# How to Get Rid of Projection Rules in Context-free Tree Grammars

Dieter Hofbauer
Technische Universität Berlin
Franklinstraße 28/29, FR 6-2
D - 10587 Berlin, Germany
e-mail: dieter@cs.tu-berlin.de

Maria Huber
Gregory Kucherov
INRIA-Lorraine & CRIN
615, rue du Jardin Botanique
54602 Villers-lès-Nancy, France
e-mail: {huber,kucherov}@loria.fr

**Abstract**

In contrast to the case of words, in context-free tree grammars projection rules cannot always be eliminated completely. In this paper we partly overcome this deficiency by approximating projection-freeness up to a given degree. It is shown that, for any given finite set of positions, we can transform a context-free tree grammar to an equivalent grammar where derivations using projections at these positions are impossible. As an immediate consequence we get a new direct proof for the decidability of the membership problem for context-free tree languages.

## 1 Introduction

Numerous classes of tree languages have been studied extensively, the most prominent classes being the counterparts to classes in the Chomsky-hierarchy of word languages. The tree language classes, however, are often quite different in nature compared to the corresponding classes of word languages. Whereas most closure properties and most decidability results carry over from words to trees for the regular languages, it is not even clear what context-sensitive grammars should look like in the tree case. In this paper we study context-free tree languages that find their applications, for example, in linguistics in connection with tree adjoining grammars [9].

Context-free tree languages, also called algebraic, are those that are generated by a finite set of context-free rewrite rules, i.e., rules of the form $A(x_1, \ldots, x_n) \to t$, where $A$ is a non-terminal symbol, $x_1, \ldots, x_n$ are pairwise different variables, and $t$ is a term built from terminal symbols, non-terminal symbols and variables $x_1, \ldots, x_n$. This includes both non-linear rules (variables occur more than once in $t$) and projection rules ($t$ is a variable), also called collapsing rules. In general, projection rules are unavoidable. This means that there is not always an equivalent grammar without projections ([10], see also [4]). Therefore size-decreasing derivation steps are necessary, and there is nothing like a Greibach normal form for context-free tree grammars. As a consequence, deciding the membership problem for the generated language is not trivial.

These unwelcome phenomena occur neither in context-free word languages nor, more generally, in linear context-free tree languages, i.e., languages generated by a grammar without non-linear rules. In this case an equivalent projection-free grammar can be constructed by adding new rules that simply do not generate those symbols which would have been deleted by a projection later on in the original derivation. This idea of "anticipating" projections, however, cannot be applied in presence of non-linear rules. Here the construction can result in a grammar generating a proper subset of the language.

In this paper we show how to avoid projections at any given finite set of positions. More precisely, we prove that a context-free tree grammar can always be transformed to an equivalent grammar where derivations using projections at positions up to a given depth are unnecessary and moreover, impossible. The main idea is to introduce new arguments for non-terminals. This makes it possible to simulate a finite number of duplications without non-linear rules, and to treat these copies independently. Consequently, applying projections to each copy can be simulated at earlier stages of the derivation.

Another important idea is to pack projection-free contexts in single non-terminals. This gives a grammar where in each derivation non-projection rules are applied at the root position unless applying a

rule which yields a term with terminal symbols at all positions in the corresponding context. Alternating steps of multiplying arguments of non-terminals and of packing projection-free contexts in "macro non-terminals" eventually yields a grammar where projection rules are impossible at all positions up to a given depth.

The derivations of the resulting grammars have an interesting structure. They consist of two parts, depending on the depth of positions, $k$ say, where we want to avoid projections. As already mentioned above, in the first part rules are only applied at the root and no projection rule is used. In the very last derivation step at the root, terminal symbols are installed at every position up to depth $k$. Therefore, in the second part of the derivation, rewriting takes place at positions below level $k$ only (using rules of the original grammar, in fact). So each derivation can be seen as a top-context-free[1] derivation with the new grammar, followed by a context-free derivation below level $k$ with the original grammar.

As an application of our construction, we get a new decidability proof for the membership problem for context-free tree languages. It is based on the fact that the construction for level $k$ yields a grammar where all terms in the generated language are right hand sides of rules, provided that their depth does not exceed $k$. Thus the membership problem is reduced to a simple reachability problem. Compared to the other proofs we are aware of, our proof is direct. The traditional way is to show first the decidability of the emptiness problem (this follows from [1] and [7]) which in its turn is reduced to the emptiness of the corresponding (string) yield language. Together with closure under intersection with regular languages (see [12, 13]), this yields the decidability of the membership problem. In [3] and [6] proofs for more general classes of languages can be found. In [5] it is shown that the emptiness problem is complete for exponential time.

In section 2 we recall some definitions concerning tree languages. After formally defining what avoiding projections means in section 3, we introduce in section 4 an important notion of $k$-level terminal grammar and then explain in section 5 how, given a $k$-level terminal grammar, an equivalent $k$-level terminal grammar can be constructed which avoids projections at level $k+1$ too. In section 6 it is shown how whole contexts can be packed in single non-terminals. Finally, section 7 is devoted to the main theorem, stating that projections can always be avoided at all positions up to a given depth.

