Lexicalization of Probabilistic Linear Context-free Rewriting Systems

Richard Mörbitz and **Thomas Ruprecht**
Faculty of Computer Science
Technische Universität Dresden
01062 Dresden, Germany
{richard.moerbitz,thomas.ruprecht}@tu-dresden.de

Abstract

In the field of constituent parsing, probabilistic grammar formalisms have been studied to model the syntactic structure of natural language. More recently, approaches utilizing neural models gained lots of traction in this field, as they achieved accurate results at high speed. We aim for a symbiosis between probabilistic linear context-free rewriting systems (PLCFRS) as a probabilistic grammar formalism and neural models to get the best of both worlds: the interpretability of grammars, and the speed and accuracy of neural models. A combination of these two could be achieved by applying supertagging to PLCFRS. This approach requires lexical grammar formalisms. Here, we present a procedure which turns any PLCFRS G into an equivalent lexical PLCFRS G'. Moreover, we show how the derivations in G' can be transformed to obtain their corresponding original derivations in G. Our construction for G' preserves the probability assignment and does not increase parsing complexity compared to G.

1 Introduction

Constituency parsing is a syntactical analysis in NLP that aims to enhance sentences with, usually tree-shaped, phrase structures (for an example cf. the left of Fig. 1). Formalisms such as context-free grammars (CFG) are used in this setting because they are conceptually simple, interpretable, and parsing is tractable (cubic in sentence length).

Discontinuous constituents span non-contiguous sets of positions in a sentence. The resulting phrase structures do not take the shape of a tree anymore, as they contain crossing branches (cf. the left of Fig. 1), and cannot be modeled by CFG. As a countermeasure, many corpora (e.g., the Penn Treebank (PTB)) denote these phrase structures as trees nevertheless and introduce designated notations for discontinuity, which is then often ignored in parsing. However, discontinuity occurs in about 20 % of the sentences in the PTB, and parsing discontinuous constituents can improve accuracy (Evang and Kallmeyer, 2011). For this, so-called "mildly context-sensitive" grammar formalisms have been investigated, e.g., tree-adjoining grammars (TAG; Joshi et al., 1975) and linear context-free rewriting systems (LCFRS; Vijay-Shanker et al., 1987). Their increased expressiveness comes at the cost of a higher parsing complexity: given a sentence of length n, parsing is in $O(n^6)$ for TAG and $O(n^{3 \cdot \mathrm{fanout}(G)})$ for an LCFRS G. The fanout is grammar-specific and reflects the degree of discontinuity in the rules of G. The expressiveness of TAG equals that of LCFRS with fanout 2. An LCFRS derivation of a discontinuous phrase is shown in the right of Fig. 1.

Supertagging has been used for more efficient parsing with lexical TAG (Bangalore and Joshi, 1999). A TAG is lexical if each rule contains one word. A supertagger selects for each position of the input sentence a subset of the rules of the TAG; these are the so-called supertags. Parsing is then performed with the much smaller grammar of supertags. Recently, the performance of supertagging has been improved by using neural classifiers for the selection of supertags (Vaswani et al., 2016). The goal of our research is to use supertagging for parsing with LCFRS, because they are more expressive than TAG.

In this paper, we lay the theoretical foundations for a supertagging-based LCFRS parser. As LCFRS obtained from corpora such as the PTB are usually not lexical, we employ a lexicalization procedure. It can be seen as an instance of the technique for lexicalization of multiple context-free tree grammars (Engelfriet et al., 2018). However, our approach is more concise and does not increase the fanout of the grammar (thus preserving parsing

Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task, pages 98–104
Virtual Meeting, July 9, 2020. ©2020 Association for Computational Linguistics

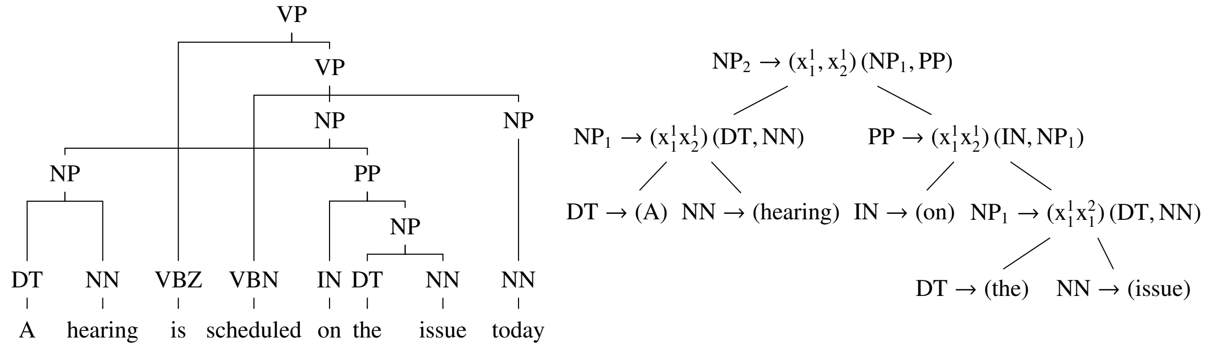
VP
VP
NP
NP
NP
PP
NP
DT NN VBZ VBN IN DT NN NN
A hearing is scheduled on the issue today
$\mathrm{NP}_2 \to (\mathrm{x}_1^1, \mathrm{x}_2^1)\,(\mathrm{NP}_1, \mathrm{PP})$
$\mathrm{NP}_1 \to (\mathrm{x}_1^1\mathrm{x}_2^1)\,(\mathrm{DT}, \mathrm{NN})$
$\mathrm{PP} \to (\mathrm{x}_1^1\mathrm{x}_2^1)\,(\mathrm{IN}, \mathrm{NP}_1)$
$\mathrm{DT} \to (\mathrm{A})$
$\mathrm{NN} \to (\mathrm{hearing})$
$\mathrm{IN} \to (\mathrm{on})$
$\mathrm{NP}_1 \to (\mathrm{x}_1^1\mathrm{x}_1^2)\,(\mathrm{DT}, \mathrm{NN})$
$\mathrm{DT} \to (\mathrm{the})$
$\mathrm{NN} \to (\mathrm{issue})$

Figure 1: Left: a discontinuous phrase structure tree of the sentence *A hearing is scheduled on the issue today*. Right: an LCFRS derivation of the discontinuous noun phrase *A hearing on the issue*.

complexity). Moreover, the approach is extended to account for probabilistic parsing. Furthermore, we introduce a procedure which recovers from each derivation of a lexicalized LCFRS all corresponding derivations of the original grammar.

