ in rules(P)$|_B$ (cf. Lemma 3.2.6) such that $A(\overline{x}) \Rightarrow_{d_1} B(\xi_{1..\ell}) \Rightarrow_r C(\xi'_{1..m})$. Note that $\xi'_{j_C} = \zeta_{j_C}[\xi_{1..\ell}]$. By the induction hypothesis, there is a derivation d'_1 such that

$$\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}) \Rightarrow_{d'_1} \xi_{j_B}[x_{j_A}/\langle A, A, \overline{x_{\mathbf{d}}}\rangle(x_{j_A})] \ .$$

Furthermore, by Claim 1, we have $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$. Then, by (4), there is a rule

$$\begin{aligned}r'\colon \langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle(x_{m_1}, \ldots, x_{m_{q'}})\\ \to \zeta_{j_C}[\xi_{1..(j_B-1)}, \langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}), \xi_{(j_B+1)..\ell}] \ .\end{aligned}$$

It can be seen that

$$\begin{aligned}&\langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle(x_{m_1}, \ldots, x_{m_{q'}})\\ &\quad\Rightarrow_{r'd'_1} \zeta_{j_C}[\xi_{1..(j_B-1)}, \xi_{j_B}[x_{j_A}/\langle A, A, \overline{x_{\mathbf{d}}}\rangle(x_{j_A})], \xi_{(j_B+1)..\ell}]\\ &\quad= \qquad \zeta_{j_C}[\xi_{1..\ell}][x_{j_A}/\langle A, A, \overline{x_{\mathbf{d}}}\rangle(x_{j_A})] \ .\end{aligned}$$

Hence, $d' = r'd'_1$ is the desired derivation of rules created due to (4).

(ii)⇒(i): We prove this by induction on the length of d'. For the base case $|d'| = 0$, we trivially get $|d| = 0$. For $|d'| \geq 1$, we let $d' = r'd'_1$ and

$$\langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle(x_{m_1}, \ldots, x_{m_{q'}}) \Rightarrow_{r'd'_1} \xi'_{j_C}[x_{j_A}/\langle A, A, \overline{x_{\mathbf{d}}}\rangle(x_{j_A})]$$

where r' is a rule due to (4) of the form

$$\begin{aligned}r'\colon \langle A, C, \xi'_{1..(j_C-1)}, \mathbf{d}, \xi'_{(j_C+1)..m}\rangle(x_{m_1}, \ldots, x_{m_{q'}})\\ \to \zeta[z/\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q})]\end{aligned}$$

for some $\zeta \in \mathrm{C}_{N\cup\Delta\cup X}(\{z\})$, $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$, and $r\colon B(\overline{x}) \to C(\zeta_{1..m})$ in R such that $B(\xi_{1..(j_B-1)}, x_{j_B}, \xi_{(j_B+1)..\ell}) \Rightarrow_r C(\xi'_{1..(j_C-1)}, \xi', \xi'_{(j_C+1)..m})$ is an outside-in derivation. We note that $\xi' = \zeta[z/x_{j_B}]$ and $\xi' = \zeta_{j_C}[\xi_{1..(j_B-1)}, x_{j_B}, \xi_{(j_B+1)..\ell}]$.

Furthermore, there is some $\xi_{j_B} \in \mathrm{T}_{N\cup\Delta}(X_{\mathrm{rk}_N(A)})$ such that

$$\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}) \Rightarrow_{d'_1} \xi_{j_B}[x_{j_A}/\langle A, A, \overline{x_{\mathbf{d}}}\rangle(x_{j_A})] \ .$$

By the induction hypothesis, there is an outside-in derivation d_1 such that $A(\overline{x}) \Rightarrow_{d_1} B(\xi_{1..\ell})$. Consider the outside-in derivation $A(\overline{x}) \Rightarrow_{d_1} B(\xi_{1..\ell}) \Rightarrow_r C(\xi''_{1..m})$. It can be seen that, for each $i \in [m] \setminus \{j_C\}$, we have $\xi''_i = \xi'_i$. We furthermore have $\xi''_{j_C} = \zeta_{j_C}[\xi_{1..\ell}] = \zeta[z/\xi_{j_B}] = \xi'_{j_C}$. Hence, $d_1 r$ is the desired outside-in derivation. □

Claim 4: Let $A, B \in M_P$, $\ell = \mathrm{rk}_N(B)$, and $\xi_{1..\ell} \in \mathrm{T}_{N\cup\Delta}(X_{\mathrm{rk}_N(A)})$. Then the following are equivalent.

(i) There is an outside-in derivation d consisting exclusively of rules in rules(P) such that $A(\overline{x}) \Rightarrow_d B(\xi_{1..\ell})$.

(ii) There is a derivation d' of rules in G' created due to (3) and (4) such that

$$\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}) \Rightarrow_{q'} \xi_{j_B}$$

where $x_{\ell_1}, \ldots, x_{\ell_q}$ are those variables of $X_{\mathrm{rk}_N(A)}$ that do not occur in the sequence of trees $\xi_{1..(j_B-1)}, \xi_{(j_B+1)..\ell}$.

Proof of Claim 4: Claim 4 follows directly from Claim 3 by using exactly one rule due to (3) at the end of d'. □

Claim 5: $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof of Claim 5: $\mathcal{L}(G) \subseteq \mathcal{L}(G')$: We let $\widetilde{\xi} \in \mathrm{T}_\Delta$ and d be a derivation such that $A_0 \Rightarrow_d \widetilde{\xi}$ holds in G. If d contains no rules from $R|_{M_P}$, then $A_0 \Rightarrow_d \widetilde{\xi}$ holds in G' as well, since all rules are in R' due to (1). Now we assume that a rule from $R|_{M_P}$ occurs in d. Then we may reorder d such that rules of $R|_{M_P}$ are executed in sequence as follows. There are $A, B \in M_P$ and some $\zeta \in \mathrm{C}_\Delta(\{z\})$ such that

$$A_0 \Rightarrow_{d_0} \zeta[A(\xi_{1..k})] \Rightarrow_{d_1} \zeta[B(\xi'_{1..\ell})] \Rightarrow_r \zeta\big[\zeta'[\xi'_{1..\ell}]\big] \Rightarrow_{d_2} \widetilde{\xi}$$

where d_0 is a sequence of rules such that $A_0 \Rightarrow_{d_0} \zeta[A(\xi_{1..k})]$ holds in both G and G', d_1 is an outside-in derivation consisting of rules from rules(P), $r\colon B(\overline{x}) \to \zeta'$ is a rule in $R|_B \setminus$ rules(P), and d_2 is the remaining sequence of rules in R. Note that rules in d_0 can be due to (1) or they can be chosen recursively by the following argument.

We consider the outside-in derivation $A(\overline{x}) \Rightarrow_{d_1} B(\zeta_{1..\ell})$. We note that $\xi'_i = \zeta_i[\xi_{1..k}]$ for each $i \in [\ell]$, i.e., the derivation is independent from its context. By Claim 1, we have $\langle A, B, \zeta_{1..(j_B-1)}, \mathbf{d}, \zeta_{(j_B+1)..\ell}\rangle \in K$. Note that r has LHS-nonterminal B and thus, there is a rule due to (2) using $\langle A, B, \zeta_{1..(j_B-1)}, \mathbf{d}, \zeta_{(j_B+1)..\ell}\rangle$ and r.

By Claim 4, there is a derivation $\langle A, B, \zeta_{1..(j_B-1)}, \mathbf{d}, \zeta_{(j_B+1)..\ell}\rangle(\overline{x}) \Rightarrow_{d'_1} \zeta_{j_B}$. Hence, we replace $d_0 d_1 r$ by

$$\begin{aligned}
A_0 &\Rightarrow_{d_0} \zeta[A(\xi_{1..k})] = \zeta[A(x_{1..k})][\xi_{1..k}] \\
&\Rightarrow \zeta\big[\zeta'[\zeta_{1..(j_B-1)}, \langle A, B, \zeta_{1..(j_B-1)}, \mathbf{d}, \zeta_{(j_B+1)..\ell}\rangle(x_{\ell_1}, \ldots, x_{\ell_q}), \zeta_{(j_B+1)..\ell}]\big][\xi_{1..k}] && (2) \\
&\Rightarrow_{d'_1} \zeta\big[\zeta'[\zeta_{1..\ell}]\big][\xi_{1..k}] && \text{(Claim 4)} \\
&= \zeta\big[\zeta'[\xi'_{1..\ell}]\big] \ .
\end{aligned}$$

Note that the above derivation does not use any rules from rules(P). By applying the above method repeatedly, we can replace the remaining rules from rules(P) in d_2 and obtain a derivation in G'.

$\mathcal{L}(G) \supseteq \mathcal{L}(G')$: A derivation in G' consists either of rules also present in G or can be reordered to contain rule sequences described by the following regular expression

$$(1)^*\Big(\underbrace{(2)\ (4)^*\,(3)}_{\text{use Claim 4}}\ (1)^*\Big)^*$$

where (i) stands for a rule due to the item (i) in the construction of R'. Using Claim 4, it can be seen that we can replace each underbraced sequence by a sequence of rules in G with the same effect. □

Next we prove that the top-recursion rank of G' is smaller than the top-recursion rank of G and show that G' is unique in argument positions. For this, we observe two properties of $\mathrm{pg}(G')$, based on the construction of the rules of G'.

(P1) For each top-recursive $P' \in \mathrm{scc}(\mathrm{pg}(G'))$, we have $M_{P'} \cap K = \emptyset$.
(Intuition: A generating cycle $p\colon (A,i) \to^* (A,i)$ in $\mathrm{pg}(G')$ where $A \in N$ can be translated into a generating cycle in $\mathrm{pg}(G)$. Note that all vertices on p belong to one top-recursive SCC. If p contains an edge from a nonterminal $\langle A, B, \xi_{1..(j_B-1)}, \mathbf{d}, \xi_{(j_B+1)..\ell}\rangle \in K$ to a nonterminal $C \in N$, then symbols are generated *above* a corresponding nonterminal occurrence of A in G (cf. rules due to (4)). In this case, by Definition 3.1.1, the translated path is a witness for G being self-embedding. This is a contradiction to the assumption of this section and thus, each top-recursive SCC cannot contain vertices with nonterminals from K.)

(P2) We let $(A,i),(B,j) \in \mathrm{V}_{\mathrm{pg}(G)}$. If $(A,i) \not\to^* (B,j)$ in $\mathrm{pg}(G)$, then $(A,i) \not\to^* (B,j)$ in $\mathrm{pg}(G')$.

Claim 6: For each top-recursive SCC $P' \in \mathrm{scc}(\mathrm{pg}(G'))$ and each $(A,i) \in \mathrm{V}_{P'}$, there is a top-recursive SCC $\widetilde{P} \in \mathrm{scc}(G)$ such that $(A,i) \in \mathrm{V}_{\widetilde{P}}$.
Proof of Claim 6: Let $P' \in \mathrm{scc}(\mathrm{pg}(G'))$ be top-recursive and $(A,i) \in \mathrm{V}_{P'}$. By (P1), $A \in N$ and thus, there is a uniquely determined $\widetilde{P} \in \mathrm{scc}(\mathrm{pg}(G))$ such that $(A,i) \in \widetilde{P}$. We show that $\widetilde{P}$ is top-recursive.

There is a rule sequence d' of rules from R' such that d' induces a generating path $p'\colon (A,i) \to_{s'} (A,i)$ in P'. We transform p' into a path $p\colon (A,i) \to^* (A,i)$ in $\mathrm{pg}(G)$ with the following case distinction on rules in d'. Let r' be a rule in d'.

If $r' \in R$, then the induced edge is not changed. Trivially, if r' induces a generating edge in p', then it induces a generating edge in p. Now assume that $r' \in R' \setminus R$ and r' induces the edge $(B_1,i_1) \to_{r'} (B_2,i_2)$ in p where $B_1, B_2 \in N$, $i_1 \in [\mathrm{rk}_N(B_1)]$, and $i_2 \in [\mathrm{rk}_N(B_2)]$. In this case, r' is due to (2). By Claim 1 and 4, there is a sequence d of rules in R such that $p_1\colon (B_1,i_1) \to_d (B_2,i_2)$. Furthermore, it can be seen that d is generating if r' is. We replace $(B_1,i_1) \to_{r'} (B_2,i_2)$ by p_1.

We transform d' rule by rule as described above and obtain the path $p\colon (A,i) \to_p (A,i)$ in $\mathrm{pg}(G)$. We have that p is generating, because p' is. Thus, $\widetilde{P}$ is top-recursive. □

Claim 7: For each $(A,i) \in \mathrm{V}_P$ and $P' \in \mathrm{scc}(\mathrm{pg}(G'))$ such that $(A,i) \in \mathrm{V}_{P'}$, we have that P' is not top-recursive.
Proof of Claim 7: Let $(A,i) \in \mathrm{V}_P$ and $P' \in \mathrm{scc}(\mathrm{pg}(G'))$ such that $(A,i) \in \mathrm{V}_{P'}$. We assume that P' is top-recursive. Then there is a generating path $p' \colon (A,i) \to^* (A,i)$ in P'. Furthermore, there are $(B,j) \in \mathrm{V}_{\mathrm{pg}(G')}$ and $r' \in R'$ such that $(A,i) \to_{r'} (B,j)$ is the first edge of p'. By (P1), we have $B \in N$. Since $A \in M_P$, we have that r' is due to (2) and, by the construction of G', we have $(B,j) \notin \mathrm{V}_P$. Thus, we have $(B,j) \not\to^* (A,i)$ in $\mathrm{pg}(G)$ and thus, by (P2), $(B,j) \not\to^* (A,i)$ in $\mathrm{pg}(G')$. This contradicts the existence of p'. Since there is no generating path $(A,i) \to (A,i)$ in $\mathrm{pg}(G)$, we obtain a contradiction to P' being top-recursive. □

Claim 8: $\mathrm{topRank}(G') < \mathrm{topRank}(G)$.
Proof of Claim 8: This is a consequence of Claims 6 and 7. □

Claim 9: The lnCFTG G' is unique in argument positions.
Proof of Claim 9: We analyze newly introduced generating SCCs. Let $P' \in \mathrm{scc}(\mathrm{pg}(G'))$ be generating. If P' is bottom-recursive, then by Observation 3.2.3, P' is trivially unique in argument positions.

If P' is top-recursive, then we analyze the rules from $\mathrm{rules}(P')$. We note that, by (P1), there are no rules due to (3) or (4). Hence, we consider rules due to (1) and (2). We let $(A,i),(A,j) \in \mathrm{V}_{P'}$ and $p' \colon (A,i) \to^* (A,j) \to^* (A,i)$ be a generating path in $\mathrm{pg}(G')$. Then, we construct the path p in $\mathrm{pg}(G)$ by modifying p' as follows. Edges induced by rules due to (1) are taken over without modification. Each edge induced by a rule due to (2) is replaced by the path induced by the corresponding sequence of rules in $\mathrm{rules}(P)$ (cf. Claim 1) followed by the single rule outside of $\mathrm{rules}(P)$. Hence, $p \colon (A,i) \to^* (A,j) \to^* (A,i)$ is a path in $\mathrm{pg}(G)$. Since G is unique in argument positions, we have $i = j$ and thus, G' is unique in argument positions. □
■

We illustrate the constructions from the proof of Lemma 3.3.6 in the following example.

Example 3.3.7 Recall the lnCFTG G_8 from Example 3.3.1. It is clear that P (cf. Figure 3.8(b)) is the selected SCC since it is the only top-recursive SCC. We note that P is reachable from one other SCC, which is not top-recursive.

The set K contains the contexts of trees that can be generated in the generating argument positions using rules from $\mathrm{rules}(P)$. We note that $j_A = 2$ and $j_B = 1$. The set K contains the following items:

$$K = \{ \quad \langle A, A, x_1, \mathbf{d}, x_3 \rangle \qquad \langle A, B, \mathbf{d}, x_3 \rangle \qquad \langle A, A, \beta, \mathbf{d}, x_3 \rangle$$
$$\langle B, B, \mathbf{d}, x_2 \rangle \qquad \langle B, A, \beta, \mathbf{d}, x_2 \rangle \qquad \}$$

Figure 3.9 presents the rules of the transformed grammar $G_8{}'$ constructed by (3) and (4).

With the help of these rules we can illustrate Claim 3 and 4. Consider the two derivations from G_8 and $G_8{}'$ in Figure 3.10. For easier comparison we denote the derivation of G_8 from left to right and the derivation of $G_8{}'$ from right to left. The two occurrences of the terminal symbol σ are created in different stages of the derivations. We link the corresponding creations by dashed curves.

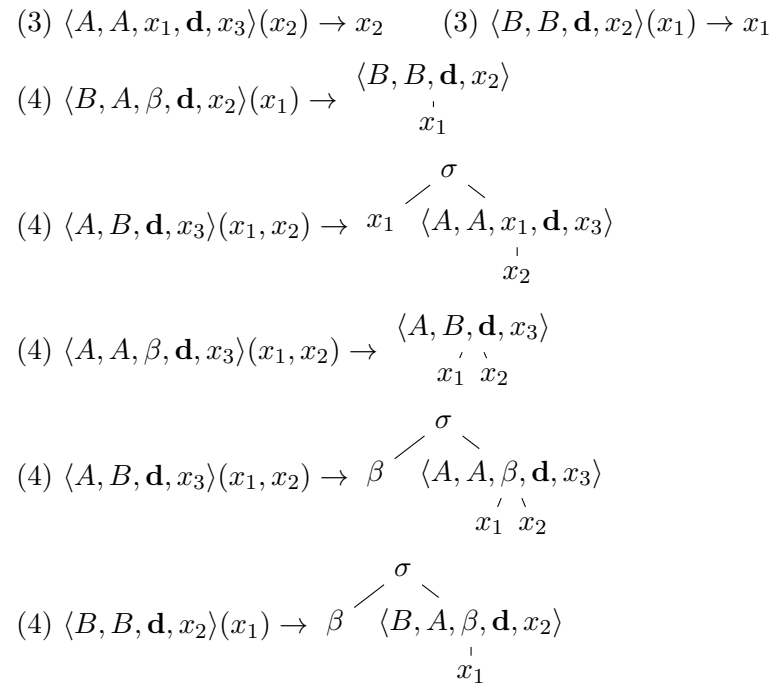


Figure 3.9: The rules of G_8' created due to (3) and (4).

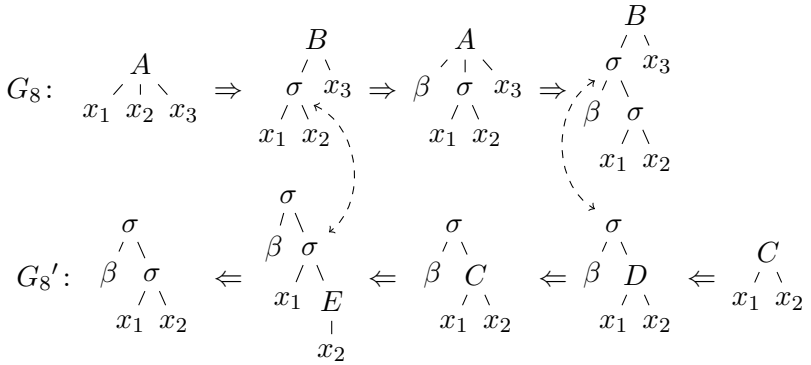


Figure 3.10: Derivations from G_8 and G_8' (Abbreviations: $C = \langle A, B, \mathbf{d}, x_3\rangle$, $D = \langle A, A, \beta, \mathbf{d}, x_3\rangle$, and $E = \langle A, A, x_1, \mathbf{d}, x_3\rangle$).

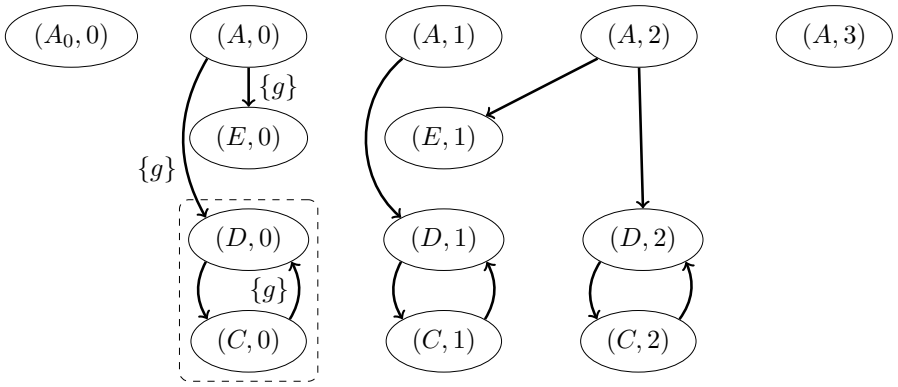


Figure 3.11: Part of the position graph of $G_8{}'$ (Abbreviations: $C = \langle A, B, \mathbf{d}, x_3\rangle$, $D = \langle A, A, \beta, \mathbf{d}, x_3\rangle$, and $E = \langle A, A, x_1, \mathbf{d}, x_3\rangle$).

We will now present the rules of $G_8{}'$ created due to (1) and (2).

(1) There is only one rule not in $R \setminus R|_{M_P}$: $A_0 \to A(\alpha, \alpha, \alpha)$.

(2) There is only one rule in $R|_{M_P} \setminus \text{rules}(P)$, viz. $r\colon A(x_1, x_2, x_3) \to \kappa(x_1, x_2, x_3)$. Hence, we create one rule for each item in K where the second component is A, viz. $\langle A, A, x_1, \mathbf{d}, x_3\rangle$, $\langle A, A, \beta, \mathbf{d}, x_3\rangle$, and $\langle B, A, \beta, \mathbf{d}, x_2\rangle$. The contextual information from each of these elements is incorporated into r.

$$A(x_1, x_2, x_3) \to x_1\ \kappa\big(\langle A, A, x_1, \mathbf{d}, x_3\rangle(x_2), x_3\big) \mid \beta\ \kappa\big(\langle A, A, \beta, \mathbf{d}, x_3\rangle(x_1, x_2), x_3\big)$$

$$B(x_1, x_2) \to \beta\ \kappa\big(\langle B, A, \beta, \mathbf{d}, x_2\rangle(x_1), x_2\big)$$

Figure 3.11 shows part of $\text{pg}(G_8{}')$. The SCC formed by nonterminals from K is bottom-recursive. We observe that the resulting lnCFTG $G_8{}'$ does not have any top-recursive SCC and thus, $\text{topRank}(G_8{}') = 0$. Claim 6 and 7 are trivial for this example and Claim 8 is illustrated. It can be seen that $\text{pg}(G_8{}')$ is unique in argument positions and this is an example of Claim 9. ○

Theorem 3.3.8 Let G be a non-self-embedding lnCFTG which is unique in argument positions. Then there is a non-self-embedding lnCFTG G' which is unique in argument positions, $\mathcal{L}(G') = \mathcal{L}(G)$, and $\text{pg}(G')$ does not contain top-recursive SCCs, i.e., $\text{topRank}(G') = 0$.

Proof. This theorem follows immediately from the repeated application of Lemma 3.3.6 to G. Since $\text{topRank}(G)$ is finite and decreases strictly with every application of the lemma, the construction terminates. ■

3.3.2 Transforming a Top-Recursion-Free lnCFTG into a RTG

In this section, *we consider a non-self-embedding lnCFTG G such that* $\mathrm{pg}(G)$ *does not contain a top-recursive SCC.* Hence, for each rule r in G, exactly one of the following three cases holds:

- $r \in \mathrm{rules}(P)$ for some bottom-recursive SCC P,
- $r \in \mathrm{rules}(P)$ for some SCC P which is not generating, and r is of the form $A(x_{1..k}) \to B(x_{j_1}, \ldots, x_{j_k})$ where $j_1, \ldots, j_k$ is a permutation of $[k]$,
- $r \in R \setminus (\bigcup_{P \in \mathrm{scc}(\mathrm{pg}(G))} \mathrm{rules}(P))$, i.e., r is not in the rules of any SCC and thus, r is not involved in any recursion.

We note that each non-self-embedding lnCFTG G such that $\mathrm{pg}(G)$ does not contain a top-recursive SCC, is unique in argument positions.

Lemma 3.3.9 We let G be a non-self-embedding lnCFTG and $\mathrm{topRank}(G) = 0$. We can construct a RTG H such that $\mathcal{L}(G) = \mathcal{L}(H)$.

Proof. Let $G = (N, \Delta, A_0, R)$. We will show that, since G contains no top-recursion, in any outside-in derivation, no unboundedly large trees will occur in any argument position of any nonterminal. In other words, the set

$$K = \{\langle \xi|_w \rangle \mid A_0 \Rightarrow_d \xi,\ w \in \mathrm{pos}_N(\xi),\ d \text{ is outside-in}\}$$

is finite.

We will construct K and use its elements as nonterminals of the RTG H. The trees in K are used to construct rules that mimic rules of G, but only use nullary nonterminals.

As an auxiliary tool we define, for each $\xi \in \mathrm{T}_{N \cup \Delta}(X)$,

$$\mathrm{cut}_N(\xi) = \{\langle \xi|_w \rangle \mid w \in \mathrm{pos}_N(\xi), w \text{ outermost in } \mathrm{pos}_N(\xi)\}\ .$$

We let $P_0 = \{\langle A_0 \rangle\}$ and define inductively, for each $i \in \mathbb{N}$,

$$P_{i+1} = P_i \cup \bigcup_{\substack{\langle \xi \rangle \in P_i \\ r \in R|_{\xi(\varepsilon)}}} \mathrm{cut}_N(\mathrm{rhs}(r)[\xi|_1, \ldots, \xi|_\ell])$$

where in each case $\ell = \mathrm{rk}_N(\xi(\varepsilon))$. Note that P_i is finite for every $i \in \mathbb{N}$.

Claim 1: There is an $n \in \mathbb{N}$ such that $K = P_n = P_{n+1}$.
Proof of Claim 1: Since no SCC in G is top-recursive, each SCC $P \in \mathrm{scc}(\mathrm{pg}(G))$ involving (A, j), for some $j \in [\mathrm{rk}_N(A)]$, is not generating, i.e., considering all outside-in derivations using rules from $\mathrm{rules}(P)$, the set of nonterminals generated below a nonterminal is finite. Hence, an item in K is a nonterminal plus a choice of argument values drawn from a finite set. We use a saturation process to find all relevant argument values.

Furthermore, since $P_0 = \{\langle A_0 \rangle\}$, it can be seen that $P_n = K$. □

Now we let $n \in \mathbb{N}$ be such that $P_n = P_{n+1}$. We construct the RTG $H = (P_n, \Delta, \langle A_0 \rangle, R')$ where R' is defined as follows. For each $\langle \xi \rangle \in P_n$ with $\ell = \mathrm{rk}_N(\xi(\varepsilon))$ and $r \in R|_{\xi(\varepsilon)}$, let $\langle \xi \rangle \to \zeta$ be in R' where ζ is obtained from $\mathrm{rhs}(r)[\xi|_1, \ldots, \xi|_\ell]$ by replacing the subtree at

each outermost position $w \in \mathrm{pos}_N(\mathrm{rhs}(r))$ by $\langle \mathrm{rhs}(r)[\xi|_1, \ldots, \xi|_\ell]|_w \rangle$. We denote the rule constructed in this way by $[r, \langle \xi \rangle]$.

Claim 2: For each $m \in \mathbb{N}$ and $\xi \in \mathrm{T}_{N \cup \Delta}$, the following are equivalent.

(i) There is an outside-in derivation d of G such that $|d| = m$ and $A_0 \Rightarrow_d \xi$.

(ii) There is a $\xi' \in \mathrm{T}_\Delta(K)$ and a derivation d' of H such that $|d'| = m$, $\langle A_0 \rangle \Rightarrow_{d'} \xi'$, and $\xi = \mathrm{removeBrackets}(\xi')$ where $\mathrm{removeBrackets}(\xi')$ is obtained from ξ' by replacing each position labeled $\langle B(\xi_{1..\ell}) \rangle$ by the tree $B(\xi_{1..\ell})$.

Proof of Claim 2: We assume that d and d' are of the form

$$\begin{aligned} d\colon\ & A_0 = \zeta_0 \Rightarrow_{r_1} \zeta_1 \Rightarrow_{r_2} \cdots \Rightarrow_{r_{m-1}} \zeta_{m-1} \Rightarrow_{r_m} \xi \quad \text{and} \\ d'\colon\ & \langle A_0 \rangle = \zeta'_0 \Rightarrow_{\tilde{r}_1, \langle \xi_1 \rangle} \zeta'_1 \Rightarrow_{\tilde{r}_2, \langle \xi_2 \rangle} \cdots \Rightarrow_{\tilde{r}_{m-1}, \langle \xi_{m-1} \rangle} \zeta'_{m-1} \Rightarrow_{\tilde{r}_m, \langle \xi_m \rangle} \xi' \ . \end{aligned}$$

(i)⇒(ii): Assume that, for each $i \in [m]$, the rule r_i is applied at position w_i in ζ_{i-1}. We obtain d' by defining, for each $i \in [m]$, that $[\tilde{r}_i, \langle \xi_i \rangle] = [r_i, \zeta'_{i-1}(w_i)]$. Then, we can show by induction that, for each $i \in [m]$, we have $\zeta_{i-1} = \mathrm{removeBrackets}(\zeta'_{i-1})$. Furthermore, since $A_0 \Rightarrow_{r_{1..(i-1)}} \zeta_{i-1}$ holds, we have by Claim 1 that $\langle \zeta_{i-1}|_{w_i} \rangle \in P_n$ and it can be seen that $\langle \zeta_{i-1}|_{w_i} \rangle = \zeta'_{i-1}(w_i)$.
(ii)⇒(i): For each $i \in [m]$, we let $r_i = \tilde{r}_i$. It can be shown by induction on m that, for each $i \in [m]$, we have $\zeta_{i-1} = \mathrm{removeBrackets}(\zeta'_{i-1})$. □

Recall from Definition 2.2.5 that $\mathcal{L}(G)$ is defined in terms of outside-in derivations. Thus, by Claim 2, we have $\mathcal{L}(H) = \mathcal{L}(G)$. ■

Example 3.3.10 We illustrate the construction in the proof of Lemma 3.3.9 by considering the lnCFTG

$$G_9 = (\{A_0{}^{(0)}, A^{(2)}\}, \{\alpha^{(0)}, \beta^{(0)}, \gamma^{(0)}, \delta^{(2)}, \kappa^{(2)}\}, A_0, R)$$

where R contains the rules

$$A_0 \to A(\alpha, \beta) \qquad \text{and} \qquad A(x_1, x_2) \to \delta(x_1, A(x_2, \gamma)) \mid \kappa(x_1, x_2) \ .$$

Applying the construction from the proof of Lemma 3.3.9 to G_9 yields the RTG H_2 with the rules

$$\langle A_0 \rangle \to \langle A(\alpha, \beta) \rangle \ , \qquad \langle A(\alpha, \beta) \rangle \to \delta(\alpha, \langle A(\beta, \gamma) \rangle) \mid \kappa(\alpha, \beta) \ ,$$

$$\langle A(\beta, \gamma) \rangle \to \delta(\beta, \langle A(\gamma, \gamma) \rangle) \mid \kappa(\beta, \gamma) \ , \text{ and} \qquad \langle A(\gamma, \gamma) \rangle \to \delta(\gamma, \langle A(\gamma, \gamma) \rangle) \mid \kappa(\gamma, \gamma) \ .$$

It can be seen that G_9 and H_2 are equivalent. ○

3.3.3 Main Theorem

Theorem 3.3.11 For each non-self-embedding lnCFTG G, we can construct a RTG H such that $\mathcal{L}(G) = \mathcal{L}(H)$, i.e., the tree language $\mathcal{L}(G)$ is regular.

Proof. By Lemma 3.2.9 we may assume that G is non-self-embedding and unique in argument positions. Furthermore, according to Theorem 3.3.8 we can assume that $\mathrm{topRank}(G) = 0$. The application of Lemma 3.3.9 yields an equivalent RTG H. Hence, $\mathcal{L}(H)$ is regular. ∎

3.4 Relationship to the String Case

We relate our result to the corresponding result for the string case. In [10, 11] it was proved that each non-self-embedding context-free (string) grammar (CFG) generates a regular string language. In [47] self-embedding was expressed as a syntactic criterion, accompanied by a direct construction of a regular string grammar (REG) starting from a non-self-embedding CFG.

Recall that CFGs and strongly monadic lnCFTGs are in a one-to-one correspondence (cf. Observation 2.2.28). For strongly monadic lnCFTGs, Property (2) of the definition of self-embedding is false. Moreover, Property (1) of that definition corresponds to the definition of self-embedding of CFG given in [11, 47]. Thus, there is a one-to-one correspondence between non-self-embedding CFGs and non-self-embedding strongly monadic lnCFTGs.

A REG can be viewed as a strongly monadic lnCFTG (N, Δ, A_0, R) in which the RHS of each rule satisfies the property that the subtree below a nonterminal only consists of x_1. Let us call such grammars *strongly monadic RTG*. There is an obvious one-to-one correspondence between REGs and strongly monadic RTGs: it is the restriction of the above mentioned one-to-one correspondence between CFGs and strongly monadic lnCFTGs (to REGs and strongly monadic RTGs, respectively).

We will now analyze the proof of Theorem 3.3.11 when G is a non-self-embedding strongly monadic lnCFTG. Note that in a strongly monadic lnCFTG, every position in the RHS is variable dominating. Hence, each occurrence of a nonterminal in the RHS of a rule induces an edge in the position graph. We compare our construction to the function *make_fa* (cf. [47, Figure 1.3]) of the string case.

The first part of the proof is concerned with the property of being unique in argument positions. In case of a strongly monadic lnCFTG, there are only two argument positions, viz. 0 and 1. By Observation 3.2.3, we have that those two argument positions are never in the same SCC. Hence, every strongly monadic lnCFTG is unique in argument positions.

The second part of the proof removes top-recursive SCCs. We discuss it for an example rule of a top-recursive SCC P. Let r be a rule $A(x_1) \to B(\gamma(C(x_1)))$ in $\mathrm{rules}(P)$ where A and B are both nonterminals in M_P. Removing top-recursion in a strongly monadic lnCFTG is similar to handling left-recursion in the string case. According to (4) from the proof of Lemma 3.3.6, we reverse r to the rule $r' \colon \langle A, B, \mathbf{d}\rangle(x_1) \to \gamma(C(\langle A, A, \mathbf{d}\rangle(x_1)))$. We compare r' to the output of *make_fa* applied to the rule $A \to B\gamma C$. This yields a rule $q_B \to \gamma C q_A$, which corresponds to r'.

Lastly, we consider the transformation of a strongly monadic non-self-embedding lnCFTG which does not contain top-recursive SCCs into a RTG. This construction corresponds to

the application of *make_fa* to A_0 and uses the case for right-recursion.