## 2 Preliminaries

A signature $\Gamma$ is a finite set of function symbols of fixed arity; for $n \geq 0$, $\Gamma_n$ denotes the set of symbols in $\Gamma$ of arity $n$. $\mathcal{T}_\Gamma(X)$ is the set of (finite) terms over $\Gamma$ and a set of variables $X$. The set of ground terms, i.e., terms without variables, over $\Gamma$ is denoted by $\mathcal{T}_\Gamma$. For $t \in \mathcal{T}_\Gamma(X)$, the set of positions in $t$ is defined in the usual way as sequences of natural numbers. $|\pi|$ denotes the length of the position $\pi$. When speaking about a position $\pi$ in a term $t$, we also call $|\pi|$ the depth of $\pi$ in $t$. If $t_1, \ldots, t_n$ are incomparable subterms of $t$, then $t$ can be written as $c[t_1, \ldots, t_n]$ where $c(\Box, \ldots, \Box)$ is a context. We call the symbol $\Box$ a hole, since it marks a leave where some other term is to be rooted. We treat contexts similar to terms. A context is called $\Gamma$-context in case it contains only symbols from $\Gamma$.

If unambiguous, we sometimes use vector notation instead of "three dots notation". For example, $A(\vec{t})$ abbreviates $A(t_1, \ldots, t_n)$ ($n$ and $t_i$ will be clear from the context), and $A(\vec{X})$ stands for $A(x_1, \ldots, x_n)$. For $A \in \Gamma_n$ and $L_1, \ldots, L_n \subseteq \mathcal{T}_\Gamma(X)$, define $A(L_1, \ldots, L_n) = \{A(t_1, \ldots, t_n) \mid t_1 \in L_1, \ldots, t_n \in L_n\}$. The latter notation is naturally extended to the "vector case" where $L_1, \ldots, L_n$ are sets of tuples. We also allow to combine individual terms (tuples) and sets of terms (tuples) in the arguments. Finally, we use the notation $\Gamma(L)$ for the set $\{A(t_1, \ldots, t_n) \mid A \in \Gamma, t_1, \ldots, t_n \in L\}$.

A *context-free tree grammar* $G = (N, \Sigma, P, S)$ consists of disjoint signatures $N$ *(nonterminals)* and $\Sigma$ *(terminals)*, a finite rewrite system $P$ over $N \cup \Sigma$, and a distinct constant symbol $S \in N_0$ *(initial symbol)*; all rules in $P$ are of the form

$$A(x_1, \ldots, x_n) \to t$$

where $A \in N_n$, $n \geq 0$, $x_1, \ldots, x_n$ are pairwise different variables, and $t \in \mathcal{T}_{N \cup \Sigma}\{x_1, \ldots, x_n\}$.

[1] Top-context-free grammars are a special kind of context-free ones, where in each right hand side of a rule non-terminal symbols occur, if at all, only at the root position, see [2, 8].

The language generated by a grammar $G = (N, \Sigma, P, S)$ is

$$\mathcal{L}(G) = \{t \in \mathcal{T}_\Sigma \mid S \stackrel{*}{\to}_P t\},$$

where $\stackrel{*}{\to}_P$ denotes the reflexive-transitive closure of the rewrite relation $\to_P$ generated by $P$. According to the usual formal language terminology, the rewriting process is called *derivation*. A language $L \subseteq \mathcal{T}_\Sigma$ is called context-free if there is a context-free grammar generating $L$.

Throughout the paper we will assume the rules of context-free grammars to have one of the following three forms:

(i) $A(x_1, \ldots, x_n) \to x_i$, (projection rules)

(ii) $A(x_1, \ldots, x_n) \to t[x_{i_1}, \ldots, x_{i_m}]$, (final rules)
where $t$ is a non-empty $\Sigma$-context,

(iii) $A(x_1, \ldots, x_n) \to B(s_1, \ldots, s_m)$,
where each $s_i$ is either some variable $x_j$, or a term $C_i(x_1, \ldots, x_{k_i})$ for some $k_i \leq n$.

Every context-free grammar can be effectively reduced to one with only rules (i)-(iii), since this form of rules subsumes the (Chomsky) normal form of Maibaum [11] (see also [7]). Thus, this restriction on the form of rules is without loss of generality. Note also that Arnold and Dauchet [2] used a normal form which is weaker than that of Maibaum but stronger than the one defined by rules (i)-(iii).

When speaking of derivations, we will often need to specify which rules at which positions are applied. This will be done using lower and upper indices respectively. For example, $\to_{(2)}$ means a one-step derivation using a rule of type (2), $\to^{\varepsilon}_{(2)}$ specifies that this derivation has been done at the root position, $\stackrel{n}{\to}^{\varepsilon}_{(2)}$ denotes $n$ derivation steps $\to^{\varepsilon}_{(2)}$, and $\stackrel{*}{\to}^{\varepsilon}_{(2)}$ stands for the reflexive-transitive closure of $\to^{\varepsilon}_{(2)}$.