This short paper is to be seen as a report on our approaches to lexicalization and recovery of derivations. An implementation of the supertagger and experimental evaluation are currently work in progress.

2 Preliminaries

The set of *non-negative* (resp. *positive*) *integers* is denoted by $\mathbb{N}$ (resp. $\mathbb{N}_+$). We abbreviate $\{1, ..., n\}$ by $[n]$ for each $n \in \mathbb{N}$. Let A be a set; the *set of (finite) strings over A* is denoted by A^*. An *alphabet* is a finite and non-empty set.

Let S be some set whose elements we call *sorts*. An *S-sorted set* is a tuple $(A, sort)$ where A is a set and $sort\colon A \to S$. Usually, we identify $(A, sort)$ with A, and denote *sort* by sort_A and $sort^{-1}(s)$ by A_s for each $s \in S$. The usual notation for sets $(\in, \subseteq, \cup, \ldots)$ is used with sorted sets in the intuitive manner. Now let A be an $(S^* \times S)$-sorted set. The *set of trees over A* is the S-sorted set T^A where $\mathrm{T}^A_s = \{a(t_1, \ldots, t_k) \mid k \in \mathbb{N}, s_1, \ldots, s_k \in S, a \in A_{(s_1 \cdots s_k, s)}, t_1 \in \mathrm{T}^A_{s_1}, \ldots, t_k \in \mathrm{T}^A_{s_k}\}$ for each $s \in S$. A *ranked set A* is an $(S^* \times S)$-sorted set where $S = \{s\}$; the notation $\mathrm{rk}_A(a) = k$ abbreviates $\mathrm{sort}_A(a) = (s^k, s)$, and A_k abbreviates $A_{(s^k, s)}$. If we use a usual set B in place of a ranked set, we will silently assume $\mathrm{rk}_B(b) = 0$ for each $b \in B$. Let X be a set. We let $A(X) = \{a(x_1, \ldots, x_k) \mid k \in \mathbb{N}, a \in A_k, x_1, \ldots, x_k \in X\}$.

LCFRS. *Linear context-free rewriting systems* extend the rule-based string rewriting mechanism of CFG to string tuples; we describe the generation process by *compositions*. Let $k \in \mathbb{N}$ and $s_1, \ldots, s_k, s \in \mathbb{N}_+$; a *Σ-composition* is a tuple $(u_1, \ldots, u_s)$ where each $u_1, \ldots, u_s$ is a non-empty string over Σ and variables of the form x_i^j where $i \in [k]$ and $j \in [s_i]$. Each of these variables must occur exactly once in $u_1 \cdots u_s$ and they are ordered such that x_i^1 occurs before x_{i+1}^1 and x_i^j occurs before x_i^{j+1} for each $i \in [k-1]$ and $j \in [s_i - 1]$. We denote the set of Σ-compositions by $\mathrm{C}^\Sigma_{(s_1 \cdots s_k, s)}$; we drop the superscript in the case $\Sigma = \emptyset$ (then $C_{(s_1 \cdots s_k, s)}$ is finite); we drop the subscript if we admit any configuration of k, $s_1, \ldots, s_k$ and s. We associate with each composition $(u_1, \ldots, u_s) \in \mathrm{C}^\Sigma_{(s_1 \cdots s_k, s)}$ a function from k string tuples, where the i-th tuple is of length s_i, to a string tuple of length s. This function is denoted by $[\![(u_1, \ldots, u_s)]\!]$. Intuitively, it replaces each variable of the form x_i^j in $u_1, \ldots, u_s$ by the j-th component of the i-th argument.

An *LCFRS* is a tuple $G = (N, \Sigma, S, R)$ where • N is a finite $\mathbb{N}_+$-sorted set (*non-terminals*), • Σ is an alphabet (*terminals*), • $S \in N_1$ (*initial non-terminal*), and • R is a finite $(N^* \times N)$-sorted set (*rules*). Each rule is of the form $A \to c(B_1, \ldots, B_k)$, where $k \in \mathbb{N}$, $A, B_1, \ldots, B_k \in N$, and $c \in \mathrm{C}^\Sigma_{(\mathrm{sort}_N(B_1) \cdots \mathrm{sort}_N(B_k), \mathrm{sort}_N(A))}$. The sort of the rule is $(B_1 \cdots B_k, A)$; we call A the *left-hand side* (*lhs*), $B_1, \ldots, B_k$ the *right-hand side* (*rhs*) and c the rule's *composition*. We drop the parentheses around the rhs if $k = 0$. We call rules of the form • $A \to c$ where $A \neq S$ *terminating*, • $A \to c(B)$ *monic*, and • $A \to c(B_1, ..., B_k)$ where $k \geq 2$ *branching*, and denote the set of these rules in R by $R^{(\mathrm{T})}$, $R^{(\mathrm{M})}$ and $R^{(\mathrm{B})}$, respectively. A rule is called *lexical*, if its composition contains *at least* one terminal. The lcfrs G is called *lexical*, if each rule is lexical. The *set of (complete) derivations in G* is $\mathrm{D}^G = \mathrm{T}^R_S$. Let $w \in \Sigma^*$ and $d = r(d_1, \ldots, d_k) \in \mathrm{T}^R$

with $r = A \to c(B_1, \ldots, B_k)$. The *yield of d* is $\mathrm{yd}(d) = [\![c]\!](\mathrm{yd}(d_1), \ldots, \mathrm{yd}(d_k))$.