3.5 Alternative Proof of Regularity

In this section, we explore a different way of transforming a top-recursion free, non-self-embedding lnCFTG G into a RTG. It is an alternative to the construction in the proof of Lemma 3.3.9. The alternative proof is inspired from the approach taken for CFG [47]. We consider the SCCs of $\mathrm{pg}(G)|_{N\times\{0\}}$. The paths between these SCCs encode the calling structure of nonterminals in H. A nonterminal A calls a nonterminal B if B occurs variable dominating in the RHS of an A-rule. In this case, there is a path from $(A,0)$ to $(B,0)$ in $\mathrm{pg}(G)|_{N\times\{0\}}$.

The alternative proof proceeds as follows. Consider a SCC P from $\mathrm{pg}(G)|_{N\times\{0\}}$. First, we construct, for each $A \in M_P$, the RTG H_A such that $\mathcal{L}(H_A) = \mathcal{L}(\mathrm{box}(G)|_{M_P}, A(\overline{x}))$ where $\mathrm{box}(G)$ is a restriction of G that only derives variable dominating occurrences of nonterminals. This step is similar to the construction in the proof of Lemma 3.3.9. Instead of deriving all nonterminals, only the nonterminals in M_P are considered. All other nonterminals are treated as terminals. Second, we join any family of RTG with the property mentioned in the first step via closure properties of regular tree languages (cf. Section 2.2) and obtain a RTG H such that $\mathcal{L}(H) = \mathcal{L}(G)$. Thus, regularity of $\mathcal{L}(G)$ is proven.

The alternative proof has two main advantages. First, each RTG H_A which is obtained by the alternative approach can be minimized individually. Since minimization of RTG is PSPACE-complete [32, Thm. 3.2], it is beneficial to minimize multiple smaller RTGs instead of one large RTG. Second, this alternative proof allows substituting the construction in Section 3.5.1 by other methods. The characterization can also be replaced by an approximation of each RTG. Thus, the alternative proof for Lemma 3.3.9 yields the justification for an approximation.

3.5.1 Construct a Family of RTGs

In this section, *we let* $G = (N, \Delta, A_0, R)$ *be a non-self-embedding lnCFTG that is unique in argument positions and such that* $\mathrm{topRank}(G) = 0$.

We construct, for each $P \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$ and each $A \in M_P$, a RTG H_A such that $\mathcal{L}(H_A) = \mathcal{L}(\mathrm{box}(G)|_{M_P}, A(\overline{x}))$ where $\mathrm{box}(G)$ will be defined in Definition 3.5.2. We differentiate variable dominating occurrences of nonterminals from those occurrences of nonterminals that are not variable dominating. Since the latter play no part in recursion, we postpone the derivation of those nonterminals in $\mathrm{box}(G)$ as follows. Each subtree ξ rooted in a nonterminal that is not variable dominating, is replaced by a new symbol $(\!|\xi|\!)$.

Definition 3.5.1 Let $\xi \in \mathrm{T}_{N\cup\Delta}(X)$ and

$$\mathbb{B}_\xi = \{(\!|\xi|_w|\!) \mid w \in \mathrm{pos}_{N\setminus N^{(0)}}(\xi), w \text{ is not variable dominating}\} \ .$$

Then the *variable dominating fragment of* ξ, denoted by $\mathrm{box}(\xi)$, is the tree $\xi' \in \mathrm{T}_{N\cup\Delta}(\mathbb{B})$ which is obtained from ξ by replacing the subtree at each outermost position $w \in \mathrm{pos}_{N\setminus N^{(0)}}(\xi)$ which is not variable dominating by the new terminal $(\!|\xi|_w|\!)$. □

If we represent a tree ξ as the symbol $(\!|\xi|\!)$, we say that we *box* the tree. Note that Definition 3.5.1 only targets nonterminals with a rank larger than zero. Since nullary nonterminals can only occur not variable dominating, we choose to allow them here rather than handling the symbols $(\!|B|\!)$ (for $B \in N^{(0)}$) later. We furthermore note that while this decision helps the intuition, it does not influence the proofs in a major way.

The variable dominating fragment of G is obtained by applying $\mathrm{box}(\cdot)$ to the right-hand side of each rule.

Definition 3.5.2 We define the *variable dominating fragment of* G, denoted by $\mathrm{box}(G)$, as the lnCFTG $\mathrm{box}(G) = (N, \Delta \cup \mathbb{B}, A_0, R_{\mathbb{B}})$ where

- $\mathbb{B} = \bigcup_{\xi \in \mathrm{rhs}(R)} \mathbb{B}_\xi$ and each $(\!|\xi|\!) \in \mathbb{B}$ has rank 0, and
- $R_{\mathbb{B}} = \{A(\overline{x}) \to \mathrm{box}(\xi) \mid A(\overline{x}) \to \xi \in R\}$. □

In the following, *we let* $\mathrm{box}(G) = (N, \Delta \cup \mathbb{B}, A_0, R_{\mathbb{B}})$.

Observation 3.5.3 Since the edges in $\mathrm{pg}(G)$ only depend on variable dominating nonterminal occurrences, we have that $\mathrm{pg}(G) = \mathrm{pg}(\mathrm{box}(G))$. Thus, we also have $\mathrm{topRank}(G) = \mathrm{topRank}(\mathrm{box}(G)) = 0$.

The proof of the following lemma is very similar to the proof of Lemma 3.3.9. We fix one SCC P of $\mathrm{pg}(\mathrm{box}(G))|_{N\times\{0\}}$ and one nonterminal $A \in M_P$. Then, we construct the RTG H_A as follows. We start with the new nonterminal $\langle A(\overline{x})\rangle$ and create new nonterminals of the form $\langle\xi\rangle$ where $\xi(\varepsilon) \in M_P$ as follows. Assume a new nonterminal $\langle\xi\rangle$ such that $\xi(\varepsilon) = B$. Then we use each rule $r \in R|_B$ to create new rules for $\langle\xi\rangle$: In $\mathrm{rhs}(r)$, we replace each variable x_i by the subtree $\xi|_i$. Furthermore, we replace each subtree ξ' that is rooted in a nonterminal from M_P by the new nonterminal $\langle\xi'\rangle$. Let ζ be the tree obtained in this way. Then we create the rule $\langle\xi\rangle \to \zeta$.

We create rules for each newly introduced nonterminal $\langle\xi\rangle$ as described above. Termination of this saturation process is guaranteed, since $\mathrm{topRank}(G) = 0$. Note that while Lemma 3.3.9 considers all nonterminal occurrences, this lemma uses only subtrees rooted in nonterminals from M_P as new nonterminals for H_A. Furthermore, we note that the terminals from H_A contain Δ, $\mathbb{B}$, as well as $X_{\mathrm{rk}_N(A)}$ and $N \setminus M_P$.

Lemma 3.5.4 Let $G = (N, \Delta, A_0, R)$ be a non-self-embedding lnCFTG that is unique in argument positions and $\mathrm{topRank}(G) = 0$. For each $P \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$ and $(A, 0) \in \mathrm{V}_P$, we can construct a RTG H_A such that $\mathcal{L}(H_A) = \mathcal{L}(\mathrm{box}(G)|_{M_P}, A(\overline{x}))$.

Proof. We let $P \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$, $(A, 0) \in \mathrm{V}_P$, and construct the RTG $H_A = (N_A, \Delta_A, A', R_A)$. As an auxiliary tool, we define the function cut_{M_P} very similar to the function cut_N from the proof of Lemma 3.3.9. For every $\xi \in \mathrm{T}_{N\cup\Delta}(X \cup \mathbb{B})$, we have

$$\mathrm{cut}_{M_P}(\xi) = \{\langle\xi|_w\rangle \mid w \in \mathrm{pos}_{M_P}(\xi), w \text{ outermost in } \mathrm{pos}_{M_P}(\xi)\} \ .$$

We let $P_0 = \{\langle A(\overline{x})\rangle\}$ and define inductively, for each $i \in \mathbb{N}$,

$$P_{i+1} = P_i \cup \bigcup_{\substack{\langle\xi\rangle \in P_i \\ r \in R_{\mathbb{B}}|_{\xi(\varepsilon)}}} \mathrm{cut}_{M_P}(\mathrm{rhs}(r)[\xi|_1, \ldots, \xi|_\ell])$$

where in each case $\ell = \mathrm{rk}_N(\xi(\varepsilon))$.
Claim 1: There is an $n \in \mathbb{N}$ such that $P_n = P_{n+1}$.
Claim 1 follows as a special case from Claim 1 of Lemma 3.3.9.

Now we let $n \in \mathbb{N}$ be such that $P_n = P_{n+1}$. We construct the desired RTG $H_A = (N_A, \Delta_A, A', R_A)$ as follows. We let

- $N_A = P_n$;
- $\Delta_A = \Delta \cup (N \setminus M_P) \cup \mathbb{B} \cup X_k$ where each symbol in $\mathbb{B}$ and X_k is nullary;
- $A' = \langle A(\overline{x}) \rangle$; and
- for each $\langle \xi \rangle \in P_n$ and $r \in R_{\mathbb{B}}|_{\xi(\varepsilon)}$, we let $\langle \xi \rangle \to \zeta$ be in R_A where ζ is obtained from $\mathrm{rhs}(r)[\xi|_1, \ldots, \xi|_\ell]$ by replacing the subtree at each outermost position $w \in \mathrm{pos}_{M_P}(\mathrm{rhs}(r))$ by $\langle \mathrm{rhs}(r)[\xi|_1, \ldots, \xi|_\ell]|_w \rangle$ where $\ell = \mathrm{rk}_N(\xi(\varepsilon))$.

Claim 2: $\mathcal{L}(\mathrm{box}(G)|_{M_P}, A(\overline{x})) = \mathcal{L}(H_A)$.
The proof of Claim 2 is analogous to the proof of Claim 2 of Lemma 3.3.9. ∎

Note that in the case of a nullary nonterminal $B \in N^{(0)}$, we have $M_P = \{B\}$ and $P_n = \{\langle B \rangle\}$. Thus, R_B is obtained from $R|_B$ by replacing each B by $\langle B \rangle$.

Example 3.5.5 As an example, consider Figure 3.12(a) displaying the lnCFTG G_{10} that is unique in argument positions and $\mathrm{topRank}(G_{10}) = 0$. Figures 3.12(b), (c), and (d) show the RTGs H_{A_0}, H_B, and H_A, respectively. Note that H_A and H_B generate variables as terminals. Furthermore, $\mathcal{L}(H_A)$ contains the symbol B as a terminal and $\mathcal{L}(H_B)$ contains the boxed tree $(\!|A(\alpha, \alpha)|\!)$, since the corresponding occurrence of A in G_{10} is not variable dominating.

3.5.2 Combine the Family of RTGs

In this section, we assume that an N-indexed family of RTGs is given that generate the tree languages induced by each nonterminal in N. We show how to combine this family and obtain one RTG that starts from the initial nonterminal $A_0 \in N$.

Throughout this section, *we let* $(H_A \mid A \in N)$ *be a family of RTGs* $H_A = (N_A, \Delta_A, A', R_A)$ *such that, for each* $P \in \mathrm{scc}(\mathrm{pg}(G)|_{N \times \{0\}})$ *and* $A \in M_P$*, the condition*

$$\mathcal{L}(\mathrm{box}(G)|_{M_P}, A(\overline{x})) = \mathcal{L}(H_A)$$

is satisfied where Δ_A *is defined as in Section 3.5.1.*

It is noteworthy that this family is not required to be the one constructed in Section 3.5.1, but it can be any family which satisfies the condition.

$$A_0 \to A(\alpha,\alpha) \qquad A(x_1,x_2) \to \sigma(B(x_1), A(x_2,\alpha)) \mid \kappa(x_1,x_2) \qquad B(x_1) \to \sigma(B(x_1), A(\alpha,\alpha)) \mid \gamma(x_1)$$

(a) The lnCFTG G_{10}

$$\langle A_0\rangle \to A(\alpha,\alpha)$$

(b) The RTG H_{A_0}

$$\langle B(x_1)\rangle \to \sigma(\langle B(x_1)\rangle, (\!|A(\alpha,\alpha)|\!)) \mid \gamma(x_1)$$

(c) The RTG H_B

$$\langle A(x_1,x_2)\rangle \to \sigma(B(x_1), \langle A(x_2,\alpha)\rangle) \qquad \langle A(x_2,\alpha)\rangle \to \sigma(B(x_2), \langle A(\alpha,\alpha)\rangle) \qquad \langle A(\alpha,\alpha)\rangle \to \sigma(B(\alpha), \langle A(\alpha,\alpha)\rangle)$$

$$\langle A(x_1,x_2)\rangle \to \kappa(x_1,x_2) \qquad \langle A(x_2,\alpha)\rangle \to \kappa(x_2,\alpha) \qquad \langle A(\alpha,\alpha)\rangle \to \kappa(\alpha,\alpha) \qquad \bigcirc$$

(d) The RTG H_A

Figure 3.12: Example of the construction of $(H_A \mid A \in N)$

Since we only consider linear nondeleting context free tree grammars, we can reorder any derivation in an arbitrary fashion (cf. Observation 2.2.16). Hence, we can observe the following.

Observation 3.5.6 $\mathcal{L}(G) = \mathcal{L}(\mathrm{box}(G)) \leftarrow \big(\langle\!|\xi|\!\rangle/\mathcal{L}(G,\xi) \mid \langle\!|\xi|\!\rangle \in \mathbb{B}\big)$.

To prove regularity of $\mathcal{L}(G)$ it thus suffices, by Theorem 2.2.23, to independently prove regularity of $\mathcal{L}(\mathrm{box}(G))$ and $\mathcal{L}(G,\xi)$ for each $\langle\!|\xi|\!\rangle \in \mathbb{B}$. In particular, we will prove that the tree language induced by $A(\overline{x})$ on $\mathrm{box}(G)$ is regular (cf. Lemma 3.5.13) and that unboxing preserves regularity (cf. Lemma 3.5.22).

We will now show the regularity of $\mathcal{L}(\mathrm{box}(G))$. For this, we observe another decomposition result, which is related to Observation 3.5.6.

Observation 3.5.7 For each $P \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$ and $A \in M_P$, we have that

$$\mathcal{L}(\mathrm{box}(G), A(\overline{x})) = \mathcal{L}(\mathrm{box}(G)|_{M_P}, A(\overline{x})) \leftarrow (B/\mathcal{L}(\mathrm{box}(G), B(\overline{x})) \mid B \in N \setminus M_P) \ .$$

We will show the regularity of $\mathcal{L}(\mathrm{box}(G), B(\overline{x}))$ for each $B \in N$.

We observe that the SCCs in $\mathrm{pg}(G)|_{N\times\{0\}}$ can be ordered w.r.t. their calling structure. A SCC P calls a SCC P' if there are $(A,0) \in \mathrm{V}_P$ and $(B,0) \in \mathrm{V}_{P'}$ such that $(A,0) \to (B,0)$ holds in $\mathrm{pg}(G)$.

Definition 3.5.8 For each $P, P' \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$, we say P' *precedes* P, denoted by $P' < P$, iff

(i) $P \neq P'$ and

(ii) there are $(A,0) \in \mathrm{V}_P$ and $(B,0) \in \mathrm{V}_{P'}$ such that $(A,0) \to_{\mathrm{pg}(G)} (B,0)$.

We let $<^+$ denote the transitive closure of $<$ and call $<^+$ the *calling order*. If $P' <^+ P$, then we say that P' is *smaller* than P, or P is *larger* than P'. □

Intuitively, larger SCCs call smaller ones.

Lemma 3.5.9 The binary relation $<^+$ is a strict order, i.e., irreflexive ($P <^+ P$ does not hold for any $P \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$), transitive ($P_1 <^+ P_2$ and $P_2 <^+ P_3$ implies $P_1 <^+ P_3$), and asymmetric (if $P' <^+ P$, then $P <^+ P'$ does not hold).

Proof. Irreflexivity follows from Condition (i) of Definition 3.5.8. The relation $<^+$ is transitive by definition. It remains to show that $<^+$ is asymmetric. We prove this by contradiction. Assume $P, P' \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$ such that $P' <^+ P$ and $P <^+ P'$. Then, by Condition (ii) we have that there are $(A,0) \in \mathrm{V}_P$ and $(B,0) \in \mathrm{V}_{P'}$ such that $(A,0) \to^*_{\mathrm{pg}(G)} (B,0)$ and $(B,0) \to^*_{\mathrm{pg}(G)} (A,0)$ and thus, $P = P'$. This contradicts Condition (i). ■

The calling order induces an enumeration of the set $\mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$ of nonterminals. Let us now consider minimal SCCs w.r.t. the calling order. The tree language generated by each nonterminal in such a minimal SCC P' does not depend on nonterminals from SCCs which are larger than P'. In other words, for each $B \in M_{P'}$, the tree language $\mathcal{L}(\mathrm{box}(G)|_{M_{P'}}, B(\overline{x}))$

does not contain any symbol from N (Recall that Δ_B contains $N \setminus M_{P'}$). We show that the tree language $\mathcal{L}(\mathrm{box}(G), B(\overline{x}))$ is regular for each nonterminal B in a minimal SCC P'. Then we proceed by induction and show that, for each larger SCC P and each nonterminal $A \in M_P$, the tree language $\mathcal{L}(\mathrm{box}(G), A(\overline{x}))$ is regular. This is done by substituting each nonterminal B from smaller SCCs by $\mathcal{L}(\mathrm{box}(G), B(\overline{x}))$.

Formally, we choose an enumeration $P_1, \ldots, P_n$ where $n = |\mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})|$ such that, for each $i, j \in [n]$, we have that $P_i <^+ P_j$ implies $i < j$. Thus, the enumeration starts with the smaller SCCs and progresses to the larger ones. Such an enumeration exists, since the SCCs of $\mathrm{pg}(G)|_{N\times\{0\}}$ have a topological ordering.

For the rest of this section, *we fix such an enumeration* $P_1, \ldots, P_n$.

We observe two intermediate properties. First, if $P_i <^+ P_\ell$, then no symbols from M_{P_ℓ} occur in the language of the $(\bigcup_{j\in[i]} M_{P_j})$-fragment of $\mathrm{box}(G)$ induced by some nonterminal from P_i.

Observation 3.5.10 We let $i \in [n]$ and $A \in M_{P_i}$. We have that

$$\mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i]} M_{P_j}}, A(\overline{x})) \subseteq \mathrm{T}_\Delta(X_{\mathrm{rk}_N(A)} \cup \mathbb{B})$$

and thus, for each $0 \leq m \leq n - i$, it holds that

$$\mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i]} M_{P_j}}, A(\overline{x})) = \mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i+m]} M_{P_j}}, A(\overline{x})) \ .$$

Since the nonterminals and rules of $\mathrm{box}(G)|_{\bigcup_{j\in[n]} M_{P_j}}$ and $\mathrm{box}(G)$ are equal, we obtain for $m = n - i$ that

$$\mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i]} M_{P_j}}, A(\overline{x})) = \mathcal{L}(\mathrm{box}(G), A(\overline{x})) \ .$$

Second, we can extend Observation 3.5.7 and obtain the following.

Observation 3.5.11 We let $i \in [n]$. Since

$$\mathrm{pos}_{M_{P_i}}\left(\bigcup\nolimits_{\ell\in[i-1]} \bigcup\nolimits_{A\in M_{P_\ell}} \mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i-1]} M_{P_j}}, A(\overline{x}))\right) = \emptyset \ ,$$

we get for each $A \in M_{P_i}$

$$\begin{aligned}
&\mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i]} M_{P_j}}, A(\overline{x})) \\
&= \mathcal{L}(\mathrm{box}(G)|_{M_{P_i}}, A(\overline{x})) \\
&\qquad \leftarrow \left(B/\mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i-1]} M_{P_j}}, B(\overline{x})) \mid B \in \bigcup\nolimits_{j\in[i-1]} M_{P_j}\right) \ .
\end{aligned}$$

Combining Observations 3.5.10 and 3.5.11, we can show that it is sufficient to substitute only nonterminals from smaller SCCs w.r.t. the calling order.

Lemma 3.5.12 For each $i \in [n]$ and $A \in M_{P_i}$, we have that

$$\begin{aligned}
\mathcal{L}(\mathrm{box}(G), A(\overline{x})) = \mathcal{L}(\mathrm{box}(G)|_{M_{P_i}}, A(\overline{x})) \\
\leftarrow \left(B/\mathcal{L}(\mathrm{box}(G), B(\overline{x})) \mid B \in \bigcup\nolimits_{j\in[i-1]} M_{P_j}\right) \ .
\end{aligned}$$

PROOF. Let $i \in [n]$ and $A \in M_{P_i}$.

$$\begin{aligned}
\mathcal{L}(\mathrm{box}(G), A(\overline{x})) &= \mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i]} M_{P_j}}, A(\overline{x})) && \text{(Obs. 3.5.10)}\\
&= \mathcal{L}(\mathrm{box}(G)|_{M_{P_i}}, A(\overline{x})) \leftarrow \Big(B/\mathcal{L}(\mathrm{box}(G)|_{\bigcup_{j\in[i-1]} M_{P_j}}, B(\overline{x})) \mid B \in \bigcup\nolimits_{j\in[i-1]} M_{P_j}\Big) && \text{(Obs. 3.5.11)}\\
&= \mathcal{L}(\mathrm{box}(G)|_{M_{P_i}}, A(\overline{x})) \leftarrow \Big(B/\mathcal{L}(\mathrm{box}(G), B(\overline{x})) \mid B \in \bigcup\nolimits_{j\in[i-1]} M_{P_j}\Big) && \text{(Obs. 3.5.10)}
\end{aligned}$$

■

Based on the fixed enumeration we can now show the desired lemma.

Lemma 3.5.13 For each $A \in N$, the tree language $\mathcal{L}(\mathrm{box}(G), A(\overline{x}))$ is regular.

PROOF. For each $i \in [n]$ and $A \in M_{P_i}$ we have

$$\begin{aligned}
\mathcal{L}(\mathrm{box}(G), A(\overline{x})) &= \mathcal{L}(\mathrm{box}(G)|_{M_{P_i}}, A(\overline{x}))\\
&\qquad \leftarrow \big(B/\mathcal{L}(\mathrm{box}(G), B(\overline{x})) \mid B \in \bigcup\nolimits_{j\in[i-1]} M_{P_j}\big) && \text{(Lemma 3.5.12)}\\
&= \mathcal{L}(H_A) \leftarrow \big(B/\mathcal{L}(\mathrm{box}(G), B(\overline{x})) \mid B \in \bigcup\nolimits_{j\in[i-1]} M_{P_j}\big)\ .
\end{aligned}$$

We prove the lemma by induction on $i \in [n]$. Let $i = 1$ and $A \in M_{P_1}$. Then, the above equation reduces to $\mathcal{L}(\mathrm{box}(G), A(\overline{x})) = \mathcal{L}(H_A)$ and thus, $\mathcal{L}(\mathrm{box}(G), A(\overline{x}))$ is regular. Now, we let $i \in [n]$, $A \in M_{P_i}$ and assume that, for each $j \in [i-1]$ and $B \in M_{P_j}$, we have that $\mathcal{L}(\mathrm{box}(G), B(\overline{x}))$ is regular. Since the substitution of regular tree languages is again a regular tree language (cf. Theorem 2.2.23), we have that $\mathcal{L}(\mathrm{box}(G), A(\overline{x}))$ is regular. ■

We have captured all variable dominating parts of G. It remains to show that unboxing preserves regularity, i.e., the tree language induced by ξ_i on G is regular for each $(\!|\xi_i|\!) \in \mathbb{B}$. We recall that

$$\begin{aligned}
\mathbb{B} &= \{\mathbb{B}_\xi \mid \xi \in \mathrm{rhs}(R)\}\\
&= \{(\!|\xi|_w|\!) \mid \xi \in \mathrm{rhs}(R), w \in \mathrm{pos}_{N\setminus N^{(0)}}(\xi), w \text{ not variable dominating}\}\ .
\end{aligned}$$

Assume that $(\!|\xi|\!) \in \mathbb{B}$ with $\xi = B(\xi_{1..k})$ for some $B \in N^{(k)}$ and $\xi_1, \ldots, \xi_k \in \mathrm{T}_\Delta$. The grammar $\mathrm{box}(G)$ regards $(\!|\xi|\!)$ as a terminal of rank 0. We want to substitute each such symbol $(\!|\xi|\!)$ by the tree language $\mathcal{L}(G, \xi)$ and show that it is regular. Since $(\!|\xi|\!)$ is rooted in B we use the tree language $\mathcal{L}(\mathrm{box}(G), B(\overline{x}))$ as basis and replace the variables x_i by $\mathrm{box}(\xi_i)$. Note that $\mathcal{L}(\mathrm{box}(G), B(\mathrm{box}(\xi_1), \ldots, \mathrm{box}(\xi_\ell)))$ might itself contain $(\!|\xi|\!)$ and hence, this substitution has to be applied repeatedly. We can achieve this by using the Kleene star ${}^{*(\!|\xi|\!)}$. Remaining occurrences of $(\!|\xi|\!)$ are eliminated by substituting trees from $\mathcal{L}(\mathrm{box}(G), B(\mathrm{box}(\xi_1), \ldots, \mathrm{box}(\xi_\ell)))$ intersected with all trees not containing $(\!|\xi|\!)$. In a general setting, ξ might contain other nonterminal occurrences (not at the root) which are thus also not variable dominating. In this case, we need to substitute tree languages for those occurrences as well. To show the regularity of the tree language $\mathcal{L}(G, \xi_i)$, for each $(\!|\xi_i|\!) \in \mathbb{B}$, we utilize a suitable order: We start with trees of a minimal size. Such a

minimal tree ξ_i cannot contain an occurrences of another tree ξ_j where $(\!|\xi_j|\!) \in \mathbb{B}$. Then, we propagate the obtained regular tree languages upwards (in terms of tree size) and thus show regularity for the tree languages induced by trees with multiple, nested nonterminal occurrences.

We fix an enumeration of the elements of $\mathbb{B}$ which is not descending in the size of its trees. Formally, if $|\mathbb{B}| = m$, then such an enumeration $(\!|\xi_1|\!)$, …, $(\!|\xi_m|\!)$ satisfies that, for every i, j, we have that $i < j$ implies $|\mathrm{pos}(\xi_i)| \leq |\mathrm{pos}(\xi_j)|$.

For the rest of this section, *we fix one such enumeration* $(\!|\xi_1|\!), \ldots, (\!|\xi_m|\!)$.

We want to undo the isolation of not variable dominating nonterminal occurrences. For this, we need a dual operation to $\mathrm{box}(\cdot)$.

Definition 3.5.14 For each $j \in [m]$ and $\zeta \in \mathrm{T}_{N\cup\Delta}(X \cup \mathbb{B})$, we define $\mathrm{unbox}_j(\zeta) = \zeta'$ where ζ' is obtained from ζ by replacing, for each $i \in [j]$, each occurrence of $(\!|\xi_i|\!)$ by the tree ξ_i.

We let $\mathrm{unbox}_j(G)$ denote the lnCFTG G' which is obtained from G by applying $\mathrm{unbox}_j(\cdot)$ to the RHS of each rule in G. Furthermore, we let $\mathrm{unbox}_0(\cdot)$ be the identity. □

If we replace the symbol $(\!|\xi|\!)$ by the tree ξ, we say that we *unbox* $(\!|\xi|\!)$.

In the following, we provide intermediate results that will prove useful for showing regularity of $\mathcal{L}(G, \xi_i)$. These results describe the relationship between the boxed and unboxed trees.

Observation 3.5.15 $\mathcal{L}(\mathrm{unbox}_m(\mathrm{box}(G))) = \mathcal{L}(G)$.

Observation 3.5.16 For every $i, i' \in [m]$, if there is a $w \in \mathrm{pos}(\xi_{i'}) \setminus \{\varepsilon\}$ such that $\xi_i = \xi_{i'}|_w$, then $i < i'$. Hence, for each $j \in [m]$, $k \in \mathbb{N}$, $B \in N^{(k)}$, and $\zeta_{1..k} \in \mathrm{T}_{N\cup\Delta}$ where $(\!|\xi_j|\!) = (\!|B(\zeta_{1..\ell})|\!)$, we have that

$$\mathrm{unbox}_{j-1}(B(\mathrm{box}(\zeta_1), \ldots, \mathrm{box}(\zeta_\ell))) = B(\zeta_1, \ldots, \zeta_\ell) = \xi_j \ .$$

As a generalization of Observation 3.5.6, we obtain the following observation.

Observation 3.5.17 For each $j \in [m]$ and $\xi \in \mathrm{T}_{N\cup\Delta}(X \cup \mathbb{B})$, we have

$$\begin{aligned}
&\mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \mathrm{unbox}_j(\xi)) \\
&= \mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \mathrm{unbox}_{j-1}(\xi)) \leftarrow_{(\!|\xi_j|\!)} (\mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_j)) \\
&= \mathcal{L}(\mathrm{box}(G), \xi) \leftarrow (\!|\xi_\ell|\!)/\mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_\ell) \mid \ell \in [j]) \ .
\end{aligned}$$

The next step is similar to an idea in [67, proof of Thm. 9] which deals with the elimination of a state in computations of a finite-state tree automaton or, equivalently, elimination of a nonterminal in the derivations of a RTG. Note that since we are only considering linear nondeleting CFTGs, these derivations can be reordered arbitrarily (cf. Lemma 2.2.15).

The idea behind the proof is to eliminate the symbol $(\!|\xi_j|\!)$ (considered as nonterminal) from the grammar $\mathrm{unbox}_{j-1}(\mathrm{box}(G))$. If only $(\!|\xi_1|\!), \ldots, (\!|\xi_{j-1}|\!)$ are unboxed, then a derivation in $\mathrm{unbox}_{j-1}(\mathrm{box}(G))$ is stopped at the symbol $(\!|\xi_j|\!)$, which has rank 0. We replace this symbol by trees derived from ξ_j in $\mathrm{unbox}_{j-1}(\mathrm{box}(G))$. However, the replacement may in

turn contain $(\!|\xi_j|\!)$ and thus, this process must be repeated. We get, for each $\xi \in \mathrm{T}_{N\cup\Delta}(X)$, that

$$\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi) \leftarrow_{(\!|\xi_j|\!)} \left(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j)^{*(\!|\xi_j|\!)}\right)$$

may contain $(\!|\xi_j|\!), \ldots, (\!|\xi_m|\!)$. To eliminate all occurrences of $(\!|\xi_j|\!)$, we lastly replace each remaining occurrence by trees from

$$\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j) \cap \mathrm{T}_\Delta(\{(\!|\xi_{j+1}|\!), \ldots, (\!|\xi_m|\!)\}) \ .$$

Note that the resulting tree language does not contain $(\!|\xi_1|\!), \ldots, (\!|\xi_j|\!)$ and thus, we successfully eliminated $(\!|\xi_j|\!)$. The above description is formalized in the following lemma.

Lemma 3.5.18 For each $\xi \in \mathrm{T}_{N\cup\Delta}(X)$ and $j \in [m]$, we have that

$$\begin{aligned}
&\mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi) \\
&\quad = \mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi) \\
&\qquad \leftarrow_{(\!|\xi_j|\!)} \left(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j)^{*(\!|\xi_j|\!)}\right) \\
&\qquad \leftarrow_{(\!|\xi_j|\!)} \left(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j) \cap \mathrm{T}_\Delta(\{(\!|\xi_{j+1}|\!), \ldots, (\!|\xi_m|\!)\})\right) \ .
\end{aligned}$$

We will use Lemma 3.5.18 for the special case where $\xi = \xi_j$, and thus, the right-hand side of the equation in Lemma 3.5.18 can be simplified. This is formalized in the following lemma using arbitrary tree languages and a nullary symbol instead of $(\!|\xi_j|\!)$.

Lemma 3.5.19 Let Σ be an arbitrary ranked alphabet, $\delta \notin \Sigma$ be a nullary symbol, and $L \subseteq \mathrm{T}_\Sigma(\{\delta\})$. Then

$$L \leftarrow_\delta L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma) = L^{*\delta} \cap \mathrm{T}_\Sigma \ .$$

Proof. We prove the lemma in three steps:
Claim 1: $L^{*\delta} = (L \leftarrow_\delta L^{*\delta}) \cup \{\delta\}$.
Proof of Claim 1: Claim 1 follows directly from the definition of $L^{*\delta}$, the associativity of $\leftarrow_\delta$, and the distributivity of $\leftarrow_\delta$ over $\cup$. □

Claim 2: $L \leftarrow_\delta L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma) = L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma)$.
Proof of Claim 2: By Claim 1, $L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma) = ((L \leftarrow_\delta L^{*\delta}) \cup \{\delta\}) \leftarrow_\delta (L \cap \mathrm{T}_\Sigma)$ holds. Distributivity of $\leftarrow_\delta$ over $\cup$ yields

$$((L \leftarrow_\delta L^{*\delta}) \cup \{\delta\}) \leftarrow_\delta (L \cap \mathrm{T}_\Sigma) = (L \leftarrow_\delta L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma)) \cup (\{\delta\} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma)) \ .$$

We observe that $L \cap \mathrm{T}_\Sigma \subseteq L \leftarrow_\delta L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma)$, since trees without occurrences of δ are not altered by the substitution. This proves Claim 2. □

Claim 3: $L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma) = L^{*\delta} \cap \mathrm{T}_\Sigma$
Proof of Claim 3: [$\subseteq$]: We replace each δ in $L^{*\delta}$ by a tree from L which does not contain occurrences of δ. This yields a tree in T_Σ.
[$\supseteq$]: We assume a tree $\xi \in L^{*\delta} \cap \mathrm{T}_\Sigma$, i.e., there is a minimal $j \in \mathbb{N}$ such that $\xi \in \{\delta\} \leftarrow_\delta L \leftarrow_\delta \ldots \leftarrow_\delta L$ (the substitution is applied j times). We do a case distinction on j. If $j = 0$, then $\xi = \delta$, which contradicts $\xi \in \mathrm{T}_\Sigma$. Hence, we have $j \geq 1$. Consider the last substitution. Since $\xi \in \mathrm{T}_\Sigma$, the trees substituted in the last step need to be in T_Σ, and hence, $\xi \in L^{*\delta} \leftarrow_\delta (L \cap \mathrm{T}_\Sigma)$. □

The lemma follows from Claims 2 and 3. ■

Now we prove that, for each $j \in [m]$ and $i \in [j]$, the tree language induced by ξ_i on the lnCFTG $\mathrm{unbox}_j(\mathrm{box}(G))$ is regular. Then, in particular, for each $i \in [m]$, we have that $\mathcal{L}(G, \xi_i)$ is regular. For this, for each $j \in [m]$ and $i \in [j]$, we define as an abbreviation

$$L_i^j = \mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_i) \ .$$

Intuitively, in the term L_i^j, the upper index j denotes the number of already unboxed symbols in the sequence $(\!|\xi_1|\!), \ldots, (\!|\xi_m|\!)$, and the lower index i refers to the index of the element $(\!|\xi_i|\!)$. Using this abbreviation, we provide the following decomposition result.