# 3 What "Avoiding Projections" Means

Without loss of generality we assume all derivations to be top-down.

**Definition 1** *A grammar $G$ is said to* avoid projections at position $\pi$ *if all terms of $\mathcal{L}(G)$ can be derived top-down without applying projection rules (i) at position $\pi$. $G$ avoids projections* up to level $k \in I\!N$ *if all terms of $\mathcal{L}(G)$ can be derived top-down without applying projection rules (i) at any position $\pi$ with $|\pi| \leq k$.*

In addition to assuming all derivations to be top-down, we assume that positions are treated consecutively. This means that a derivation at a given position consists in applying rules (i) and (iii) unless a rule of type (ii) is applied. The rule of type (ii) installs a terminal at this position which is then retained "forever". Avoiding projections at some position implies that only rules (iii) can be applied at this position before an application of a rule (ii).

In [2] Arnold and Dauchet proved an important lemma which shows that one can transform a context-free grammar into an equivalent one which avoids projections at the root position (level 0). The transformation consists in taking the closure of the set of rules with respect to the following operation: for every rule $A(x_1, \ldots, x_n) \to B(s_1, \ldots, s_i, \ldots, s_m)$ of type (iii), and every projection rule $B(x_1, \ldots, x_m) \to x_i$, add the rule $A(x_1, \ldots, x_n) \to s_i$. We say that the extended grammar is *closed under top projections*. The following lemma is a direct consequence of the construction.

**Lemma 2** *Let $G$ be a grammar closed under top projections. Then for $t_1, t_2, t_3 \in \mathcal{T}_{N \cup \Sigma}$, if $t_1 \to^{\varepsilon}_{(iii)} t_2 \to^{\varepsilon}_{(i)} t_3$, then $t_1 \to^{\varepsilon}_{(i)(iii)} t_3$.*

Thus, the extended grammar permits to replace an application of a projection rule preceded by an application of a rule (iii) by a single derivation step. By induction, the construction above allows us to apply projections only at the beginning of the derivation at a given position, that is before the first application of a non-projection rule. As a particular case, the construction allows us to avoid applying projections at the root position, since the first rule of this derivation applies to the initial symbol and cannot be a projection rule.

However, the construction above does not allow us to avoid projections completely at any inner position. Note that by the above remark, only applications of projection rules at the beginning of the derivation at each position are concerned. We cannot just apply projections "in advance" as in the Arnold and Dauchet construction, since terms may result from duplication. In this case, applying projections in advance would mean doing the same for another term, which can restrict the resulting language. In contrast to the root case, in the case of internal positions the set of non-terminals should be changed.

# 4 $k$-level Terminal Grammars

We introduce a special class of grammars, called *$k$-level terminal* grammars, which in a sense generalizes grammars avoiding projections at the root position. These grammars play a central role in our presentation.

**Definition 3** *For $k \in I\!N$, a grammar $G$ is said to be a* $k$-level terminal grammar *if $G$ satisfies the following conditions.*

1. *$G = (N \cup \bar{N}, \Sigma, P \cup \bar{P}, \bar{S})$, where $N \cap \bar{N} = \emptyset$, $\bar{S} \in \bar{N}$,*
2. *every rule of $P$ is of type (i), (ii), or (iii) and does not use non-terminals from $\bar{N}$,*
3. *every rule of $\bar{P}$ has one of the following two forms:*

   - $$\bar{A}(x_1, \ldots, x_n) \to \bar{B}(s_1, \ldots, s_m), \tag{1}$$

     *where $\bar{A}, \bar{B} \in \bar{N}$, and for every $i, 1 \leq i \leq m$, either $s_i = x_j$ for some $j, 1 \leq j \leq n$, or $s_i = C_i(x_1, \ldots, x_{k_i})$ for some $C_i \in N$, $k_i \leq n$,*
   - $$\bar{A}(x_1, \ldots, x_n) \to t[x_{i_1}, \ldots, x_{i_m}], \tag{2}$$

     *where $t$ is a $\Sigma$-context and all variable occurrences of $t$ have depth $k+1$.*

A $k$-level terminal grammar trivially avoids projections at the root position since there are no projection rules for non-terminals of $\bar{N}$, and only these non-terminals can occur at the root position. On the other hand, when leaving the root position in the derivation, the grammar generates at once a term containing terminals up to level $k$. Thus, no application of projection rules is possible up to level $k$. To put it more formally, all $G$-derivations of a term $t \in \mathcal{T}_{N \cup \Sigma}$ have the form

$$\bar{S} \overset{*}{\to}^{\varepsilon}_{(1)} \circ \to^{\varepsilon}_{(2)} s \overset{*}{\to}^{>k}_{P} t,$$

where $\to^{>k}$ means that the derivation step is applied at a position deeper than $k$. Since $s$ contains only terminals up to level $k$, non-terminals in $s$ occur only at depth greater than $k$. Therefore, the last derivation step at the root position is followed by a derivation step at a position deeper than $k$. This immediately implies that $k$-level terminal grammars allow no projections up to level $k$. Note also that all non-terminals of $s$ belong to $N$.

