Feature Unification in TAG Derivation Trees

Sylvain Schmitz
LORIA, INRIA Nancy Grand Est, France
Sylvain.Schmitz@loria.fr

Joseph Le Roux
LORIA, Nancy Université, France
Joseph.Leroux@loria.fr

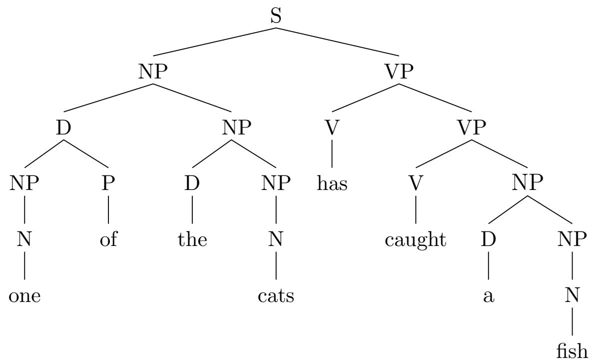
Abstract

The derivation trees of a tree adjoining grammar provide a first insight into the sentence semantics, and are thus prime targets for generation systems. We define a formalism, *feature-based regular tree grammars*, and a translation from feature based tree adjoining grammars into this new formalism. The translation preserves the derivation structures of the original grammar, and accounts for feature unification.

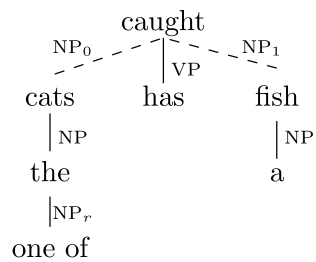
1 Introduction

Each sentence derivation in a tree adjoining grammar (Joshi and Schabes, 1997, TAG) results in two parse trees: a *derived tree* (Figure 1a), that represents the phrase structure of the sentence, and a *derivation tree* (Figure 1b), that records how the elementary trees of the grammar were combined. Each type of parse tree is better suited for a different set of language processing tasks: the derived tree is closely related to the lexical elements of the sentence, and the derivation tree offers a first insight into the sentence semantics (Candito and Kahane, 1998). Furthermore, the derivation tree language of a TAG, being a regular tree language, is much simpler to manipulate than the corresponding derived tree language.

Derivation trees are thus the cornerstone of several approaches to sentence generation (Koller and Striegnitz, 2002; Koller and Stone, 2007), that rely crucially on the ease of encoding regular tree grammars, as dependency grammars and planning problems respectively. Derivation trees also serve as intermediate representations from which both derived trees (and thus the linear order information) and semantics can be computed, e.g. with the abstract categorial grammars of de Groote (2002),



(a) Derived tree.



(b) Derivation tree.

Figure 1: Parse trees for "One of the cats has caught a fish." using the grammar of Figure 2.

Pogodalla (2004), and Kanazawa (2007), or similarly with the bimorphisms of Shieber (2006).

Nevertheless, these results do not directly apply to many real-world grammars, which are expressed in a feature-based variant of TAGs (Vijay-Shanker, 1992). Each elementary tree node of these grammars carries two feature structures that constrain the allowed substitution or adjunction operations at this node (see for instance Figure 2). In theory, such structures are unproblematic, because the possible feature values are drawn from finite domains, and thus the number of grammar categories could be increased in order to account for all the possible structures. In practice, the sheer number of structures precludes such a naive im-

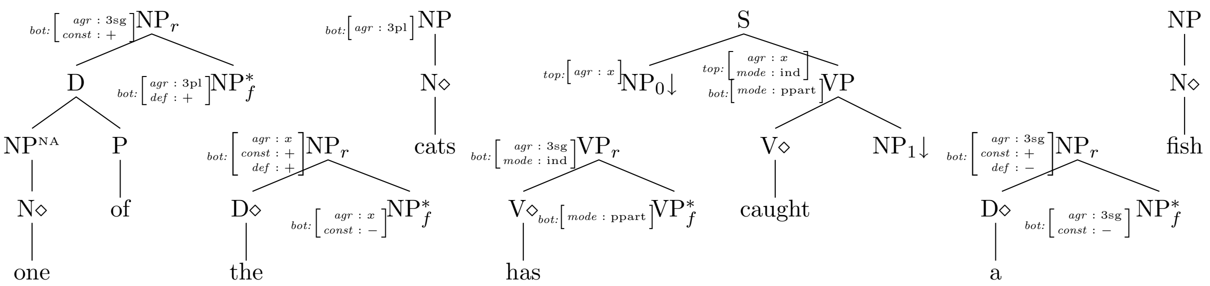


Figure 2: A feature-based tree adjoining grammar. For the sake of clarity, we identify elementary trees with their anchors in our examples.

plementation: for instance, the 50 features used in the XTAG English grammar (XTAG Research Group, 2001) together define a domain containing more than 10^{19} different structures. Furthermore, finiteness does not hold for some grammars, for instance with the semantic features of Gardent and Kallmeyer (2003).