Lemma 3.5.20 For each $j \in [m]$, $k \in \mathbb{N}$, $B \in N^{(k)}$ and $\zeta_1, \ldots, \zeta_k \in \mathrm{T}_{N \cup \Delta}$ where $(\!|\xi_j|\!) = (\!|B(\zeta_{1..k})|\!)$, we have

(i) $L_j^j = \Big((\mathcal{L}(\mathrm{box}(G), B(\overline{x}))[x_\ell / \mathrm{box}(\zeta_\ell) \mid \ell \in [k]]) \leftarrow \big((\!|\xi_i|\!) / L_i^{j-1} \mid i \in [j-1] \big) \Big)^{*(\!|\xi_j|\!)} \cap \mathrm{T}_\Delta(\{(\!|\xi_{j+1}|\!), \ldots, (\!|\xi_m|\!)\})$ and

(ii) for each $i \in [j-1]$, it holds that $L_i^j = L_i^{j-1} \leftarrow_{(\!|\xi_j|\!)} L_j^j$.

Proof. We let $j \in [m]$. For Case (i), we let $B \in N$ and $\zeta_{1..k} \in \mathrm{T}_{N \cup \Delta}$ such that $(\!|\xi_j|\!) = (\!|B(\zeta_{1..k})|\!)$, and we abbreviate $\mathrm{T}_\Delta(\{(\!|\xi_{j+1}|\!), \ldots, (\!|\xi_m|\!)\})$ by T^j.

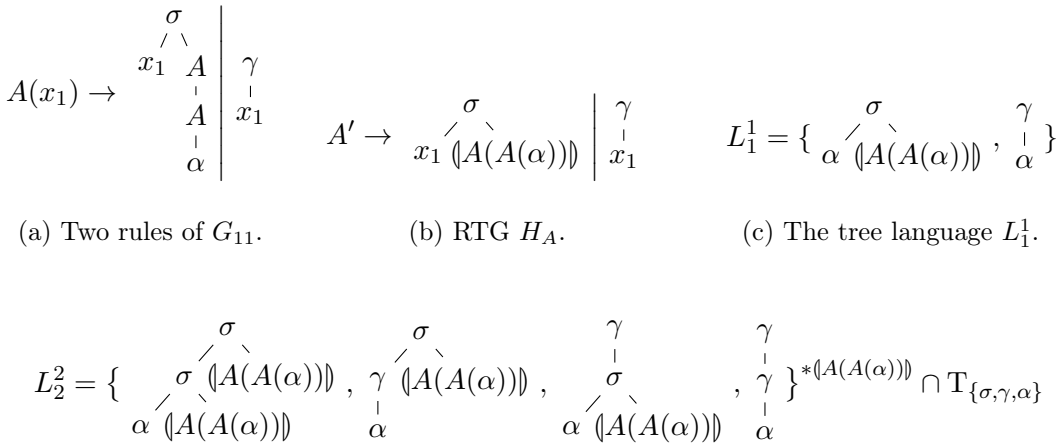
$$\begin{aligned}
&\big((\mathcal{L}(\mathrm{box}(G), B(\overline{x}))[x_\ell / \mathrm{box}(\zeta_\ell) \mid \ell \in [k]]) \leftarrow ((\!|\xi_i|\!) / L_i^{j-1} \mid i \in [j-1]) \big)^{*(\!|\xi_j|\!)} \cap \mathrm{T}^j \\
&= \big(\mathcal{L}(\mathrm{box}(G), B(\mathrm{box}(\zeta_1), \ldots, \mathrm{box}(\zeta_k))) \leftarrow ((\!|\xi_i|\!) / L_i^{j-1} \mid i \in [j-1]) \big)^{*(\!|\xi_j|\!)} \cap \mathrm{T}^j \\
&= \big(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \mathrm{unbox}_{j-1}(B(\mathrm{box}(\zeta_1), \ldots, \mathrm{box}(\zeta_k)))) \big)^{*(\!|\xi_j|\!)} \cap \mathrm{T}^j && \text{(Obs. 3.5.17)} \\
&= \big(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j) \big)^{*(\!|\xi_j|\!)} \cap \mathrm{T}^j && \text{(Obs. 3.5.16)} \\
&= \big(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j) \big) \\
&\quad \leftarrow \big(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j)^{*(\!|\xi_j|\!)} \big) \\
&\quad \leftarrow \big(\mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_j) \cap \mathrm{T}^j \big) && \text{(Lm. 3.5.19)} \\
&= \mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_j) (= L_j^j) && \text{(Lm. 3.5.18)}
\end{aligned}$$

For Case (ii), we have, for each $i \in [j-1]$, that

$$\begin{aligned}
L_i^{j-1} \leftarrow_{(\!|\xi_j|\!)} L_j^j &= \mathcal{L}(\mathrm{unbox}_{j-1}(\mathrm{box}(G)), \xi_i) \leftarrow_{(\!|\xi_j|\!)} \mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_j) \\
&= \mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \mathrm{unbox}_j(\xi_i)) && \text{(Obs. 3.5.17)} \\
&= \mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_i)
\end{aligned}$$

■

Example 3.5.21 As an example, we consider the part of the lnCFTG G_{11} depicted in Figure 3.13(a). In $\mathrm{pg}(G_{11})$, the vertex $(A, 0)$ is in a trivial SCC, since there are no variable dominating occurrences of nonterminals in the RHSs. The corresponding RTG H_A is depicted in Figure 3.13(b). We have $\mathbb{B} = \{(\!|\xi_1|\!), (\!|\xi_2|\!)\}$ where $\xi_1 = A(\alpha)$ and $\xi_2 = A(A(\alpha))$ and thus, $m = 2$. In this simple scenario, Figure 3.13(c) depicts L_1^1 completely, since it is finite. In Figure 3.13(d) we indicate how L_2^2 is obtained from $\mathcal{L}(H_A)$ and L_1^1. ○



(a) Two rules of G_{11}.

(b) RTG H_A.

(c) The tree language L_1^1.

(d) The tree language L_2^2.

Figure 3.13: Unboxing $\mathrm{box}(G_{11})$.

Lemma 3.5.22 The tree language $\mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_i)$ is regular for every $j \in [m]$ and $i \in [j]$.

Proof. By induction on $j \in [m]$ we can prove the following claim: For each $i \in [j]$, the tree language $\mathcal{L}(\mathrm{unbox}_j(\mathrm{box}(G)), \xi_i)$ is regular. We start with $i = j$ and use Lemma 3.5.20 (i). Then, we apply the decomposition result from Lemma 3.5.13 and the closure properties of regular tree languages, i.e., we apply Theorems 2.2.20, 2.2.22, and 2.2.23. Then, with decreasing $i \in [j]$, we employ Lemma 3.5.20 (ii) and again use the closure of regular tree languages under substitution (cf. Theorem 2.2.23). ∎

We summarize the findings of this section in the following theorem. Combined with Lemma 3.5.4, it is an alternative proof of Theorem 3.3.11.

Theorem 3.5.23 Let $G = (N, \Delta, A_0, R)$ be a non-self-embedding lnCFTG that is unique in argument positions and $\mathrm{topRank}(G) = 0$. Given a family $(H_A \mid A \in N)$ of RTGs such that, for each $P \in \mathrm{scc}(\mathrm{pg}(G)|_{N\times\{0\}})$ and $A \in M_P$, we have $\mathcal{L}(\mathrm{box}(G)|_{M_P}, A(\overline{x})) = \mathcal{L}(H_A)$, we can construct a RTG H such that $L(G) = \mathcal{L}(H)$, i.e., $\mathcal{L}(G)$ is a regular tree language.

Proof. We let $(H_A \mid A \in N)$ be a family of RTG as required in the theorem. Furthermore, we let $m = |\mathbb{B}|$ where $\mathbb{B}$ is defined in Definition 3.5.2. We get the following:

$$\begin{aligned}
\mathcal{L}(G) &= \mathcal{L}(G, A_0) && \text{(Def. of } \mathcal{L}(\cdot)\text{)}\\
&= \mathcal{L}(\mathrm{unbox}_m(\mathrm{box}(G)), \mathrm{unbox}_m(\mathrm{box}(A_0))) && \text{(Def. of box}(\cdot)\text{, and unbox}(\cdot)\text{)}\\
&= \mathcal{L}(\mathrm{box}(G), \mathrm{box}(A_0)) \leftarrow (\langle\!|\xi_i|\!\rangle/\mathcal{L}(\mathrm{unbox}_m(\mathrm{box}(G)), \xi_i) \mid i \in [m]) && \text{(Obs. 3.5.17)}
\end{aligned}$$

Since A_0 is nullary, $\mathrm{box}(A_0) = A_0$. Using Lemma 3.5.13, $\mathcal{L}(\mathrm{box}(G), A_0)$ is a regular tree language. By Lemma 3.5.22, for each $i \in [m]$, we have that $\mathcal{L}(\mathrm{unbox}_m(\mathrm{box}(G)), \xi_i)$ is a regular tree language. Hence, using Theorem 2.2.23, the substitution is regular as well. ∎

3.6 Non-Self-Embedding deleting lCFTGs

Since each linear (and deleting) CFTG can be transformed into an equivalent lnCFTG (cf. Theorem 2.2.17), an obvious question is whether the tree language induced by non-self-embedding linear and deleting CFTG can be expressed by a RTG. We recall that Definition 3.1.1 applies to all CFTG and thus, the notion of non-self-embedding lCFTG is defined. Furthermore, the position pair graph (cf. Definition 3.1.5) and the position graph (cf. Definition 3.2.2) are also defined on lCFTG. Hence, according to Corollary 3.1.8 it can be decided in polynomial time whether a lCFTG is self-embedding or non-self-embedding.

To show that the tree language induced by a non-self-embedding lCFTG can be expressed by a RTG, it suffices to show that the construction in the proof of Theorem 2.2.17, which replaces deleting rules, preserves the property of being non-self-embedding.

Lemma 3.6.1 For each non-self-embedding lCFTG G, there is an equivalent non-self-embedding lnCFTG G'.

Proof. We show that the transformation described in the proof of Theorem 2.2.17 preserves the property of non-self-embedding by contraposition. Assume that the transformation applied on a lCFTG G results in a lnCFTG G' that is self-embedding. Then we have a cycle in $\mathrm{ppg}(G')$ from a vertex (A_α, i, j) to (A_α, i, j) such that the union of all labels in the cycle contains 1 and 2. Due to the nature of the transformation, which does no more than systematically remove argument positions, we must then have a cycle in $\mathrm{ppg}(G)$ from some vertex (A, i', j') to (A, i', j') such that the union of all labels in the cycle once more contains 1 and 2. The argument positions $i' \geq i$ and $j' \geq j$ are straightforwardly obtained from i and j by accounting for the removed positions as recorded in α. Thereby G must be self-embedding as well. ∎

Theorem 3.6.2 Each non-self-embedding lCFTG induces a regular tree language.

Proof. By Lemma 3.6.1 for each non-self-embedding lCFTG G, there is a non-self-embedding lnCFTG G' such that $\mathcal{L}(G) = \mathcal{L}(G')$ and by Theorem 3.3.11 we have that $\mathcal{L}(G')$ and thus, $\mathcal{L}(G)$ is a regular tree language. ∎

Note, that applying the construction to remove deletion from a lCFTG G (cf. Theorem 2.2.17) combined with the removal of useless rules (Corollary 2.2.19) may transform a self-embedding lCFTG into a non-self-embedding lnCFTG. This also holds if G contained no useless rules before the removal as the following example shows.

Example 3.6.3 We define the lCFTG $G_{12} = (\{{A_0}^{(0)}, A^{(2)}\}, \{\gamma^{(1)}, \alpha^{(0)}\}, A_0, R)$ where R contains the following rules.

$$A_0 \to \overset{A}{\underset{\alpha \quad \alpha}{/ \ \backslash}} \qquad\qquad A(x_1, x_2) \to \left.\begin{array}{c} \overset{A}{/ \ \backslash} \\ \underset{x_1 \quad x_2}{\gamma \quad \gamma} \end{array}\right| x_1$$

Note that G_{12} is self-embedding since A produces symbols synchronously in two of its argument positions. Intuitively, it is clear that the second argument position of A does

not contribute to any terminal tree and thus, G_{12} can easily be modified to obtain a non-self-embedding lnCFTG.

Applying the construction of Theorem 2.2.17 yields the lnCFTG G_{13} with nonterminals A_0, $A_\emptyset$, $A_{\{1\}}$, $A_{\{2\}}$, and $A_{\{1,2\}}$ and the following rules.

$$A_0 \to \begin{array}{c} A_\emptyset \\ \alpha \quad \alpha \end{array} \;\Big|\; \begin{array}{c} A_{\{1\}} \\ \alpha \end{array} \;\Big|\; \begin{array}{c} A_{\{2\}} \\ \alpha \end{array} \;\Big|\; A_{\{1,2\}} \qquad A_\emptyset(x_1, x_2) \to \begin{array}{c} A_\emptyset \\ \gamma \quad \gamma \\ x_1 \quad x_2 \end{array}$$

$$A_{\{1\}}(x_2) \to \begin{array}{c} A_{\{1\}} \\ \gamma \\ x_2 \end{array} \qquad A_{\{2\}}(x_1) \to \begin{array}{c} A_{\{2\}} \\ \gamma \\ x_1 \end{array} \;\Big|\; x_1 \qquad A_{\{1,2\}} \to A_{\{1,2\}}$$

Note that G_{13} is self-embedding. However, it is clear that $A_\emptyset$, $A_{\{1\}}$, and $A_{\{1,2\}}$ cannot be used to derive any terminal tree. Hence, if we prune all useless rules (cf. Corollary 2.2.19), we see that the remaining part of G_{13} is non-self-embedding. ○

Observation 3.6.4 Example 3.6.3 and Lemma 3.6.1 suggest that testing for the property of being non-self-embedding should be done after applying Theorem 2.2.17 and Corollary 2.2.19.

3.7 Non-Weakly-Self-Embedding CFTGs

Theorems 3.3.11 and 3.6.2 only apply to lnCFTG and lCFTG, respectively, but the proofs cannot be applied to an unrestricted CFTG. We illustrate this by using the following example.

Example 3.7.1 Let $G_{14} = (\{{A_0}^{(0)}, A^{(1)}\}, \{\kappa^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}, A_0, R)$ be a CFTG where R contains the rules

$$A_0 \to \begin{array}{c} A \\ \alpha \end{array} \qquad \text{and} \qquad A(x_1) \to \begin{array}{c} A \\ \gamma \\ x_1 \end{array} \;\Big|\; \begin{array}{c} \kappa \\ x_1 \quad x_1 \end{array} .$$

We note that G_{14} is not linear. By considering $\mathrm{ppg}(G_{14})$, it can be seen that G_{14} is non-self-embedding (cf. Theorem 3.1.7). However, we have $\mathcal{L}(G_{14}) = \{\kappa(\gamma^n(\alpha), \gamma^n(\alpha)) \mid n \in \mathbb{N}\}$ which is not a regular tree language. ○

In the light of Example 3.7.1, it is interesting to find a criterion that guarantees regularity of the induced tree language for the full class of CFTGs.

In [54], a self-embedding property for indexed grammars is defined that guarantees that a non-self-embedding indexed grammar induces a context-free string language. We describe this property and compare it to our definition of non-self-embedding lnCFTG. Furthermore, we indicate how the property of indexed grammars can be transferred to (unrestricted) CFTGs. For this, we define weakly-self-embedding CFTGs and show that each non-weakly-self-embedding CFTG induces a regular tree language.

Indexed grammars are introduced in [1, Ch. 2]. For notational convenience, we use the characterization of indexed grammars as CFG with pushdown storage [19]. CFTG with storage will be recalled in Section 4.1. Here, we use grammars with storage for CFG instead of CFTG. Since the required changes to the definitions in Section 4.1 are minimal, we refrain from giving a formal definition for CFG with pushdown storage (denoted by CF(PD(Γ))-grammar). Intuitively, each nonterminal in the derivation is associated with a pushdown containing symbols from the pushdown alphabet Γ. A rule may test for the pushdown and modify it at nonterminals in the RHS of the rule.

Definition 3.7.2 An *indexed grammar* M is a CF(PD(Γ))-grammar using a finite pushdown alphabet Γ.[1]

The *string language induced by an indexed grammar* M, denoted by $\mathcal{L}(M)$, is defined as the language of the extended CFG $\mathcal{G}(M)$ associated with M (similar to Definition 4.1.5).□

Example 3.7.3 Consider the indexed grammar M_6 where

- the set of nonterminals is $\{A, B\}$,
- the set of terminals is $\{a, b, \sharp\}$,
- the initial nonterminal is A
- the pushdown alphabet is $\Gamma = \{\gamma\}$, and
- M_6 uses the rules

$$A \to A(\text{push}(\gamma)) \mid B(\text{id}) \ , \qquad \begin{array}{l} B \to \textbf{if top} = \gamma \textbf{ then } aB(\text{pop})b \ , \text{ and} \\ B \to \textbf{if top} = \varepsilon \textbf{ then } \sharp \end{array}$$

The nonterminal A pushes a number of γ's onto the pushdown. Each such γ is then popped by the nonterminal B while a and b are synchronously generated to the left and right of B, respectively. The derivation

$$A(\varepsilon) \Rightarrow A(\gamma) \Rightarrow A(\gamma\gamma) \Rightarrow B(\gamma\gamma) \Rightarrow aB(\gamma)b \Rightarrow aaB(\varepsilon)bb \Rightarrow aa\sharp bb$$

exemplifies how the pushdown is used in a derivation. It can be seen that the string language induced by M_6 is $\mathcal{L}(M_6) = \{a^n\sharp b^n \mid n \in \mathbb{N}\}$. ○

[1] The original definition in [1, pp. 648f] requires the rules of an indexed grammar to be of a special form. This restriction is not necessary (cf. [19, p. 281, Rm. (v)], [17, p. 18]).

The following definition is based on [54, Def. 2.5].

Definition 3.7.4 An indexed grammar M is *weakly-self-embedding*[2], if there are $\gamma \in \Gamma$, $v \in \Gamma^+$, and arbitrary words s and t over terminals and nonterminals with associated pushdown configurations such that

- $A(\varepsilon) \Rightarrow^* sA(v)t$ or
- $A(\gamma) \Rightarrow^* sA(v\gamma)t$

is a derivation in $\mathcal{G}(M)$. □

It is decidable whether an indexed grammar M is weakly-self-embedding by constructing, for each nonterminal A and each pushdown symbol $\gamma \in \Gamma \cup \{\varepsilon\}$, an indexed grammar $M_{A,\gamma}$ such that $\mathcal{L}(M_{A,\gamma}) \neq \emptyset$ iff $A(\gamma) \Rightarrow^* sA(v\gamma)t$ holds for some $v \in \Gamma^+$ and arbitrary words s, t over terminals and nonterminals with associated pushdown configuration [54, Remark (2)]. Since emptiness of indexed grammars is decidable [1, Thm. 4.1], weakly-self-embedding for indexed grammars is decidable.

Example 3.7.5 The indexed grammar M_6 (cf. Example 3.7.3) is weakly-self-embedding, since A pushes arbitrarily many symbols onto the pushdown. Formally, $A(\varepsilon) \Rightarrow A(\gamma)$ is a derivation in $\mathcal{G}(M_6)$ and is a witness for M_6 being weakly-self-embedding. Note that M_6 is weakly-self-embedding, although $\mathcal{L}(M_6)$ is a context-free string language. ○

Intuitively, an indexed grammar is weakly-self-embedding if there is a cycle from a nonterminal A to another occurrence of A such that the pushdown associated with the nonterminal occurrences grows and has the same top-most symbol. Hence, if this is not the case, then there are only finitely many pushdown configurations reachable from the initial nonterminal. This finite information can be encoded into the nonterminals of a CFG.

Theorem 3.7.6 [54, Thm. 3.1] If an indexed grammar is non-weakly-self-embedding, then it induces a context-free string language.

Note that since any CFG can be seen as an indexed grammar with an empty pushdown alphabet (i.e., CF(PD($\emptyset$)))-grammars, each CFG is trivially an indexed grammar which is non-weakly-self-embedding.

Since the yield of a (outside-in) context-free tree language is an outside-in macro language [58, pp. 113] (cf. Section 3.8 for details) which in turn is an indexed language [20, Thm. 4.2.8], it seems worthwhile to compare weakly-self-embedding indexed grammars and self-embedding lnCFTG. In [20, Thm. 4.2.8] the equivalence between indexed grammars and outside-in macro grammars is shown. The constructions rely on normal forms and introduce copying and deletion. Thus, our definition of self-embedding for the special case of lnCFTG cannot be transferred directly. To give an intuition about the difference between the properties, we relate them on an informal level using the indexed grammar M_6 and the lnCFTG G_{15} which is defined in the following example.

[2] In [54] this property is called *self-embedding*. Since our property of self-embedding lnCFTG is different (in the sense discussed later in this section), we chose another name here.

Example 3.7.7 We let $G_{15} = (N, \Delta, A_0, R)$ be a lnCFTG where $N = \{{A_0}^{(0)}, A^{(1)}, B^{(1)}\}$, $\Delta = \{\sigma^{(3)}, \alpha^{(0)}, \beta^{(0)}, \sharp^{(0)}\}$, and R contains the rules

$$A_0 \to A(\sharp) \ , \qquad A(x) \to A(B(x)) \mid x \ , \text{ and} \qquad B(x) \to \sigma(\alpha, x, \beta) \ .$$

It can be seen that $\text{yield}(\mathcal{L}(G_{15})) = \{\alpha^n \sharp \beta^n \mid n \in \mathbb{N}\}$. The nonterminal A accumulates a tower of B's in its argument position. Each B is then derived into a synchronized α and β using σ as concatenation symbol. The σ's are removed by taking the yield. ○

Intuitively, M_6 (cf. Example 3.7.3) and G_{15} (cf. Example 3.7.7) have a similar derivation behavior. The indexed grammar M_6 accumulates a tower of γ's while G_{15} accumulates the same number of B's. Then these symbols are derived into the corresponding number of a's and b's, or α's and β's, respectively. Concerning the induced string languages, we have that $\mathcal{L}(M_6)$ is equal to $\text{yield}(\mathcal{L}(G_{15}))$ if each α is replaced by an a and each β by b. We note that $\mathcal{L}(M_6)$ is a context-free string language and $\mathcal{L}(G_{15})$ is a regular tree language. Due to our definition of self-embedding for lnCFTG, we can detect the regularity of $\mathcal{L}(G_{15})$, since G_{15} is a non-self-embedding lnCFTG. In contrast, the property of weakly-self-embedding for indexed grammars is not sufficient to detect that M_6 induces a context-free string language.

There is a conceptual difference between the property of weakly-self-embedding for indexed languages which applies to arbitrary indexed grammars and the property of self-embedding lnCFTG which only applies to linear nondeleting CFTGs. This motivates the definition of a weakly-self-embedding property for arbitrary CFTGs. Its idea is very similar to the idea of weakly-self-embedding indexed grammars.

Definition 3.7.8 A CFTG $G = (N, \Delta, A_0, R)$ is called *weakly-self-embedding*, if there are $k \in \mathbb{N}$, $A \in N^{(k)}$, $i \in [k]$, $\xi \in \mathrm{C}_{N \cup \Delta \cup X_k}(\{z\})$, and $\xi_{1..k} \in \mathrm{T}_{N \cup \Delta}(X_k)$ such that

- $A(\overline{x}) \Rightarrow^* \xi[A(\xi_{1..k})]$,
- ξ_i contains x_i, and
- $\xi_i \neq x_i$. □

Intuitively, a CFTG is weakly-self-embedding if an unbounded number of symbols can be generated below a repeated occurrence of a nonterminal. Note that weakly-self-embedding is a semantic property. Checking for generation below a nonterminal corresponds to determining whether $\text{topRank}(G) = 0$.

Lemma 3.7.9 For a CFTG G, we have that G is weakly-self-embedding iff there is a top-recursive SCC in $\text{pg}(G)$.

Proof. The proof is very similar to the proof of Theorem 3.1.7, but only considers one argument position (instead of two in parallel) and thus operates on the position graph instead of the position pair graph. Let G be weakly-self-embedding. Then there are $\xi \in \mathrm{C}_{N \cup \Delta \cup X_k}(z)$ and $\xi_{1..k} \in \mathrm{T}_{N \cup \Delta}(X_k)$ such that $A(\overline{x}) \Rightarrow^* \xi[A(\xi_{1..k})]$, x_i occurs in ξ_i, and $\xi_i \neq x_i$. This derivation gives rise to a cycle in $\text{pg}(G)$ from (A, i) to (A, i) which is top-recursive, since $\xi_i \neq x_i$.

It can be seen that a generating cycle from (A, i) to (A, i) in a top-recursive SCC of $\text{pg}(G)$ has to be induced by a derivation as specified in Definition 3.7.8. ■

The following two observations follow directly from the definitions of the properties of self-embedding CFTG and weakly-self-embedding CFTG.

Observation 3.7.10 Each self-embedding CFTG is also weakly-self-embedding, since each self-embedding CFTG has at least one top-recursive SCC (cf. Lemma 3.7.9). However, there are non-self-embedding CFTG which are weakly-self-embedding, e.g., the lnCFTG G_{15} (cf. Example 3.7.7).

Observation 3.7.11 Each RTG H is non-weakly-self-embedding.

We will now present the main theorem of this section.

Theorem 3.7.12 For each CFTG G which is non-weakly-self-embedding, we can construct a RTG H such that $\mathcal{L}(G) = \mathcal{L}(H)$, i.e., $\mathcal{L}(G)$ is a regular tree language.

Proof. By Lemma 3.7.9 we have that $\mathrm{pg}(G)$ does not contain a top-recursive SCC. Thus, we can use the exact same proof as for Lemma 3.3.9, since the proof of Lemma 3.3.9 does not make use of G being linear nondeleting and solely relies on $\mathrm{topRank}(G) = 0$. Thus, it applies to G in the setting of this lemma and we can construct the desired RTG H which is equivalent to G. ∎

3.8 Non-Self-Embedding MACs

The yield of a context free tree language (using outside-in derivation mode) is an outside-in macro language (cf. [58, p. 113] and [15, Thm. 7.17]). Thus, it seems natural that the results from Sections 3.3 and 3.7 can be transferred to linear nondeleting macro grammars and arbitrary macro grammars (MACs), respectively. In this section, we recall MACs and show that non-self-embedding linear MACs and non-weakly-self-embedding MACs induce context-free string languages.

Macro grammars were first introduced by Fischer [20]. They use nested terms [20, Def. 2.2.1] as right-hand sides. We recall the definition here.

Definition 3.8.1 Let Σ be an alphabet, N be a ranked alphabet, and U be a set. The set of *nested terms over Σ and N* using variables from U is defined as the smallest set T satisfying that

(i) $\Sigma \cup U \subseteq T$,

(ii) for each $k \in \mathbb{N}$ and $t_1, \ldots, t_k \in T$, we have $t_1 \ldots t_k \in T$, and

(iii) for each $k \in \mathbb{N}$, $t_1, \ldots, t_k \in T$, and $A \in N^{(k)}$, we have $A(t_1, \ldots, t_k) \in T$.

If Σ, N, and U are understood from the context, then we call a nested term over Σ and N using variables from U just a *nested term*.

The *width of a nested term t*, denoted by $|t|$ is inductively defined by (i) $|t| = 1$ if $t \in \Sigma \cup U$, (ii) $|t| = |t_1| + \ldots + |t_k|$ if $t = t_1 \ldots t_k$ for nested terms $t_{1..k}$, and (iii) $|t| = 1$ if $t = A(t_1, \ldots, t_k)$ for some $A \in N$ and nested terms $t_{1..k}$.

A *nested context* is a nested term using variables from U such that it contains each element in U exactly once. If t is a nested term using variables from a finite set U, $k = |U|$, $U = \{u_1, \ldots, u_k\}$, and $t_1, \ldots, t_k$ are nested terms, then $t[t_1, \ldots, t_k]$ denotes the nested term obtained by simultaneously replacing, for each $i \in [k]$, each occurrence of u_i in t by t_i. □

As a preliminary tool, we extend the notion of the yield of a tree in such a way that we keep all (possibly ranked) symbols in a set Σ.

Definition 3.8.2 Let Δ be a ranked alphabet, U be a finite set, $\xi \in \mathrm{T}_\Delta(U)$, and $\Sigma \subseteq \Delta \cup U$ where $\Delta \cap U = \emptyset$, we consider each element in U as a symbol of rank 0, and Σ as a ranked alphabet inheriting the ranks of Δ and U. The *Σ-yield of ξ*, denoted by $\mathrm{yield}_\Sigma(\xi)$, is a nested term over $\Sigma \setminus U$ using variables from $U \cap \Sigma$ defined by induction over the structure of ξ as

$$\mathrm{yield}_\Sigma(\xi) = \begin{cases} u & \text{if } \xi = u \text{ for some } u \in (\Sigma \cap U), \\ \varepsilon & \text{if } \xi = u \text{ for some } u \in U \setminus \Sigma, \\ \delta(\mathrm{yield}_\Sigma(\xi_1), \ldots, \mathrm{yield}_\Sigma(\xi_k)) & \text{if } \xi = \delta(\xi_{1..k}) \text{ for some } \delta \in \Sigma^{(k)} \cap \Delta \\ & \quad \text{and } \xi_{1..k} \in \mathrm{T}_\Delta(U), \text{ and} \\ \mathrm{yield}_\Sigma(\xi_1) \ldots \mathrm{yield}_\Sigma(\xi_k) & \text{if } \xi = \delta(\xi_{1..k}) \text{ for some } \delta \in \Delta^{(k)} \setminus \Sigma. \end{cases}$$ □

Note that if $\Sigma = \Delta^{(0)} \cup U$, then, for each $\xi \in \mathrm{T}_\Delta(U)$, we have $\mathrm{yield}_\Sigma(\xi) = \mathrm{yield}(\xi)$.

Now we recall the definition of a macro grammar [20, Def. 2.2.12].

Definition 3.8.3 An *outside-in macro grammar* (MAC) is a tuple $M = (N, \Sigma, A_0, R)$ where

- N is a ranked alphabet (*nonterminals*),
- Σ is an alphabet (*terminals*),
- $A_0 \in N^{(0)}$ (*initial nonterminal*), and
- R is a finite set of *rules* of the form $A(x_{1..k}) \to t$ where $k \in \mathbb{N}$, $A \in N^{(k)}$, and t is a nested term over Σ and N using variables X_k.

Let $M = (N, \Sigma, A_0, R)$ be a MAC. We say that M is a *linear* MAC (lMAC) if, for each rule $A(x_{1..k}) \to t$ in R, we have that, for each $i \in [k]$, the variable x_i occurs at most once in t. We say that M is a *nondeleting* MAC if, for each rule $A(x_{1..k}) \to t$ in R, we have that, for each $i \in [k]$, the variable x_i occurs at least once in t. If M is a linear and nondeleting MAC, then we say it is a linear nondeleting MAC (lnMAC).

The *outside-in derivation relation* of M, denoted by $\Rightarrow$, is defined in the following similarly to the definition of the derivation relation for CFTG (cf. Definition 2.2.2). For nested terms t_1, t_2, we have $t_1 \Rightarrow_r t_2$ if there are a rule $r\colon A(x_{1..k}) \to t$ in R, a nested context s using the variable $\{z\}$, and nested terms $s_1, \ldots, s_k$ such that

- $t_1 = s[A(s_1, \ldots, s_k)]$,
- s does not contain a nonterminal occurrence that is z-dominating, and
- $t_2 = s[t[s_1, \ldots, s_k]]$.

We let $\Rightarrow^*$ denote the reflexive closure of $\Rightarrow$.

The string language induced by M, denoted by $\mathcal{L}(M)$, is defined as

$$\mathcal{L}(M) = \{s \in \Sigma^* | A_0 \Rightarrow^* s\} \ .$$ □

We will explicitly prove the following lemma here by giving the corresponding constructions, since we rely on them later. The idea is taken from [58, p. 113] and formalized in [15, Thm. 7.17].

Lemma 3.8.4 The following two statements hold.

(i) For each CFTG G, we can construct a MAC M such that $\mathcal{L}(M) = \text{yield}(\mathcal{L}(G))$.

(ii) Let Σ be an alphabet. For each MAC M using terminals in Σ, we can construct a CFTG G such that $\text{yield}_\Sigma(\mathcal{L}(G)) = \mathcal{L}(M)$.

Furthermore, both constructions preserve the properties of each formalism being linear and being nondeleting.

Proof. (i): We let $G = (N, \Delta, A_0, R)$ be a CFTG and $\Sigma = N \cup \Delta^{(0)} \cup X$ where each variable $x \in X$ is considered to be nullary. Then we construct the MAC $M = (N, \Delta^{(0)}, A_0, R')$ by letting R' contain the rule $A(\overline{x}) \to \text{yield}_\Sigma(\xi)$ for each rule $A(\overline{x}) \to \xi$ in R. Note that the construction preserves the properties of being linear and being nondeleting. It can be shown that $A_0 \Rightarrow^* \xi$ is a derivation in G iff $A_0 \Rightarrow^* \text{yield}_\Sigma(\xi)$ is a derivation in M. Hence, $\mathcal{L}(M) = \text{yield}(\mathcal{L}(G))$.

(ii): We let $M = (N, \Sigma, A_0, R)$ be a MAC. Furthermore, we let m be the maximal width of any nested term in the right-hand side of a rule in R. Then we construct a CFTG $G = (N, \Delta, A_0, R')$ where $\Delta = \Sigma \cup \Delta' \cup \{e\}$ where $\Delta' = \{\delta_i \mid i \in [m]\}$, for each $i \in [m]$, $\text{rk}_\Delta(\delta_i) = i$, $\Delta' \cap \Sigma = \emptyset$, $e \notin \Sigma \cup \Delta'$, $\text{rk}_\Delta(e) = 0$, and R' is defined in the following. If $A(\overline{x}) \to t$ is in R, then we let $A(\overline{x}) \to f(t)$ be in R' where f is the recursive function

$$f(t) = \begin{cases} A(f(t_1), \ldots, f(t_k)) & \text{if } t = A(t_1, \ldots, t_k) \text{ for some } A \in N \text{ and nested terms } t_{1..k}, \\ \delta_k(f(t_1), \ldots, f(t_k)) & \text{if } t = t_1 \ldots t_k \text{ and } k \geq 2 \text{ for some nested terms } t_{1..k}, \\ t & \text{if } t \in \Sigma \cup X_{\text{rk}_N(A)}, \text{ and} \\ e & \text{if } t = \varepsilon. \end{cases}$$

Note that the construction preserves the properties of being linear and being nondeleting. As in the previous case, it can be shown that $A_0 \Rightarrow^* \xi$ is a derivation in G iff $A_0 \Rightarrow^* \text{yield}_{\Sigma \cup N}(\xi)$ is a derivation in M. Hence, $\mathcal{L}(M) = \text{yield}_\Sigma(\mathcal{L}(G))$. ∎

Observation 3.8.5 Note that a MAC where each nonterminal has rank 0 is a CFG. Thus, Lemma 3.8.4 (i) applied to a RTG yields a CFG.