The particular structure of the derivations suggests another interesting observation. If we treat symbols of $N$ as terminals, the grammar $(\bar{N}, \Sigma \cup N, \bar{P}, \bar{S})$ becomes a top-context-free grammar [2, 8]. Thus, any $G$-derivation can be regarded as a top-context-free $\bar{P}$-derivation generating all terminal symbols up to level $k$ followed by a context-free $P$-derivation.

Let us first turn to 0-level terminal grammars. A grammar closed under top projections avoids projections at the root but is not a 0-level terminal grammar since generally we cannot separate non-terminals which appear only at the root ($\bar{N}$) and those which appear only at inner positions ($N$). However, a grammar $G = (N, \Sigma, P, S)$ closed under top projections can be easily extended to a 0-level terminal grammar. This is achieved by defining $\bar{N} = \{\bar{A} \mid A \in N\}$, and defining $\bar{P}$ as follows. For each rule $A(x_1, \ldots, x_n) \to t[x_{i_1}, \ldots, x_{i_m}]$ of $P$, $\bar{P}$ contains the rule $\bar{A}(x_1, \ldots, x_n) \to t[x_{i_1}, \ldots, x_{i_m}]$, and for each rule $A(x_1, \ldots, x_n) \to B(s_1, \ldots, s_m)$ of $P$, $\bar{P}$ contains the rule $\bar{A}(x_1, \ldots, x_n) \to \bar{B}(s_1, \ldots, s_m)$.

# 5 Avoiding Projections at the Next Level

In this section we show how to avoid projections at level $k+1$ for a given $k$-level terminal grammar. For the sake of clarity, we proceed in two steps: first we show how to avoid projections at *one* arbitrary position at level $k+1$ and then generalize the construction to *all* positions of level $k+1$.

## 5.1 Avoiding Projections at One Position of Level $k+1$

Consider a $k$-level terminal grammar $G = (N \cup \bar{N}, \Sigma, P \cup \bar{P}, \bar{S})$. We define another $k$-level terminal grammar $G' = (N \cup N', \Sigma, P \cup P', S')$ by transforming grammar $G$ as follows. For every non-terminal $\bar{A} \in \bar{N}$, introduce a non-terminal $A' \in N'$ which has twice the arity of $\bar{A}$. The initial symbol $S' \in G'$ correponds to the initial symbol $\bar{S} \in \bar{N}$. The set of production rules $P'$ is constructed by transforming the rules of type (1) and (2) of $\bar{P}$. We first show how the rules of type (1) are transformed.

*Rules of type (1).* For every rule of type (1)

$$\bar{A}(x_1, \ldots, x_n) \to \bar{B}(s_1, \ldots, s_m),$$

$P'$ contains the following rules.

1. $P'$ contains the rule

$$A'(x_1, \ldots, x_n, y_1, \ldots, y_n) \to B'(s_1, \ldots, s_m, s_1, \ldots, s_m) \tag{3}$$

   Note that $y_i$'s don't occur in the right-hand side of (3).

2. for every $i, 1 \leq i \leq m$,

   - if $s_i = x_j$, then $P'$ contains the rule

$$A'(x_1, \ldots, x_n, y_1, \ldots, y_n) \to B'(s_1, \ldots, s_i, \ldots, s_m, s_1, \ldots, y_j, \ldots, s_m) \tag{4}$$

   - if $s_i = C_i(x_1, \ldots, x_{k_i})$, then for every projection rule $C_i(x_1, \ldots, x_{k_i}) \to x_j$ in $P$, $P'$ contains the rule

$$A'(x_1, \ldots, x_n, y_1, \ldots, y_n) \to B'(s_1, \ldots, s_i, \ldots, s_m, s_1, \ldots, y_j, \ldots, s_m) \tag{5}$$

Note the following properties of rules (3), (4), (5):

- rules (3), (4), (5) have the form (iii),
- variables $y_1, \ldots, y_n$ occur only at level 1,
- the tuple of first $n$ arguments of the right-hand side is the same for all rules (3), (4), (5) and does not contain $y_j$'s.

Before showing how to transform rules of type (2) we state the following key lemma which illustrates the intended meaning of rules (3), (4), (5).

