Expressing Context-Free Tree Languages by Regular Tree Grammars

Extended Abstract

Markus Teichmann

In machine translation, there is a classical tradeoff between the expressivity of a grammar formalism and its computational complexity. The tree languages induced by context-free tree grammars (CFTGs) subsume the formal languages of many important grammar formalisms that are used in the area of machine translation. However, the computational cost required to work with CFTGs is high and the formalism might be unfeasible for certain applications. Hence, it is worthwhile to investigate the expressivity of CFTGs and find ways to reduce the complexity.

In the thesis, three methods are explored to express context-free tree languages, i.e., languages that are induced by CFTGs, by regular tree grammars (RTGs). A RTG is a CFTG with strong syntactic restrictions and thus, it can be efficiently used for computations. Expressing context-free tree languages by RTGs unveils details about the complexity of CFTGs and provides knowledge about the two fundamental grammar formalisms.

The first method finds restrictions for CFTGs such that each CFTG which complies with these restrictions can be *characterized* by a RTG. The restrictions are given in form of syntactic and decidable properties. Furthermore, it is shown how to construct a RTG which is equivalent to a given restricted CFTG. The properties are inspired by *non-self-embedding* context-free grammars [3, 10]. The second method *approximates* the tree language induced by an arbitrary CFTG. An approximation in this sense is a RTG that induces a superset of the tree language generated by the CFTG. In this way, non-regular dependencies within a tree are broken. This method uses an already known result that shows how context-free tree languages can be characterized by RTG extended by a pushdown storage [9, 6]. This characterization is turned into an approximation by restricting the pushdown storage and only allow a finite amount of storage configurations. To obtain the approximation, the finite information from the pushdown storage is incorporated into the nonterminals of the RTG. Whenever information is lost by the restriction of the pushdown, the missing data is guessed. Hence, a superset approximation is obtained (cf. [1, 2, 15, 8, 10] for similar results for strings). The third method relates weighted variants of CFTG and RTG. Weighted CFTG induce a weighted tree grammar, i.e., each tree from the tree language induced by the grammar is associated with a weight from the reals. The third method describes how the weights for a weighted RTG can be *approximated* such that the induced weighted tree language is as close as possible to the weighted tree language induced by a given weighted linear nondeleting CFTG.

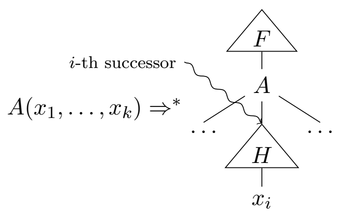
In the following, each of the three methods will be explained in more detail.

1 Characterization by RTG

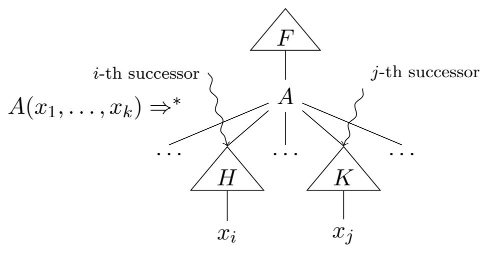
This part of the thesis is based on the publication [12]. The first method finds restrictions such that each CFTG that fulfills the restrictions induces a regular tree language, i.e., there is a RTG that induces the same tree language. For this, phenomena are considered that correspond to non-regular behavior. Since copying of argument values is a potential source of non-regularity, the investigation is initially restricted to linear CFTGs (lCFTGs), i.e., CFTGs that cannot copy values in argument positions. In derivations of a lCFTG, there are two remaining phenomena that indicate possible non-regular behavior.

The first phenomenon can be seen by considering a nonterminal A with one argument position. If, in a derivation starting from $A(x_1)$, the nonterminal A occurs again with x_1 in its argument position and symbols have been produced above *and* below the repeated occurrence, then these symbols are regarded as *synchronously* generated. The synchronous generation can be repeated an unbounded number of times. Each repetition separates the symbols that were produced earlier. Hence, there are arbitrary many synchronized and separated symbols. Such generation cannot be modeled by a RTG, since a RTG cannot produce symbols below a nonterminal.

The second phenomenon can be illustrated by considering a nonterminal A with two argument positions. If there is a derivation starting from $A(x_1, x_2)$ such that (i) the nonterminal A occurs again, (ii) x_1 occurs in its first argument positions, (iii) x_2 occurs in its second argument position, and (iv) in both argument positions there are more symbols than just the respective variable, then the symbols in the argument positions are regarded as synchronously generated. Repeating this synchronous production leads to an arbitrary amount of such synchronized symbols. Hence, since RTG cannot produce symbols in the argument positions of nonterminals, this is an indicator for non-regular behavior. If a lCFTG includes one of both phenomena, then it is called *self-embedding* (cf. [3, 10] for a similar result for context-free grammars). Figure 1 illustrates both phenomena in a general setting.