Now we define self-embedding for MAC very similar to Definition 3.1.1.

Definition 3.8.6 We let $M = (N, \Sigma, A_0, R)$ be a MAC. Then M is *self-embedding* if there are $k \in \mathbb{N}_+$, $A \in N^{(k)}$, nested terms $t_{1..k}$, and a nested context t using the variable z such that $A(\overline{x}) \Rightarrow^* t[A(t_{1..k})]$ and at least one of the following two properties holds:

(1) $t \neq z$, and there is an $i \in [k]$ such that t_i contains x_i and $t_i \neq x_i$.

(2) There are $i, j \in [k]$ with $i \neq j$ such that t_i contains x_i and $t_i \neq x_i$, and t_j contains x_j and $t_j \neq x_j$. □

Definition 3.8.6 and Definition 3.1.1 are very similar on a syntactic level. Hence, we observe the following.

Observation 3.8.7 We let M be a MAC and G be the CFTG G constructed for M as in the proof of Lemma 3.8.4 (ii). If M is self-embedding, then G is self-embedding. Furthermore, if M is non-self-embedding, then G is also non-self-embedding.

Note that Observation 3.8.7 only holds in this direction as demonstrated in the following example.

Example 3.8.8 We let $G_{16} = (N, \Delta, A_0, R)$ be a lnCFTG where $N = \{A_0, A\}$, $\Delta = \{\delta^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$, and R contains the rules

$$A_0 \to \begin{array}{c} A \\ / \ \backslash \\ \alpha \quad \alpha \end{array} \qquad\qquad A(x_1, x_2) \to \begin{array}{c} A \\ / \ \backslash \\ \gamma \quad \gamma \\ | \quad | \\ x_1 \quad x_2 \end{array} \;\Big|\; \begin{array}{c} \delta \\ / \ \backslash \\ x_1 \quad x_2 \end{array} \quad .$$

By inspecting the rules, we can see that $\mathcal{L}(G_{16}) = \{\delta(\gamma^n(\alpha), \gamma^n(\alpha)) \mid n \in \mathbb{N}\}$.

It is clear that G_{16} is self-embedding since γ's are synchronously generated in the first and second argument position of A. However, if we apply the construction of the proof of Lemma 3.8.4 (i) to G_{16}, we obtain the lnMAC $M_7 = (N, \Delta^{(0)}, A_0, R')$ where R' contains the rules

$$A_0 \to A(\alpha, \alpha) \qquad\qquad A(x_1, x_2) \to A(x_1, x_2) \mid x_1 x_2 \ .$$

We have $\mathcal{L}(M_7) = \text{yield}(\mathcal{L}(G_{16})) = \{\alpha\alpha\}$ and note that M_7 is non-self-embedding. ○

Let M be a non-self-embedding lMAC and G be the lCFTG such that $\mathcal{L}(M) = \text{yield}(\mathcal{L}(G))$ (cf. Lemma 3.8.4 (ii)). By Observation 3.8.7, G is non-self-embedding. Using Theorem 3.6.2, we can construct a RTG H such that $\mathcal{L}(G) = \mathcal{L}(H)$. By Observation 3.8.5, we can transform H into a CFG M' such that $\mathcal{L}(M') = \text{yield}(\mathcal{L}(H))$. Hence, the following corollary holds.

Corollary 3.8.9 For each non-self-embedding lMAC M, we can construct a CFG M' such that $\mathcal{L}(M) = \mathcal{L}(M')$, i.e., M induces a context-free string language.

Corollary 3.8.9 makes use of Theorem 3.6.2 and the fact that we only consider linear MACs. We will now obtain a property for arbitrary MACs similar to non-weakly-self-embedding CFTG. We can straightforwardly give the definition of a weakly-self-embedding MAC very similar to Definition 3.7.8.

Definition 3.8.10 A MAC $M = (N, \Sigma, A_0, R)$ is *weakly-self-embedding* if there are $A \in N$, a nested term t, and nested terms $t_{1..k}$ such that $A(\overline{x}) \Rightarrow^* t$ and t contains an occurrence of $A(t_{1..k})$ such that x_i occurs in t_i and $t_i \neq x_i$. □

The definition of weakly-self-embedding CFTG and weakly-self-embedding MAC are very similar. The relation is formalized in the following observation.

Observation 3.8.11 Let M be a MAC and G be the CFTG constructed for M using the construction in the proof of Lemma 3.8.4 (ii). If M is weakly-self-embedding, then G is weakly-self-embedding. Furthermore, if M is non-weakly-self-embedding, then G is non-weakly-self-embedding.

Note that reverse direction does not hold, i.e., there is a weakly-self-embedding lnCFTG G such that the MAC M obtained by applying the construction in the proof of Lemma 3.8.4 is non-weakly-self-embedding. As an illustration recall the lnCFTG G_{16} and the lnMAC M_7 from Example 3.8.8. Clearly, G_{16} is weakly-self-embedding whereas M_7 is non-weakly-self-embedding.

It is easy to see that each lMAC that is self-embedding is also weakly-self-embedding. However, the reverse is not true, i.e., there is a lMAC that is weakly-self-embedding but non-self-embedding. We illustrate this using the following example.

Example 3.8.12 We define the lnMAC $M_8 = (N, \Sigma, A_0, R)$ where $N = \{A_0, A^{(1)}, B^{(1)}\}$, $\Sigma = \{\alpha, \beta, \sharp\}$, and R contains the rules

$$A_0 \to A(\sharp) \ , \qquad A(x_1) \to A(B(x_1)) \ , \text{ and } \qquad B(x_1) \to \alpha x_1 \beta \ .$$

Note that M_8 and G_{15} are connected via Lemma 3.8.4. ○

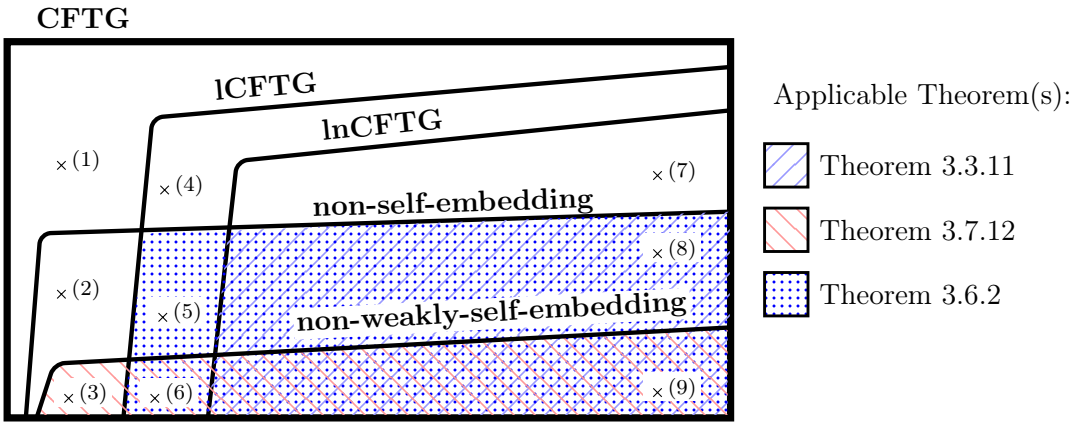
Using Observation 3.8.11, the following corollary follows from Theorem 3.7.12 and Observation 3.8.5.

Corollary 3.8.13 For each non-weakly-self-embedding MAC M, we can construct a CFG M' such that $\mathcal{L}(M) = \mathcal{L}(M')$, i.e., the language induced by M is context-free.

3.9 Overview

We visualize all results concerning CFTGs that induce regular tree languages in Figure 3.14(a). Context-free tree grammars belonging into the hatched areas induce a regular tree language. The pattern denotes which theorems prove the regularity of the induced language. For each CFTG G which is not in the hatched areas, our results cannot determine whether $\mathcal{L}(G)$ is regular or not. Furthermore, for each area in Figure 3.14(a), we give a very easy example in Figure 3.14(b). We use nonterminals ${A_0}^{(0)}$, $A^{(2)}$, and $B^{(1)}$, and terminals $\alpha^{(0)}$, $\gamma^{(1)}$, $\sigma^{(1)}$, $\delta^{(2)}$, and $\kappa^{(2)}$.

Figure 3.14(a) can also be seen as an overview for MACs. We consider non-weakly-self-embedding MACs, lMACs, lnMACs, non-self-embedding lMACs, and non-self-embedding lnMACs instead of the corresponding CFTG areas. In this setting, Theorems 3.3.11 and 3.6.2 are both replaced by Corollary 3.8.9 and Theorem 3.7.12 is replaced by Corollary 3.8.13.



(a) Overview of self-embedding relations for CFTG.

(1) $A_0 \to \begin{smallmatrix} B \\ | \\ \alpha \end{smallmatrix} \quad B(x_1) \to \begin{smallmatrix} \sigma \\ | \\ B \\ | \\ \gamma \\ | \\ x_1 \end{smallmatrix} \Bigg| \begin{smallmatrix} \kappa \\ / \ \backslash \\ x_1 \ x_1 \end{smallmatrix}$

(2) $A_0 \to \begin{smallmatrix} B \\ | \\ \alpha \end{smallmatrix} \quad B(x_1) \to \begin{smallmatrix} B \\ | \\ \gamma \\ | \\ x_1 \end{smallmatrix} \Bigg| \begin{smallmatrix} \kappa \\ / \ \backslash \\ x_1 \ x_1 \end{smallmatrix}$

(3) $A_0 \to \begin{smallmatrix} B \\ | \\ \alpha \end{smallmatrix} \quad B(x_1) \to \begin{smallmatrix} \delta \\ / \ \backslash \\ B \ B \\ | \ | \\ x_1 \ x_1 \end{smallmatrix} \Bigg| \begin{smallmatrix} \kappa \\ / \ \backslash \\ x_1 \ x_1 \end{smallmatrix}$

(4) $A_0 \to \begin{smallmatrix} A \\ / \ \backslash \\ \alpha \ \alpha \end{smallmatrix}$

$A(x_1, x_2) \to \begin{smallmatrix} \sigma \\ | \\ A \\ / \ \backslash \\ \gamma \ \alpha \\ | \\ x_1 \end{smallmatrix} \Bigg| \begin{smallmatrix} \sigma \\ | \\ x_1 \end{smallmatrix}$

(5) $A_0 \to \begin{smallmatrix} A \\ / \ \backslash \\ \alpha \ \alpha \end{smallmatrix}$

$A(x_1, x_2) \to \begin{smallmatrix} A \\ / \ \backslash \\ \gamma \ \alpha \\ | \\ x_1 \end{smallmatrix} \Bigg| \begin{smallmatrix} \sigma \\ | \\ x_1 \end{smallmatrix}$

(6) $A_0 \to \begin{smallmatrix} A \\ / \ \backslash \\ \alpha \ \alpha \end{smallmatrix}$

$A(x_1, x_2) \to \begin{smallmatrix} \sigma \\ | \\ A \\ / \ \backslash \\ x_1 \ \alpha \end{smallmatrix} \Bigg| \begin{smallmatrix} \sigma \\ | \\ x_1 \end{smallmatrix}$

(7) $A_0 \to \begin{smallmatrix} A \\ / \ \backslash \\ \alpha \ \alpha \end{smallmatrix}$

$A(x_1, x_2) \to \begin{smallmatrix} \sigma \\ | \\ A \\ / \ \backslash \\ \gamma \ \gamma \\ | \ | \\ x_1 \ x_2 \end{smallmatrix} \Bigg| \begin{smallmatrix} \kappa \\ / \ \backslash \\ x_1 \ x_2 \end{smallmatrix}$

(8) $A_0 \to \begin{smallmatrix} B \\ | \\ \alpha \end{smallmatrix} \quad B(x_1) \to \begin{smallmatrix} B \\ | \\ \gamma \\ | \\ x_1 \end{smallmatrix} \Bigg| \begin{smallmatrix} \sigma \\ | \\ x_1 \end{smallmatrix}$

(9) $A_0 \to \begin{smallmatrix} A \\ / \ \backslash \\ \alpha \ \alpha \end{smallmatrix}$

$A(x_1, x_2) \to \begin{smallmatrix} \sigma \\ | \\ A \\ / \ \backslash \\ x_1 \ x_2 \end{smallmatrix} \Bigg| \begin{smallmatrix} \kappa \\ / \ \backslash \\ x_1 \ x_2 \end{smallmatrix}$

(b) Examples for each area of Figure 3.14(a).

Figure 3.14: An overview over the classes of CFTG.

3.10 Remarks on Non-Self-Embedding lnCFTGs

In this section, we present two results related to non-self-embedding lnCFTG. First, the succinctness of non-self-embedding lnCFTGs is compared to RTGs which induce the same tree language. It turns out that non-self-embedding lnCFTGs can express certain tree languages exponentially smaller than RTGs. Second, we instantiate Theorem 3.3.11 in the light of coregular grammars as introduced in [2].

Non-Self-Embedding lnCFTGs may Represent Regular Tree Languages More Succinctly than RTG

In this section, we outline that there is a regular tree language L that is induced by a non-self-embedding lnCFTG G such that each RTG H inducing L is exponentially larger than G. The lnCFTG G derives trees using a fixed amount of nondeterministic choices as follows. Each choice synchronously generates two copies of either a γ or a δ in two separate argument positions of a nonterminal. We use different nonterminals to ensure that each choice is only taken once and there is no recursion. This very limited nondeterminism can be encoded into the nonterminals and rules of a RTG. However, since the generation happens in different argument positions, this requires exponentially more nonterminals or rules and the size of the RTG grows. We will actually present an infinite family of regular tree languages induced by non-self-embedding lnCFTG that have the succinctness property explained before.

First, we formally define the size of a CFTG and then present the key lemma showing the exponential blow-up.

Definition 3.10.1 Let $G = (N, \Delta, A_0, R)$ be a CFTG. Then the *size of* G, denoted by $|G|$ is defined as $|G| = |N| + |R|$. □

The proof of the following lemma contains the formal definition of the lnCFTGs outlined at the beginning of this section. They illustrate the blow-up. The first part of the proof may thus be considered as an example.

Lemma 3.10.2 There is a family of non-self-embedding lnCFTG $(G_n \mid n \in \mathbb{N}_+)$ such that for each $n \in \mathbb{N}_+$ and each RTG H with $\mathcal{L}(H) = \mathcal{L}(G_n)$, we have that $|H| > 2^{|G_n|}$.

Proof. We let $\Delta = \{\kappa^{(2)}, \gamma^{(1)}, \delta^{(1)}, \alpha^{(0)}\}$. The family of lnCFTG $(G_n \mid n \in \mathbb{N}_+)$ is defined as follows. For each $n \in \mathbb{N}_+$, we define $G_n = (N_n, \Delta, A_0, R_n)$ where $N_n = \{A_0{}^{(0)}\} \cup \{A_i^{(2)} \mid i \in [n]\}$ and the rules in R_n are depicted in Figure 3.15(a). For each $i \in \mathbb{N}$, the nonterminal A_i offers the choice between γ or δ. The chosen terminal is then synchronously produced in *both* argument positions of the next nonterminal A_{i+1} (or κ in the case of $i = n$).

We fix an $n \in \mathbb{N}_+$ and note $|N_n| = n + 1$, $|R_n| = 2n + 1$, and thus, $|G_n| = 3n + 2$. Furthermore, we let $\mu\colon [n] \to \{\gamma, \delta\}$ be a function that represents the choice we took in each nonterminal A_i. Figure 3.15(b) depicts the general structure of a tree in $\mathcal{L}(G_n)$ for μ.

By a close inspection of $\mathcal{L}(G_n)$, we can see that $|\mathcal{L}(G_n)| = 2^n$ and that $\mathcal{L}(G)$ has exactly 2^n Nerode equivalence classes[3]. Thus, the minimal deterministic finite tree automaton

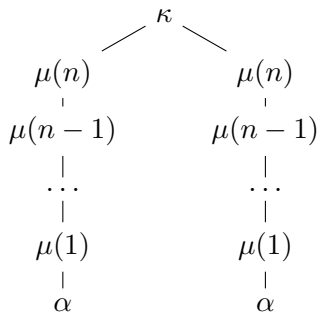
[3]The Nerode equivalence describes two trees $\xi_1, \xi_2 \in \mathrm{T}_\Delta$ as equivalent if, for each context $\zeta \in \mathrm{C}_\Delta(\{z\})$, we

$$A_0 \to A_1(\alpha, \alpha),$$

$$A_i(x_1, x_2) \to \begin{array}{c|c} A_{i+1} & A_{i+1} \\ / \ \backslash & / \ \backslash \\ \gamma \quad \gamma & \delta \quad \delta \\ | \quad | & | \quad | \\ x_1 \quad x_2 & x_1 \quad x_2 \end{array} \quad (i \in [n-1]),$$

$$A_n(x_1, x_2) \to \begin{array}{c|c} \kappa & \kappa \\ / \ \backslash & / \ \backslash \\ \gamma \quad \gamma & \delta \quad \delta \\ | \quad | & | \quad | \\ x_1 \quad x_2 & x_1 \quad x_2 \end{array} .$$

(a) Rules of the lnCFTG G_n.



(b) Abstract shape of a tree in $\mathcal{L}(G_n)$.

Figure 3.15: A lnCFTG showing the succinctness of non-self-embedding lnCFTG.

recognizing this tree language has at least 2^n states [25, Theorem 2.7.1]. We note that this result only yields a minimal size for *deterministic* finite tree automata.

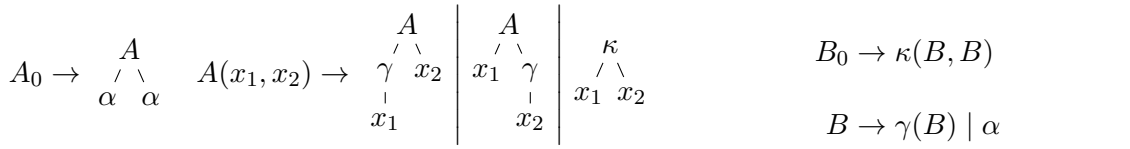
Now we show that each RTG H which is equivalent to G_n has at least size $2^n + 1$. For this, we first show a RTG H_{en} such that $\mathcal{L}(G_n) = \mathcal{L}(H_{\text{en}})$ and $|H_{\text{en}}| = 2^{n+1}$, and then show that there cannot be a smaller one. Since $\mathcal{L}(G_n)$ is finite, it can be enumerated by a RTG H_{en} with 2^n rules. Each rule produces exactly one terminal tree. Thus, using the initial nonterminal as only nonterminal, we have $|H_{\text{en}}| = 2^n + 1$. Now we assume that there is a RTG $H = (N_H, \Delta, B_0, R_H)$ such that $\mathcal{L}(G_n) = \mathcal{L}(H)$ and $|H| \leq 2^n$. Furthermore, we assume that H is the smallest such grammar and thus, each nonterminal and each rule are useful. It can be seen that H cannot enumerate all trees from $\mathcal{L}(G_n)$ in individual rules similar to H_{en}, since then $|H| > 2^n$.

We prove that the B_0-rules of H have the form $B_0 \to \kappa(\xi_1, \xi_2)$ for some $\xi_1, \xi_2 \in \mathrm{T}_{N_H \cup \Delta}$. For this, we assume that there is a rule $B_0 \to B$ for some $B \in N_H$. Then, for each $B' \in N_H$ such that $B_0 \Rightarrow^* B' \Rightarrow \kappa(\xi_1, \xi_2)$ for some $\xi_1, \xi_2 \in \mathrm{T}_{N_H \cup \Delta}$, we have that B' cannot occur in derivations starting from ξ_1 and ξ_2. Hence, we may omit all those nonterminals B' and their rules and directly add the rules $B_0 \to \kappa(\xi_1, \xi_2)$. The obtained grammar is equivalent to H, but has a smaller size than H. This contradicts the minimality of H and thus, each B_0-rule has the form $B_0 \to \kappa(\xi_1, \xi_2)$.

Since $|H| \leq 2^n$, there is a rule $B_0 \to \kappa(\xi_1, \xi_2)$ with $\xi_1, \xi_2 \in \mathrm{T}_{N_H \cup \Delta}$ such that we have $|\mathcal{L}(H, \kappa(\xi_1, \xi_2))| > 1$. We assume that $|\mathcal{L}(H, \xi_1)| > 1$. We consider the derivation $B_0 \Rightarrow \kappa(\xi_1, \xi_2) \Rightarrow^* \kappa(\xi_1, \xi_2')$ where $\xi_2' \in T_\Delta$. The tree ξ_2' uniquely determines the tree that is derived from ξ_1. This contradicts $|\mathcal{L}(H, \xi_1)| > 1$. Hence, we have $|H| > 2^n$ for each RTG H that is equivalent to G_n. ∎

We note that the non-self-embedding lnCFTGs shown in the proof of Lemma 3.10.2 can be extended to induce infinite tree languages or to induce different tree shapes. The blowup explained in this section stems from the synchronous nondeterministic choice.

have $\zeta[\xi_1] \in \mathcal{L}(G)$ iff $\zeta[\xi_2] \in \mathcal{L}(G)$. In other words, ξ_1 and ξ_2 are indistinguishable in terms of inclusion in $\mathcal{L}(G)$. For a detailed definition we refer to, e.g., [25, p. 86].



(a) The coregular lnCFTG G_{17}.

(b) A RTG equivalent to G_{17}.

Figure 3.16: A coregular lnCFTG that induces a regular tree language.

Non-self-embedding coregular lnCFTG

When considering regularity of the languages induced by lnCFTGs, another restriction of CFTGs comes to mind, viz. coregular grammars. In a coregular CFTG a nonterminal may only occur at the root of the right-hand side of a rule. As the name coregular CFTG might suggest, this restriction might be related to non-self-embedding lnCFTG. We will investigate the relationship between coregular lnCFTG and non-self-embedding lnCFTG. For this, we recall the notion of coregular CFTG from [2, p. 21].

Definition 3.10.3 A CFTG $G = (N, \Delta, A_0, R)$ is *coregular* if, for each rule $r \in R$, we have $\mathrm{pos}_N(\mathrm{rhs}(r)) \subseteq \{\varepsilon\}$. □

In this chapter, we focus on linear nondeleting CFTG. We apply the same restrictions to coregular CFTG and obtain coregular lnCFTG. Since each coregular lnCFTG is also a lnCFTG, we can apply Theorem 3.3.11 and immediately obtain the following corollary.

Corollary 3.10.4 Each non-self-embedding coregular lnCFTG induces a regular tree language.

A natural question is whether for coregular lnCFTG the property of being non-self-embedding is a necessary criterion for the regularity of the induced tree language. The following lemma answers this question negatively.

Lemma 3.10.5 There is a self-embedding coregular lnCFTG that induces a regular tree language.

Proof. Consider the coregular lnCFTG G_{17} given in Figure 3.16(a). By Theorem 3.1.7 it can be verified that G_{17} is self-embedding, since it fulfills Property (2) of self-embedding. However, $\mathcal{L}(G_{17})$ is regular. An equivalent RTG can be found in Figure 3.16(b). ■

4 Approximation of Arbitrary CFTGs

In Chapter 3 we have shown that there are tree languages induced by CFTGs that can be characterized by RTGs. However, it is clear that there are context-free tree languages where such a characterization is not possible. Furthermore, the potential exponential blowup similar to the phenomenon described in Lemma 3.10.2 may be undesired. Since RTGs can be parsed efficiently, it is worthwhile to investigate how the tree languages induced by arbitrary CFTGs can be expressed by RTG. This can be achieved by means of an approximation, i.e., some precision on the language model is waived for a better parsing time.

Approximation of languages includes in particular subset approximation (generating only parts of the language) and superset approximation (generating all elements of the language and potentially more). We focus on superset approximation and call it just *approximation*. By definition, T_Δ is an approximation for each tree language. This motivates to talk about the quality of an approximation. We use the subset relation as partial order on approximations as follows. If $L \subseteq \mathrm{T}_\Delta$ and $L_1, L_2 \supseteq L$ are approximations of L, then we say that L_1 is *better* than L_2 if $L \subseteq L_1 \subseteq L_2$. In this sense, an approximation improves compared to another approximation, if less trees are wrongly generated. We note that there are incomparable approximations regarding our quality measure.

In the literature, there are similar investigations that show how to express the language induced by a CFG using a finite state formalism, e.g., a REG. The research is closely linked to parsing of CFG and the results include LR-parsing [3, 55], recursive transition networks [48], and the direct construction of an approximating REG [28, 47]. An overview of these results can be found in [47, Sec. 6].

In this chapter we present a method to approximate the tree language induced by an arbitrary CFTG by a RTG inspired by a similar result for the string case [55]. For this, we utilize the concept of a pushdown storage as introduced by [17] and the known characterization that each context-free tree language can be generated by a RTG using a pushdown storage [29, 17]. We describe how to characterize the language induced by a CFTG G by a RTG H using a pushdown in the following. The RHS of a rule r in G is traversed from top to bottom. We distinguish three different cases depending on the label of the current position. A terminal symbol is just output by the RTG and the traversal continues at the arguments of that terminal. If a nonterminal B occurs at the current position w in $\mathrm{rhs}(r)$, then we store r and w onto the pushdown as a return address and continue the traversal at the root of a nondeterministically chosen rule with LHS-nonterminal B. Lastly, if a variable x_i occurs, then we pop a return address (r', w') from the top of the pushdown and continue the derivation process in $\mathrm{rhs}(r')$ at position $w'i$.

This characterization can be turned into an approximation by restricting the number of pushdown configurations that can occur in derivations. We obtain a (superset) approximation by forgetting the return addresses at the bottom of the stack. Then, whenever an empty stack occurs and a variable is met, we know that a symbol was forgotten. In this

case, we nondeterministically choose one of the possible return addresses.

As an example, we consider the CFTG $G_{18} = (\{A_0{}^{(0)}, A^{(2)}\}, \Delta, A_0, R)$ where $\Delta = \{\kappa^{(3)}, \delta^{(1)}, \gamma^{(1)}, \alpha^{(0)}\}$ and R contains the rules

$$A_0 \to \begin{array}{c} A \\ / \ \backslash \\ \alpha \quad \alpha \end{array} \qquad \text{and} \qquad A(x_1, x_2) \to \begin{array}{c} A \\ / \ \backslash \\ \gamma \quad \delta \\ | \quad | \\ x_1 \quad x_2 \end{array} \Bigg| \begin{array}{c} \kappa \\ / \ | \ \backslash \\ x_1 \ x_2 \ x_1 \end{array} .$$

We call the rules from left to right r_1, r_2, and r_3 and note that r_3 is copying. The derivation

$$A_0 \Rightarrow \begin{array}{c} A \\ / \ \backslash \\ \alpha \quad \alpha \end{array} \Rightarrow \begin{array}{c} A \\ / \ \backslash \\ \gamma \quad \delta \\ | \quad | \\ \alpha \quad \alpha \end{array} \Rightarrow \begin{array}{c} \kappa \\ / \ | \ \backslash \\ \gamma \quad \delta \quad \gamma \\ | \quad | \quad | \\ \alpha \quad \alpha \quad \alpha \end{array}$$

of G_{18} shows that γ's and δ's are synchronously generated. By generalizing this derivation, we can see that $\mathcal{L}(G_{18}) = \{\kappa(\gamma^j(\alpha), \delta^j(\alpha), \gamma^j(\alpha) \mid j \in \mathbb{N}\}$.

We will illustrate the rule traversal and show how to approximate the language induced by G_{18} by a RTG without a pushdown storage, i.e. the pushdown is restricted to contain only the empty configuration (not depicted here). Furthermore, we outline at which points a less restricted pushdown would be used to improve the approximation regarding the subset relation. We define the RTG $H_3 = (\{B_0, B_1, B_2\}, \Delta, B_0, R')$ where R' contains the rules

$$B_0 \to B_A \ , \quad B_A \to \ B_A \Bigg| \begin{array}{c} \kappa \\ / \ | \ \backslash \\ B_1 \ B_2 \ B_1 \end{array} \ , \quad B_1 \to \begin{array}{c} \gamma \\ | \\ B_1 \end{array} \Bigg| \alpha \ , \text{ and } \quad B_2 \to \begin{array}{c} \delta \\ | \\ B_2 \end{array} \Bigg| \alpha \ .$$

We note that in this example, we use simplified rules which do not correspond to the formal construction that will be presented in Definition 4.2.2. The simplifications allow for a compact representation of H_3 and more intuitive explanations. The nonterminal B_0 symbolizes the root of the RHS of r_1. The label at this position is the nonterminal A, hence, traversal must continue at the RHS of either r_2 or r_3. The nonterminal B_A represents the roots of the RHS of r_2 and r_3, respectively. Hence, there are two possibilities. First, the nonterminal B_A might be called again, which corresponds to the recursive rule application by r_2. Since H_3 does not utilize a pushdown, information about the number of recursive calls is lost in this step. Second, the nonterminal B_A might produce the terminal κ and traversal continues in its argument positions at positions below an occurrence of the nonterminal A. For this, the nonterminal B_1 symbolizes to continue in the first argument position, and B_2 to continue in the second argument position of an occurrence of A. We note that the position of B_1 and B_2 correspond to the position of the variables x_1 and x_2, respectively, in r_3. Hence, B_1 and B_2 either recursively generate γ's and δ's, respectively, or abort recursion by producing an α. If a pushdown is used, then the number of required γ's or δ's can be tracked up to a certain bound. Considering the rules, we can see that $\mathcal{L}(H_3) = \{\kappa(\gamma^j(\alpha), \delta^k(\alpha), \gamma^\ell(\alpha)) \mid j, k, \ell \in \mathbb{N}\}$. Thus, we have $\mathcal{L}(G_{18}) \subseteq \mathcal{L}(H_3)$ and H_3 is indeed a (superset) approximation.

In this chapter, we first recall the definitions related to CFTGs with storage (cf. Section 4.1). Afterwards, we present the actual approximation that gives rise to an infinite hierarchy of improving approximations (cf. Section 4.2).

4.1 Context-Free Tree Grammars with Storage

In this section, we recall basic notions required to equip a CFTG with a notion of storage. For this, we recall storage types as a simplification of the definitions in [17]. Furthermore, we recall the special storage type that describes a pushdown and show how the pushdown storage type can be used within the rules of a CFTG.

As a preliminary definition, we need to extend the definition of a CFTG to allow for infinite sets of nonterminals and rules, because nonterminals will be paired with a possibly infinite amount of storage configurations (also cf. Definition 4.1.5).

Definition 4.1.1 An *extended CFTG* is a tuple (N, Δ, A_0, R) where

- N is a (possibly infinite) set (*nonterminals*),
- Δ is a ranked alphabet such that $N \cap \Delta = \emptyset$,
- $A_0 \in N^{(0)}$ (*initial nonterminal*), and
- R is a (possibly infinite) set (*rules*), such that for every $A \in N$ we have that $R|_A$ is finite.

Let G_{ex} be an extended CFTG. We denote the *derivation relation of* G_{ex} by $\Rightarrow$ and define it as for CFTG. The *tree language of* G_{ex}, denoted by $\mathcal{L}(G_{\mathrm{ex}})$, is obtained as in the case of CFTG by $\mathcal{L}(G_{\mathrm{ex}}) = \{\xi \in \mathrm{T}_\Delta \mid A_0 \Rightarrow_d \xi,\ d \text{ is outside-in}\}$. □

We note that if N is finite, then R is finite as well and the definition of an extended CFTG matches the definition of a CFTG. We observe the following.

Observation 4.1.2 Let G_{ex} be an extended CFTG. Then, for each $\zeta \in \mathrm{T}_{N\cup\Delta}(X)$, we have that $\{\xi \in \mathrm{T}_{N\cup\Delta}(X) \mid \zeta \Rightarrow \xi\}$ is finite (note that we only consider a single derivation step). Thus, for each $n \in \mathbb{N}$, we have that $\{\xi \in \mathrm{T}_{N\cup\Delta}(X) \mid A_0 \Rightarrow^n \xi\}$ is a finite set.

The idea of grammar with storage, inspired by [60], was introduced in and generalized by [17] (see also [19]). We recall their definitions and adapt them to our scenario.

Definition 4.1.3 A *storage type* is a tuple $S = (C, C_P, C_F, c_{\mathrm{in}}, \mu)$ where

- C is a set (*configurations*)
- C_P is a finite set (*predicates*)
- C_F is a finite set (*functions*)
- $c_{\mathrm{in}} \in C$ (*initial configuration*)
- μ is a mapping (*meaning function*) which interprets every predicate and function as follows. For $p \in C_P$ we have a total function $\mu(p)\colon C \to \{\mathrm{true}, \mathrm{false}\}$. For $f \in C_F$ we have a partial function $\mu(f)\colon C \nrightarrow C$.

We call S a *finite storage type* if C is finite. □

Note that this definition is close to the definition of a "datatype" in [19, Definition 3.1], but more restrictive in the following sense. We do not allow for arbitrary Boolean combinations of predicates. Furthermore, a storage type includes the initial configuration as a special instance of [19, Definition 2.3]. This simplifies notation for the particular case handled here.

In the following let $S = (C, C_P, C_F, c_{\text{in}}, \mu)$ be an arbitrary but fixed storage type.

Definition 4.1.4 A CFT(S)*-grammar* (read: context-free tree grammar using the storage S) is a tuple $G_S = (N, \Delta, A_0, R_S)$ where

- N, Δ, and A_0 are defined as for CFTGs, and
- R_S is a finite set of rules of the form $A(\overline{x}) \to \textbf{if } p \textbf{ then } \xi'$ where $A(\overline{x}) \to \xi$ is a CFTG rule, $p \in C_P$, and ξ' is obtained from ξ by replacing every occurrence of a nonterminal B in ξ by $B(f)$ for some $f \in C_F$.

A RT(S)*-grammar* (read: regular tree grammar using storage S) is a CFT(S)-grammar where $N = N^{(0)}$. □

We define the ranked alphabet $N(C) = \{A(c) \mid A \in N, c \in C\}$ where for each $A(c) \in N(C)$ we define $\text{rk}_{N(C)}(A(c)) = \text{rk}_N(A)$.