PLCFRS. A *probabilistic LCFRS* is a tuple (G, μ) where $G = (N, \Sigma, S, R)$ is an LCFRS and $\mu\colon R \to [0, 1]$ maps each rule to a probability. The *weight of a derivation* $d = r(d_1, \ldots, d_k) \in \mathrm{T}^R$ is $\mathrm{wt}(d) = \mu(r) \cdot \mathrm{wt}(d_1) \cdots \mathrm{wt}(d_k)$.

Top-down Tree Transducers. A *top-down tree transducer* (*TT*) is a tuple $A = (Q, \Sigma, \Delta, q_\mathrm{i}, \delta)$ where • Q is a finite set (*states*), • Σ and Δ are ranked alphabets (*input* and *output alphabet*), • $q_\mathrm{i} \in Q$ (*initial state*), and • δ is a finite set (*transitions*). Each transition is of the form $q(\sigma(\mathrm{x}_1, \ldots, \mathrm{x}_k)) \to t$, where $\sigma \in \Sigma_k$ and $t \in \mathrm{T}^{\Delta \cup Q(\{\mathrm{x}_1, \ldots, \mathrm{x}_k\})}$. We call a transition *linear* (resp. *non-deleting*) if each variable in $\{\mathrm{x}_1, \ldots, \mathrm{x}_k\}$ occurs at most once (resp. at least once) in t. We call A deterministic if, for each $q \in Q$ an $\sigma \in \Sigma$, there is at most one transition of the form $q(\sigma(\mathrm{x}_1, \ldots, \mathrm{x}_k)) \to t$ in δ. We call it *linear* (resp. *non-deleting*) if each transition is linear (resp. non-deleting).

In a *TT with ε-rules* (*εTT*), the set δ may also contain transitions of the form $q(\mathrm{x}_1) \to t$ where $t \in \mathrm{T}^{\Delta \cup Q(\{\mathrm{x}_1\})}$ and $q \in Q$. We treat them analogously to transitions where $k = 1$.

The transduction of A is expressed in terms of the binary derivation relation $\Rightarrow_A$ over $\mathrm{T}^{\Delta \cup Q(\mathrm{T}^\Sigma)}$. We write $s \Rightarrow_A t$ if there is a subtree $q(\sigma(s_1, \ldots, s_k))$ in s and a transition $q(\sigma(\mathrm{x}_1, \ldots, \mathrm{x}_k)) \to t'$ such that t is obtained by replacing $q(\sigma(s_1, \ldots, s_k))$ in s by t' and replacing x_i by s_i for each $i \in [k]$. The *transduction of A* is the relation $[\![A]\!] = \{(s, t) \in \mathrm{T}^\Sigma \times \mathrm{T}^\Delta \mid q_\mathrm{i}(s) \Rightarrow_A^* t\}$. If A is deterministic, we consider $[\![A]\!]$ a function.

TTs are a special case of TTs with regular lookahead (*TT^R*) that were used by Engelfriet et al. (2018). For an excellent overview on tree transducers, we refer to Maletti (2010).

Equivalence of (P)LCFRS. Two LCFRS G and G' are called *ldTT^R-equivalent* if there are two linear and deterministic TT^R, T and T', such that • $[\![T]\!](d) \in \mathrm{D}^{G'}$ and $\mathrm{yd}([\![T]\!](d)) = \mathrm{yd}(d)$ for each $d \in \mathrm{D}^G$, and • $[\![T']\!](d') \in \mathrm{D}^G$ and $\mathrm{yd}([\![T']\!](d')) = \mathrm{yd}(d')$ for each $d' \in \mathrm{D}^{G'}$. Two PLCFRS (G, μ) and (G', μ') are called *ldTT^R-equivalent* if G and G' are ldTT^R-equivalent and • $\mathrm{wt}([\![T]\!](d)) \geq \mathrm{wt}(d)$ for each $d \in \mathrm{D}^G$, and • $\mathrm{wt}([\![T']\!](d')) \geq \mathrm{wt}(d')$ for each $d' \in \mathrm{D}^{G'}$.

$[\![T]\!]$, as well as $[\![T']\!]$, may map multiple derivations to one single derivation. The relation between the weights assures that this mapped derivation assumes the greatest of the original derivations' weights.

3 Lexicalizing LCFRS

For the remainder, we assume (without loss of generality; cf. Seki et al., 1991) a PLCFRS (G, μ) where $G = (N, \Sigma, S, R)$ is *terminal-* and *initial-separated*, i.e. each rule is either of the form $A \to (\sigma)$ with $\sigma \in \Sigma$ or $A \to c(B_1, \ldots, B_k)$ where $c \in \mathrm{C}$ (i.e. c contains no terminal symbols) and none of $B_1, \ldots, B_k$ is S. Starting with (G, μ), we incrementally construct a lexical PLCFRS in three steps:

1. Monic rules are removed.
2. Terminating rules are removed and, for each branching rule, each subset of rhs nonterminals is replaced with lexical symbols of matching terminating rules. This construction obtains terminating rules with at least two lexical symbols and each constructed monic rule contains at least one lexical symbol.
3. A terminal is cut from each terminating rule and pasted into a remaining non-lexical branching rule if a derivation with the branching rule reaches the terminating rule at some point. At the end of this step, each rule contains at least one terminal.

The first two steps are direct instances of the lexicalization for multiple context-free tree grammars as introduced by Engelfriet et al. (2018).

Step 1 (Dechain). In the first step, we remove each monic rule and chain its composition with the composition of each other reachable rule.