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



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

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

Both phenomena are defined on a semantic level, i.e., they require to prove the existence of a certain derivation. However, it is also shown how to syntactically detect whether a lCFTG is self-embedding by a graph analysis. The graph in question is build from the rules of the lCFTG and represents which nonterminals synchronously generate symbols in which of their argument positions. Thus, the position graph is an indicator for the complexity of a lCFTG.

Since the second phenomenon has no equivalent in the string case [3, 10], novel approaches are presented to show that each non-self-embedding lCFTG induces a regular tree language. This seems intuitively clear, since non-regular behavior is forbidden. However, the proof requires attention to detail. First, unbounded generation of symbols that happens in two argument positions of the same nonterminal is separated into distinct nonterminals. This is possible, because such generation must be independent. Afterwards, unbounded generation below one nonterminal is transformed such that it happens above a new nonterminal in reversed order. After these two steps, there is no more unbounded generation below a nonterminal and thus, the finite information that can occur below a nonterminal can be encoded into the nonterminals of a RTG. The constructed RTG induces the same language as the original non-self-embedding lCFTG.

For the above considerations, copying of argument values is explicitly forbidden by investigating lCFTGs. A second decidable property, called *non-weakly-self-embedding*, is shown that applies to all CFTGs and guarantees that a non-weakly-self-embedding CFTG induces a regular tree language. The property disallows unbounded generation below a nonterminal, even if it only occurs in one argument position. Thus, a RTG that induces the tree language of a non-weakly-self-embedding CFTG can be constructed, since in all derivations only finitely many trees may occur below a nonterminal occurrence.

It is known that the yields of context-free tree languages correspond to the languages induced by macro grammars (introduced by [7]). This constructive result (cf. [16, p. 113] and [5, Thm. 7.17]) is used to transfer the properties of being non-self-embedding and being non-weakly-self-embedding to the realm of macro grammars. It is shown that non-self-embedding linear macro grammars and non-weakly-self-embedding macro grammars induce context-free string grammars.

An overview of the results can be found in Figure 2 for different restrictions of CFTGs. The CFTGs belonging to the hatched areas induce regular tree languages.

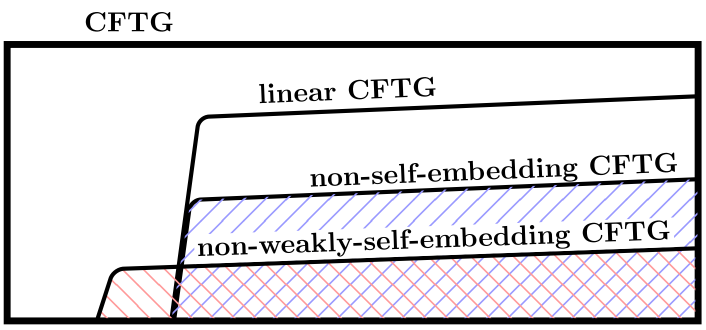


Figure 2: An overview of the characterization results.

2 Approximation

The second method to express a context-free tree language by a RTG approximates the given tree language. For this, the already known characterization of a context-free tree language in terms of a RTG with pushdown storage is used [9, 6]. The characterization is realized by traversing the right-hand sides of the rules of a given CFTG. Whenever a nonterminal A is processed, the traversal of the current rule is interrupted. The rule and position within that rule is stored as a *return address* onto the pushdown and traversal continues at the root of a rule for A. If a variable is processed, then the traversal continues at the position determined by the most recently pushed return address. In this way, using pushdown configurations of arbitrary size, each context-free tree language can be characterized by a RTG with an associated pushdown.

Following analogous results for the string case [1, 2, 15, 8, 10], this approach can be modified to obtain an approximation of context-free tree languages. In more detail, a RTG that induces a superset of the tree language induced by an arbitrary CFTG can be obtained as follows. The size of the pushdown is limited to a fixed height. Whenever a return addressed is pushed such that the pushdown would exceed the allowed height, the return address on the bottom of the pushdown is forgotten. The derivation of the RTG continues as discussed above and the pushdown associated with the nonterminals remains smaller than the limit. If the traversal requires a return address but the pushdown storage is empty, then one return address is nondeterministically guessed from all possible return addresses and the traversal continues. Since there are only finitely many return addresses and the pushdown never exceeds a fixed height, only finite information can be represented in the pushdown storage. This finite information can be encoded into the nonterminals of a RTG. The RTG obtained in this way is called a pushdown approximation.