Definition 4.1.5 Let $G_S = (N, \Delta, A_0, R_S)$ be a CFT(S)-grammar. We define the *CFTG associated with* G_S, denoted by $\mathcal{G}(G_S)$, as the extended CFTG

$$\mathcal{G}(G_S) = (N(C), \Delta, A_0(c_{\text{in}}), R')$$

where R' is obtained as follows: If $A(\overline{x}) \to \textbf{if } p \textbf{ then } \xi$ is in R_S, then, for every $c \in C$, such that $\mu(p)(c) = \text{true}$ and $\mu(f)(c)$ is defined for each $f \in C_F$ that occurs in ξ, we let the rule $A(c)(\overline{x}) \to \xi'$ be in R' where ξ' is obtained from ξ by replacing every occurrence of $B(f)$ for some nonterminal $B \in N$ and function $f \in C_F$ by $B(\mu(f)(c))$. □

Note that the requirement of extended CFTGs is met: For every nonterminal of $\mathcal{G}(G_S)$, i.e., every nonterminal $A \in N$ from G_S and every configuration $c \in C$, the set $R'|_{A(c)}$ is finite.

Definition 4.1.6 Let G_S be a CFT(S)-grammar. The *derivation relation of* G_S, denoted by $\Rightarrow_{G_S}$, is defined by $\Rightarrow_{G_S} = \Rightarrow_{\mathcal{G}(G_S)}$. The *tree language of* G_S, denoted by $\mathcal{L}(G_S)$, is defined as $\mathcal{L}(G_S) = \mathcal{L}(\mathcal{G}(G_S))$. □

We observe the special case when S is a finite storage type.

Lemma 4.1.7 Let G_S be a CFT(S)-grammar. If S is a finite storage type, then $\mathcal{G}(G_S)$ is a CFTG. If additionally G is an RT(S)-grammar, then $\mathcal{G}(G)$ is a RTG.

Proof. Clearly, $N(C)$ is finite if N and C are finite. Moreover, since, for each $A \in N$, the set $R|_A$ is finite, we have that the set of rules from $\mathcal{G}(G_S)$ is finite as well. ■

Note that in Definition 4.1.5 and consequently in the proof of Lemma 4.1.7 we incorporate all pushdown configurations into the nonterminals of $\mathcal{G}(G_S)$. This can be optimized by only considering reachable pushdown configurations in the construction.

In the following we recall a special storage type, namely the pushdown storage [19, Def. 3.28]. Note that, in contrast to the original definition in [19], we allow the pushdown to be empty.

Definition 4.1.8 Let Γ be an alphabet. We define the *Γ-pushdown storage*, denoted by $\mathrm{PD}(\Gamma)$, as the storage type $\mathrm{PD}(\Gamma) = (C, C_P, C_F, c_0, \mu)$ where

- $C = \Gamma^*$,
- $C_P = \{\mathrm{top} = \gamma \mid \gamma \in \Gamma \cup \{\varepsilon\}\}$,
- $C_F = \{\mathrm{push}(\gamma) \mid \gamma \in \Gamma\} \cup \{\mathrm{pop}, \mathrm{id}\}$,
- $c_0 = \varepsilon$, and
- μ is defined, for each $c \in C$, as
  - $\mu(\mathrm{top} = \varepsilon)(c) = \begin{cases} \mathrm{true} & \text{if } c = \varepsilon, \\ \mathrm{false} & \text{otherwise,} \end{cases}$
  - $\mu(\mathrm{pop})(c) = \begin{cases} w & \text{if } c = w\gamma \text{ with } w \in \Gamma^* \text{ and } \gamma \in \Gamma, \\ \mathrm{undefined} & \text{otherwise,} \end{cases}$
  - $\mu(\mathrm{id})(c) = c$.

  and for each $\gamma \in \Gamma$, as
  - $\mu(\mathrm{top} = \gamma)(c) = \begin{cases} \mathrm{true} & \text{if } c = w\gamma \text{ for some } w \in \Gamma^*, \\ \mathrm{false} & \text{otherwise,} \end{cases}$
  - $\mu(\mathrm{push}(\gamma))(c) = c\gamma$, □

The configurations of $\mathrm{PD}(\Gamma)$ are denoted as strings. The symbol at the right end of such a configuration is considered the topmost symbol on the pushdown.

4.2 Approximation of a CFTG by a RTG

In this section, we will see how the language induced by an arbitrary CFTG can be expressed by an RTG by means of an approximation. The approximation result is split into two parts. First, we show that any CFTG is equivalent to a RTG with pushdown storage. Second, we present restrictions to the storage such that the number of possible storage configurations becomes finite. This finite information can be incorporated into the nonterminals of the RTG. We obtain a hierarchy of approximations that improve concerning the subset relation. Lastly, we relate this approximation result to the string case, i.e., the approximation of context-free languages by REG.

Throughout this section, *we let $G = (N, \Delta, A_0, R)$ be an arbitrary, but fixed CFTG.*

Characterization of a CFTG by a RTG with Pushdown Storage

As a first step on the way to an approximation we characterize the CFTG G by a $\mathrm{RT}(S_{\mathrm{PD}(\Gamma)})$-grammar H where Γ is the pushdown alphabet of a lnCFTG defined in the following.

Definition 4.2.1 We define the *pushdown alphabet of G*, denoted by Γ_G, as the set $\Gamma_G = \{\langle r, w\rangle \mid r \in R, w \in \mathrm{pos}_{N \setminus N^{(0)}}(\mathrm{rhs}(r))\}$. We abbreviate $\mathrm{PD}(\Gamma_G)$ to $\mathrm{PD}(G)$ and call it the *pushdown storage associated with G*. □

In the following, we let $\mathrm{PD}(G) = (C, C_P, C_F, c_0, \mu)$ be the pushdown storage associated with G. Furthermore, we let Γ_G be defined as in Definition 4.2.1 and we call elements in Γ_G *return addresses* of G. Return addresses are used as pushdown symbols in the following construction, which is based on [29, Thm. 3, p. 257].

Definition 4.2.2 We define the *pushdown characterization of* G, denoted by $\mathrm{pd}(G)$, as the $\mathrm{RT}(\mathrm{PD}(G))$-grammar $\mathrm{pd}(G) = (N', \Delta, A_0{}', R')$ where N', $A_0{}'$, and R' are defined as follows. We let $N' = \{A_0{}'\} \cup \{\langle r, w\rangle \mid r \in R, w \in \mathrm{pos}(\mathrm{rhs}(r))\}$ and define R' as the smallest set satisfying the following five properties.

(1) For each $r_{\mathrm{in}} \in R|_{A_0}$, we let $A_0{}' \to \langle r_{\mathrm{in}}, \varepsilon\rangle(\mathrm{id})$ be in R'.

Let $r \in R$ be of the form $A(\overline{x}) \to \xi$ and $w \in \mathrm{pos}(\xi)$.

(2) If $\xi(w) = \delta$ for some $k \in \mathbb{N}$ and $\delta \in \Delta^{(k)}$, then we let, for each predicate $p \in C_P$

$$\langle r, w\rangle \to \textbf{if } p \textbf{ then } \delta(\langle r, w1\rangle(\mathrm{id}), \ldots, \langle r, wk\rangle(\mathrm{id}))$$

be in R'.

(3) If $\xi(w) = B$ for some $B \in N^{(0)}$, then, for each $r' \in R|_B$ and predicate $p \in C_P$, we let

$$\langle r, w\rangle \to \textbf{if } p \textbf{ then } \langle r', \varepsilon\rangle(\mathrm{id})$$

be in R'.

(4) If $\xi(w) = B$ for some $B \in N$ with $\mathrm{rk}_N(B) > 0$, then, for each $r' \in R|_B$ and predicate $p \in C_P$, we let

$$\langle r, w\rangle \to \textbf{if } p \textbf{ then } \langle r', \varepsilon\rangle(\mathrm{push}(\langle r, w\rangle))$$

be in R'.

(5) If $\xi(w) = x_i$, for some $i \in [\mathrm{rk}_N(A)]$, then, for each $r' \in R$ and $v \in \mathrm{pos}_A(\mathrm{rhs}(r'))$, we let

$$\langle r, w\rangle \to \textbf{if } \mathrm{top} = \langle r', v\rangle \textbf{ then } \langle r', vi\rangle(\mathrm{pop})$$

be in R'.

Each rule that is created using case $i \in [5]$ is called a *Type (i)* rule. □

Note that, for each rule $r \in R'$, we have that $\mathrm{lhn}(r)$ determines the type of the rule. For each rule r' in $\mathcal{G}(\mathrm{pd}(G))$ such that $\mathrm{lhn}(r') = B(c)$ for some $B \in N'$ and $c \in C$, we say that r' is a Type (i) rule ($i \in [5]$) if this is the type determined by the nonterminal B in $\mathrm{pd}(G)$. There is a connection between the nonterminals in N' and the return addresses used in Type (4) and (5) rules.

Intuitively, we traverse the RHSs of the rules of G from top to bottom. For this, we consider a rule r together with a position in its RHS. This information is encoded into the nonterminals of $\mathrm{pd}(G)$. A rule in $\mathrm{pd}(G)$ with LHS-nonterminal $\langle r, w\rangle$ corresponds to the symbol $\mathrm{rhs}(r)(w)$. We make a case analysis of the different types of symbols. If we

encounter a terminal (Type (2) rule), then it is output as usual and the processing continues in each subtree. If we encounter a nonterminal A (Type (4)), then we put the current rule and position onto the pushdown as a return address and continue at the root of an A-rule. If A is nullary (Type (3)), the return address is not needed. On meeting an x_i, we continue at the i-th successor of the return address encoded in the topmost pushdown symbol.

Observation 4.2.3 In each rule of $\mathrm{pd}(G)$, the predicate of the rules only refers to the topmost symbol of the pushdown configuration. More precisely, if $A(c\gamma)(\overline{x}) \Rightarrow \zeta$ is a derivation in $\mathrm{pd}(G)$, then $A(c'\gamma)(\overline{x}) \Rightarrow \zeta'$ is a derivation in $\mathrm{pd}(G)$ where $c' \in \Gamma_G^*$ such that $c'\gamma \in C$, ζ' is obtained as defined in Definition 4.1.5, and ζ and ζ' only differ in the associated configurations.

We relate derivations in $\mathcal{G}(\mathrm{pd}(G))$ and derivations in G by considering a rule $r \in R$ and a position $w \in \mathrm{rhs}(r)$. If we start a derivation with $\langle r, w\rangle(\varepsilon)$ and apply rules from $\mathrm{pd}(G)$ as long as any rule is applicable, the resulting tree consists of terminals and nonterminals of the form $\langle r, w'\rangle(\varepsilon)$ for some $w' \in \mathrm{pos}(\mathrm{rhs}(r))$. A similar tree can be obtained by starting a derivation with $\mathrm{rhs}(r)|_w$, but instead of the nonterminals, variables occur. This is formalized in the following observation, where we use a variable z_j (j is a unique index) for each occurrence of a variable x_i ($i \in [\mathrm{rk}_N(\mathrm{lhn}(r))]$) in $\mathrm{rhs}(r)$.

Observation 4.2.4 [29, p. 258] We let $r \in R$, $\mathrm{rhs}(r) = \zeta$, $m = |\mathrm{pos}_X(\zeta)|$, and $Z_m = \{z_1, \ldots, z_m\}$. Furthermore, we let $w_1, \ldots, w_m \in \mathrm{pos}_X(\zeta)$ be pairwise disjoint, $\xi = \zeta[w_1/z_1, \ldots, w_m/z_m]$, $w \in \mathrm{pos}(\xi)$, and $\xi' \in \mathrm{C}_\Delta(Z_m)$. We modify G to treat elements from Z_m as terminals. Then it holds that

$$\langle r, w\rangle(\varepsilon) \Rightarrow^* \xi'[z_i/\langle r, w_i\rangle(\varepsilon) \mid i \in [m]] \text{ holds in } \mathcal{G}(\mathrm{pd}(G)) \quad \text{iff} \quad \xi|_w \Rightarrow^* \xi' \text{ holds in } G \ .$$

Theorem 4.2.5 It holds that $\mathcal{L}(G) = \mathcal{L}(\mathrm{pd}(G))$.

Proof. $\mathcal{L}(G) \subseteq \mathcal{L}(\mathrm{pd}(G))$: We let $r \in R|_{A_0}$ be of the form $A_0 \to \zeta$, $\xi \in \mathrm{T}_\Delta$, and consider a derivation $A_0 \Rightarrow_r \zeta \Rightarrow^* \xi$ in G. Note that any derivation of G has this form. There is a Type (1) rule ${A_0}' \to \langle r, \varepsilon\rangle(\mathrm{id})$ in $\mathrm{pd}(G)$. Furthermore, note that ξ and ζ do not contain any variables. By Observation 4.2.4, we have $\langle r, \varepsilon\rangle(\varepsilon) \Rightarrow^* \xi$ is a derivation in $\mathcal{G}(\mathrm{pd}(G))$ and thus, $\xi \in \mathcal{L}(\mathrm{pd}(G))$.

$\mathcal{L}(G) \supseteq \mathcal{L}(\mathrm{pd}(G))$: Let $\xi \in \mathrm{T}_\Delta$, $r \in R|_{A_0}$, and consider a derivation ${A_0}'(\varepsilon) \Rightarrow_{r_1} \langle r, \varepsilon\rangle(\varepsilon) \Rightarrow^* \xi$ in $\mathcal{G}(\mathrm{pd}(G))$. Note that each derivation in $\mathcal{G}(\mathrm{pd}(G))$ has this form. Since r_1 is a Type (1) rule and by Definition 4.2.2, the rule r has the form $A_0 \to \zeta$ for some $\zeta \in \mathrm{T}_{N\cup\Delta}$. By Observation 4.2.4, there is a derivation $\zeta \Rightarrow^* \xi$ in G and thus, $\xi \in \mathcal{L}(G)$. ∎

Approximation of a CFTG by a RTG with a Restricted Pushdown

We will now restrict the pushdown storage to only use finitely many configurations. Then, we can incorporate the set of configurations into the state space of a RTG. Thus, we get an approximation for G. We introduce a depth-limited pushdown and then define an approximation RTG. The depths of the pushdown is limited to a natural number m and if a push occurs such that the pushdown configuration would become larger than m, then the pushdown forgets the oldest symbol.

Afterwards, we will consider the special case of the trivial pushdown storage ($m = 0$), i.e., no pushdown is used at all. Then, we formally show the intuitive fact that the pushdown approximation improves if m increases, i.e., more return addresses can be stored. For this, an infinite hierarchy of pushdown approximations is considered.

In the following, *we let* $m \in \mathbb{N}$.

Definition 4.2.6 Let $\mathrm{PD}(G) = (C, C_P, C_F, \varepsilon, \mu)$. We define the *$m$-depth limited pushdown storage associated with G*, denoted by $\mathrm{PD}^m(G)$, as the storage type $\mathrm{PD}^m(G) = (C^m, C_P, C_F, \varepsilon, \mu^m)$ where C^m and μ^m are defined as follows.

- $C^m = \{c \in \Gamma_G^* \mid |c| \leq m\}$.
- The meaning function μ^m is defined as in Definition 4.1.8 for all but the following case: For each $c \in C^m$ where $c = \gamma_1 \dots \gamma_{|c|}$ with $\gamma_i \in \Gamma_G$, for all $i \in [|c|]$, and for each $\gamma \in \Gamma_G$ we define
  - $\mu^m(\mathrm{push}(\gamma))(c) = \begin{cases} c\gamma & \text{if } |c\gamma| \leq m, \\ \gamma_2 \dots \gamma_{|c|}\gamma & \text{if } |\gamma_2 \dots \gamma_{|c|}\gamma| \leq m, \\ \varepsilon & \text{if } m = 0. \end{cases}$ □

Note that C^m is a finite set.

In the following, we let $\mathrm{PD}^m(G) = (C^m, C_P, C_F, \varepsilon, \mu^m)$ be the m-depth limited pushdown storage associated with G.

We can use the m-depth limited pushdown to obtain a pushdown approximation of G. Since $\mathrm{PD}^m(G)$ forgets pushdown symbols, the approximation needs to guess the forgotten symbols if it wants to read from an empty pushdown as follows.

Definition 4.2.7 We define the *m-pushdown RTG associated with G*, denoted by $\mathrm{pd}^m(G)$, as the $\mathrm{RT}(\mathrm{PD}^m(G))$-grammar $\mathrm{pd}^m(G) = (N', \Delta, A_0{}', R'')$ where N' is defined as in Definition 4.2.2 and R'' contains all rules of R' from Definition 4.2.2 as well as the following additional rules: Let $r \in R$ be of the form $A(\overline{x}) \to \zeta$, $i \in [\mathrm{rk}_N(A)]$, and $w \in \mathrm{pos}_{x_i}(\zeta)$.

(6) For each $r' \in R$ and $w' \in \mathrm{pos}_A(\mathrm{rhs}(r'))$, we let

$$\langle r, w\rangle \to \textbf{if } \mathrm{top} = \varepsilon \textbf{ then } \langle r', w'i\rangle(\mathrm{id})$$

be in R''. A rule created by this case is called a *Type (6)* rule. □

Since $\mathrm{PD}^m(G)$ is a finite storage type, we have that $\mathcal{G}(\mathrm{pd}^m(G))$ is a RTG (cf. Lemma 4.1.7). Recall that the LHS-nonterminal of a rule in $\mathrm{pd}(G)$ determines the type of that rule. This property is no longer true for rules in $\mathrm{pd}^m(G)$, because rules of Type (5) and (6) have the same kind of LHS-nonterminals, viz. $\langle r, w\rangle$ where $r \in R$ and $w \in \mathrm{pos}_X(\mathrm{rhs}(r))$. However, we can differentiate such rules as follows. In a derivation of $\mathrm{pd}^m(G)$ and thus in $\mathcal{G}(\mathrm{pd}^m(G))$, we consider a rule r' in $\mathcal{G}(\mathrm{pd}^m(G))$ such that $\mathrm{lhn}(r') = \langle r, w\rangle(c)$ for some $r \in R$, $w \in \mathrm{pos}_X(\mathrm{rhs}(r))$, and $c \in C^m$. Then r' is of Type (5), if $c \neq \varepsilon$ and r' is of Type (6) if $c = \varepsilon$. This holds, because Type (5) rules require the pushdown to be nonempty.

Now we want to prove that $\mathrm{pd}^m(G)$ is indeed an approximation of $\mathrm{pd}(G)$ and thus of G. For this, we relate derivations in $\mathcal{G}(\mathrm{pd}^m(G))$ to derivations in $\mathcal{G}(\mathrm{pd}(G))$. Consider a derivation d in $\mathcal{G}(\mathrm{pd}(G))$. Intuitively, if no pushdown symbol is forgotten, i.e., the size of each pushdown is less than m, then d is also a derivation in $\mathcal{G}(\mathrm{pd}^m(G))$. If a symbol is forgotten, then we can use the corresponding Type (6) rule.

Lemma 4.2.8 $\mathcal{L}(\mathrm{pd}(G)) \subseteq \mathcal{L}(\mathrm{pd}^m(G))$

Proof. We let $\xi \in \mathrm{T}_{N' \cup \Delta}$, $n \in \mathbb{N}$, $\xi_{0..n} \in \mathrm{T}_\Delta(N'(C))$, and $r_{1..n} \in R'$. If there is a derivation

$$d\colon A_0'(\varepsilon) = \xi_0 \Rightarrow_{r_1} \xi_1 \Rightarrow_{r_2} \ldots \Rightarrow_{r_n} \xi_n = \xi$$

in $\mathcal{G}(\mathrm{pd}(G))$, then there are $\xi'_{0..n} \in \mathrm{T}_\Delta(N'(C^m))$ and $r'_{1..n} \in R''$ such that

$$d'\colon A_0'(\varepsilon) = \xi_0 \Rightarrow_{r'_1} \xi'_1 \Rightarrow_{r'_2} \ldots \Rightarrow_{r'_n} \xi'_n = \xi$$

is a derivation in $\mathcal{G}(\mathrm{pd}^m(G))$ where, for each $i \in [n]$, r'_i and ξ'_i are defined by induction in the following. For each $i \in [n]$, we assume that r_i is applied at w_i. We let $r'_1 = r_1$, we obtain ξ'_1 from ξ_0 by applying r'_1 at w_1, and for each $i \in [n] \setminus \{1\}$, we define

$$r'_i = \begin{cases} B_1(c') \to \delta(B_2(c'), \ldots, B_k(c')) & \text{if } r_i \text{ is } B_1(c) \to \delta(B_2(c), \ldots, B_k(c)) \\ B_1(c') \to B_2(c') & \text{if } r_i \text{ is } B_1(c) \to B_2(c) \\ B_1(c') \to B_2(\mu^m(\mathrm{push}(\gamma))(c')) & \text{if } r_i \text{ is } B_1(c) \to B_2(c\gamma) \\ B_1(c') \to B_2(c'_1) & \text{if } r_i \text{ is } B_1(c) \to B_2(c_1),\ c = c_1\gamma,\ c' = c'_1\gamma \\ & \quad \text{for some } c_1 \in C,\ c'_1 \in C^m, \text{ and } \gamma \in \Gamma_G \\ B_1(\varepsilon) \to B_2(\varepsilon) & \text{if } r_i \text{ is } B_1(c) \to B_2(c_1),\ c = c_1\gamma \\ & \quad \text{for some } c_1 \in C \text{ and } \gamma \in \Gamma_G \end{cases}$$

where in each case c' is such that $\xi'_{i-1}(w_i) = B_1(c')$. Furthermore, we define ξ'_i such that $\xi'_{i-1} \Rightarrow_{r'_i} \xi'_i$ holds where r'_i is applied at w_i. Note that the case distinction for r'_i covers each rule type not concerning the initial nonterminal, i.e. Types (2) to (5), where Type (5) is split into two cases.

By Observation 4.2.3, only the topmost symbol on the pushdown determines the applicable rules. Hence, disregarding the pushdown at each nonterminal, the only difference between a derivation in $\mathcal{G}(\mathrm{pd}(G))$ and a derivation in $\mathcal{G}(\mathrm{pd}^m(G))$ may occur when the pushdown is empty. In this case, we can find an appropriate Type (6) rule. It remains to adapt the pushdown configurations. It can be seen that the construction of r'_i does exactly that. ∎

No Pushdown. To illustrate the concept of a pushdown approximation, we consider the special case of using no pushdown at all. We present an example, show the connection of a RTG H to $\mathrm{pd}^0(H)$, and relate this setting to the case for strings.

Here, we let $m = 0$ and hence we get $C^0 = \{\varepsilon\}$. Thus, in $\mathrm{pd}^0(G)$ the predicate of a Type (5) rules is never satisfied, since the pushdown is always empty. If a return address is needed, it is guessed from all valid possibilities using Type (6) rules.

Example 4.2.9 We construct the CFTG $G_{19} = (\{A_0, A\}, \Delta, A_0, R)$ as follows. We let $\Delta = \{\gamma^{(1)}, \delta^{(1)}, \alpha^{(0)}, \beta^{(0)}\}$ and we let R contain the rules

$$r_1 : A_0 \to \gamma(A(\alpha, \alpha)), \qquad r_3 : A(x_1, x_2) \to \gamma(A(\gamma(x_1), \gamma(x_2))),$$
$$r_2 : A_0 \to \delta(A(\beta, \beta)), \qquad r_4 : A(x_1, x_2) \to \kappa(x_1, x_2)\ .$$

We want to highlight some properties of this grammar. First, we note that G_{19} is linear nondeleting. Second, it can be seen that

$$\mathcal{L}(G_{19}) = \{\ \gamma(\gamma^n(\kappa(\gamma^n(\alpha), \gamma^n(\alpha)))),\ \delta(\gamma^n(\kappa(\gamma^n(\beta), \gamma^n(\beta)))) \mid n \in \mathbb{N}\}\ .$$

Third, we observe that trees in $\mathcal{L}(G_{19})$ are either rooted in γ and have an α at each leaf, or they are rooted in δ and have a β at each leaf.

We construct the 0-pushdown RTG $\mathrm{pd}^0(G_{19}) = (N', \Delta, A_0{}', R')$ as follows. Since $C^0 = \{\varepsilon\}$, only one predicate is satisfiable, viz. the predicate $(\mathrm{top} = \varepsilon)$. Hence, all rules testing for other predicates are not relevant. Consequently, Type (5) rules cannot be used in $\mathrm{pd}^0(G_{19})$. Furthermore, all push instructions result in the configuration ε and are thus omitted in the following. In Type (6) rules, the predicate is always true. Thus, we omit the predicate as well as the function. We define

$$\begin{aligned} N' = {} & \{\langle r_1, \varepsilon\rangle, \langle r_1, 1\rangle, \langle r_1, 11\rangle, \langle r_1, 12\rangle, A_0{}'\} \\ & \cup \{\langle r_2, \varepsilon\rangle, \langle r_2, 1\rangle, \langle r_2, 11\rangle, \langle r_2, 12\rangle\} \\ & \cup \{\langle r_3, \varepsilon\rangle, \langle r_3, 1\rangle, \langle r_3, 11\rangle, \langle r_3, 111\rangle, \langle r_3, 12\rangle, \langle r_3, 121\rangle\} \\ & \cup \{\langle r_4, \varepsilon\rangle, \langle r_4, 1\rangle, \langle r_4, 2\rangle\} \end{aligned}$$

and point out some of the rules of R' which are of interest:

Type (1) rules:

$$A_0{}' \to \langle r_1, \varepsilon\rangle \qquad A_0{}' \to \langle r_2, \varepsilon\rangle$$

Type (2) rules:

$$\begin{array}{lll} \langle r_1, \varepsilon\rangle \to \gamma(\langle r_1, 1\rangle) & \langle r_1, 11\rangle \to \alpha & \langle r_2, 11\rangle \to \beta \\ \langle r_4, \varepsilon\rangle \to \kappa(\langle r_4, 1\rangle, \langle r_4, 2\rangle) & \langle r_1, 12\rangle \to \alpha & \langle r_2, 12\rangle \to \beta \end{array}$$

Type (4) rules:

$$\begin{array}{lll} \langle r_1, 1\rangle \to \langle r_3, \varepsilon\rangle & \langle r_2, 1\rangle \to \langle r_3, \varepsilon\rangle & \langle r_3, 1\rangle \to \langle r_3, \varepsilon\rangle \\ \langle r_1, 1\rangle \to \langle r_4, \varepsilon\rangle & \langle r_2, 1\rangle \to \langle r_4, \varepsilon\rangle & \langle r_3, 1\rangle \to \langle r_4, \varepsilon\rangle \end{array}$$

Type (6) rules:

$$\begin{array}{lll} \langle r_3, 111\rangle \to \langle r_1, 11\rangle & \langle r_3, 111\rangle \to \langle r_2, 11\rangle & \langle r_3, 111\rangle \to \langle r_3, 11\rangle \\ \langle r_3, 121\rangle \to \langle r_1, 12\rangle & \langle r_3, 121\rangle \to \langle r_2, 12\rangle & \langle r_3, 121\rangle \to \langle r_3, 12\rangle \\ \langle r_4, 1\rangle \to \langle r_1, 11\rangle & \langle r_4, 1\rangle \to \langle r_2, 11\rangle & \langle r_4, 1\rangle \to \langle r_3, 11\rangle \\ \langle r_4, 2\rangle \to \langle r_1, 12\rangle & \langle r_4, 2\rangle \to \langle r_2, 12\rangle & \langle r_4, 2\rangle \to \langle r_3, 12\rangle \end{array}$$

Note that by using Type (6) rules, information is lost since the grammar nondeterministically chooses a possible return address. We illustrate this using the tree $\xi = \gamma(\kappa(\alpha, \beta))$ with

the following derivation in $\mathcal{G}(\mathrm{pd}^0(G_{19}))$ (recall that the only configuration $c = \varepsilon$ is not depicted):

$$\begin{aligned} A_0{}' &\Rightarrow \langle r_1, \varepsilon\rangle \Rightarrow \gamma(\langle r_1, 1\rangle) \Rightarrow \gamma(\langle r_4, \varepsilon\rangle) \Rightarrow \gamma(\kappa(\langle r_4, 1\rangle, \langle r_4, 2\rangle)) \\ &\Rightarrow^2 \gamma(\kappa(\langle r_1, 11\rangle, \langle r_2, 12\rangle)) \Rightarrow^2 \gamma(\kappa(\alpha, \beta)) \ . \end{aligned}$$

Thus, we have $\xi \in \mathcal{L}(\mathrm{pd}^0(G_{19}))$, but $\xi \notin \mathcal{L}(G_{19})$. Recall from Theorem 4.2.5 that $\mathcal{L}(G_{19}) = \mathcal{L}(\mathrm{pd}(G_{19}))$. It is easy to see that for each derivation of $\mathrm{pd}(G_{19})$, there is a corresponding one in $\mathrm{pd}^0(G_{19})$ where each return address is correctly guessed (cf. Lemma 4.2.8). Hence, we get that $\mathcal{L}(\mathrm{pd}^0(G_{19})) \supset \mathcal{L}(G_{19})$. ○

It is intuitively clear, that the pushdown is only needed if return addresses must be stored. Since RTGs contain no variables, return addresses are not needed in a derivation. Hence, we obtain the following lemma.

Lemma 4.2.10 For each RTG H, we have $\mathcal{L}(H) = \mathcal{L}(\mathrm{pd}^0(H))$.

Proof. To prove this lemma, we analyze the rules of $\mathrm{pd}(H)$. Since H is a RTG, $\mathrm{pd}(H)$ does not contain Type (4) or Type (5) rules and $\mathrm{pd}^0(H)$ does not contain Type (6) rules.

Hence, derivations in both $\mathrm{pd}(H)$ and $\mathrm{pd}^0(H)$ only contain the configuration ε and only use the function id. Since the meaning functions of $\mathrm{PD}(H)$ and $\mathrm{PD}^0(H)$ applied to id behave in the same way concerning the configuration ε, each derivation in $\mathcal{G}(\mathrm{pd}(H))$ is a derivation of $\mathcal{G}(\mathrm{pd}^0(H))$ and vice versa. Thus, we get that $\mathcal{L}(\mathrm{pd}(H)) = \mathcal{L}(\mathrm{pd}^0(H))$ and, by Theorem 4.2.5, it holds that $\mathcal{L}(H) = \mathcal{L}(\mathrm{pd}(H))$. ■

Restricted Pushdown. Now, we consider non-trivial pushdown storages, i.e., we choose $m > 0$. We investigate how this effects the m-pushdown RTG. We give an example and show how increasing the pushdown limit improves the approximation.

Example 4.2.11 Recall the lnCFTG G_{19} from Example 4.2.9. By using the 1-pushdown RTG $\mathrm{pd}^1(G_{19})$, we can observe that $\gamma(\kappa(\alpha, \beta)) \notin \mathcal{L}(\mathrm{pd}^1(G_{19}))$ since one return address is stored. This can be illustrated using the following rules from $\mathrm{pd}^1(G_{19})$ (rules without predicate are shortcuts for the rules with all possible predicates):

$$\begin{aligned}
&\langle r_1, \varepsilon\rangle \to \gamma(\langle r_1, 1\rangle(\mathrm{id})) && \langle r_3, 111\rangle \to \textbf{if } \mathrm{top} = \langle r_1, 1\rangle \textbf{ then } \langle r_1, 11\rangle(\mathrm{pop}) \\
&\langle r_1, 1\rangle \to \langle r_3, \varepsilon\rangle(\mathrm{push}(\langle r_1, 1\rangle)) && \langle r_3, 121\rangle \to \textbf{if } \mathrm{top} = \langle r_1, 1\rangle \textbf{ then } \langle r_1, 12\rangle(\mathrm{pop}) \\
&\langle r_1, 1\rangle \to \langle r_4, \varepsilon\rangle(\mathrm{push}(\langle r_1, 1\rangle)) && \langle r_3, 111\rangle \to \textbf{if } \mathrm{top} = \langle r_2, 1\rangle \textbf{ then } \langle r_2, 11\rangle(\mathrm{pop}) \\
&\langle r_2, \varepsilon\rangle \to \delta(\langle r_2, 1\rangle(\mathrm{id})) && \langle r_3, 121\rangle \to \textbf{if } \mathrm{top} = \langle r_2, 1\rangle \textbf{ then } \langle r_2, 12\rangle(\mathrm{pop}) \\
&\langle r_2, 1\rangle \to \langle r_3, \varepsilon\rangle(\mathrm{push}(\langle r_2, 1\rangle)) && \\
&\langle r_2, 1\rangle \to \langle r_4, \varepsilon\rangle(\mathrm{push}(\langle r_2, 1\rangle)) &&
\end{aligned}$$

A derivation in $\mathrm{pd}^1(G_{19})$ contains non-trivial pushdown configurations as for example in the derivation of the tree $\xi = \gamma(\kappa(\alpha, \alpha))$ depicted in Figure 4.1 Since $\xi \notin \mathcal{L}(\mathrm{pd}^1(G_{19}))$ and $\xi \in \mathcal{L}(\mathrm{pd}^0(G_{19}))$, we conclude that $\mathrm{pd}^1(G_{19})$ is a better approximation of G_{19} than $\mathrm{pd}^0(G_{19})$ from Example 4.2.9. ○

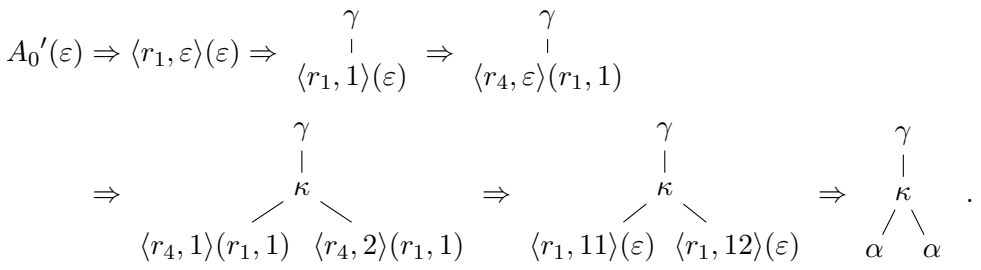


Figure 4.1: A derivation in $\mathrm{pd}^1(G_{19})$.

There are CFTGs that use only finitely many return addresses. If G is such a CFTG, the m-limited pushdown approximation of G is equivalent to G, since all needed pushdown approximations can be stored. We formalize this in the following observation.

Observation 4.2.12 We let G be a CFTG and $m_G \in \mathbb{N}$. Furthermore, we let d be a derivation in $\mathcal{G}(\mathrm{pd}(G))$, $n = |d|$, $\xi_{1..(n-1)} \in \mathrm{T}_\Delta(N(C))$, and $\xi_n \in \mathrm{T}_\Delta$ be such that $d\colon A_0'(\varepsilon) \Rightarrow \xi_1 \Rightarrow \ldots \Rightarrow \xi_n$. If, for each $i \in [n]$ and $c \in C$ with $\mathrm{pos}_{N(\{c\})}(\xi_i) \neq \emptyset$, we have $|c| \leq m_G$, then d is a derivation of $\mathcal{G}(\mathrm{pd}^{m_G}(G))$.