Definition 1. Let $k \in \mathbb{N}$, $s, s', s_1, \ldots, s_k \in \mathbb{N}_+$, $c \in \mathrm{C}_{(s_1 \cdots s_k, s)}$ and $c' \in \mathrm{C}_{(s, s')}$. We denote the composition $[\![c']\!](c) \in \mathrm{C}_{s_1 \cdots s_k, s'}$ by $c' \circ c$. □

The set of rules R_dc is the smallest set $\hat{R}$ such that • $R \setminus R^{(\mathrm{M})} \subseteq \hat{R}$ • for each $A \to c_1(B) \in R^{(\mathrm{M})}$ and $B \to c_2(C_1, ..., C_k) \in \hat{R}$, the rule $A \to c_1 \circ c_2\,(C_1, ..., C_k)$ is in $\hat{R}$. We define the function $\mu_\mathrm{dc}\colon R_\mathrm{dc} \to [0, 1]$ such that, for each rule $r = A \to c\,(C_1, ..., C_k) \in R_\mathrm{dc}$, the value is

$$\begin{aligned}\mu_\mathrm{dc}(r) = \max\, &\{\mu(r)\} \cup \\ &\{\mu(r_1) \cdot \mu_\mathrm{dc}(r_2) \mid r_1 = A \to c_1(B) \in R^{(\mathrm{M})}, \\ &\quad r_2 = B \to c_2(C_1, ..., C_k) \in R_\mathrm{dc}\colon c_1 \circ c_2 = c\}.\end{aligned}$$

The set R_dc, as well the function μ_dc, can be efficiently computed with an instance of the algorithm by Aho et al. (1974, alg. 5.5), cf. alg. 1 in app. A.

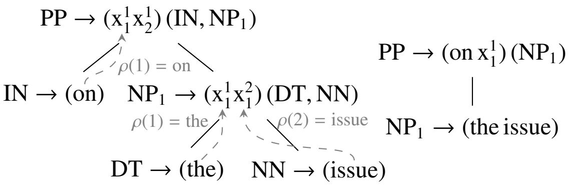
PP → (x$_1^1$x$_2^1$) (IN, NP$_1$)
ρ(1) = on
IN → (on)
NP$_1$ → (x$_1^1$x$_1^2$) (DT, NN)
ρ(1) = the
ρ(2) = issue
DT → (the)
NN → (issue)
PP → (on x$_1^1$) (NP$_1$)
NP$_1$ → (the issue)

Figure 2: Fusing of terminals. Left: derivation of G_{dc}. Right: corresponding derivation of G_{ft}, where the terminals of the leaves are inserted into their parent rules.

Step 2 (Fuse Terminals). In this step, the symbols of the terminating rules are inserted into non-terminating (terminal-free) rules. The intuition is given in Fig. 2. For the formal definition we first introduce the insertion operation.

Definition 2. Let $A \to c(B_1, \dots, B_k)$ be a rule, $\pi \subseteq \{i \in [k] \mid B_i \in N_1\}$ and $\rho\colon \pi \to \Sigma$. Moreover, let $i_1, \dots, i_{k-|\pi|}$ be the indices in $[k] \setminus \pi$ in ascending order. The composition $c[\rho]$ is obtained from c by replacing each variable of the form x_i^j • by $\rho(i)$ if $i \in \pi$, or • by x_ℓ^j if $i_\ell = i$. We denote $A \to c[\rho](B_{i_1}, \dots, B_{i_{k-|\pi|}})$ by $A \to c(B_1, \dots, B_k)[\rho]$. □

For each rule $r = A \to c(B_1, \dots, B_k)$, we define the set $F(r) = \{\rho\colon \pi \to \Sigma \mid \pi \subseteq [k], \forall i \in \pi\colon B_i \to (\rho(i)) \in R_{\mathrm{dc}}^{(\mathrm{T})}\}$. It contains each function that replaces a subset of r's rhs nonterminals with terminal symbols respecting the terminating rules; we use it to define the set of rules R_{ft} and the function $\mu_{\mathrm{ft}}\colon R_{\mathrm{ft}} \to [0, 1]$:

$$R_{\mathrm{ft}} = \{r[\rho] \mid r \in R_{\mathrm{dc}} \setminus R_{\mathrm{dc}}^{(\mathrm{T})}, \rho \in F(r)\},$$
$$\mu_{\mathrm{ft}}(r') = \max \{\mu_{\mathrm{dc}}(r) \cdot \prod\nolimits_{i \in \pi} \mu_{\mathrm{dc}}(B_i \to (\rho(i)))$$
$$\mid r \in R_{\mathrm{dc}} \setminus R_{\mathrm{dc}}^{(\mathrm{T})}, \rho \in F(r)\colon r[\rho] = r'\}.$$

Step 3 (Propagate Terminals). In this final step, terminal symbols are inserted into the remaining non-lexical branching rules. Intuitively, one terminal symbol is *cut* from each terminating rule, which now has at least two occurrences of terminal symbols. This symbol is then *pasted* into a non-lexical branching rule that reaches the rule it was cut from via its second right-hand side nonterminal. If there are rules between the terminating rule the symbol was cut from and the rule it shall be pasted into, then the information that a terminal can be pasted through these rules is propagated via the nonterminals. For this we extend the set of nonterminals to $N_{\mathrm{pt}} = N \cup N \times \Sigma$, where $(A, \sigma) \in N \times \Sigma$ indicates that σ was cut from a rule with left-hand side A ($\mathrm{sort}_{N_{\mathrm{pt}}}(A, \sigma) = \mathrm{sort}_N(A)$ for each $(A, \sigma) \in N \times \Sigma$). Fig. 3 shows an example of how the terminal symbols are propagated through a derivation.