**Lemma 4** *Assume that for some $l \in \mathbb{N}$, $\bar{S} \overset{l}{\to}{}^{\varepsilon}_{(1)} \bar{A}(t_1, \ldots, t_n)$ and for some $j, 1 \leq j \leq n$, $t_j \overset{*}{\to}{}^{\varepsilon}_{(i)} \widehat{t_j}$. Then*

$$S' \overset{l}{\to}{}^{\varepsilon}_{(3)(4)(5)} A'(t_1, \ldots, t_j, \ldots, t_n, t_1, \ldots, \widehat{t_j}, \ldots, t_n) \tag{6}$$

Proof. The proof goes by induction on $l$. Assuming that the lemma holds for some $l$, we prove that for a derivation of length $(l+1)$

$$\bar{S} \overset{l}{\to}{}^{\varepsilon}_{(1)} \bar{A}(t_1, \ldots, t_n) \to^{\varepsilon}_{(1)} \bar{B}(p_1, \ldots, p_i, \ldots, p_m), \tag{7}$$

if

$$p_i \overset{*}{\to}{}^{\varepsilon}_{(i)} \widehat{p_i}, \tag{8}$$

then there exists a $P'$-derivation

$$S' \overset{1+1}{\Rightarrow}{}^{\varepsilon}_{(3)(4)(5)} B'(p_1,\ldots,p_i,\ldots,p_m,p_1,\ldots,\widehat{p_i},\ldots,p_m). \tag{9}$$

Consider the rule

$$\bar{A}(x_1,\ldots,x_n) \to \bar{B}(s_1,\ldots,s_i,\ldots,s_m) \tag{10}$$

applied at the last step of (7). We proceed by case analysis of $s_i$.

*Case 1.* $s_i = x_j$ for some $j, 1 \le j \le n$ and hence $p_i = t_j$. By induction hypothesis, there exists a derivation

$$S' \overset{1}{\to}{}^{\varepsilon}_{(3)(4)(5)} A'(t_1,\ldots,t_j,\ldots,t_n,t_1,\ldots,\widehat{p_i},\ldots,t_n). \tag{11}$$

Derivation (9) is then obtained by applying the corresponding rule (4) of $P'$.

*Case 2.* $s_i = C_i(x_1,\ldots,x_{k_i})$ and hence $p_i = C_i(t_1,\ldots,t_{k_i})$. First observe that the case $p_i = \widehat{p_i}$ (that is, zero projections is applied in (8)) does not represent a difficulty, since derivation (9) is constructed easily by using only rules (3). Rewrite (8) as

$$p_i = C_i(t_1,\ldots,t_{k_i}) \to^{\varepsilon}_{(i)} t_j \overset{*}{\to}{}^{\varepsilon}_{(i)} \widehat{p_i}, \tag{12}$$

where the first step is done by the projection rule $C_i(x_1,\ldots,x_{k_i}) \to x_j$ of $P$. By induction hypothesis, there exists a derivation

$$S' \overset{1}{\to}{}^{\varepsilon}_{(3)(4)(5)} A'(t_1,\ldots,t_j,\ldots,t_n,t_1,\ldots,\widehat{p_i},\ldots,t_n), \tag{13}$$

and derivation (9) is obtained by applying the corresponding rule (5) of $P'$. □

To complete the construction of $P'$, we show now how rules of type (2) are transformed.

*Final rules.* For every rule of type (2)

$$\bar{A}(x_1,\ldots,x_n) \to t[x_{i_1},\ldots,x_{i_m}],$$

and for every $j, 1 \le j \le m$, $P'$ contains the rule

$$A'(x_1,\ldots,x_n,y_1,\ldots,y_n) \to t[x_{i_1},\ldots,y_{i_j},\ldots,x_{i_m}] \tag{14}$$

Note that variable $x_{i_j}$ may occur several times in the right-hand side of (2). However, only one occurrence of it is replaced by $y_{i_j}$ in (14).

Trivially, $G'$ is a $k$-level terminal grammar. It is easy to see that any $\bar{P}$-derivation at the root

$$\bar{S} \overset{1}{\to}{}^{\varepsilon}_{(1)} \bar{A}(s_1,\ldots,s_n) \to^{\varepsilon}_{(2)} t[s_{i_1},\ldots,s_{i_m}],$$

for $s_{i_1},\ldots,s_{i_m} \in \mathcal{T}_{N\cup\Sigma}$, can be one-to-one simulated by the following $P'$-derivation using corresponding rules (3)

$$S' \overset{1}{\to}{}^{\varepsilon}_{(3)} A'(s_1,\ldots,s_n,s_1,\ldots,s_n) \to^{\varepsilon}_{(14)} t[s_{i_1},\ldots,s_{i_m}].$$

On the other hand, from the construction of $P'$ it follows that a $P'$-derivation can be simulated by a (possibly not top-down) $(\bar{P}\cup P)$-derivation. We then conclude that $\mathcal{L}(G') = \mathcal{L}(G)$.

Finally, lemma 4 together with rules (14) implies that $G'$ avoids projections at any given position of level $k+1$. Clearly, $G'$ avoids projection up to level $k$ by definition of a $k$-level terminal grammar. We summarize the construction as follows.

**Theorem 5** *Given a $k$-level terminal grammar $G$, there effectively exists an equivalent $k$-level terminal grammar $G'$ such that $G'$ avoids projections at any given position up to level $k+1$.*

Note that theorem 5 does not imply that $G'$ avoids projections at *all* positions of level $k+1$. Constructing such a grammar is the next step of our development.