Ignoring feature structures typically results in massive over-generation in derivation-centric systems. We define a formalism, *feature-based regular tree grammars*, that produces derivation trees that account for the feature structures found in a tree adjoining grammar. In more details,

- we recall how to generate the derivation trees of a tree adjoining grammar through a regular tree grammar (Section 2), then
- we define feature-based regular tree grammars and present the translation from feature-based TAG (Section 3); finally,
- we provide an improved translation inspired by left corner transformations (Section 4).

We assume the reader is familiar with the theory of tree-adjoining grammars (Joshi and Schabes, 1997), regular tree grammars (Comon et al., 2007), and feature unification (Robinson, 1965).

2 Regular Tree Grammars of Derivations

In this section, we define an encoding of the set of derivation trees of a tree adjoining grammar as the language of a regular tree grammar (RTG). Several encodings equivalent to regular tree grammars have been described in the literature; we follow here the one of de Groote (2002), but explicitly construct a regular tree grammar.

Formally, a tree adjoining grammar is a tuple $\langle \Sigma, N, I, A, S \rangle$ where Σ is a terminal alphabet, N is a nonterminal alphabet, I is a set of initial trees α, A is a set of auxiliary trees β and S is a distinguished nonterminal from N. We note γ_r the root node of the elementary tree γ and β_f the foot node of the auxiliary tree β. Let us denote by $\gamma_1, \ldots, \gamma_n$ the *active* nodes of an elementary tree γ, where a substitution or an adjunction can be performed;[1] we call n the *rank* of γ, denoted by $\mathsf{rk}(\gamma)$. We set γ_1 to be the root node of γ, i.e. $\gamma_1 = \gamma_r$. Finally, $\mathsf{lab}(\gamma_i)$ denotes the label of node γ_i.

Each elementary tree γ of the TAG will be converted into a single rule $X \to \gamma(Y_1, \ldots, Y_n)$ of our RTG, such that $\mathsf{rk}(\gamma) = n$ and each of the Y_i symbols represents the possible adjunctions or substitutions of node γ_i. We introduce accordingly two duplicates $N_A = \{X_A \mid X \in N\}$ and $N_S = \{X_S \mid X \in N\}$ of N, and a nonterminal labeling function defined for any active node γ_i with label $\mathsf{lab}(\gamma_i) = X$ as

$$\mathsf{nt}(\gamma_i) = \begin{cases} X_A & \text{if } \gamma_i \text{ is an adjunction site} \\ X_S & \text{if } \gamma_i \text{ is a substitution site} \end{cases} \quad (1)$$

The rule corresponding to the tree "one of" in Figure 2 is then $NP_A \to \text{one of}(NP_A, D_A, P_A, N_A)$, meaning that this tree adjoins into an NP labeled node, and expects adjunctions on its nodes NP_r, D, P, and N. Given our set of elementary TAG trees, only the first one of these four will be useful in a reduced RTG.

Definition 1. The *regular derivation tree grammar* $G = \langle S_S, \mathcal{N}, \mathcal{F}, R \rangle$ of a TAG $\langle \Sigma, N, I, A, S \rangle$ is a RTG with axiom S_S, nonterminal alphabet $\mathcal{N} = N_S \cup N_A$, terminal alphabet $\mathcal{F} = I \cup A \cup$

[1] We consider in particular that no adjunction can occur at a foot node. We do not consider null adjunctions constraints on root nodes and feature structures on null adjoining nodes, which would rather obscure the presentation, and we do not treat other adjunction constraints either.

$\{\varepsilon_A\}$ with ranks $\mathsf{rk}(\gamma)$ for elementary trees γ in $I \cup A$ and rank 0 for ε_A, and with set of rules

$$
\begin{aligned}
R = \{&X_S \to \alpha(\mathsf{nt}(\alpha_1), \ldots, \mathsf{nt}(\alpha_n)) \\
&\quad \mid \alpha \in I, n = \mathsf{rk}(\alpha), X = \mathsf{lab}(\alpha_r)\} \\
\cup\ \{&X_A \to \beta(\mathsf{nt}(\beta_1), \ldots, \mathsf{nt}(\beta_n)) \\
&\quad \mid \beta \in A, n = \mathsf{rk}(\beta), X = \mathsf{lab}(\beta_r)\} \\
\cup\ \{&X_A \to \varepsilon_A \mid X_A \in N_A\}
\end{aligned}
$$

□

The ε-rules $X_A \to \varepsilon_A$ for each symbol X_A account for adjunction sites where no adjunction takes place. The RTG has the same size as the original TAG and the translation can be computed in linear time.