Definition 3. Let $\sigma \in \Sigma$ and r be a rule of the form $A \to (\sigma u_1, u_2, ..., u_s)(B_1, ..., B_k)$. We denote $(A, \sigma) \to (u_1, ..., u_s)(B_1, ..., B_k)$ by $\mathrm{cut}(r)$. □

Definition 4. Let $r = A \to c\,(B_1, ..., B_k)$ be a rule and $i \in [k]$. We obtain c' from c by replacing the variable x_i^1 with $\sigma \mathrm{x}_i^1$ and denote $A \to c'\,(B_1, ..., (B_i, \sigma), ..., B_k)$ by $\mathrm{paste}_\sigma^i(r)$. □

Let $R_{\mathrm{ft}}^{(\not\Sigma)}$ denote the set of rules in $R_{\mathrm{ft}}^{(\mathrm{B})}$ without terminals. We define the sets of rules

$$R' = \{\,\mathrm{paste}_\sigma^2(r) \mid r \in R_{\mathrm{ft}}^{(\not\Sigma)}, \sigma \in \Sigma\} \cup (R_{\mathrm{ft}}^{(\mathrm{B})} \setminus R_{\mathrm{ft}}^{(\not\Sigma)})$$
$$R_{\mathrm{pt}} = (R_{\mathrm{ft}} \setminus R_{\mathrm{ft}}^{(\mathrm{B})}) \cup R'$$
$$\cup \{\,\mathrm{cut}(A \to c) \mid A \to c \in R_{\mathrm{ft}}^{(\mathrm{T})}\}$$
$$\cup \{\,\mathrm{cut}(\mathrm{paste}_\sigma^1(r)) \mid r \in R' \cup R_{\mathrm{ft}}^{(\mathrm{M})}, \sigma \in \Sigma\}.$$

Note that, in contrast to Engelfriet et al. (2018), we do not need to split the rules' compositions, because we always cut the first symbol. Therefore, the fanout of the grammar is unchanged.

The applications of cut and paste that led to a rule in R_{pt} are unambiguously determined from the lhs and rhs nonterminals in $N \times \Sigma$. These operations are unambiguously reversible; for each $r' \in R_{\mathrm{pt}}$, there is a rule in $r \in R_{\mathrm{ft}}$ uniquely determined such that r' is exactly one of • r, • $\mathrm{paste}_\sigma^2(r)$, • $\mathrm{cut}(r)$, • $\mathrm{cut}(\mathrm{paste}_\sigma^1(r))$, or • $\mathrm{cut}(\mathrm{paste}_{\sigma_1}^1(\mathrm{paste}_{\sigma_2}^2(r)))$. We define the function $\mu_{\mathrm{pt}}\colon R_{\mathrm{pt}} \to [0, 1]$ such that $\mu_{\mathrm{pt}}(r') = \mu_{\mathrm{ft}}(r)$ for each $r' \in R_{\mathrm{pt}}$.

Theorem 5. The PLCFRS $((N_{\mathrm{pt}}, \Sigma, S, R_{\mathrm{pt}}), \mu_{\mathrm{pt}})$ and (G, μ) are ldTT$^{\mathrm{R}}$-equivalent.

4 Unlexicalizing Derivations

The ultimate goal of parsing is to obtain derivations of the original grammar (G, μ), which are quite different from the derivations of the transformed grammar $((N_{\mathrm{pt}}, \Sigma, S, R_{\mathrm{pt}}), \mu_{\mathrm{pt}})$. Therefore we seek a transformation from $\mathrm{T}^{R_{\mathrm{pt}}}$ to T^{R}.

Engelfriet et al. (2018) have introduced deterministic linear and non-deleting TT$^{\mathrm{R}}$ from $\mathrm{T}^{R_{\mathrm{pt}}}$ to $\mathrm{T}^{R_{\mathrm{ft}}}$, from $\mathrm{T}^{R_{\mathrm{ft}}}$ to $\mathrm{T}^{R_{\mathrm{dc}}}$, and from $\mathrm{T}^{R_{\mathrm{dc}}}$ to T^{R}; they were used to show ldTT$^{\mathrm{R}}$-equivalence of the corresponding grammars. The composition of these transducers yields a transduction from $\mathrm{T}^{R_{\mathrm{pt}}}$ to T^{R}. However, for recovering derivations, these transducers are not adequate, as a derivation in $\mathrm{T}^{R_{\mathrm{ft}}}$ or $\mathrm{T}^{R_{\mathrm{dc}}}$ may have multiple possible originals in $\mathrm{T}^{R_{\mathrm{dc}}}$ or T^{R}, respectively. We want to be able to obtain all of these derivations, as this may be beneficial for

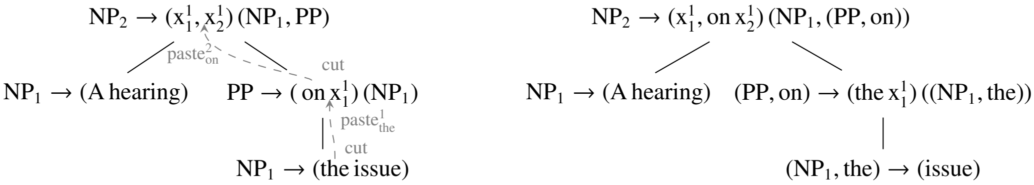

Figure 3: Propagation of terminals. Left: derivation of a (partly lexicalized) LCFRS before the application of step 3. Right: derivation after step 3, where *the* is propagated from the leaf to the PP rule and *on* is propagated from there to the NP_2 rule, thus lexicalizing it.

later stages of an application (e.g., when selecting k best derivations is desired).

For this, we employ two approaches. We map each derivation in $\mathrm{T}^{R_{\mathrm{pt}}}$ to its unique original derivation by using the transducer of (Engelfriet et al., 2018); we denote this transducer by $T_{\overleftarrow{\mathrm{pt}}}$. It realizes a deterministic tree relabeling which is already indicated at the end of the previous section. For the other two transductions, we define novel non-deterministic tree transducers.

Transduction $\mathrm{T}^{R_{\mathrm{ft}}} \rightarrow \mathrm{T}^{R_{\mathrm{dc}}}$**.** Let $r = A \rightarrow c(B_1, ..., B_k) \in R_{\mathrm{ft}}$, $\pi \subseteq \{i \in [k] \mid B_i \in N_1\}$, $\rho\colon \pi \rightarrow \Sigma$ and $i_1, \ldots, i_{k-|\pi|}$ be the elements of $[k] \setminus \pi$ in ascending order. We denote the tree $r(d_1, \ldots, d_k)$, where $d_i = B_i \rightarrow (\rho(i))$ if $i \in \pi$ and $d_i = *(x_\ell)$ if $i = i_\ell$, by $r(x_1, \ldots, x_k)[\rho]$.