## 5.2 Avoiding Projections at All Positions of Level $k+1$

The guiding observation here is that to avoid projections at one position, one additional block of arguments is introduced for each new non-terminal. This block is used to prepare a copy of the arguments except that some number of projections have been applied to one of them (see lemma 4). To avoid projection at the whole level $k+1$, we need as many additional blocks as copies that we must have.

Let $G = (N \cup \bar{N}, \Sigma, P \cup \bar{P}, S)$ be a $k$-level terminal grammar. Assume that $M$ is the maximal number of variable occurrences in the right-hand sides of rules (2) of $\bar{P}$. We now define a $k$-level terminal grammar $G'' = (N \cup N'', \Sigma, P \cup P'', S'')$ as the following generalization of $G'$. Non-terminals of $N''$ are in one-to-one correspondence with the non-terminals of $\bar{N}$ but have $M$ additional blocks of arguments with respect to their counterparts. Here, the non-terminal of $N''$ corresponding to $\bar{A} \in \bar{N}$ will be denoted by $A''$.

Similar to $G'$, rules of $P''$ are constructed by transforming the rules of type (1) and (2) of $\bar{P}$.

*Rules of type (1).* To define the rules of $P''$ corresponding to rules (1), let us group together all the rules (3), (4), (5) corresponding to a rule (1) and write them in the following vector notation:

$$A'(\vec{X}, \vec{Y}) \to B'(\vec{s}, \vec{S}), \tag{15}$$

where $\vec{X} =< x_1, \ldots, x_n >$, $\vec{Y} =< y_1, \ldots, y_n >$, $\vec{s} =< s_1, \ldots, s_m >$, and $\vec{S}$ is the set of tuples corresponding to the second blocks of arguments in right-hand sides of the rules (3), (4), (5). Note that $\vec{s}$ contains only variables of $\vec{X}$ while $\vec{S}$ contains variables from both $\vec{X}$ and $\vec{Y}$.

Now introduce $M$ tuples of distinct variables $\vec{Y}_1, \ldots, \vec{Y}_M$, where $\vec{Y}_i =< y_i^1, \ldots, y_i^n >$, $1 \le i \le M$. Using $\vec{Y}_i$'s, the rules of $P''$ corresponding to the rules (1) of $\bar{P}$ are defined as the following set of rules.

$$A''(\vec{X}, \vec{Y}_1, \ldots, \vec{Y}_M) \to B''(\vec{s}, \vec{S}[\vec{Y} \mapsto \vec{Y}_1], \ldots, \vec{S}[\vec{Y} \mapsto \vec{Y}_M]) \tag{16}$$

where $\vec{S}[\vec{Y} \mapsto \vec{Y}_i]$ means renaming variables $Y =< y_1, \ldots, y_n >$ by variables $Y_i =< y_i^1, \ldots, y_i^n >$ respectively in every tuple of $\vec{S}$.

Lemma 4 is now generalized as follows.

**Lemma 6** *Assume that for some $l \in I\!N$, $S \overset{l}{\to}{}^{\varepsilon}_{(1)} A(t_1, \ldots, t_n)$. Take some (not necessarily distinct) indices $i_1, \ldots, i_M \in [1:n]$, and assume that for every $j$, $1 \le j \le M$, $t_{i_j} \overset{*}{\to}{}^{\varepsilon}_{(i)} \widehat{t_{i_j}}$. Then*

$$\bar{S} \overset{l}{\to}{}^{\varepsilon}_{(16)} \bar{A}(t_1, \ldots, t_n, t_1, \ldots, \widehat{t_{i_1}}, \ldots, t_n, \ldots, t_1, \ldots, \widehat{t_{i_M}}, \ldots, t_n) \tag{17}$$

PROOF. Apply lemma 4 to each of the indices $i_1, \ldots, i_M$ and construct $M$ corresponding derivations (6). Note that all of these derivations have the same length. Clearly, rules (16) allow us to simulate $M$ independent derivations (6) by a single derivation (17). □

*Final rules.* To complete the construction of $G''$, we introduce the rules of $P''$ corresponding to the rules (2) of $\bar{P}$. For every rule (2)

$$\bar{A}(x_1, \ldots, x_n) \to t[x_{i_1}, \ldots, x_{i_m}], \tag{18}$$

$P''$ contains the rule

$$A''(x_1, \ldots, x_n, y_1^1, \ldots, y_1^n, \ldots, y_M^1, \ldots, y_M^n) \to t[y_1^{i_1}, \ldots, y_M^{i_m}] \tag{19}$$

Again, by combining lemma 6 with the definition of rules (19) we get the desired generalization of theorem 5.

**Theorem 7** *Given a $k$-level terminal grammar $G$, there effectively exists an equivalent $k$-level terminal grammar $G''$ such that $G''$ avoids projections up to level $k+1$.*

# 6 Encoding Contexts as Non-terminals

As a next step, we will show how a $(k+1)$-level terminal grammar is constructed from a given $k$-level terminal grammar that avoids projections at one additional level, namely at level $k+1$.