A derivation of an arbitrary CFTG can be modeled by a pushdown approximation if at each nondeterministic choice, the correct return address is guessed. Hence, each tree that can be derived by the CFTG can also be derived by the pushdown approximation. However, since the pushdown is copied to all nonterminals in the right-hand-side of a rule but each nonterminal guesses a return address independently when reading an empty pushdown, the pushdown approximation may derive trees that do not belong to the context-free tree language. Thus, a pushdown approximation is a superset approximation.

One factor of the complexity of a CFTG is the size of the pushdown that is required to characterize it. If the limit of the pushdown approximation is never reached, then the approximation becomes a characterization. However, it is shown that there are CFTGs that require arbitrary large pushdown configurations. Furthermore, for such CFTGs, an increased limit on the pushdown storage improves the approximation, i.e., some trees are not generated that would have been produced using a lower limit. Formally, a hierarchy of improving pushdown approximations is presented.

3 Training

This part of the thesis is based on the conference article [18]. The third method shows how certain weighted context-free tree languages can be approximated by RTGs. For this, weighted linear nondeleting CFTGs are considered which are explained in the following. A linear nondeleting CFTG (lnCFTG) may neither delete nor copy argument values in a derivation. A weighted CFTG is a CFTG together with an assignment from its rules to the positive reals, called the *weight* of a rule. For each derivation of a weighted CFTG, a weight is obtained as follows. At each derivation step, the weight of the applied rule is multiplied with the value obtained from the remaining steps. The weight of a tree is then defined by summing up the weights of all derivations that produce this tree. A weighted RTG is defined analogously.

Given a weighted lnCFTG and a RTG, it is shown how weights for the rules of the RTG can be obtained such that the induced weighted tree languages of both grammars are as close as possible considering the Kullback-Leibler divergence. This divergence is 0, if the induced weighted tree languages are equal and greater than 0 otherwise. The Kullback-Leibler divergence is chosen, since it is broadly used in the existing literature on this topic. The approach is based on a similar result for the training of a finite string automaton [11] and for the training of a context-free string grammar [4].

In more detail, the approximation is calculated as follows. First, a given RTG H is enriched with a trivial weight assignment $\mathbb{1}$ that assigns the weight 1 to each rule. Then, the weighted intersection between the weighted RTG $(H, \mathbb{1})$ and the given weighted lnCFTG (G, p_G) is constructed. This intersection is represented by a weighted lnCFTG (K, p_K) which, for each tree in the intersection of the tree languages of G and H, assigns the product of the weights assigned by each weighted grammar (cf. [16, 17, 13, 14] for similar constructions).

Due to a special requirement upon H, it turns out that (K, p_K) assigns to each tree that is in the tree language of both G and H, the value that is assigned by G. Furthermore, considering the weighs as probabilities, it is possible to approximate expected frequencies for each rule in derivations of (K, p_K).

The construction of K couples derivations from G and H. The coupling allows transferring the expected frequencies from (K, p_K) to the rules of H. Then, using the obtained expected frequencies, weights for the rules of H can be calculated. It is shown that the weights obtained in this way are optimal regarding the Kullback-Leibler divergence to the weighted tree language induced by (G, p_G).

In machine translation, this approximation of the optimal weights can be regarded as training of a RTG where the given weighted lnCFTG is considered as a language model. An advantage is that any RTG can be trained in this way as long as its tree language intersected with the tree language induced by the lnCFTG is nonempty. Thus, the method applies to characterizations and approximations. Thus, the result extends the two previous methods to a weighted variant.

References

[1] 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.

[2] 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.

[3] N. Chomsky. On Certain Formal Properties of Grammars. *Information and Control* 2.2 (1959), 137–167.

[4] 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.

[5] W. Damm. The IO- and OI-Hierarchies. *Theoretical Computer Science* 20.2 (1982), 95–207. DOI: 10.1016/0304-3975(82)90009-3.

[6] J. Engelfriet. *Context-Free Grammars with Storage.* Tech. rep. 86-11. Republished 2014. University of Leiden, 1986. URL: http://arxiv.org/abs/1408.0683.

[7] M. Fischer. Grammars with Macro-Like Productions. PhD thesis. Harvard University, Massachusetts, 1968.

[8] 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.

[9] I. Guessarian. Pushdown Tree Automata. *Mathematical Systems Theory* 16.1 (1983), 237–263. DOI: 10.1007/BF01744582.

[10] 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.

[11] M.-J. Nederhof. A General Technique to Train Language Models on Language Models. *Computational Linguistics* 31.2 (2005), 173–186. DOI: 10.1162/0891201054223986.

[12] 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* (2016). Acceppted for publication.

[13] 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.

[14] 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.

[15] 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`.

[16] 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`.

[17] W. C. Rounds. Mappings and Grammars on Trees. *Mathematical Systems Theory* 4.3 (1970), 257–287. DOI: `10.1007/BF01695769`.

[18] 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`.