We define the linear and non-deleting TT $T_{\overleftarrow{\mathrm{ft}}} = (\{*\}, R_{\mathrm{ft}}, R_{\mathrm{dc}}, *, \delta)$, where δ is the smallest set such that, for each rule $r = A \rightarrow c(B_1, ..., B_k) \in R_{\mathrm{dc}} \backslash R_{\mathrm{dc}}^{(\mathrm{T})}$ and $\rho \in F(r)$, the transition $*(r[\rho](\mathrm{x}_1, ..., \mathrm{x}_{k-|\pi|})) \rightarrow r(\mathrm{x}_1, ..., \mathrm{x}_k)[\rho]$ is in δ.

Transduction $\mathrm{T}^{R_{\mathrm{dc}}} \rightarrow \mathrm{T}^{R}$**.** Let us denote the composition $(\mathrm{x}_1, ..., \mathrm{x}_{\mathrm{sort}_N(A)})$ by id_A for each $A \in N$. We define the linear and non-deleting εTT $T_{\overleftarrow{\mathrm{dc}}} = (Q, R_{\mathrm{dc}}, R, (S, \mathrm{id}_S), \delta)$ where $Q = \bigcup_{A,B \in N} \{B\} \times \mathrm{C}_{(\mathrm{sort}_N(B), \mathrm{sort}_N(A))}$ and δ is the smallest set that contains,

- for each $(B, c) \in Q$, $r = B \rightarrow c'(B_1, \ldots B_k) \in R \setminus R^{(\mathrm{M})}$ and $A \in N$, the transition
$$(B, c)\,(A \rightarrow c \circ c'\,(B_1, ..., B_k)\,(\mathrm{x}_1, ..., \mathrm{x}_k)) \\ \rightarrow r\,((B_1, \mathrm{id}_{B_1})(\mathrm{x}_1), ..., (B_k, \mathrm{id}_{B_k})(\mathrm{x}_k)),$$
- for each $(A, c) \in Q$ and $r = A \rightarrow c'(B) \in R^{(\mathrm{M})}$, the transition $(A, c)(\mathrm{x}_1) \rightarrow r\,((B, c \circ c')(\mathrm{x}_1))$.

The transduction $[\![T_{\overleftarrow{\mathrm{dc}}}]\!] \circ [\![T_{\overleftarrow{\mathrm{ft}}}]\!] \circ [\![T_{\overleftarrow{\mathrm{pt}}}]\!]\colon \mathrm{T}^{R_{\mathrm{pt}}} \rightarrow \mathrm{T}^{R}$, i.e., the composition of the three transductions introduced in this section, is the inverse of the transduction $[\![T]\!]$ from the proof of Theorem 5 (cf. App. B, p. 7). Therefore, the k best derivations in (G, μ) must be among the transductions of the k best derivations in $((N_{\mathrm{pt}}, \Sigma, S, R_{\mathrm{pt}}), \mu_{\mathrm{pt}})$, which benefits the enumeration of k best derivations.

5 Conclusion

Based on Engelfriet et al. (2018), we have introduced a procedure which constructs for every PLCFRS G an equivalent lexicalized PLCFRS G'. Moreover, we have described how to recover from each derivation of G' all corresponding derivations of G. In future work, we will use our approach to implement a supertagging-based LCFRS parser.

Acknowledgements

We thank our colleague Kilian Gebhardt as well as the anonymous reviewers for their insightful comments on drafts of this paper.

References

Alfred V Aho, John E. Hopcroft, and Jeffrey D. Ullman. 1974. *The design and analysis of computer algorithms*. Pearson Education India.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational linguistics*, 25(2):237–265.

Joost Engelfriet, Andreas Maletti, and Sebastian Maneth. 2018. Multiple context-free tree grammars: Lexicalization and characterization. *Theoretical Computer Science*, 728:29–99.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. 10(1):136–163.

Andreas Maletti. 2010. Survey: Tree transducers in machine translation. In *NCMA*, pages 11–32. Citeseer.

Andreas Maletti and Giorgio Satta. 2009. Parsing algorithms based on tree automata. In *Proceedings of* *the 11th International Conference on Parsing Technologies*, IWPT '09, pages 1–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. 88(2):191–229.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with LSTMs. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–237.

Krishnamurti Vijay-Shanker, David Jeremy Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, ACL '87, pages 104–111, Stroudsburg, PA, USA. Association for Computational Linguistics.

A Supplemental Algorithms

Algorithm 1 COMPS shows an instance of the alg. by (Aho et al., 1974, alg. 5.5) that we use to compute R_{dc} and μ_{dc} efficiently.

```
Require: terminal-separated PLCFRS $(G, \mu)$ where $G = (N, \Sigma, S, R)$
Ensure: DECHAIN$(N, R, \mu) = (R_{\mathrm{dc}}, \mu_{\mathrm{dc}})$
max $v \leftarrow w$ denotes $v \leftarrow \max(v, w)$
function COMPS$(N, R, \mu)$
    let $(m_{A\to c(B)} = 0 \mid A, B \in N, c \in C)$
    $m_r \leftarrow \mu(r) \quad \forall r \in R^{(\mathrm{M})}$
    $m_{A\to \mathrm{id}_A(A)} \leftarrow 1 \quad \forall A$
    for $B \in N$ do
        for $A, C \in N \setminus \{B\}$ do
            for $c_1, c_2 \in \mathrm{C}$ do
                $w' \leftarrow m_{A\to c_1(B)} \cdot m_{B\to c_2(C)}$
                max $m_{A\to c_1\circ c_2(C)} \leftarrow w'$
    return $m$
function DECHAIN$(N, R, \mu)$
    $m \leftarrow$ COMPS$(N, R, \mu)$
    let $(\hat{\mu}_{A\to c(B_1,...,B_k)} = 0 \mid A, B_1, ..., B_k \in N, c \in C)$
    $R' \leftarrow R \setminus R^{(\mathrm{M})}$
    $\hat{\mu}_{r'} = \mu(r') \quad \forall r' \in R'$
    for $r = B \to c_2(C_1, ..., C_k) \in R \setminus R^{(\mathrm{M})}$ do
        for $A \in N, c_1 \in \mathrm{C}\colon m_{A\to c_1(B)} \neq 0$ do
            $r' \leftarrow A \to c_1 \circ c_2(C_1, ..., C_k)$
            $R' \leftarrow R' \cup \{r'\}$
            max $\hat{\mu}_{r'} \leftarrow m_{A\to c_1(B)} \cdot \mu(r)$
    let $\mu\colon R' \to [0, 1]$ with $\mu'(r') = \hat{\mu}_{r'} \forall r' \in R'$
    return $(R', \mu')$
```