**Theorem 8** *Given a $k$-level terminal grammar $G_k$ that avoids projections up to level $k+1$, there effectively exists an equivalent $(k+1)$-level terminal grammar $G_{k+1}$.*

Hence, $G_{k+1}$ avoids projections up to level $k+1$.

In order to construct this $(k+1)$-level terminal grammar we treat whole contexts as single non-terminals. To represent a term $t$ as a composition of such a non-terminal corresponding to a context of depth $k$, and a number of arguments, we introduce the notation $[\![t]\!]_k$. For defining $[\![t]\!]_k$ and the inverse "unpack" operation we introduce some notation.

For a signature $\Gamma$ and a set of terms or a set of contexts $T$, the set $\Gamma^k(T)$ is defined by $\Gamma^0(T) = T$ and $\Gamma^{k+1}(T) = \Gamma(\Gamma^k(T))$. For example, $\Sigma^k(\{\Box\})$ denotes the set of $\Sigma$-contexts of depth $k$ where every hole $\Box$ occurs at depth $k$.

Consider the finite set of contexts $\Gamma^k(\{\Box\})$. The idea is to treat every context $c \in \Gamma^k(\{\Box\})$ as a new symbol of the arity equal to the number of hole occurrences in $c$. Formally, for a term $t \in \mathcal{T}_\Gamma(X)$ without variables at positions of length smaller or equal to $k$, the term $[\![t]\!]_k \in \Gamma^k(\{\Box\})(\mathcal{T}_\Gamma(X))$ is defined as

$$[\![t]\!]_k = c(t_1, \ldots, t_n) \text{ iff } t = c[t_1, \ldots, t_n]$$

for $c \in \Gamma^k(\{\Box\})$ and $t_1, \ldots, t_n \in \mathcal{T}_\Gamma(X)$. Note that for given $t$ and $k$, both $c$ and $t_1, \ldots, t_n$ (and thus $[\![t]\!]_k$) are uniquely determined. For example, $[\![f(a, g(b))]\!]_1 = [\![f(\Box, \Box)[a, g(b)]]\!]_1 = f(\Box, \Box)(a, g(b))$.

The inverse operation is defined for any $c \in \Gamma^k(\{\Box\})$ and $t_1, \ldots, t_n \in \mathcal{T}_\Gamma(X)$ by

$$[\![c(t_1, \ldots, t_n)]\!]^{-1} = c[t_1, \ldots, t_n].$$

Note that $[\![[\![t]\!]_k]\!]^{-1} = t$ holds. E.g., $[\![[\![[\![f(a, g(b))]\!]_1]\!]^{-1} = [\![f(\Box, \Box)(a, g(b))]\!]^{-1} = f(\Box, \Box)[a, g(b)] = f(a, g(b)) = f(a, g(\Box))[b] = [\![f(a, g(\Box))(b)]\!]^{-1} = [\![[\![f(a, g(b))]\!]_2]\!]^{-1}$.

Using this notation, we now explain how the proof of theorem 8 works. Consider a $k$-level terminal grammar $G_k = (N \cup \bar{N}, \Sigma, P \cup \bar{P}, \bar{S})$ which avoids projections up to level $k+1$. We introduce non-terminals $N_1 = \bar{N}(N(\{\Box\}))$ and $N_2 = \Sigma^k((N \cup \Sigma)(\{\Box\}))$; note that $\Sigma^{k+1}(\{\Box\}) \subseteq N_2$. Replacing $\bar{N}$ by the new non-terminals $\tilde{N} = N_1 \cup N_2$ and $\bar{P}$ by new rules $\tilde{P}$, we obtain the grammar $G_{k+1} = (N \cup \tilde{N}, \Sigma, P \cup \tilde{P}, S)$. The set $\tilde{P}$ contains all rules according to the following schemes, where $c_1, c_1' \in N_1$ and $c_2, c_2' \in N_2$:

- $$c_1(x_1, \ldots, x_n) \to c_1'(t_1, \ldots, t_m) \tag{20}$$
  if $[\![c_1(x_1, \ldots, x_n)]\!]^{-1} \to_{\bar{P}}^{\varepsilon} [\![c_1'(t_1, \ldots, t_m)]\!]^{-1}$ by a rule of type (1).

- $$c_1(x_1, \ldots, x_n) \to c_2'(t_1, \ldots, t_m) \tag{21}$$
  if $[\![c_1(x_1, \ldots, x_n)]\!]^{-1} \to_{\bar{P}}^{\varepsilon} [\![c_2'(t_1, \ldots, t_m)]\!]^{-1}$ by a rule of type (2).

- $$c_2(x_1, \ldots, x_n) \to c_2'(t_1, \ldots, t_m) \tag{22}$$
  if $[\![c_2(x_1, \ldots, x_n)]\!]^{-1} \to_P [\![c_2'(t_1, \ldots, t_m)]\!]^{-1}$ by a non-projection step (rule of type (ii) or (iii)) at a position of depth $k+1$.