If the above holds for each derivation d of $\mathcal{G}(\mathrm{pd}(G))$, then $\mathcal{L}(\mathrm{pd}(G)) = \mathcal{L}(\mathrm{pd}^{m_G}(G))$.

The following lemma connects Observation 4.2.12 to the notion of a self-embedding lnCFTG as introduced in Chapter 3.

Lemma 4.2.13 We let $m \in \mathbb{N}$, and G be a CFTG such that, for each derivation of $\mathcal{G}(\mathrm{pd}(G))$ of the form $d\colon A_0'(\varepsilon) \Rightarrow \xi_1 \Rightarrow \ldots \Rightarrow \xi_n$ where $n = |d|$, $\xi_{1..(n-1)} \in \mathrm{T}_\Delta(N(C))$, and $\xi_n \in \mathrm{T}_\Delta$ the following holds. For each $i \in [n]$ and $c \in C$ with $\mathrm{pos}_{N(\{c\})}(\xi_i) \neq \emptyset$ we have $|c| \leq m$. If G is linear nondeleting, then G is non-self-embedding.

Proof. We assume that G is self-embedding. Then, there are a rule r in $\mathcal{G}(\mathrm{pd}(G))$ with $\mathrm{lhn}(r) = A(c_1)$, $c' \in \Gamma_G^* \setminus \{\varepsilon\}$, and $\xi \in \mathrm{C}_\Delta(\{z\})$ such that $\langle r, \varepsilon\rangle(c_1) \Rightarrow^* \xi[\langle r, \varepsilon\rangle(c_1 c')]$ is a derivation in $\mathcal{G}(\mathrm{pd}(G))$. By Observation 4.2.3, for each $n \in \mathbb{N}$, we have that $\langle r, \varepsilon\rangle(c_1) \Rightarrow^* \xi^n[\langle r, \varepsilon\rangle(c_1(c')^n)]$ is a derivation in $\mathcal{G}(\mathrm{pd}(G))$. Since $c' \neq \varepsilon$, we have $|c_1(c')^n| > m_G$ for some $n \in \mathbb{N}$. This contradicts the requirement of this lemma and thus, by contradiction, G is non-self-embedding. ∎

We examine the effect of increasing the pushdown limit. Intuitively, the more return addresses a pushdown approximation can store, the less trees are wrongly generated. We show this intuition formally in the following lemma.

Lemma 4.2.14 For each CFTG G and every $m \in \mathbb{N}$ we have

$$\mathcal{L}(\mathrm{pd}^{m+1}(G)) \subseteq \mathcal{L}(\mathrm{pd}^m(G)) \ .$$

Proof (sketch). Let $\xi \in \mathcal{L}(\mathrm{pd}^{m+1}(G))$ and d be a derivation in $\mathcal{G}(\mathrm{pd}^{m+1}(G))$ that derives ξ. We consider each configuration c occurring in d and do a case distinction on $|c|$.

If $|c| \leq m$, then it is clear that d is a derivation in $\mathcal{G}(\mathrm{pd}^m(G))$ and hence, $\xi \in \mathcal{L}(\mathrm{pd}^m(G))$. If $|c| > m$, then there is at least one configuration c occurring in d such that $|c| = m + 1$.

Following Observation 4.2.3, it can be seen that we can construct a derivation d' of $\mathcal{G}(\mathrm{pd}^m(G))$ shortening each configuration c_1 occurring in a sentential form of d and propagate this change through the derivation. If a nonempty configuration c_2 occurs in d and the associated configuration in d' is empty, then we can use a Type (6) rule to continue the derivation. The adaption of configurations in d' can be achieved very similar as in the proof of Lemma 4.2.8. ∎

The inclusion of Lemma 4.2.14 becomes strict if the considered CFTG is complex enough. However, a formal definition of such grammars is very technical while the intuition is easily described. A CFTG is complex in this sense, if there are derivations that use pushdown configurations of arbitrary size and furthermore, forgetting some pushdown symbols changes the induced tree language. The following lemma states that there are such complex grammars.

Lemma 4.2.15 There is a CFTG G such that, for each $m \in \mathbb{N}$, we have

$$\begin{aligned} &\mathcal{L}(\mathrm{pd}^m(G)) \setminus \mathcal{L}(\mathrm{pd}(G)) \quad \neq \emptyset \text{ and} \\ &\mathcal{L}(\mathrm{pd}^m(G)) \setminus \mathcal{L}(\mathrm{pd}^{m+1}(G)) \neq \emptyset \ . \end{aligned}$$

Proof. It is easy to see that the CFTG G_{19} from Example 4.2.9 can be used to show that, for each $m \in \mathbb{N}$ and tree $\xi_m = \gamma(\gamma^{m+1}(\kappa(\gamma^{m+1}(\alpha), \gamma^{m+1}(\beta))))$, we have $\xi_m \in \mathcal{L}(\mathrm{pd}^m(G_{19}))$, but $\xi_m \notin \mathcal{L}(\mathrm{pd}^{m+1}(G_{19}))$. Furthermore, it is clear that $\xi_m \notin \mathcal{L}(\mathrm{pd}(G_{19})) = \mathcal{L}(G_{19})$. ∎

Example 4.2.11 is an illustration of Lemma 4.2.15 for the case $m = 0$.

We combine results regarding approximation of context-free tree languages by RTGs and obtain the following theorem.

Theorem 4.2.16 Let G be a CFTG. The following holds:

$$\begin{aligned} \mathcal{L}(G) = \mathcal{L}(\mathrm{pd}(G)) \subseteq \ldots \subseteq \mathcal{L}(\mathrm{pd}^{m+1}(G)) \subseteq \mathcal{L}(\mathrm{pd}^m(G)) \subseteq \ldots \\ \subseteq \mathcal{L}(\mathrm{pd}^1(G)) \subseteq \mathcal{L}(\mathrm{pd}^0(G)) \end{aligned}$$

Furthermore, there are CFTGs G' for which each of the inclusions is strict.

Proof. The first part of the theorem is a consequence of Theorem 4.2.5 and Lemmas 4.2.8 and 4.2.14. The second part follows from Lemma 4.2.15. ∎

Relation to the String Case

To conclude this section, we relate Theorem 4.2.16 to the string case. It turns out that our approximation of a context-free tree language by a RTG with pushdown corresponds to an already existing result in which the string language of a CFG is approximated by a finite state automata using a pushdown [4, 55]. We note that the way in which return addresses are forgotten in [55] is different from the approach taken here. Instead of just forgetting the oldest return address, the authors of [55] drop reoccurring sequences of return addresses. This approach allows for a better approximation of synchronized symbols at the beginning

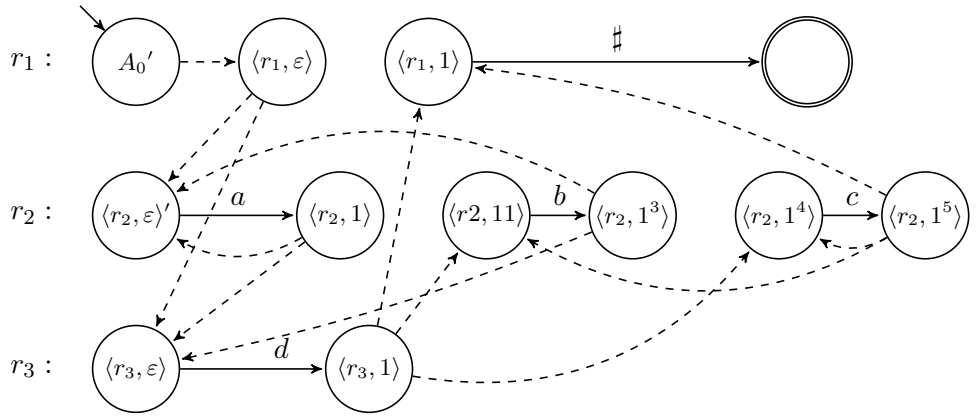


Figure 4.2: Example in the String case.

and end of a string. However, deleting only repeating return addresses restricts the control over the size of the state space, since the pushdown may grow very large before a return address is repeated.

Furthermore, we want to exemplarily show that the special case of our result with $m = 0$, restricted to the tree representations of a REG, corresponds to an approximation of context-free string languages by REG described in [46]. For this, we recall the tree representation of a CFG (cf. Definition 2.2.24) and that each CFG M can be seen as a lnCFTG $\widetilde{M}$ (cf. Observation 2.2.25). If we consider the tree representation of strings, then our definition, using a 0-depth limited pushdown storage, corresponds to [46, Section 9.3.2]. This can be seen in the following example.

Example 4.2.17 We let $M = (\{B_0\}, \{a, b, c, d\}, B_0, R)$ be a CFG where R contains the two rules $B_0 \to aB_0bB_0c$ and $B_0 \to d$. The CFTG $\widetilde{M} = (\{{A_0}^{(0)}, B^{(1)}\}, \widetilde{\Sigma}, A_0, R')$ contains the rules

$$\begin{aligned} r_1 &\colon A_0 \to B_0(\sharp) \ , \\ r_2 &\colon B_0(x_1) \to a(B_0(b(B_0(c(x_1))))) \ , \text{ and} \\ r_3 &\colon B_0(x_1) \to d(x_1) \ . \end{aligned}$$

Constructing the associated RTG $\mathcal{G}(\mathrm{pd}^0(\widetilde{M}))$ results in the state behavior depicted in Figure 4.2. The resulting RTG is illustrated in the form of a finite state string automaton, although it formally recognizes unary trees. Dashed lines symbolize ε-transitions. ○

5 Training of Regular Tree Grammars

This chapter is mainly based on my conference publication "Regular Approximation of Weighted Linear Nondeleting Context-Free Tree Languages" [66]. Some parts are taken over verbatim.

In machine translation, a language model can be extended with weights to measure the grammaticality of sentences. The task of obtaining suitable weights is called *training*. The weights can be extracted from finite sets of samples, called corpora. Well-known linguistic corpora stem from, e.g., sentences found in newspapers, transcriptions of speeches held in the European parliament, or just a collection of sentences found in the world wide web. Alternatively, one grammar can be trained based on an infinite language, e.g., based on the induced language of another grammar. The latter possibility corresponds to expressing the weighted language of one grammar by means of another grammar. For example, Nederhof describes a method to approximate weights for a given finite automaton using a given weighted CFG such that the Kullback-Leibler divergence between the induced weighted string languages is minimal [49]. Another approach is, considering the derivation trees of a CFG as its induced tree language, to find a weight assignment for a CFG such that its induced weighted tree language is close to an arbitrary, infinite tree distribution [12]. Concerning more expressive tree-generating grammars, it has been shown how to obtain a weight assignment for a given RTG to approximate the weighted tree language induced by a weighed tree adjoining grammar [50].

In this chapter, we show how weighted context-free tree languages can be expressed by weighted RTGs (wRTGs). More concretely, we focus on weighted tree languages induced by weighted lnCFTGs (wlnCFTGs) and show how to approximate optimal weights for a given RTG. An optimal weight assignment in this sense is determined by minimizing the Kullback-Leibler (KL) divergence between the weighted tree languages induced by the wlnCFTG and the wRTG. The choice of utilizing the KL divergence is motivated by its broad use in the existing literature (cf., e.g., [49, 12, 50]).

Before explaining the details on how to express the weighted tree language induced by a wlnCFTG by means of a wRTG, we illustrate the involved weighted tree grammars. The following example describes the wlnCFTG $(G_{20}, p_{G_{20}})$ and the RTG H_4 (we refer to Section 5.2 for a formal definition of wlnCFTGs and wRTGs). These two grammars are used as a running example throughout this chapter. We let $\Delta = \{\kappa^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and define the lnCFTG $G_{20} = (N_{G_{20}}, \Delta, A_0, R_{G_{20}})$ where $N_{G_{20}} = \{A_0, A\}$ and the rules in $R_{G_{20}}$ are depicted in Figure 5.1(a) where we ignore the annotated weights above each RHS. By inspecting the rules, it can be seen that $\mathcal{L}(G_{20}) = \{\kappa(\gamma^{2n}(\alpha), \gamma^n(\alpha)) \mid n \in \mathbb{N}\}$. A *weight assignment* for G_{20} is a mapping $p_{G_{20}}$ that assigns a value, called *weight* to each rule. The weight assignment $p_{G_{20}}$ is depicted in Figure 5.1 as an annotation above the RHS of each rule. The pair $(G_{20}, p_{G_{20}})$ is a *weighted lnCFTG* (wlnCFTG). It induces a *weighted tree language*, i.e., a mapping from T_Δ into $\mathbb{R}_{\geq 0}$ as follows. The weight of a tree $\xi \in \mathrm{T}_\Delta$ is the sum of the weight of each derivation resulting in ξ. The weight of a derivation is obtained

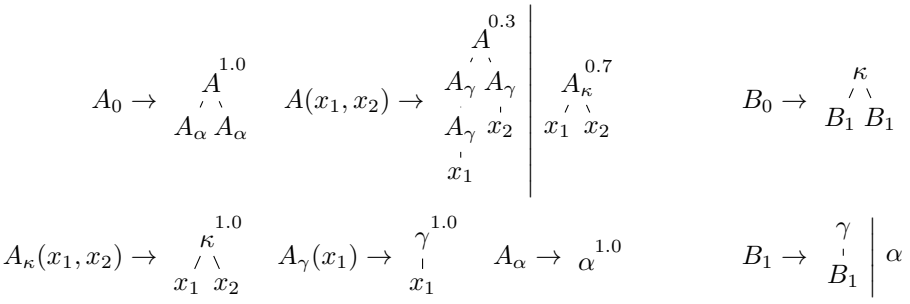
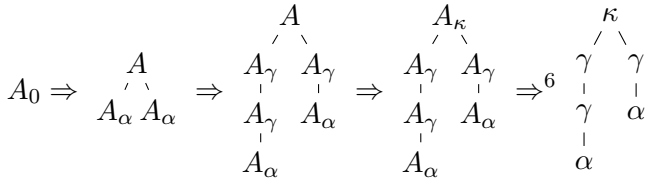


Figure 5.1: (a) The wlnCFTG $(G_{20}, p_{G_{20}})$. (b) The RTG H_4.

by multiplying the weight of a rule upon its application during the derivation. As an illustration, the derivation



is assigned the weight $1 \cdot 0.3 \cdot 0.7 \cdot 1^6 = 0.21$. Each factor in the product corresponds to one application of a rule. By inspecting the rules of G_{20}, it can be seen that, for each $n \in \mathbb{N}$ and tree $\xi_n = \kappa(\gamma^{2n}(\alpha), \gamma^n(\alpha))$, there is exactly one derivation for ξ_n and thus, $p_{G_{20}}(\xi_n) = 0.3^n \cdot 0.7$. For each tree ξ' of a different shape, we have $p_{G_{20}}(\xi') = 0$.

We want to approximate the weighted tree language induced by $(G_{20}, p_{G_{20}})$ by a wRTG (H_4, p_{H_4}) where $H_4 = (N_{H_4}, \Delta, B_0, R_{H_4})$, $N_{H_4} = \{B_0, B_1\}$, and R_{H_4} is given in Figure 5.1(b). It can be seen that $\mathcal{L}(H_4) = \{\kappa(\gamma^k(\alpha), \gamma^\ell(\alpha)) \mid k, \ell \in \mathbb{N}\}$ and thus, $\mathcal{L}(G_{H_4}) \subseteq \mathcal{L}(H_4)$. We will show how to approximate the best weight assignment for H_4 regarding the KL divergence.

In more detail, we proceed as follows. Given a wlnCFTG (G, p_G) and a RTG H, we intersect (G, p_G) and H enriched by a trivial weight assignment. In the obtained wlnCFTG (K, p_K), we determine the expected frequencies of some rules of K by fixpoint iteration up to an arbitrary precision. Then, we use these expected frequencies to find an optimal weight assignment p_H for H such that the KL divergence between (G, p_G) and (H, p_H) is minimized.

The chapter is structured as follows. We first introduce the notion of tree shaped derivations (cf. Section 5.1), followed by a formal definition of wlnCFTGs and wRTGs (cf. Section 5.2). Furthermore, for each rule r of a wlnCFTG, we formalize the concept of the expected frequency of r and three characterizations of this value (cf. Section 5.3). Afterwards, we describe the intersection of a lnCFTG and a RTG in the unweighted case and lift the result to the weighted case (cf. Section 5.4). Then, we describe how the best weight assignment for H can be approximated (cf. Section 5.5).

5.1 Tree-Shaped Derivations

In this section we develop an alternative representation of the derivations of a lnCFTG. We let r be a rule of the lnCFTG with RHS ζ. For each nonterminal occurrence in ζ, one tree needs to be substituted. Each such tree is obtained by a derivation starting with a rule from the respective LHS-nonterminal. We encode this information in a tree over the rules, by considering the rules of a lnCFTG as a ranked alphabet. Each rule's rank is determined by the number of nonterminal occurrences in its RHS.

Throughout this section, *we let $G = (N, \Delta, A_0, R)$ be an arbitrary lnCFTG.*

Definition 5.1.1 We consider R as a ranked alphabet by letting, for each $r \in R$, the rank of r be defined as $\mathrm{rk}_R(r) = |\mathrm{pos}_N(\mathrm{rhs}(r))|$.

Let $r \in R$, $\xi = \mathrm{rhs}(r)$, $n = |\mathrm{pos}_N(\xi)|$, and $w_1, \ldots, w_n \in \mathrm{pos}_N(\xi)$ be pairwise distinct and ordered lexicographically. We define the *nonterminal word of* r, denoted by $\mathrm{ntw}(r)$, as the string $\xi(w_1) \ldots \xi(w_n)$. We note, for each $r \in R$, that $\mathrm{ntw}(r) \in N^*$ and $\mathrm{rk}_R(r) = |\mathrm{ntw}(r)|$.

We will define an N-indexed family over $\mathcal{P}(\mathrm{T}_R)$, denoted by $(\mathcal{D}_G(A) \mid A \in N)$, by simultaneously defining the sets $\mathcal{D}_G(A)$ as follows. Let $A \in N$, $d \in \mathrm{T}_R$, and $r = d(\varepsilon)$. Then $d \in \mathcal{D}_G(A)$ if

(i) $\mathrm{lhn}(r) = A$ and,

(ii) for each $i \in [\mathrm{rk}_R(r)]$, we have $d|_i \in \mathcal{D}_G(A_i)$ where $A_i = \mathrm{ntw}(r)(i)$.

We call each element in $\mathcal{D}_G(A)$ a *tree-shaped derivation starting in* A. For each $A \in N$ and $d \in \mathcal{D}_G(A)$, the size of d, denoted by $|d|$, is defined as $|d| = |\mathrm{pos}(d)|$. □

Observation 5.1.2 Let $A \in N$. There is a one-to-one correspondence between $\mathcal{D}_G(A)$ and the set of complete leftmost outermost derivations starting in $A(\overline{x})$.

Let $d \in \mathcal{D}_G(A)$ and d' be the corresponding complete leftmost outermost derivation starting in $A(\overline{x})$ according to Obs. 5.1.2. We write $A(\overline{x}) \Rightarrow_d \xi$ instead of $A(\overline{x}) \Rightarrow_{d'} \xi$. Furthermore, $\mathcal{D}_G(A_0)$ is abbreviated by $\mathcal{D}_G$.

Observation 5.1.2 states the connection between a complete leftmost outermost derivation and the tree representation of a derivation. It is however not necessary to use the leftmost outermost ordering of rules. In fact, complete derivations using any fixed order on the rules have the one-to-one correspondence to tree shaped derivations as mentioned in Observation 5.1.2. Thus, the tree representation of the derivations of a lnCFTG abstracts from the derivation mode (cf. unrestricted derivation from Section 2.2).

The next definition describes sets of specific derivations that will help us later. We define the set of derivations that derive a specific tree, the subset of derivations starting with a specific rule, and we define derivations where exactly one nonterminal occurrence is not derived.

Definition 5.1.3 For each $\xi \in \mathrm{T}_\Delta$ and $r \in R$, we define the sets

- $\mathcal{D}_G(\xi)$ of *derivations ending in* ξ as $\mathcal{D}_G(\xi) = \{d \in \mathcal{D}_G \mid A_0 \Rightarrow_d \xi\}$ and
- $\mathcal{D}_G(r)$ of *derivations starting with* r as $\mathcal{D}_G(r) = \{d \in \mathcal{D}_G(\mathrm{lhn}(r)) \mid r = d(\varepsilon)\}$.

For $A, B \in N$, we define the *set of B-partial derivations starting in A*, denoted by $\mathcal{D}_G^B(A)$, as follows. For each $d \in \mathrm{C}_R(\{z\})$, we let $d \in \mathcal{D}_G^B(A)$ if there is a derivation $d_B \in \mathcal{D}_G(B)$ such that $d[d_B] \in \mathcal{D}_G(A)$. □

As mentioned before, we abstract from the order in which nonterminals are derived. However, there is still some nondeterminism in the choice of rules which are applied at each nonterminal occurrence during the derivation process. Consider two distinct derivations. Either the derivations end in different trees, or both derivations end in the same tree. In the literature, the latter phenomenon is called the ambiguity of a grammar. The following definition disallows such ambiguity. Intuitively, if a terminal tree is in the tree language of G, then there is exactly one (tree-shaped) derivation for it.

Definition 5.1.4 We say that the lnCFTG G is *unambiguous* if, for each $\xi \in \mathrm{T}_\Delta$, we have that $|\mathcal{D}_G(\xi)| \in \{0, 1\}$. □

5.2 Weighted lnCFTGs and Weighted RTGs

In this section, we introduce weighted lnCFTGs and weighted RTGs as a special case. We utilize weights from $\mathbb{R}_{\geq 0}$. A weighted CFTG is obtained from a CFTG by assigning a weight to each rule. The weight of a derivation is obtained taking, for each occurrence of a rule in the derivation, the weight of that rule and multiplying those weights. Then, the weight of a tree is the sum of the weights of all derivations for this tree. We give formal definitions in the following.

Definition 5.2.1 A *weighted lnCFTG over $\mathbb{R}_{\geq 0}$ (wlnCFTG)* is a tuple (G, p_G) where $G = (N, \Delta, A_0, R)$ is a lnCFTG and p_G is a mapping $p_G \colon R \to \mathbb{R}_{\geq 0}$, which we call *weight assignment for G*. □

In this section, *we let (G, p_G) be a wlnCFTG where $G = (N, \Delta, A_0, R)$.*

A weighted wlnCFTG assigns weights to each derived tree. Thus, in analogy to a tree language, we define the concept of a weighted tree language. As a technical tool, we use weights from the positive reals $\mathbb{R}_{\geq 0}$ extended by the value ∞, denoted by $\mathbb{R}_{\geq 0}^{\infty}$. We utilize the complete semiring $(\mathbb{R}_{\geq 0}^{\infty}, +, \cdot, 0, 1)$ with an infinitary sum operation that allows to sum over infinite index sets (cf., e.g., [16, Sec. 2] for detailed definitions).

Definition 5.2.2 A *weighted tree language (over Δ)* is a mapping $p \colon \mathrm{T}_\Delta \to \mathbb{R}_{\geq 0}^{\infty}$. □

We now formally define the weight of a derivation in a wlnCFTG and the weight of a tree in a wlnCFTG. Furthermore, the concept is extended to partial derivations by not considering the gap in the derivation.

Definition 5.2.3 We let $A, B \in N$ and $d \in \mathcal{D}_G(A) \cup \mathcal{D}_G^B(A)$. Then the *weight of d in* (G, p_G), denoted by $p_G(d)$, is defined as

$$p_G(d) = \prod\nolimits_{w \in \mathrm{pos}_R(d)} p_G(d(w)) \ .$$

For each $\xi \in \mathrm{T}_\Delta$, we define the *weight of ξ in* (G, p_G), denoted by $p_G(\xi)$, as

$$p_G(\xi) = \sum\nolimits_{d \in \mathcal{D}_G(\xi)} p_G(d) \ .$$

We call p_G the *weighted tree language induced by* (G, p_G). □

We note that if $\xi \in \mathrm{T}_\Delta \setminus \mathcal{L}(G)$, then $p_G(\xi) = 0$, because $\mathcal{D}_G(\xi) = \emptyset$. Furthermore, we remark that p_G denotes the weight assignment for G as well as the weighted tree language induced by (G, p_G). However, a distinction is always possible from the context.

Definition 5.2.4 Let p be a weighted tree language. We call p

- *proper* if for each $A \in N$ we have $\sum_{r \in R|_A} p_G(r) = 1$, and
- *consistent* if $\sum_{\xi \in \mathrm{T}_\Delta} p_G(\xi) = 1$.

We call (G, p_G) *proper* (*consistent*) if p_G is proper (consistent).

The *support of* (G, p_G), denoted by $\mathrm{supp}(G, p_G)$, is defined as

$$\mathrm{supp}(G, p_G) = \{\xi \in \mathrm{T}_\Delta \mid p_G(\xi) \neq 0\} \ . \qquad \square$$

Definition 5.2.5 A *weighted regular tree grammar* (wRTG) is a wlnCFTG (H, p_H) where H is a RTG. □

The normal forms of CFTGs and RTGs are described in Section 2.2. These can easily be extended to the weighted case.

Definition 5.2.6 A wlnCFTG (G, p_G) is in *nonterminal form* if G is. □

The following lemma follows as a simple extension from the unweighted case proven in [58, p. 113].

Lemma 5.2.7 For every wlnCFTG (G, p_G), there is a wlnCFTG $(G', p_{G'})$ in nonterminal form such that, for each $\xi \in \mathrm{T}_\Delta$, we have $p_G(\xi) = p_{G'}(\xi)$. The construction preserves properness and consistency.

Definition 5.2.8 A wRTG (H, p_H) is *producing* if H is. □

5.3 Expected Frequencies

In this section, we recall the concept of the expected frequency of a rule similarly to the case for weighted CFG [49, Sec. 3]. Intuitively, this is the number of times, a rule occurs in a derivation discounted by the weight of the derivation. Thus, the expected frequency does not need to be a natural number. We present an example, and then give the formal definition of the expected frequency of a rule.

Example 5.3.1 This example uses the lnCFTG G_{20} from the running example at the beginning of this chapter. It is easy to see that, in each derivation, the rule $r_\kappa \colon A(x_1, x_2) \to \kappa(x_1, x_2)$ occurs exactly once. Hence, the expected frequency of r_κ is 1, denoted by $E(r_\kappa) = 1$. We let $n \in \mathbb{N}$ and let ξ_n denote the tree $\kappa(\gamma^{2n}(\alpha), \gamma^n(\alpha))$. The rule $r_\gamma \colon A(x_1) \to \gamma(x_1)$ occurs $3n$ times in a derivation of ξ_n. We obtain the expected frequency of r_γ by summing

over all derivations and discounting each summand by the weight of the respective derivation. Thus, we have that

$$E(r_\gamma) = \underbrace{\sum_{n\in\mathbb{N}}}_{\text{all derivations}} \underbrace{0.3^n \cdot 0.7}_{\text{weight of derivation}} \cdot \underbrace{3n}_{\text{occurrences of } r_\gamma} = \frac{9}{7} \ .$$

Note that for each $n \in \mathbb{N}$, there is a unique derivation in G_{20} deriving ξ_n. ○

Throughout this section, *we let* (G, p_G) *be a consistent wlnCFTG where* $G = (N, \Delta, A_0, R)$, $A \in N$, *and* $r \in R|_A$.

Definition 5.3.2 We define the *expected rule frequency of* r as

$$E(r) = \sum_{\substack{d_1 \in \mathcal{D}_G^A(A_0) \\ d_2 \in \mathcal{D}_G(r)}} p_G(d_1[d_2]) \ . \tag{5.1}$$

□

In the following we present three characterizations of the expected rule frequency of r. From Definition 5.1.3, we recall that A-partial derivations are those partial derivations that can be completed by using a derivation starting from A. In (5.1), the gap in the A-partial derivation d_1 is closed using derivations starting specifically with r. Hence, Equation (5.1) enumerates all the distinct ways the rule r can be used in a derivation. Each such way is discounted by the weight of the respective derivation. This intuition can be used to characterize the expected rule frequency of r by the number of occurrences of r in all derivations. We first define this number and then formally state the observation.

Definition 5.3.3 For each $r' \in R$ and $d \in \mathcal{D}_G$, the *number of occurrences of* r' *in* d, denoted by $\sharp_{r'}(d)$, is defined as

$$\sharp_{r'}(d) = |\mathrm{pos}_{r'}(d)| \ .$$

□

Observation 5.3.4

$$E(r) = \sum_{d\in\mathcal{D}_G} p_G(d) \cdot \sharp_r(d) \ . \tag{5.2}$$

The second characterization of the expected rule frequency is obtained by using inner and outer values (cf. [41]). We consider the nonterminal $A' \in N$. The outer value of A' is the sum of the weights of all A'-partial derivations starting in A_0. The inner value of A' in turn is the weight of all derivations starting from A'.

Definition 5.3.5 For each $A' \in N$, we define the *outer value of* A', denoted by $\mathrm{outer}(A')$, and the *inner value of* A', denoted by $\mathrm{inner}(A')$, as

$$\mathrm{outer}(A') = \sum_{d\in\mathcal{D}_G^{A'}(A_0)} p_G(d) \qquad \text{and} \qquad \mathrm{inner}(A') = \sum_{d\in\mathcal{D}_G(A')} p_G(d) \ .$$

□

Inner and outer values allow to rephrase the expected frequency of r. We consider the outer value of A. This is the weight of all partial derivations that can continue with A. Instead of continuing the derivation with any A-rule, we continue exclusively using the rule r. Thus, we multiply the outer value of A by the weight of r. In order to obtain a complete derivation, we then multiply with the inner value of each nonterminal occurrence in the RHS of r, i.e., the inner weight of each nonterminal in $\mathrm{ntw}(r)$.

Observation 5.3.6 Assume that $\mathrm{ntw}(r) = A_1 \dots A_k$. Then it holds that

$$E(r) = \mathrm{outer}(A) \cdot p_G(r) \cdot \prod_{i \in [k]} \mathrm{inner}(A_i) \ . \tag{5.3}$$

The third characterization of the expected rule frequency of r is obtained using the second one and rephrasing inner and outer values. We first give an alternative for the outer value of a nonterminal $A' \in N$. Instead of considering the weight of an A'-partial derivation, we consider all rules r' such that A' occurs in the nonterminal word of r'. For each such r', we take the outer value of $\mathrm{lhn}(r')$ and multiply it with the weight of the rule as well as the product of the inner weights of all other nonterminal occurrences in $\mathrm{ntw}(r')$. Intuitively, these two approaches yield the same result. However, we need to consider the special case of the initial nonterminal A_0. In the string case discussed in [49], the initial nonterminal may not occur in any right-hand side and, by definition, $\mathrm{outer}(A_0) = 1$. We do not impose this restriction and allow for A_0 to occur in right-hand sides. Since G is proper, the outer value of the initial nonterminal is at least one, even if it does not occur on any RHS. We define, for every $A' \in N$, the value $d(A' = A_0)$ to be 1 if $A' = A_0$ and 0 otherwise. We add $d(A' = A_0)$ to the outer value of A'. Thus, if A_0 does not occur in any right-hand side, then $\mathrm{outer}(A_0) = 1$. Otherwise, $\mathrm{outer}(A_0) \geq 1$. We reword the inner value of A' by considering again the sum of all rules r' where A' occurs in the nonterminal word of r' and multiply the weight of r' with the product of the inner value of each nonterminal in $\mathrm{ntw}(r')$.

Observation 5.3.7

$$\begin{aligned} \mathrm{outer}(A') = d(A' = A_0) + \sum_{\substack{\ell \in \mathbb{N},\ r' \in R,\ j \in [\ell] \\ \mathrm{ntw}(r') = A'_1 \dots A'_\ell,\ A' = A'_j}} & \mathrm{outer}(\mathrm{lhn}(r')) \cdot p_G(r') \\ & \cdot \prod_{i \in ([\ell] \setminus \{j\})} \mathrm{inner}(A'_i) \quad \text{and} \\ \mathrm{inner}(A') = \sum_{\substack{r' \in R|_{A'} \\ \mathrm{ntw}(r') = A'_1 \dots A'_\ell}} & p_G(r') \cdot \prod_{i \in [\ell]} \mathrm{inner}(A'_i) \ . \end{aligned} \tag{5.4}$$

As in the string case (cf. [49, p. 5]), the values for $\mathrm{inner}(A)$ and $\mathrm{outer}(A)$ can be approximated by fixed-point iteration. Thus, using Observation 5.3.6, we can approximate $E(r)$.

5.4 Intersection of a (w)lnCFTG and a (w)RTG

In this section, we recall how a lnCFTG can be constructed that induces the intersection of the tree language induced by another lnCFTG and the tree language induced by a RTG. Furthermore, we recall the same construction for the case that both grammars are weighted. In the latter case, the intersection is replaced by the pointwise multiplication of the respective tree languages. Details of the construction will be used later for the main result of the chapter (cf. Section 5.5). Thus, we present the construction in full detail.

The intersection of a context-free tree language with a regular tree language is again a context-free tree language (cf. [58, p. 114] and [59, Thm. 7]). The original proof of Thm. 7 in [59] introduces copying and deleting rules. It is possible to exchange Thm. 7 of [59] by a construction which preserves linearity and nondeletion (cf., e.g., [53, Lm. 3] for the more general case of linear and nondeleting one-state weighted pushdown-extended tree transducers). A complete construction for the intersection is described in [52, p. 60], which,

given a (synchronous) lnCFTG G and RTG H, yields a (synchronous) lnCFTG K such that the tree language of K is the intersection of the tree languages of G and H.

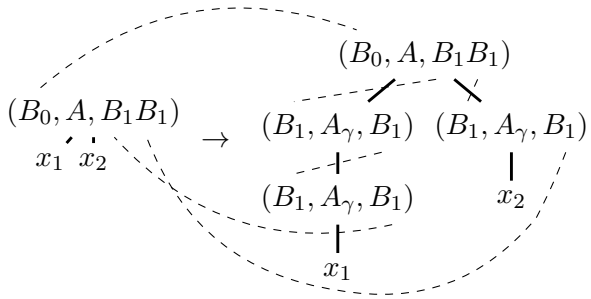
Here, we show a slightly modified version of the construction in [52] and identify properties of the constructed grammar which will prove useful in the remainder of this chapter. The main idea is, as in both [52] and [53], to annotate nonterminals of G with nonterminals of H. Since we require G to be in nonterminal form and H to be producing, we can describe the intersection similar to the string case [49]. Each nonterminal in G is annotated with all possible combinations of nonterminals from H. Terminal symbols can only be generated if the guess was correct. In [52], this checking is partly done using additional RTGs.