B Supplemental Proofs

We prove thm. 5 in two steps.

First lem. 7 shall prove that the (unweighted) underlying grammars, G and $(N_{\mathrm{pt}}, \Sigma, S, R_{\mathrm{pt}})$ are ldTT$^{\mathrm{R}}$-equivalent. As linear and deterministic TT$^{\mathrm{R}}$ are closed under composition, the idea is to construct two of them for each step, one that transduces derivations in the original to derivations in the constructed grammar and vice versa. The three transducers for each direction are then composed to obtain transductions from derivations in G to derivations in $(N_{\mathrm{pt}}, \Sigma, S, R_{\mathrm{pt}})$ and vice versa.

Thm. 5 additionally proves the preservation of the weights. Similarly to the above, we show that the PLCFRS constructed in each of the three steps are ldTT$^{\mathrm{R}}$-equivalent. The property is clearly transitive.

Definition 6. A *TT^R* is a tuple $A = (Q, \Sigma, \Delta, q_{\mathrm{i}}, \delta)$ where Q, Σ, Δ and q_{i} are as in TT, and transitions in δ are of the form $q(\sigma(\mathrm{x}_1\colon L_1, \dots, \mathrm{x}_k\colon L_k)\colon L_0) \to t$ where $q \in Q$, $\sigma \in \Sigma_k$, $t \in \mathrm{T}^{\Delta\cup Q(\{\mathrm{x}_1,\dots,\mathrm{x}_k\})}$ and $L_0, \dots, L_k$ are regular tree languages.[1] We omit some of the languages $L_0, \dots, L_k$ if they are T^{Σ}. We define the binary relation $\Rightarrow_A$ over $\mathrm{T}^{\Delta\cup Q(\mathrm{T}^{\Sigma})}$ such that $s \Rightarrow_A t$ if there is a subtree $q(\sigma(s_1, \dots, s_k))$ in s and a transition $q(\sigma(\mathrm{x}_1\colon L_1, \dots, \mathrm{x}_k\colon L_k)\colon L_0) \to t'$ such that • $\sigma(s_1, \dots, s_k) \in L_0$, $s_1 \in L_1, \dots, s_k \in L_k$, and • t is obtained by replacing $q(\sigma(s_1, \dots, s_k))$ in s by t' and replacing x_i by s_i for each $i \in [k]$. □

Lemma 7. The LCFRS G and $(N_{\mathrm{pt}}, \Sigma, S, R_{\mathrm{pt}})$ are ldTT$^{\mathrm{R}}$-equivalent.

Proof. The first two steps are instances of lems. 32 and 37 by Engelfriet et al. (2018), therefore we will only show the third step.

Step 3. For the construction, we consider R_{ft} and R_{pt} as *ranked* sets. For each $\hat{R} \in \{R_{\mathrm{ft}}, R_{\mathrm{pt}}\}$ and rule of the form $r = A \to c(B_1, \dots, B_k) \in \hat{R}$, we let $\mathrm{rk}_{\hat{R}}(r) = k$.

Let, for each $\sigma \in \Sigma$, R_σ and R_{X} be the sets of all rules in R_{ft} of the form $A \to c(B_1, \dots, B_k)$ where c is of the form $(\sigma u_1, u_2, \dots, u_s)$ and $(\mathrm{x}_1^1 u_1, u_2, \dots, u_s)$, respectively. To decide whether a terminal symbol may be propagated through a derivation, we define the look-ahead language L_σ

[1] Regular tree languages are recognized by regular tree automata. We refer to Maletti and Satta (2009) for an overview.

for each $\sigma \in \Sigma$ as the smallest set L such that

$$\begin{aligned} L = \{r(d_1, \dots, d_k) \mid r \in R_\sigma, d_1, \dots, d_k \in \mathrm{T}^{R_{\mathrm{ft}}}\} \\ \cup \{r(d_1, \dots, d_k) \mid r \in R_{\mathrm{x}}, d_1 \in L, \\ d_2, \dots, d_k \in \mathrm{T}^{R_{\mathrm{ft}}}\}. \end{aligned}$$

Observation 8. Let $d \in \mathrm{T}^{R_{\mathrm{ft}}}$ and $\sigma \in \Sigma$. The following are equal: 1. $d \in L_\sigma$, and 2. $\mathrm{yd}(d)$ is of the form $(\sigma u_1, u_2, \dots, u_s)$.