- $$c_2(x_1, \ldots, x_n) \to [\![c_2(x_1, \ldots, x_n)]\!]^{-1} \tag{23}$$
  if $c_2 \in \Sigma^{k+1}(\{\Box\})$.

It can easily be seen that the new rules of type (20), (21), and (22) are of the form (1), and that the new rules (23) are of the form (2) with all variable occurrences at level $k+1$. Therefore, $G_{k+1}$ is a $(k+1)$-level terminal grammar. The derivations in $G_k$ and $G_{k+1}$ correspond to each other in the following way:

$$\begin{array}{cccccccccc} \bar{S} & \overset{*}{\to}{}^{\varepsilon}_{(1)} & t_1 & \to^{\varepsilon}_{(2)} & t_2 & \overset{*}{\to}{}^{=k+1}_{P} & t_3 & \overset{*}{\to}{}^{>k+1}_{P} & t_4 \\ [\![\bar{S}]\!]_2 & \overset{*}{\to}{}^{\varepsilon}_{(20)} & [\![t_1]\!]_2 & \to^{\varepsilon}_{(21)} & [\![t_2]\!]_{k+1} & \overset{*}{\to}{}^{\varepsilon}_{(22)} [\![t_3]\!]_{k+1} \to^{\varepsilon}_{(23)} & t_3 & \overset{*}{\to}{}^{>k+1}_{P} & t_4 \end{array}$$

This correspondence especially implies that $\mathcal{L}(G_k) = \mathcal{L}(G_{k+1})$ and thus proofs theorem 8. For showing that derivations in $G_k$ and in $G_{k+1}$ can be mutually simulated in this way, it is important that there are no projections at level $k+1$ in $G_k$.

# 7 Main Result

Putting together the results of previous sections, we obtain the following main result.

**Theorem 9** *For every context-free grammar $G$ and every $k \in I\!N$, there effectively exists an equivalent $k$-level terminal grammar $G_k$.*

PROOF. By induction on $k$. $G_0$ is obtained as described in section 4. Assume that $G_k$ is a $k$-level terminal grammar equivalent to $G$. By theorem 7, an equivalent $k$-level terminal grammar $G'_k$ can be constructed which avoids projections up to level $k+1$. Then by theorem 8, an equivalent $(k+1)$-level terminal grammar $G_{k+1}$ can be effectively constructed. □

As a corollary of theorem 9, we obtain a new direct proof of the decidability of membership problem for context-free tree languages.

**Corollary 10** *It is decidable if for a context-free tree grammar $G = (N, \Sigma, P, S)$ and a term $t \in \mathcal{T}_\Sigma$, $t \in \mathcal{L}(G)$.*

# References


[1] A. V. Aho. Indexed Grammars–An Extension of Context-Free-Grammars. *Journal of the ACM*, 15(4):641–671, 1968.

[2] A. Arnold and M. Dauchet. Un Théorème de Duplication pour les Forêts Algébriques. *Journal of Computer and System Sciences*, 13:223–244, 1976.

[3] W. Damm. The IO- and OI-Hierarchies. *Theoretical Computer Science*, 20:95–207, 1982.

[4] M. Dauchet and S. Tison. Structural Complexity of Classes of Tree Languages. In *Tree Automata and Languages*, pages 327–353. Elsevier Science Publishers B. V. (North-Holland), 1992.

[5] J. Engelfriet. Iterated Stack Automata and Complexity Classes. *Information and Computation*, 95:21–75, 1991.

[6] J. Engelfriet and H. Vogler. Macro Tree Transducers. *Journal of Computer and System Sciences*, 31:71–146, 1985.

[7] M. Fischer. Grammars with Macro-Like Productions. In *9th Symposium on Switching and Automata Theory*, pages 131–142, 1968. Long version in the PhD–theses, Harvard University, 1968.

[8] D. Hofbauer, M. Huber, and G. Kucherov. Some Results on Top-context-free Tree Languages. In S. Tison, editor, *Proceedings 19th International Colloquium on Trees in Algebra and Programming, Edinburgh (U.K.)*, volume 787 of *Lecture Notes in Computer Science*, pages 157–171. Springer-Verlag, Apr. 1994.

[9] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:136–163, 1975.

[10] B. Leguy. *Réductions, Transformations et Classification des Grammaires Algébriques d'Arbres.* Thèse de Doctorat de Troisième Cycle, Université des Sciences et Techniques de Lille (France), 1980.

[11] T. S. E. Maibaum. A Generalized Approach to Formal Languages. *Journal of Computer and System Sciences*, 8:409–439, 1974.

[12] W. C. Rounds. Mappings and Grammars on Trees. *Mathematical Systems Theory*, 4(3):257–287, 1970.

[13] W. C. Rounds. Tree-Oriented Proofs of some Theorems on Context-Free and Indexed Languages. In *Proceedings of 2nd Annual Symposium on Theory of Computing*, pages 109–116, 1970.