We first present an example and then formally define the intersection. The example can then be reconsidered while reading the proof.

Example 5.4.1 We recall the lnCFTG G_{20} and the RTG H_4 from the beginning of this chapter. We construct the lnCFTG $K = (N', \Delta, A_0{}', R')$ such that $\mathcal{L}(K) = \mathcal{L}(G_{20}) \cap \mathcal{L}(H_4)$ as follows. Each nonterminal of K consists of a nonterminal of G_{20} annotated by nonterminals of H_4. For example, N' contains the nonterminal symbol (B_0, A, B_1B_1). The intuition behind the name of the nonterminal is that G_{20} can derive a tree $\xi \in \mathrm{C}_\Delta(X_2)$ starting from $A(x_1, x_2)$, and that B_0 can derive the tree $\xi[B_1, B_1]$, i.e., the variables of ξ are replaced by the respective nonterminal symbols from H_4. Following the intuition, the initial nonterminal for K is (B_0, A_0, ε) since A_0 is nullary and both initial nonterminals should derive the same tree for the intersection.

The rules of K are build in one of two cases for each rule $r \in R$ of G_{20}:

(1) If r is a Type I rule, then we obtain a set of new rules where each nonterminal in r (in both the LHS and RHS) is annotated by nonterminals from H_4. The annotations must be compatible in the following way. Consider the annotated nonterminal (B, A, B_1B_2) at position w in the RHS of a newly created rule. Then the nonterminal at position $w1$ needs to be such that its first component is B_1. Similarly, the first component of the nonterminal at $w2$ needs to be B_2. If $w = \varepsilon$, then B must occur as the first component of the LHS-nonterminal of the newly constructed rule. If $w1$ is labeled by the variable x_1, then the first element of the third component of the LHS-nonterminal is B_1. For example, the following rule is created (dashed lines symbolize compatible parts):



(2) If r is a Type II rule, then it produces exactly one terminal, say δ. Since H_4 is producing, all rules in H_4 also produce exactly one terminal. For each rule from H_4 of the form $B \to \delta(B_1, \ldots, B_k)$, we let $A = \mathrm{lhn}(r)$ and construct the rule

$$(B, A, B_1 \ldots B_k)(\overline{x}) \to \delta(x_1, \ldots, x_k) \ .$$

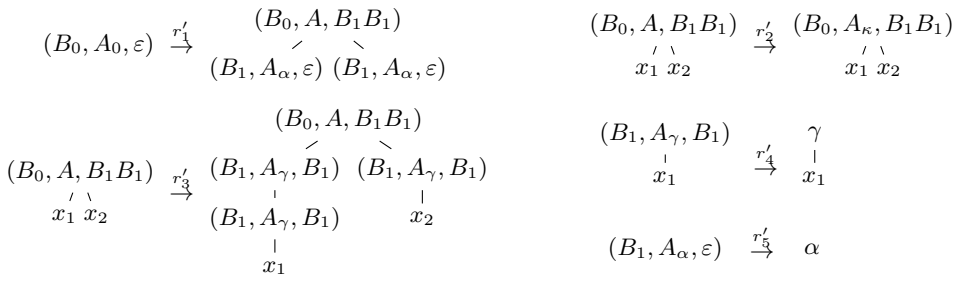


Figure 5.2: Some rules of the lnCFTG K which is the intersection of G_{20} and H_4.

Figure 5.2 depicts some of the useful rules of K. A shorthand for each rule is annotated above the arrow separating LHS and RHS. We note that we also construct some useless rules, e.g., the rule $(B_0, A, B_0B_1)(x_1, x_2) \to (B_0, A_\kappa, B_0B_1)(x_1, x_2)$. These can be eliminated after the construction (cf. Corollary 2.2.19).

It is easy to see that there is a close connection between the rules in K and the rules of G_{20} as well as the rules in H_4. For example, we say that r_2' from Figure 5.2 corresponds to the rule $A(x_1, x_2) \to \kappa(x_1, x_2)$ from G_{20}. Furthermore, the rule r_4' corresponds to both $A_\gamma(x_1) \to \gamma(x_1)$ from G_{20} and $B_1 \to \gamma(B_1)$ from H_4. ○

The proof of the following theorem contains the formal construction of the idea outlined in Example 5.4.1 as well as the proof of correctness.

Theorem 5.4.2 For each lnCFTG G and RTG H, there is a lnCFTG K in nonterminal form such that $\mathcal{L}(K) = \mathcal{L}(G) \cap \mathcal{L}(H)$.

Proof. By Lemmas 2.2.12 and 2.2.14, we can assume that G is in nonterminal form and H is producing. We proceed by constructing the lnCFTG K and then prove that $\mathcal{L}(K)$ is the intersection of $\mathcal{L}(G)$ and $\mathcal{L}(H)$.

We let $G = (N_G, \Delta, A_0, R_G)$, $H = (N_H, \Delta, B_0, R_H)$, and define

$$N' = \bigcup\nolimits_{k \in \mathbb{N}} N_H \times N_G^{(k)} \times (N_H)^k$$

as a ranked alphabet where $\mathrm{rk}_{N'}((B, A, B_1 \dots B_k)) = \mathrm{rk}_{N_G}(A) = k$.

For each right hand side ζ of a rule from G of Type I, we represent the selection of nonterminals from H that are assigned to nonterminals in ζ by an assignment τ that assigns one nonterminal from H to each position in ζ. Formally, for each $\zeta \in \mathrm{T}_{N_G}(X)$ and function $\tau \colon \mathrm{pos}(\zeta) \to N_H$, we define $\zeta_\tau \in T_{N'}(X)$ as follows. We let $\mathrm{pos}(\zeta_\tau) = \mathrm{pos}(\zeta)$ and, for each $w \in \mathrm{pos}(\zeta)$, we let

$$\zeta_\tau(w) = \begin{cases} \big(\tau(w), \zeta(w), \tau(w1) \dots \tau(w\ell)\big) & \text{if } \zeta(w) \in N_G^{(\ell)}, \\ \zeta(w) & \text{if } \zeta(w) \in X. \end{cases}$$

We define the lnCFTG $K = (N', \Delta, {A_0}', R')$ where ${A_0}' = (B_0, A_0, \varepsilon)$ and R' is defined as follows. For each Type I rule $r_G \in R_G$ of the form $A(x_{1..k}) \to \zeta$, $B, B_1, \dots, B_k \in N_H$,

and $\tau : \mathrm{pos}(\zeta) \to N_H$ such that (i) $\tau(\varepsilon) = B$ and (ii) for each $i \in [k]$ and $w_i \in \mathrm{pos}_{x_i}(\zeta)$, it holds that $\tau(w_i) = B_i$, we let

$$r' : (B, A, B_1 \dots B_k)(x_{1..k}) \to \zeta_\tau \tag{5.5}$$

be in R'. Note that r' is a rule of Type I. We call r_G the rule *corresponding* to r', denoted by $\mathrm{cor}(r') = r_G$.

For each Type II rule $r_G \in R_G$ of the form $A(x_{1..k}) \to \delta(x_{1..k})$ and each $r_H \in R_H$ of the form $B \to \delta(B_{1..k})$, we let

$$r' : (B, A, B_1 \dots B_k)(x_{1..k}) \to \delta(x_{1..k}) \tag{5.6}$$

be in R'. Note that r' is a rule of Type II. We call r_G and r_H the rules *corresponding* to r', denoted by $\mathrm{cor}(r') = (r_G, r_H)$.

We note that, for each $r' \in R'$, we denote different objects by $\mathrm{cor}(r')$ depending on the type of r. If r' is of Type-I, then $\mathrm{cor}(r')$ denotes one rule from R_G. If r' is of Type II, then $\mathrm{cor}(r')$ denotes the pair (r_G, r_H) of corresponding rules. We do not explicitly distinguish between the two cases, since the choice is clear from the context.

We prove $\mathcal{L}(K) = \mathcal{L}(G) \cap \mathcal{L}(H)$ by relating the derivations of K, G, and H in the following two claims.

Claim 1: Let $\xi \in \mathrm{T}_\Delta$ and $n = |\mathrm{pos}(\xi)|$. For each $d \in \mathcal{D}_K(\xi)$, there are unique derivations $d_G \in \mathcal{D}_G(\xi)$ and $d_H \in \mathcal{D}_H(\xi)$ such that

- $\mathrm{pos}(d) = \mathrm{pos}(d_G)$,
- for each $w \in \mathrm{pos}(d)$ where w is not a leaf, we have that $\mathrm{cor}(d(w)) = d_G(w)$,
- $n = |\mathrm{pos}(\xi)| = |d_H| = |\mathrm{yield}(d)|$, and
- there is a one-to-one correspondence φ between the positions in $\mathrm{yield}(d)$ and positions in d_H such that, for each $i \in [n]$, we have that

$$\mathrm{cor}\big(\mathrm{yield}(d)(i)\big) = \big(\mathrm{yield}(d_G)(i), d_H(\varphi(i))\big) \ .$$

Proof of Claim 1: It can be seen that for each rule $r' \in R'$, there is exactly one corresponding rule $r_G \in R_G$. We can obtain d_G from d by replacing each nonterminal $(B, A, B_1 \dots B_k)$ occurring in d by the projection to its second component A and thus, $\mathrm{pos}(d) = \mathrm{pos}(d_G)$. Since in each position $w \in \mathrm{pos}(d)$ the shape of the RHSs of $d(w)$ and $d_G(w)$, the nonterminal structure, and the terminals in the RHSs coincide, we have that $d_G \in D_G(\xi)$. Hence, d uniquely determines d_G.

Now we show how to obtain d_H. For this, we consider how terminals of the tree ξ are derived in K and H, respectively. Since K is in nonterminal form, each terminal of ξ is generated by exactly one rule of Type II, i.e., a leaf of d. Since H is producing, each terminal of ξ corresponds to exactly one position in d_H. Hence, we let $|d_H| = |\mathrm{pos}(\xi)| = |\mathrm{yield}(d)|$. Furthermore, there is a unique bijection φ between the positions of $\mathrm{yield}(d)$ and the positions of d_H. It follows that $\mathrm{cor}\big(\mathrm{yield}(d)(i)\big) = \big(\mathrm{yield}(d_G)(i), d_H(\varphi(i))\big)$. This uniquely determines d_H. □

We let $d \in \mathcal{D}_K$ and we uniquely obtain $d_G \in \mathcal{D}_G$ and $d_H \in \mathcal{D}_H$ according to Claim 1. We denote this fact by $\mathrm{cor}(d) = (d_G, d_H)$.

Claim 2: Let $\xi \in \mathrm{T}_\Delta$, $d_G \in \mathcal{D}_G(\xi)$, and $d_H \in \mathcal{D}_H(\xi)$. There is a unique $d \in \mathcal{D}_K(\xi)$ such that $cor(d) = (d_G, d_H)$.
Proof of Claim 2: We construct d as follows. By Claim 1, it holds that $\mathrm{pos}(d) = \mathrm{pos}(d_G)$. The shape of each rule in d is uniquely determined by the rule in d_G at the same position. It remains to consider the annotated nonterminals from N_H. For each position $w \in \mathrm{pos}(d)$ with $d(w) = r$, we let $\mathrm{lhn}(r) = (B, A, B_1 \dots B_k)$ and

$$\mathrm{ntw}(r) = (B^1, A^1, B_1^1 \dots B_{k_1}^1) \dots (B^\ell, A^\ell, B_1^\ell \dots B_{k_\ell}^\ell)$$

where $B, B_1, \dots, B_k, B_1^1, \dots, B_{k_1}^1, \dots, B_1^\ell, \dots, B_{k_\ell}^\ell \in N_H$ are determined by d_H as described in the following. If $w = \varepsilon$, then $B = B_0$, $A = A_0$, and $k = 0$. By definition of ζ_τ, we have that, for each $i \in [\ell]$, the nonterminals B^i,A^i, and $B_1^i \dots B_{k_i}^i$ are uniquely determined by $\mathrm{lhn}(d(wi))$. Hence, it remains to assign the left hand side nonterminals of every leaf in d. Note that if w is a leaf, then $\ell = 0$. By Claim 1, for each $w \in \mathrm{pos}(d)$ such that w is a leaf and $\mathrm{lhn}(d(w)) = (B, A, B_1 \dots B_k)$, we have that $B, B_1, \dots, B_k$ are uniquely determined by d_H. Furthermore, the nonterminal A is uniquely determined by $\mathrm{lhn}(d_G(w))$. Thus, d is completely determined by rules from G and H, and we get from the construction that $d \in \mathcal{D}_K(\xi)$. □

The set inclusion $\mathcal{L}(K) \subseteq \mathcal{L}(G) \cap \mathcal{L}(H)$ follows from Claim 1. The other direction $\mathcal{L}(K) \subseteq \mathcal{L}(G) \cap \mathcal{L}(H)$ is a consequence of Claim 2. Considering (5.5) and (5.6), it can be seen that K is in nonterminal form. ■

Weighted Intersection

We extend Theorem 5.4.2 to the weighted case, i.e., for a wlnCFTG (G, p_G) and a wRTG (H, p_H), we construct a wlnCFTG (K, p_K) such that, for each tree ξ, it holds that $p_K(\xi) = p_G(\xi) \cdot p_H(\xi)$. For this, we use the same construction as in the proof of Theorem 5.4.2 and first obtain the lnCFTG K. We extend K with a suitable weight assignment p_K to the wlnCFTG (K, p_K). The weight assignment p_K can be obtained by considering the weight of the corresponding rules. We illustrate the construction by an example.

Example 5.4.3 Recall the lnCFTG G_{20} and the RTG H_4 from the beginning of the chapter as well as the constructed lnCFTG K from Example 5.4.1. We consider three related derivations as shown in Figure 5.3. All three derivations generate the same tree $\xi = \kappa(\gamma^2(\alpha), \gamma(\alpha))$.

The first derivation, $d_{G_{20}}$, is a derivation of $(G_{20}, p_{G_{20}})$. The first three derivation steps use Type I rules and create the structure of ξ, where each terminal $\delta \in \Delta$ is replaced by the nonterminal A_δ. These three rule applications also determine the weight of the derivation, as the following six derivation steps (condensed into one depicted step) are just creating the terminals with weight 1.

The second derivation, d_{H_4}, is a derivation of (H_4, p') where p' is a weight assignment for the rules of H_4 defined as follows. The B_0-rule has weight 1, $B_1 \to \gamma(B_1)$ has weight 0.3 and $B_1 \to \alpha$ has weight 0.7. We note that p' is not the optimal weight assignment p_{H_4} that we are trying to approximate. The derivation of the tree ξ takes six derivation steps (some are condensed into one depicted step). These six steps are closely related to the six last steps of $d_{G_{20}}$.

The last derivation, d_K, is a derivation of (K, p_K) that corresponds to $d_{G_{20}}$ and d_{H_4}. It shows how the weights of $p_{G_{20}}$ and p' need to be combined such that $p_K(\xi) = p_{G_{20}}(\xi) \cdot p'(\xi)$. Since d_K corresponds in each step to $d_{G_{20}}$, it is clear that p_K should take over all the weights of $p_{G_{20}}$. Since the last six steps of d_K correspond to the entire derivation d_{H_4}, the weight of those rules in p_K are multiplied by the corresponding weights of p'.

The following theorem formalizes the intuition of weighted intersection described in Example 5.4.3.

Theorem 5.4.4 For each wlnCFTG (G, p_G) and producing wRTG (H, p_H), there is a wlnCFTG (K, p_K) in nonterminal form such that $p_K(\xi) = p_G(\xi) \cdot p_H(\xi)$ for each $\xi \in \mathrm{T}_\Delta$.

Proof. By Lemma 5.2.7, we can assume that (G, p_G) is in nonterminal form. We let K be the lnCFTG obtained by intersecting G and H according to Theorem 5.4.2. We extend K to a wlnCFTG (K, p_K) as follows. For each $r' \in R'$, we define

$$p_K(r') = \begin{cases} p_G(\mathrm{cor}(r')) & \text{if } r' \text{ is of Type I, and} \\ p_G(r_G) \cdot p_H(r_H) & \text{if } r' \text{ is of Type II and } \mathrm{cor}(r') = (r_G, r_H). \end{cases}$$

By Theorem 5.4.2, we have $\mathcal{L}(K) = \mathcal{L}(G) \cap \mathcal{L}(H)$ and K is in nonterminal form. For each $\xi \in \mathrm{T}_\Delta \setminus \mathcal{L}(K)$, it holds that $p_K(\xi) = 0$ by definition. Hence, it remains to prove that, for each $\xi \in \mathcal{L}(K)$, we have $p_K(\xi) = p_G(\xi) \cdot p_H(\xi)$.

From Claims 1 and 2 of the proof of Theorem 5.4.2, we get that there is a one-to-one connection between each $d \in \mathcal{D}_K(\xi)$ and (d_G, d_H) where $d_G \in \mathcal{D}_G(\xi)$ and $d_H \in \mathcal{D}_H(\xi)$ such that $\mathrm{cor}(d) = (d_G, d_H)$. From the proof of Claim 2 in the proof of Theorem 5.4.2, we get the connection between the rule occurrences in the three derivations. Since multiplication is commutative, the order of weight multiplication does not matter. Hence, the lemma holds. ∎

In the proof of Theorem 5.4.4 we used the fact that corresponding derivations are closely linked. We separate this fact here, since we will need it later on.

Corollary 5.4.5 For each $\xi \in \mathrm{T}_\Delta$ and each $d \in \mathcal{D}_K(\xi)$ such that $\mathrm{cor}(d) = (d_G, d_H)$ we have $p_K(d) = p_G(d_G) \cdot p_H(d_H)$.

5.5 Training of the Optimal Weight Assignment

In [12] it is shown how to train a weighted CFG based on an infinite set of derivation trees or an infinite set of strings. We use the result in the realm of trees similar to [50] as follows. Given an unambiguous and producing RTG H, our goal is to approximate the weighted tree language induced by a wlnCFTG (G, p_G) by means of the wRTG (H, p_H). For this, we construct the intersection wlnCFTG (K, p_K) of (G, p_G) and $(H, \mathbb{1})$ where $\mathbb{1}$ is the trivial weight assignment assigning 1 to each rule. Then, the training data is the set of derivations of K. From this set, we will approximate expected frequencies for the rules in K and use them to obtain an optimal p_H.

We note that each RTG can be made unambiguous by the following steps. First, we construct the associated nondeterministic finite tree automaton, then determinize it using standard techniques [25, Ch. II, Thm. 2.6] and finally transform it back to an RTG. The determinization might lead to an exponential blowup of the size of the finite automaton.

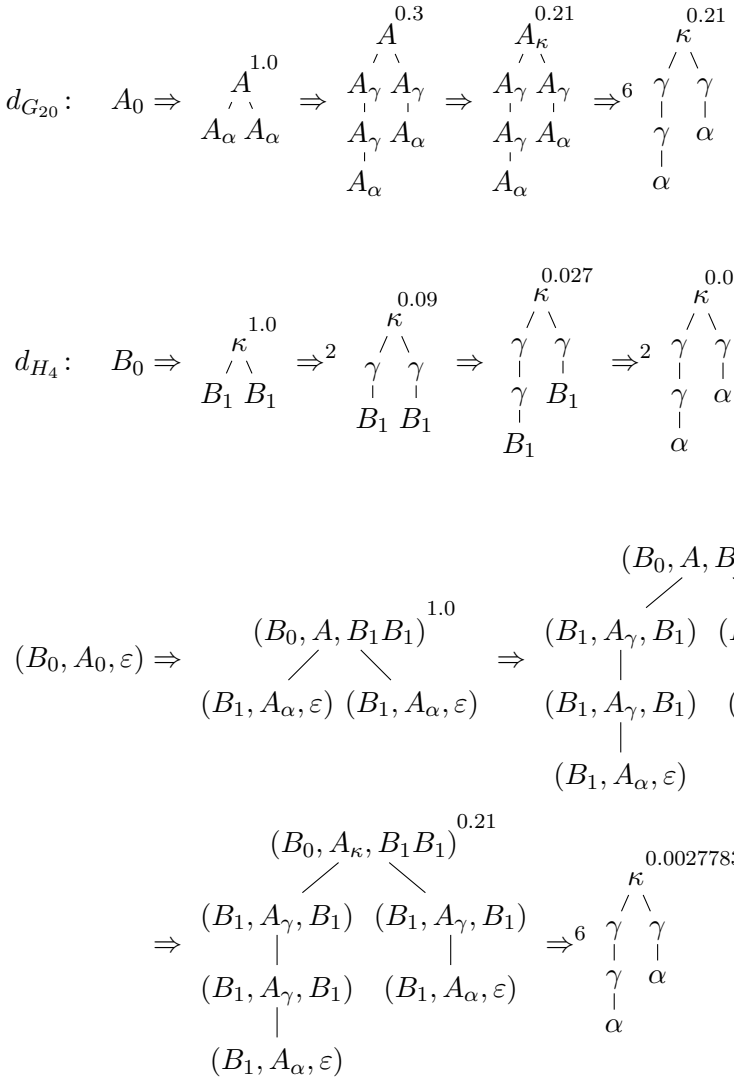


Figure 5.3: Derivations of $(G_{20}, p_{G_{20}})$, (H_4, p'), and their intersection wlnCFTG (K, p_K).

Throughout this section, *we let* (G, p_G) *be a consistent wlnCFTG where* $G = (N_G, \Delta, A_0, R_G)$, *and we let* $H = (N_H, \Delta, B_0, R_H)$ *be an unambiguous and producing RTG such that* $\mathrm{supp}(G, p_G) \cap \mathcal{L}(H) \neq \emptyset$.

According to Theorem 5.4.4, we choose (K, p_K) to be the wlnCFTG obtained by intersecting (G, p_G) and $(H, \mathbb{1})$ where $\mathbb{1}(r_H) = 1$ for each $r_H \in R_H$. We let $K = (N_K, \Delta, A_0{}', R')$ and according to Theorem 5.4.4, we have, for each $\xi \in \mathcal{L}(G) \cap \mathcal{L}(H)$, that $p_K(\xi) = p_G(\xi) \cdot \mathbb{1}(\xi)$, i.e., $p_K(\xi) = p_G(\xi)$. Since not all trees of $\mathcal{L}(G)$ need to occur in $\mathcal{L}(H)$, we normalize p_G such that only trees in $\mathcal{L}(H)$ are assigned a non-null weight.

Definition 5.5.1 We define the $\mathcal{L}(H)$*-restriction of* p_G, denoted by $p_G|_H$, for each $\xi \in \mathrm{T}_\Delta$, as

$$p_G|_H(\xi) = \begin{cases} \dfrac{p_G(\xi)}{\sum_{\xi' \in \mathcal{L}(H)} p_G(\xi')} & \text{if } \xi \in \mathcal{L}(H), \\ 0 & \text{otherwise.} \end{cases}$$ □

We note that, since (G, p_G) is consistent, $p_G|_H$ is well-defined and Definition 5.5.1 implies that $p_G|_H$ is consistent. Furthermore, if $\mathrm{supp}(G, p_G) \subseteq \mathcal{L}(H)$, then $p_G|_H = p_G$.

Lemma 5.5.2 For each $\xi \in \mathrm{T}_\Delta$, we have $p_G|_H(\xi) = \frac{p_K(\xi)}{\sum_{\xi' \in \mathrm{T}_\Delta} p_K(\xi')}$.

Proof. We observe that $\sum_{\xi' \in \mathrm{T}_\Delta} p_K(\xi') > 0$ since in this section we required that $\mathrm{supp}(G, p_G) \cap \mathcal{L}(H) \neq \emptyset$ holds. We show the following for each $\xi \in \mathcal{L}(H)$.

$$\begin{aligned} p_G(\xi) &= \sum\nolimits_{d_G \in \mathcal{D}_G(\xi)} p_G(d_G) \\ &= \sum\nolimits_{d_G \in \mathcal{D}_G(\xi), d_H \in \mathcal{D}_H(\xi)} p_G(d_G) \cdot \mathbb{1}(d_H) && (H \text{ is unambiguous}) \\ &= \sum\nolimits_{d \in \mathcal{D}_K(\xi)} p_K(d) && (\text{Clm. 2 of Thm. 5.4.2, Cor. 5.4.5}) \\ &= p_K(\xi) \end{aligned}$$

Let $\xi \in \mathrm{T}_\Delta$. If $\xi \notin \mathcal{L}(H)$, then we have $p_G|_H(\xi) = 0$ and, since $\mathcal{D}_H(\xi) = \emptyset$ and thus $\mathbb{1}(\xi) = 0$, we also have $\frac{p_K(\xi)}{\sum_{\xi' \in \mathrm{T}_\Delta} p_K(\xi')} = 0$. Hence, for each $\xi \in \mathrm{T}_\Delta$, we have

$$p_G|_H(\xi) = \frac{p_G(\xi)}{\sum_{\xi' \in \mathrm{T}_{\mathcal{L}(H)}} p_G(\xi')} = \frac{p_K(\xi)}{\sum_{\xi' \in \mathrm{T}_{\mathcal{L}(H)}} p_K(\xi')} = \frac{p_K(\xi)}{\sum_{\xi' \in \mathrm{T}_\Delta} p_K(\xi')} \ .$$ ■

In the following we will obtain a weight assignment p_H such that (H, p_H) best approximates the weighted tree language induced by (G, p_G). We measure the quality of such an approximation using the Kullback-Leibler divergence (cf. [40, Eq. 2.4]).

Definition 5.5.3 Let p and p' be two arbitrary consistent weighted tree languages. The *Kullback-Leibler (KL) divergence of* p *and* p', denoted by $\mathrm{KL}(\mathrm{p} \mid\mid \mathrm{p}')$, is given by

$$\mathrm{KL}(p \mid\mid p') = \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p(\xi) \cdot \log \frac{p(\xi)}{p'(\xi)} \ .$$ □

Note that the KL divergence is closely connected to the notion of cross-entropy. In fact, the KL divergence of p and p' is equal to the difference of the cross-entropy of p and p', and the entropy of p.

We will now formally define our goal. For this, we let P_H be the set of all proper and consistent weight assignments for H, i.e.,

$$P_H = \{p'_H \mid p'_H \colon R_H \to \mathbb{R}_{\geq 0},\ p'_H \text{ is proper and consistent}\}$$

and determine p_H such that we have

$$p_H = \operatorname{argmin}_{p'_H \in P_H} \mathrm{KL}(p_G|_H \mid\mid p'_H) \ .$$

We rephrase the KL divergence using the cross-entropy since this will help us later. We abbreviate $\sum_{\xi' \in \mathrm{T}_\Delta} p_K(\xi')$ to Z and obtain the following equations.

$$\begin{aligned}
\operatorname{argmin}_{p'_H \in P_H} \mathrm{KL}(p_G|_H \mid\mid p'_H) &= \operatorname{argmin}_{p'_H \in P_H} \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_G|_H(\xi) \cdot \log \frac{p_G|_H(\xi)}{p'_H(\xi)} \\
&= \operatorname{argmin}_{p'_H \in P_H} \sum\nolimits_{\xi \in \mathrm{T}_\Delta} \frac{p_K(\xi)}{Z} \cdot \log \frac{p_K(\xi)}{p'_H(\xi) \cdot Z} && \text{(Lm. 5.5.2)} \\
&= \operatorname{argmin}_{p'_H \in P_H} \Big(\sum\nolimits_{\xi \in \mathrm{T}_\Delta} \frac{p_K(\xi)}{Z} \cdot \log \frac{p_K(\xi)}{Z} \Big) \\
&\qquad - \Big(\frac{1}{Z} \cdot \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \log p'_H(\xi) \Big) \\
&= \operatorname{argmin}_{p'_H \in P_H} - \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \log p'_H(\xi) && (5.7)
\end{aligned}$$

Equation (5.7) holds, since Z and p_K are independent of p'_H.

In [12] it is shown how to use a weighted CFG to approximate an infinite corpus of trees. The corpus is regarded as set of derivations of the weighted CFG and is given as a probability distribution p_T over the derivations. Using the technique of Lagrange multipliers, it is shown how to choose a weight assignment $p_{\min}$ such that the cross-entropy between p_T and $p_{\min}$ is minimized. The cross-entropy corresponds to (5.7). Informally, $p_{\min}$ is defined for every rule r of the CFG as (cf. Eq. (9) of [12])

$$p_{\min}(r) = \frac{\sum_{\text{derivation tree } d} p_T(d) \cdot \sharp_r(d)}{\sum_{\text{derivation tree } d} \sum_{r' \text{ with } \mathrm{lhn}(r') = \mathrm{lhn}(r)} p_T(d) \cdot \sharp_{r'}(d)} \ . \tag{5.8}$$

This result cannot be straightforwardly applied to our scenario for two reasons. First, we are not given a distribution over derivations of H and second, in contrast to (5.8), our setting would require an unsupervised training (similar to [12, Sec. 7.2]). However, we can use the intersection grammar (K, p_K) and infer derivations for H. This approach corresponds to the string case [49] and the case of TAGs [50].

From Claim 2 of Theorem 5.4.2, we get that for each derivation in H there are multiple derivations in K, viz. one derivation $d \in \mathcal{D}_K(\xi)$ for each derivation $d_G \in \mathcal{D}_G(\xi)$ discounted by the weight of the corresponding d_G. We can thus infer the expected frequencies of rules in H by analyzing the expected frequencies of rules in K. For this to work, we need to define the following restriction of the rules in K.

Definition 5.5.4 For each $r_H \in R_H$, we define the *subset of R' corresponding to r_H* as

$$R'|_{r_H} = \{r' \in R' \mid \mathrm{cor}(r') = (r_G, r_H) \text{ for some } r_G \in R_G\} \ . \qquad \square$$

Using the set of corresponding rules allows instantiating the Lagrange multipliers in [12, Sec. 3] for our scenario. They will be used in the proof of the following lemma.

Lemma 5.5.5 We define p_H, for each $r_H \in R_H$, as

$$p_H(r_H) = \frac{\sum_{r' \in R'|_{r_H}} E(r')}{\sum_{r \in R_H|_{\mathrm{lhn}(r_H)}} \sum_{r' \in R'|_r} E(r')} \ . \tag{5.9}$$

Then it holds that $p_H = \mathrm{argmin}_{p'_H \in P_H} \mathrm{KL}(p_G|_H \mid\mid p'_H)$.

Proof. Recall that H is unambiguous, i.e., for each tree $\xi \in \mathcal{L}(H)$ there is exactly one derivation in $\mathcal{D}_H(\xi)$. This single element of $\mathcal{D}_H(\xi)$ is denoted by d_ξ.

We want to obtain $p' = \mathrm{argmin}_{p'_H \in P_H} \mathrm{KL}(p_G|_H \mid\mid p'_H)$ using the constraint that for each $A \in N$, we have $\sum_{r_H \in R_H|_A} p'(r_H) = 1$. By using the technique of Lagrange multipliers with the side condition of properness, we minimize (5.7) and thus define

$$\nabla = \sum\nolimits_{A \in N} \lambda_A \cdot \Big(\big(\sum\nolimits_{r \in R_H|_A} p'(r) \big) - 1 \Big) - \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \log p'(\xi) \ .$$

For each $A \in N$, we obtain the derivative $\frac{\partial \nabla}{\partial \lambda_A} = \sum_{r \in R_H|_A} p'(r) - 1$. Furthermore, for each $r_H \in R_H|_A$, we obtain the derivative

$$\begin{aligned}
\frac{\partial \nabla}{\partial p'(r_H)} &= \lambda_A - \frac{\partial}{\partial p'(r_H)} \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \log p'(\xi) \\
&= \lambda_A - \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \frac{\partial}{\partial p'(r_H)} \log \prod\nolimits_{r \in R_H} p'(r)^{\sharp_r(d_\xi)} \\
&= \lambda_A - \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sum\nolimits_{r \in R_H} \frac{\partial}{\partial p'(r_H)} \sharp_r(d_\xi) \cdot \log p'(r) \\
&= \lambda_A - \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_{r_H}(d_\xi) \cdot \frac{1}{\ln 2} \cdot \frac{1}{p'(r_H)} \\
&= \lambda_A - \frac{1}{\ln 2} \cdot \frac{1}{p'(r_H)} \cdot \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_{r_H}(d_\xi)
\end{aligned} \tag{5.10}$$

Setting these derivatives to 0, we obtain

$$\sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_{r_H}(d_\xi) = \lambda_A \cdot \ln 2 \cdot p'(r_H) \ . \tag{5.11}$$

Summing over all rules r in $R_H|_A$ then yields

$$\sum\nolimits_{r \in R_H|_A} \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_r(d_\xi) = \lambda_A \cdot \ln 2 \cdot \sum\nolimits_{r \in R_H|_A} p'(r)$$

and, using the constraint $\sum_{r_H \in R_H|_A} p'(r_H) = 1$, we obtain

$$\sum\nolimits_{r \in R_H|_A} \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_r(d_\xi) = \lambda_A \cdot \ln 2 \ . \tag{5.12}$$

Replacing $\lambda_A \cdot \ln 2$ in (5.11) with the value obtained in (5.12), we get

$$\sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_{r_H}(d_\xi) = \sum\nolimits_{r \in R_H|_A} \sum\nolimits_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_r(d_\xi) \cdot p'(r_H) \ .$$

Solving this equation for $p'(r_H)$, we obtain

$$p'(r_H) = \frac{\sum_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_{r_H}(d_\xi)}{\sum_{r \in R_H|_A} \sum_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_r(d_\xi)} \ .$$

Since, for each $r \in R_H$, we have that $\sum_{\xi \in \mathrm{T}_\Delta} p_K(\xi) \cdot \sharp_r(d_\xi) = \sum_{d \in \mathcal{D}_K} \sum_{r' \in R'|_r} p_K(d) \cdot \sharp_{r'}(d)$, we obtain

$$p'(r_H) = \frac{\sum_{d \in \mathcal{D}_K} \sum_{r' \in R'|_{r_H}} p_K(d) \cdot \sharp_{r'}(d)}{\sum_{r \in R_H|_A} \sum_{d \in \mathcal{D}_K} \sum_{r' \in R'|_r} p_K(d) \cdot \sharp_{r'}(d)} \ . \tag{5.13}$$

If we use the characterization of the expected frequency of a rule from (5.2), we see that (5.13) is equal to (5.9), because the sums can be reordered.