We define the ldTT$^{\mathrm{R}}$ $T_{\mathrm{pt}} = (\{\mathrm{p}, *\}, R_{\mathrm{ft}}, R_{\mathrm{pt}}, *, \delta)$ where δ contains the following transitions

1. $*(r(\mathrm{x}_1, \dots, \mathrm{x}_k)) \rightarrow r(*(\mathrm{x}_1), \dots, *(\mathrm{x}_k))$ for each $r \in R_{\mathrm{ft}} \setminus R_{\mathrm{ft}}^{(\not\Sigma)}$,
2. for each $r \in R_{\mathrm{ft}}^{(\not\Sigma)}$, $*(r(\mathrm{x}_1, \mathrm{x}_2 \colon L_\sigma, \mathrm{x}_3, \dots, \mathrm{x}_k))$ $\rightarrow \mathrm{paste}_\sigma^2(r)(*(\mathrm{x}_1), \mathrm{p}(\mathrm{x}_2), *(\mathrm{x}_3), \dots, *(\mathrm{x}_k))$,
3. $\mathrm{p}(r) \rightarrow \mathrm{cut}(r)$ for each $r \in R_{\mathrm{ft}}^{(\mathrm{T})}$,
4. for $r \in R_{\mathrm{ft}}^{(\mathrm{M})} \cup (R_{\mathrm{ft}}^{(\mathrm{B})} \setminus R_{\mathrm{ft}}^{(\not\Sigma)})$ and $\sigma \in \Sigma$, $\mathrm{p}(r(\mathrm{x}_1 \colon L_\sigma, \mathrm{x}_2, \dots, \mathrm{x}_k))$ $\rightarrow \mathrm{cut}(\mathrm{paste}_\sigma^1(r))(\mathrm{p}(\mathrm{x}_1), *(\mathrm{x}_2), \dots, *(\mathrm{x}_k))$, and
5. for each $r \in R_{\mathrm{ft}}^{(\not\Sigma)}$ and $\sigma_1, \sigma_2 \in \Sigma$, $\mathrm{p}(r(\mathrm{x}_1 \colon L_{\sigma_1}, \mathrm{x}_2 \colon L_{\sigma_2}, \mathrm{x}_3, \dots, \mathrm{x}_k))$ $\rightarrow r'(\mathrm{p}(\mathrm{x}_1), \mathrm{p}(\mathrm{x}_2), *(\mathrm{x}_3), \dots, *(\mathrm{x}_k))$ where $r' = \mathrm{cut}(\mathrm{paste}_{\sigma_1}^1(\mathrm{paste}_{\sigma_2}^2(r)))$.

Let us have a close look at some constructions in these transitions. Let $r = A \rightarrow c(B_1, \dots, B_k)$ be a rule and $(v_i \in (\Sigma^*)^{\mathrm{fo}(B_i)} \mid i \in [k])$. We denote the composition of $\mathrm{cut}(r)$ by $\mathrm{cut}(c)$ and the composition of $\mathrm{paste}_\sigma^i(r)$ by $\mathrm{paste}_\sigma^i(c)$.

Observation 9. Let $\mathrm{cut}(c)$ be defined, then $[\![\mathrm{cut}(c)]\!](v_1, \dots, v_k) = \mathrm{cut}([\![c]\!](v_1, \dots, v_k))$.

Observation 10. Let v_i be of the form $(\sigma u_1, u_2, \dots, u_s)$ for $i \in [k]$ and $\sigma \in \Sigma$. $[\![\mathrm{paste}_\sigma^i(c)]\!](v_1, \dots, \mathrm{cut}(v_i), \dots, v_k) = [\![c]\!](v_1, \dots, v_k)$.

Using the observations 8–10 one can easily show that, for each $d \in \mathrm{T}^{R_{\mathrm{ft}}}$ and $d' \in \mathrm{T}^{R_{\mathrm{pt}}}$: 1. if $*(d) \Rightarrow_{T_{\mathrm{pt}}}^* d'$, then $\mathrm{yd}(d) = \mathrm{yd}(d')$, and 2. if $\mathrm{p}(d) \Rightarrow_{T_{\mathrm{pt}}}^* d'$, then $\mathrm{yd}(d') = (u_1, \dots, u_s)$ and $d \in L_\sigma$ where $(\sigma u_1, \dots, u_s) = \mathrm{yd}(d)$. This concludes the proof in the direction $\mathrm{T}^{R_{\mathrm{ft}}} \rightarrow \mathrm{T}^{R_{\mathrm{pt}}}$.

For the other direction, we observe that the applications of cut and paste to obtain the rules in R_{pt} can easily be determined by the occurrences of the nonterminals in $N \times \Sigma$. The transduction $\mathrm{T}^{R_{\mathrm{pt}}} \rightarrow \mathrm{T}^{R_{\mathrm{ft}}}$ is thus a deterministic relabeling and can be implemented by a deterministic top-down tree transducer. Engelfriet et al. (2018, lem. 42, pg. 39) came to the same conclusion for their construction. ■

Theorem 5. The PLCFRS $((N_{\mathrm{pt}}, \Sigma, S, R_{\mathrm{pt}}), \mu_{\mathrm{pt}})$ and (G, μ) are ldTT$^{\mathrm{R}}$-equivalent.

Proof sketch. We split the proof into the two remaining properties to show:

1. $\mathrm{wt}([\![T]\!](d)) \geq \mathrm{wt}(d)$ for each $d \in \mathrm{T}^R$, and
2. $\mathrm{wt}([\![T']\!](d')) \geq \mathrm{wt}(d')$[2] for each $d' \in \mathrm{T}^{R_{\mathrm{pt}}}$.

(item 1) The weight functions in sec. 3 are obviously defined in such a way that the weight of a constructed rule is the greatest product of weights for all involved rules in the construction. For each step, the transducer T replaces the rules in the derivation d with an unambiguous constructed rule (this follows from the inductive structure of the previous proof). Therefore, the weight $\mathrm{wt}([\![T]\!](d))$ must be greater than $\mathrm{wt}(d)$.

(item 2) For each step, the transducer T' gives us some derivation that contains the rules that were used to construct the rules in d'. The construction of T' for the steps 1 and 2 is ambiguous, so that we can choose any combination of rules used to construct r' for each rule r' in d'. $[\![T']\!](d')$ then contains each rule in these combinations. Intuitively, we choose the combination of rules with greatest product of weights. Then the weight $\mathrm{wt}([\![T']\!](d'))$ is $\mathrm{wt}(d')$. ■

[2] More specifically, we prove $\mathrm{wt}([\![T']\!](d')) = \mathrm{wt}(d')$ for each $d' \in \mathrm{T}^{R_{\mathrm{pt}}}$.