It remains to show that (5.9) is consistent. We follow the argument of the string case described in [8] for finite sample sets and [12] for infinite samples and transfer the reasoning to the tree case. We recall from Claim 1 in the proof of Theorem 5.4.2 that there is a one-to-one correspondence between positions in yield(d) and positions in d_H. As an intermediate step, we express specific counts of nonterminal occurrences in a derivation d_H of H by the corresponding derivation d of K as follows. We let $d \in \mathcal{D}_K$ be a tree-shaped derivation, $B \in N_H$, and $v = \mathrm{yield}(d)$ be of the form $v = r_1 \dots r_n$ where for each $i \in [n]$, we let $\mathrm{lhn}(r_i) = (B^i, A^i, B^i_1 \dots B^i_{\ell_i})$. Then we let

- ${}_B\sharp(d) = |\{i \in [n] \mid B^i = B\}|$ (number of leaves of d such that B is the first component of its LHS-nonterminal),
- $\sharp_{B'}(d) = |\{(i,j) \mid i \in [n],\ j \in [\ell_i],\ B^i_j = B'\}|$ (number of occurrences of B' in the third component of a LHS-nonterminal in a leaf of d), and
- ${}_B\sharp_{B'}(d) = |\{(i,j) \mid i \in [n],\ B^i = B,\ j \in [\ell_i],\ B^i_j = B'\}|$ (number of occurrences of B' in the third component of a LHS-nonterminal in a leaf of d where B is the first component).

Furthermore, for each $B \in N_H$, we let q_B denote the probability that derivations starting in B will not finish. Hence, if $q_{A_0} = 0$, then p_H is consistent. We can estimate q_B in terms of occurrences of nonterminals in the RHS of each rule in $R_H|_B$ using the inequality

$$q_B \leq \sum\nolimits_{r_H \in R_H|_B} p_H(r_H) \cdot \sum_{\substack{i \in [\ell] \\ \mathrm{ntw}(r_H) = B_1 \dots B_\ell}} q_{B_i} \ . \tag{5.14}$$

We combine (5.9) and (5.14) and obtain

$$q_B \leq \sum\nolimits_{r_H \in R_H|_B} \frac{\sum_{r' \in R'|_{r_H}} E(r')}{\sum_{r \in R_H|_{\mathrm{lhn}(r_H)}} \sum_{r' \in R'|_r} E(r')} \cdot \sum_{\substack{i \in [\ell] \\ \mathrm{ntw}(r_H) = B_1 \dots B_\ell}} q_{B_i} \ .$$

Since, for each $r_H \in R_H|_B$, we trivially have $\mathrm{lhn}(r_H) = B$, we get

$$q_B \cdot \sum\nolimits_{r \in R_H|_B} \sum\nolimits_{r' \in R'|_r} E(r') \leq \sum\nolimits_{r_H \in R_H|_B} \sum\nolimits_{r' \in R'|_{r_H}} E(r') \cdot \sum\nolimits_{\substack{i \in [\ell] \\ \mathrm{ntw}(r_H) = B_1 \dots B_\ell}} q_{B_i} \ .$$

Recall from (5.2) that $E(r')$ counts the occurrences of r in all derivations. Since we sum over all leaves where B is the first component of the LHS-nonterminal, we can simplify the inequality and obtain

$$q_B \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \cdot {}_B\sharp(d) \leq \sum\nolimits_{B' \in N_H} q_{B'} \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \cdot {}_B\sharp_{B'}(d) \ .$$

By summing over all nonterminals $B \in N_H$ we obtain

$$\begin{aligned} \sum\nolimits_{B \in N_H} q_B \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \cdot {}_B\sharp(d) &\leq \sum\nolimits_{B' \in N_H} q_{B'} \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \cdot \sum\nolimits_{B \in N_H} {}_B\sharp_{B'}(d) \\ &= \sum\nolimits_{B' \in N_H} q_{B'} \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \cdot \sharp_{B'}(d) \ . \end{aligned} \tag{5.15}$$

We let $B \in N_H \setminus \{B_0\}$ and $d \in \mathcal{D}_K$. It is intuitively clear that the number of occurrences of B's in the LHSs of rules in d equals the number of occurrences of B's in the RHSs of the rules in d. The only exception is the initial nonterminal B_0, which occurs exactly one time more is the LHSs. Thus, we observe the equations

$${}_B\sharp(d) = \sharp_B(d) \qquad \text{and} \qquad {}_{B_0}\sharp(d) = \sharp_{B_0}(d) + 1 \ . \tag{5.16}$$

If we use (5.16) in (5.15) and subtract all summands except for B_0, then we obtain

$$q_{B_0} \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \cdot {}_{B_0}\sharp(d) \leq q_{B_0} \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \cdot ({}_{B_0}\sharp(d) - 1)$$

and thus

$$q_{B_0} \cdot \sum\nolimits_{d \in \mathcal{D}_K} p_K(d) \leq 0 \ .$$

Since (G, p_G) is consistent and thus $\sum_{d \in \mathcal{D}_K} p_K(d) = 1$, we get that $q_{B_0} = 0$. Hence, p_H is a consistent weighted tree language.

This completes the proof. ∎

Note that (5.13) corresponds to (5.8) and thus gives the connection to the string case. Furthermore, note that since R_H and R' are finite sets, we obtain p_H based on $E(r')$ for each $r_H \in R_H$ and $r' \in R'|_{r_H}$. Since, for each $r_H \in R_H$ and each $r' \in R'|_{r_H}$, the rule r' is of Type II, we have $E(r') = \mathrm{outer}(\mathrm{lhn}(r')) \cdot p_K(r')$. Furthermore, because inner and outer values can be approximated to an arbitrary precision (cf. Sec. 5.3), we can effectively obtain an approximation of p_H.

Theorem 5.5.6 For each proper and consistent wlnCFTG (G, p_G) and each unambiguous and producing RTG H such that $\mathcal{L}(G) \cap \mathcal{L}(H) \neq \emptyset$, the weight assignment p_H from (5.9) is such that $p_H = \mathrm{argmin}_{p'_H \in P_H} \mathrm{KL}(p_G|_H \mid\mid p'_H)$ holds.

Proof. By Theorem 5.4.4, we obtain (K, p_K) as intersection of (G, p_G) and $(H, \mathbb{1})$. Then Lemma 5.5.5 applies and defines the minimal p_H.

Example 5.5.7 Recall G_{20} and H_4 from the beginning of this chapter as well as their intersection K from Example 5.4.1. Consider some rules of K depicted in Figure 5.2. It is easy to see that r'_1 and r'_2 occur exactly once in each derivation and thus, $E(r'_1) = E(r'_2) = 1$. The rule r'_3 occurs n times in a derivation of $\kappa(\gamma^{(2n)}(\alpha), \gamma^n(\alpha))$ and we get $E(r'_3) = \sum_{n\in\mathbb{N}} 0.3^n \cdot 0.7 \cdot n = \frac{3}{7}$. Since r'_4 occurs three times as often as r'_3, we get $E(r'_4) = 3 \cdot E(r'_3) = \frac{9}{7}$. The rule r'_5 occurs exactly twice, so $E(r'_5) = 2$. Note that these expected frequencies of the rules from K correspond to the expected frequencies of the rules of G_{20} as discussed in Example 5.3.1.

We denote the rules of H_4 from Figure 5.1(b) by r_1, r_2, and r_3. We note that r_2 and r_3 correspond to r'_3 and r'_5, respectively. Hence, we obtain an optimal p_{H_4} as

$$p_{H_4}(r_1) = 1, \qquad p_{H_4}(r_2) = \frac{\frac{9}{7}}{2 + \frac{9}{7}} = \frac{9}{23} \approx 0.39, \qquad p_{H_4}(r_3) = \frac{2}{2 + \frac{9}{7}} = \frac{14}{23} \approx 0.61 \ .$$

Although p_{H_4} is the optimal weight assignment for H_4 concerning the KL divergence to the weighted tree language induced by $(G_{20}, p_{G_{20}})$, there is a considerable approximation error since probability mass is lost on trees of shape $\kappa(\gamma^{n_1}(\alpha), \gamma^{n_2}(\alpha))$ where $n_1 \neq 2 \cdot n_2$. Improved approximation results can be obtained by considering other RTGs with a nonterminal structure that better approximates the structure of trees in $\mathcal{L}(G_{20})$. Such RTGs usually have an increased parsing time, since they contain more nonterminals and rules. ○

5.6 Remarks

The results in this chapter rely on a given unambiguous RTG. A natural question is how such a grammar can be obtained. This question brings together the results of this chapter with the results in Chapters 3 and 4. Each of the latter chapters describes a method that allows to obtain a RTG given a lnCFTG.

For the first method, we let (G, p_G) be a wlnCFTG such that G is a non-self-embedding lnCFTG. In this case, by Theorem 3.3.11, we obtain a RTG H such that $\mathcal{L}(G) = \mathcal{L}(H)$. Then H can be made unambiguous if needed through the determinization of the associated finite tree automaton (cf. [25, Ch. II, Thm. 2.6]). Since $\mathcal{L}(G) \cap \mathcal{L}(H) = \mathcal{L}(G)$, we can apply Theorem 5.5.6 if $\mathcal{L}(G) \neq \emptyset$. Thus, we obtain a weight assignment p_H such that the KL divergence between (G, p_G) and (H, p_H) is minimal.

Since our proofs in Chapter 3 are constructive, it might alternatively be possible to extend the result to the weighted case and, given a consistent (G, p_G), obtain a RTG (H, p_H) such that p_G and p_H induce the same weighted tree language. Whether this is possible for all non-self-embedding lnCFTG with a consistent and proper weight assignment, remains an open problem.

We note that the property $\mathcal{L}(G) = \mathcal{L}(H)$ does not imply equality of the weighted tree languages induced by (G, p_G) and the trained (H, p_H), respectively. This is demonstrated in the following example.

Example 5.6.1 Let $(G_{21}, p_{G_{21}})$ be a wlnCFTG such that $G_{21} = (\{A_0, A\}, \{\gamma, \alpha\}, A_0, R)$ where R contains the rules

$$A_0 \to A(\alpha) \qquad\qquad A(x_1) \to A(\gamma(\gamma(x_1))) \mid \gamma(x_1) \mid x_1$$

and $p_{G_{21}}$ assigns the weights 1, 0.7, 0.2, and 0.1 to the rules (in the order of their depiction from left to right). By inspecting the rules it is clear that $\mathcal{L}(G_{21}) = \{\gamma^n(\alpha) \mid n \in \mathbb{N}\}$ and, for each tree $\gamma^n(\alpha) \in \mathcal{L}(G)$, we have

$$p_G(\gamma^n(\alpha)) = \begin{cases} 0.7^{\lfloor \frac{n}{2} \rfloor} \cdot 0.2 & \text{if } n \text{ is odd and} \\ 0.7^{\lfloor \frac{n}{2} \rfloor} \cdot 0.1 & \text{if } n \text{ is even} \end{cases}$$

where $\lfloor \frac{n}{2} \rfloor$ is the natural number obtained by rounding $\frac{n}{2}$ downwards.

We want to characterize G_{21} by the unambiguous RTG $H_5 = (\{B_0\}, \{\gamma, \alpha\}, B_0, R')$ where R' contains the two rules

$$B_0 \to \gamma(B_0) \mid \alpha \ .$$

We denote the rules of R' by r_1 and r_2. It is easy to see that $\mathcal{L}(G_{21}) = \mathcal{L}(H_5)$. However, there is no weight assignment p_{H_5} for H_5 such that $p_{G_{21}}$ and p_{H_5} induce the same weighted tree language. This can be seen by considering the trees α, $\gamma(\alpha)$, and $\gamma^2(\alpha)$. We have $p_{G_{21}}(\alpha) = 0.1$, $p_{G_{21}}(\gamma(\alpha)) = 0.2$, and $p_{G_{21}}(\gamma^2(\alpha)) = 0.07$. All three trees have a unique derivation in H_5 and thus, if $p_{G_{21}} = p_{H_5}$, then the equations

$$p_{H_5}(r_2) = 0.1 \ , \qquad p_{H_5}(r_1) \cdot p_{H_5}(r_2) = 0.2 \ , \qquad \text{and} \qquad p_{H_5}(r_1)^2 \cdot p_{H_5}(r_2) = 0.07$$

must be true. It can be seen that there is no solution to this system of equations. Hence, there is no weight assignment p_{H_5} for H_5 such that (H_5, p_{H_5}) induces the same weighted tree language as $(G_{21}, p_{G_{21}})$. However, by using Lemma 5.5.5, the best possible p_{H_5} according to the KL divergence can be approximated.

We note that, for this example, there is a wRTG (H_5', p'_{H_5}) such that $p_{G_{21}} = p'_{H_5}$. We leave the question open, whether there is such a wRTG for all CFTGs that induce a regular tree language. ○

The second method to obtain a suitable RTG H given the wlnCFTG (G, p_G) stems from Chapter 4. This method is applicable to arbitrary wlnCFTG, even if G self-embedding. Furthermore, depending on the requirements on the quality of the approximation, using the pushdown approximation from Chapter 4 might yield a smaller (and thus more efficient) result than the characterization described in Chapter 3. The method described in Chapter 4 applied to (G, p_G) results in a pushdown approximation H. The RTG H can be made unambiguous using again determinization of the associated finite state tree automaton. Since $\mathcal{L}(H) \supseteq \mathcal{L}(G)$, Theorem 5.5.6 applies if $\mathcal{L}(G) \neq \emptyset$. As explained in Lemma 4.2.15, choosing a suitable pushdown limit allows adjusting the tradeoff between expressiveness and computational cost.

As a final remark, we consider the described method to make a RTG unambiguous. The standard construction for determinizing the equivalent finite tree automaton [25, Ch. II, Thm. 2.6] might lead to an exponential blowup, because it involves a powerset construction. Hence, this construction should only be applied if necessary. Checking, whether a finite tree automaton is unambiguous can be done in time quadratic in the size of the automaton [61, Prop. 1.2]. Furthermore, it might be worthwhile to investigate other methods how a RTG can be made unambiguous. This is not considered in this thesis.

6 Conclusion and Further Research

In the introduction, the classical tradeoff in machine translation between the expressiveness of a formal grammar and its computational complexity was described (cf. Section 1.1). It was argued that this tradeoff explicitly applies to CFTGs, since they generalize multiple formal grammars used in machine translation, but are difficult to parse. To investigate the tradeoff, this thesis offered three methods to express context-free tree languages by RTG. The general idea of each method was described in Sections 1.2, 1.3, and 1.4, respectively. In the following, the results obtained for each methods are broadly summarized and further research opportunities are addressed.

A detailed list of the main contributions presented in this thesis can be found in the corresponding sections of the introduction (cf. Sections 1.2, 1.3, and 1.4).

Characterization by RTG

In Chapter 3, the tree languages induced by restricted CFTGs were *characterized* by RTGs. It was shown that non-regular phenomena in trees may occur (i) above and below repeated nonterminal occurrences, or (ii) in two argument positions of recursive nonterminal occurrences. By excluding both possible sources of non-regularity, two restrictions for CFTGs were obtained that imply regularity of the induced tree languages. In more detail, it was shown that non-self-embedding linear CFTGs (cf. Theorem 3.6.2) and non-weakly-self-embedding CFTGs (cf. Theorem 3.7.12) induce regular tree languages. Furthermore, the involved proofs are constructive, i.e., for each restricted CFTG, an equivalent RTG can be obtained effectively. As an extension of this thesis, it is interesting to investigate whether such a characterization is also possible if weighted CFTGs are given. While doing this, it might be worthwhile to study whether the same properties of non-self-embedding lCFTG and non-weakly-self-embedding CFTG can be used.

As a main analysis tool, the position graph of a CFTG was introduced. This graph can be used to analyze which nonterminals are participating in the recursive generation of symbols. It was shown that each non-self-embedding lnCFTG can be broken down in different parts that induce a regular tree language. The combination of these parts is an alternative proof for the regularity of the tree language induced by the non-self-embedding lnCFTG (cf. Theorem 3.5.23). Thus, the position graph is a tool to analyze which nonterminals contribute to the complexity of the induced tree language and thus gives insight about the structure of the CFTG. In future work, an approximation of the language induced by an arbitrary CFTG can be investigated that is obtained by approximating only some (difficult) parts of the CFTG identified by the position graph.

Approximation

In Chapter 4, the tree languages induced by arbitrary CFTG were *approximated* by RTGs. For this, the characterization of the tree language induced by a CFTG in terms of a RTG

with a pushdown storage was restricted. It was shown that this restriction can be chosen such that it yields a RTG that is a superset approximation (cf. Lemma 4.2.8). Furthermore, there is a hierarchy of improving approximations depending on the size of the restricted pushdown (cf. Theorem 4.2.16). Hence, the size of the pushdown adjusts the tradeoff between the quality of the approximation and the computational complexity.

Future work involves formally investigating more strategies to restrict the pushdown size, and finding a weighted version of the result. Furthermore, the approximation can be used to express parts of non-self-embedding lCFTG as explained above. It might also be interesting whether different pushdown symbols can be used. The choice of pushdown symbols may possibly respect information obtained from the position graph of a CFTG, e.g., the calling order or cycles in a SCC.

Training of a Weighted RTG

In Chapter 5, it was shown how weighted context-free tree languages can be *approximated* by weighted RTGs. In particular, it was outlined how weights for a given unambiguous RTG can be approximated such that the induced weighted tree language is close to the weighted tree language induced by a weighted lnCFTG (cf. Theorem 5.5.6). This can be seen as part of the training step in machine translation by considering the weighted lnCFTG as training data.

Further research might address more ways how both given grammars can be obtained. For instance, the weighted lnCFTG might be hand-crafted by linguists, or automatically obtained from a set of sample trees. The unambiguous RTG can be approximated from the unweighted lnCFTG or obtained independently from the same set of sample trees. The latter approach seems useful if the weights for the lnCFTG are provided by experts, since the representation of a tree language as a CFTG might be smaller than its representation by a RTG (cf. Lemma 3.10.2), and thus, CFTGs are more feasible for hand-crafted annotations.

Furthermore, unambiguous RTGs can be investigated. For instance, better transformations from RTGs into unambiguous RTGs are useful to improve the result presented in this thesis. Alternatively, it is interesting whether the property of unambiguity used for the results can be relaxed, e.g., by considering finite degrees of ambiguity.

Bibliography

[1] A. V. Aho. Indexed Grammars - An Extension of Context-Free Grammars. *Journal of the ACM* 15.4 (1968), 647–671. DOI: 10.1145/321479.321488.

[2] A. Arnold and M. Dauchet. Transductions de Forets Reconnaissables Monadiques Forets Coregulieres. *Informatique Théorique et Applications* 10.1 (1976), 5–28.

[3] T. P. Baker. Extending Lookahead for LR Parsers. *Journal of Computer and System Sciences* 22.2 (1981), 243–259. ISSN: 0022-0000. DOI: 10.1016/0022-0000(81)90030-1.

[4] M. E. Bermudez and K. M. Schimpf. Practical Arbitrary Lookahead LR Parsing. *Journal of Computer and System Sciences* 41.2 (1990), 230–250. DOI: 10.1016/0022-0000(90)90037-L.

[5] S. Bozapalidis. Context-Free Series on Trees. *Information and Computation* 169.2 (2001), 186–229. DOI: 10.1006/inco.2001.2890.

[6] W. S. Brainerd. Tree Generating Regular Systems. *Information and Control* 14.2 (1969), 217–231. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(69)90065-5.

[7] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics* 19.2 (June 1993), 263–311. ISSN: 0891-2017.

[8] Z. Chi and S. Geman. Estimation of Probabilistic Context-free Grammars. *Journal Computational Linguistics* 24.2 (1998), 299–305. ISSN: 0891-2017.

[9] N. Chomsky. Three Models for the Description of Language. *IRE Transactions on Information Theory* 2 (1956), 113–124.

[10] N. Chomsky. A Note on Phrase Structure Grammars. *Information and Control* 2.4 (1959), 393–395. DOI: 10.1016/S0019-9958(59)80017-6.

[11] N. Chomsky. On Certain Formal Properties of Grammars. *Information and Control* 2.2 (1959), 137–167.

[12] A. Corazza and G. Satta. Probabilistic Context-Free Grammars Estimated from Infinite Distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.8 (2007), 1379–1393. DOI: 10.1109/TPAMI.2007.1065.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 3rd edition. The MIT Press, 2009. ISBN: 0262033844, 9780262033848.

[14] B. Courcelle. An axiomatic definition of context-free rewriting and its application to NLC graph grammars. *Theoretical Computer Science* 55.2 (1987), 141–181. ISSN: 0304-3975. DOI: 10.1016/0304-3975(87)90102-2.

[15] W. Damm. The IO- and OI-Hierarchies. *Theoretical Computer Science* 20.2 (1982), 95–207. DOI: 10.1016/0304-3975(82)90009-3.

[16] M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. 1st. Springer Publishing Company, Incorporated, 2009. ISBN: 3642014917, 9783642014918.

[17] J. Engelfriet. *Context-Free Grammars with Storage*. Tech. rep. 86-11. Republished 2014. University of Leiden, 1986. URL: http://arxiv.org/abs/1408.0683.

[18] J. Engelfriet and E. M. Schmidt. IO and OI. I. *Journal of Computer and System Sciences* 15.3 (1977), 328–353. DOI: 10.1016/S0022-0000(77)80034-2.

[19] J. Engelfriet and H. Vogler. Pushdown Machines for the Macro Tree Transducer. *Theoretical Computer Science* 42 (1986), 251–368. DOI: 10.1016/0304-3975(86)90052-6.

[20] M. Fischer. Grammars with Macro-Like Productions. PhD thesis. Harvard University, Massachusetts, 1968.

[21] A. Fujiyoshi and T Kasai. Spinal-Formed Context-Free Tree Grammars. *Theory of Computing Systems* 33.1 (2000), 59–83. ISSN: 1432-4350. DOI: 10.1007/s002249910004.

[22] A. Fujiyoshi. Restrictions on Monadic Context-Free Tree Grammars. In: *Proceedings of Coling 2004*. COLING, Aug. 2004, 78–84. DOI: 10.3115/1220355.1220367.

[23] A. Fujiyoshi. Linearity and Nondeletion on Monadic Context-Free Tree Grammars. *Information Processing Letters* 93.3 (2005), 103–107. DOI: 10.1016/j.ipl.2004.10.008.

[24] K. Gebhardt and J. Osterholzer. A Direct Link Between Tree-Adjoining and Context-Free Tree Grammars. In: *Proceedings of the 12th International Conference on Finite-State Methods and Natural Language Processing*. Ed. by T. Hanneforth and C. Wurm. 2015. URL: http://aclweb.org/anthology/W15-4805.

[25] F. Gécseg and M. Steinby. *Tree Automata*. Ed. by T. Dunkelberger. See also arXiv:1509.06233. Akadémiai Kiadó, Budapest, 1984. ISBN: 963-05-3170-4.

[26] F. Gécseg and M. Steinby. Tree Languages. In: *Handbook of Formal Languages*. Ed. by G. Rozenberg and A. Salomaa. Vol. 3. Springer, 1997, 1–68.

[27] D. Gildea. Optimal Parsing Strategies for Linear Context-Free Rewriting Systems. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Ed. by C. P. Rosé. Association for Computational Linguistics, 2010, 769–776. ISBN: 1-932432-65-5.

[28] E. Grimley-Evans. Approximating Context-Free Grammars with a Finite-State Calculus. In: *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1997, 452–459. DOI: 10.3115/979617.979675.

[29] I. Guessarian. Pushdown Tree Automata. *Mathematical Systems Theory* 16.1 (1983), 237–263. DOI: 10.1007/BF01744582.

[30] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Cambridge, 1979.

[31] J. E. Hopcroft and J. D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley Longman Publishing Co., Inc., 1969.

[32] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. In: *Automata, Languages and Programming: 18th International Colloquium Madrid, Spain, July 8–12, 1991 Proceedings*. Ed. by J. L. Albert, B. Monien, and M. R. Artalejo. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, 629–640. ISBN: 978-3-540-47516-3. DOI: 10.1007/3-540-54233-7_169.

[33] A. Joshi and Y. Schabes. Tree-Adjoining Grammars. In: *Handbook of Formal Languages*. Ed. by G. Rozenberg and A. Salomaa. Vol. 3. Springer, 1997, 69–123.

[34] A. K. Joshi. Tree Adjoining Grammars: How much Context-Sensitivity is Required to Provide Reasonable Structural Descriptions? In: *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*. Ed. by D. R. Dowty, L. Karttunen, and A. M. Zwicky. Cambridge: Cambridge University Press, 1985, 206–250. DOI: 10.1017/CBO9780511597855.007.

[35] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences* 10.1 (1975), 136–163. DOI: 10.1016/S0022-0000(75)80019-5.

[36] M. Kanazawa. Multidimensional Trees and a Chomsky-Schützenberger-Weir Representation Theorem for Simple Context-Free Tree Grammars. *Journal of Logic and Computation* (2014). DOI: 10.1093/logcom/exu043.

[37] M. Kanazawa. A Generalization of Linear Indexed Grammars Equivalent to Simple Context-Free Tree Grammars. In: *Formal Grammar*. Ed. by G. Morrill, R. Muskens, R. Osswald, and F. Richter. Vol. 8612. Lecture Notes in Computer Science. Springer, 2014, 86–103. DOI: 10.1007/978-3-662-44121-3_6.

[38] 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 (2011), 361–384. ISSN: 0925-8531. DOI: 10.1007/s10849-011-9134-0.

[39] W. Kuich. Tree Transducers and Formal Tree Series. *Acta Cybernetica* 14.1 (1999), 135–149.

[40] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22.1 (1951), 79–86. DOI: 10.1214/aoms/1177729694.

[41] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language* 4 (1990), 35–56.

[42] P. M. Lewis II and R. E. Stearns. Syntax-Directed Transduction. *Journal of the ACM* 15.3 (July 1968), 465–488. DOI: 10.1145/321466.321477.

[43] A. Maletti and J. Engelfriet. Strong Lexicalization of Tree Adjoining Grammars. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Ed. by H. Li, C.-Y. Lin, M. Osborne, G. G. Lee, and J. C. Park. Association for Computational Linguistics, 2012, 506–515.

[44] A. Maletti and G. Satta. Parsing Algorithms Based on Tree Automata. In: *Proceedings of the 11th International Conference on Parsing Technologies*. Association for Computational Linguistics, 2009, 1–12. URL: http://dl.acm.org/citation.cfm?id=1697236.1697238.

[45] A. Maletti, J. Graehl, M. Hopkins, and K. Knight. The Power of Extended Top-Down Tree Transducers. *SIAM Journal on Computing* 39.2 (2009), 410–430. DOI: 10.1137/070699160.

[46] M. Mohri and M.-J. Nederhof. Regular Approximation of Context-Free Grammars through Transformation. In: *Robustness in Language and Speech Technology*. Ed. by J. Junqua and G. van Noord. Kluwer Academic Publishers, 2000, 251–261.

[47] M.-J. Nederhof. Regular Approximation of CFLs: A Grammatical View. In: *Advances in Probabilistic and Other Parsing Technologies*. Ed. by H. Bunt and A. Nijholt. Vol. 16. Text, Speech and Language Technology. Springer, 2000, 221–241. DOI: 10.1007/978-94-015-9470-7_12.

[48] M.-J. Nederhof. Practical Experiments with Regular Approximation of Context-free Languages. *Computational Linguistics* 26.26 (1 2000), 17–44. DOI: 10.1162/089120100561610.

[49] M.-J. Nederhof. A General Technique to Train Language Models on Language Models. *Computational Linguistics* 31.2 (2005), 173–186. DOI: 10.1162/0891201054223986.

[50] M.-J. Nederhof. Weighted Parsing of Trees. In: *Proceedings of the 11th International Conference on Parsing Technologies*. Association for Computational Linguistics, 2009, 13–24. URL: http://dl.acm.org/citation.cfm?id=1697236.1697239.

[51] 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 (2016), 203–246.

[52] 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*. 2012, 55–63.

[53] J. Osterholzer. Pushdown Machines for Weighted Context-Free Tree Translation. In: *Proceedings of 19th International Conference on Implementation and Application of Automata*. Ed. by M. Holzer and M. Kutrib. Vol. 8587. Lecture Notes in Computer Science. 2014, 290–303.

[54] R. Parchmann and J. Duske. Self-Embedding Indexed Grammars. *Theoretical Computer Science* 47 (1986), 219–223. DOI: 10.1016/0304-3975(86)90147-7.

[55] F. C. N. Pereira and R. N. Wright. Finite-state Approximation of Phrase Structure Grammars. In: *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1991, 246–255. DOI: 10.3115/981344.981376.

[56] G. K. Pullum and G. Gazdar. Natural Languages and Context-Free Languages. *Linguistics and Philosophy* 4.4 (1982), 471–504.

[57] W. C. Rounds. Complexity of Recognition in Intermediate Level languages. In: *IEEE Conference Record of 14th Annual Symposium on Switching and Automata Theory*. 1973, 145–158. DOI: 10.1109/SWAT.1973.5.

[58] 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*. 1970, 109–116. DOI: 10.1145/800161.805156.

[59] W. C. Rounds. Mappings and Grammars on Trees. *Mathematical Systems Theory* 4.3 (1970), 257–287. DOI: 10.1007/BF01695769.

[60] D. Scott. Some Definitional Suggestions for Automata Theory. *Journal of Computer and System Sciences* 1.2 (1967), 187–212. DOI: 10.1016/S0022-0000(67)80014-X.

[61] H. Seidl. On the Finite Degree of Ambiguity of Finite Tree Automata. *Acta Informatica* 26.6 (1989), 527–542. DOI: 10.1007/BF00263578.

[62] H. Seki and Y. Kato. On the Generative Power of Multiple Context-Free Grammars and Macro Grammars. *IEICE Transactions on Information and Systems* E91.D.2 (2008), 209–221. DOI: 10.1093/ietisy/e91-d.2.209.

[63] H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On Multiple Context-Free Grammars. *Theoretical Computer Science* 88.2 (1991), 191–229. ISSN: 0304-3975. DOI: 10.1016/0304-3975(91)90374-B.

[64] S. Shieber. Evidence Against the Context-Freeness of Natural Language. *Linguistics and Philosophy* 8 (1985), 333–343.

[65] H. Stamer. *Restarting Tree Automata. Formal Properties and Possible Variations.* kassel university press GmbH, 2008. ISBN: 9783899586350.

[66] M. Teichmann. Regular Approximation of Weighted Linear Nondeleting Context-Free Tree Languages. In: *Proceedings of the 21st International Conference on Implementation and Application of Automata*. Ed. by Y.-S. Han and K. Salomaa. Vol. 9705. Lecture Notes in Computer Science. Springer, 2016, 273–284. DOI: 10.1007/978-3-319-40946-7_23.

[67] 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 (1968), 57–81. DOI: 10.1007/BF01691346.

[68] K. Vijay-Shankar and A. K. Joshi. Some Computational Properties of Tree Adjoining Grammars. In: *Proceedings of the 23rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1985, 82–93. DOI: 10.3115/981210.981221.

[69] K. Vijay-Shanker, D. Weir, and A. Joshi. Characterizing Structural Descriptions Produced by Various Grammatical Formalisms. In: *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1987, 104–111.

Index

Naming Scheme

Symbol	Explanation
a, b	Unranked symbols
α, β, γ, δ, σ, κ	Ranked symbols
A, B, (C, D)	Nonterminals (C, D only in Chapter 3)
A_0, B_0	Initial nonterminal
$\mathbb{B}$	Set of boxed symbols (only Section 3.5)
c_{in}	Initial configuration of a storage type
C	No subscript: Set of configurations of a storage type
	C_P: Finite set of predicates
	C_F: Finite set of functions
$\mathrm{C}_\Delta(U)$	Context over Δ where each symbol of U occurs exactly once
d	Derivation
$\mathbf{d}$	Placeholder (dynamic position)
$\mathcal{D}$	Set of derivations
Δ	Ranked alphabet
ε	Empty word
E	Edges of a graph (V, E)
f	Function
g	Label for a generating edge
γ	Chapter 4: Pushdown symbol
	Otherwise: ranked terminal
G	Context-free tree grammar
G_S	Chapter 4: Context-free tree grammar using storage S
$\mathcal{G}(G_S)$	Chapter 4: CFTG associated with G_S
Γ, Γ_Gu	Chapter 4: Pushdown alphabet, Pushdown alphabet of G
H	Regular tree grammar
i, j, k, ℓ, m, n, q	Natural number
j_B	Dynamic argument position of B (only Chapter 3)
K	Chapter 3: Finite set (of nonterminals)
	Chapter 5: lnCFTG obtained by intersection
L	Language
$\mathcal{L}(\_)$	Induced language of a grammar
log, ln	Binary Logarithm; natural logarithm
μ	Chapter 4: meaning function
M	String grammar (e.g. REG, CFG, or MAC)
M_P	Set of nonterminals in the SCC P (only Chapter 3)
N	Set of nonterminals (ranked or unranked)
$\mathbb{N}$, $\mathbb{N}_+$	The set of natural numbers 0, 1, 2, ...; with subscript $_+$: excluding 0

Symbol (cont.)	Explanation (cont.)
p	Chapter 3: Path in a graph
	Chapter 5: Weighted tree language
p_H, p_G, ...	Weight assignment, weighted tree language
φ	Function
π	Permutation
P	SCC of a graph
P_0, P_i, P_n	Finite set of nonterminals (only Chapter 3)
P_H	Set of proper weight assignments for RTG H (only Chapter 5)
$\mathcal{P}(\_)$	Powerset
$\mathrm{pos}(\_)$	The set of positions (of a tree)
r	Rule of a grammar
$\mathrm{rk}_\Delta(\_)$	Rank function
R	Set of rules
$\mathbb{R}$	Real numbers; with subscript $_{\geq 0}$: positive real numbers
s, t	String over any alphabet
	Section 3.8: Nested term
S	Storage Type
T	Set of Terms
$\mathrm{T}_\Delta(U)$	Trees using ranked alphabet Δ and nullary symbols from the set U
u	Chapter 2: Element of a set U
	Section 3.8: Nested term
U	A finite set
Σ	Unranked alphabet
v	String over any alphabet
V	Vertices of a graph (V, E)
w	Position (in a tree)
W	Set of positions
x, z	Variable (z especially for contexts)
$\overline{x}$	Sequence of variables
ξ, ζ	Tree
X	Set of all variables x_i; with subscript k: only containing k variables
$\emptyset$	Empty set
$[k]$	Set of the numbers $1, \ldots, k$

Acknowledgments

My sincerest thanks are dedicated to my supervisor Heiko Vogler for his continued support, guidance, criticism, encouragement, time spent discussing my ideas, and for putting his trust into me! I wish to thank Mark-Jan Nederhof and Heiko Vogler as my coauthors for the result about non-self-embedding CFTG. The results in this level of detail would not have been achieved without their ideas, stamina, and fruitful discussions.

A special thanks goes to my colleagues and friends who supported me with valuable input, commented on my preliminary results, and supported me in various different ways.

I want to thank the research training group QuantLA (DFG Graduiertenkolleg 1763) for the possibility to exchange ideas with many researchers worldwide and for the financial support during the time I spent researching. I also like to thank the Graduate Academy of Technische Universität Dresden for financing the final months of writing this thesis.