Example 2. The reduced regular tree grammar corresponding to the tree adjoining grammar of Figure 2 is then:

$$
\begin{aligned}
&\langle S_S, \{S_S, VP_S, VP_A, NP_S, NP_A\}, \\
&\ \{\text{one of}, \text{the}, \text{cats}, \text{has}, \text{caught}, \text{a}, \text{fish}, \varepsilon_A\}, \\
&\ \{\ S_S \to \text{caught}(NP_S, VP_A, NP_S), \\
&\ \ NP_S \to \text{cats}(NP_A), \\
&\ \ NP_S \to \text{fish}(NP_A), \\
&\ \ NP_A \to \text{the}(NP_A), \\
&\ \ NP_A \to \text{a}(NP_A), \\
&\ \ NP_A \to \text{one of}(NP_A), \\
&\ \ NP_A \to \varepsilon_A, \\
&\ \ VP_A \to \text{has}(VP_A), \\
&\ \ VP_A \to \varepsilon_A\}\rangle
\end{aligned}
$$

□

Let us recall that the derivation relation induced by a regular tree grammar $G = \langle S_S, \mathcal{N}, \mathcal{F}, R\rangle$ relates terms[2] of $T(\mathcal{F}, \mathcal{N})$, so that $t \Rightarrow t'$ holds iff there exists a context[3] C and a rule $A \to a(B_1, \ldots, B_n)$ such that $t = C[A]$ and $t' = C[a(B_1, \ldots, B_n)]$. The language of the RTG is $L(G) = \{t \in T(\mathcal{F}) \mid S_S \Rightarrow^* t\}$.

One can check that the grammar of Example 2 generates trees with a root labeled with "caught", and three subtrees, the leftmost and rightmost of which labeled with "cats" or "fish" followed by an arbitrary long combination of nodes labeled with "one of", "a" or "the". The central subtree is an arbitrary long combination of nodes labeled with "has". Each branch terminates with ε_A. Two of these trees can be seen on Figure 3. Our RTG generates the derivation trees of a version of the original TAG expunged from its feature structures.

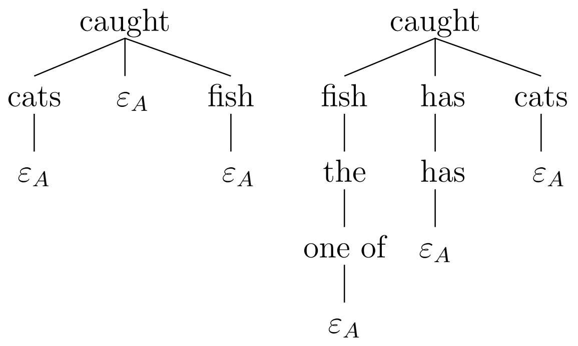


Figure 3: Some trees generated by the regular tree grammar of Example 2.

3 Unification on TAG Derivation Trees

3.1 Feature-based Regular Tree Grammars

In order to extend the previous construction to feature-based TAGs, our RTGs use combinations of rewrites and unifications—also dubbed *narrowings* (Hanus, 1994)—of terms with variables in $\mathcal{N} \times \mathcal{D}$, where $\mathcal{N}$ denotes the nonterminal alphabet and $\mathcal{D}$ the set of feature structures.[4]

Definition 3. A *feature-based regular tree grammar* $\langle S, \mathcal{N}, \mathcal{F}, \mathcal{D}, R\rangle$ comprises an axiom S, a set $\mathcal{N}$ of nonterminal symbols that includes S, a ranked terminal alphabet $\mathcal{F}$, a set $\mathcal{D}$ of feature structures, and a set R of rules of form $(A, d) \to a((B_1, d'_1), \ldots, (B_n, d'_n))$, where $A, B_1, \ldots, B_n$ are nonterminals, $d, d'_1, \ldots, d'_n$ are feature structures, and a is a terminal with rank n.

The *derivation* relation $\Rightarrow$ for a feature-based RTG $G = \langle S, \mathcal{N}, \mathcal{F}, \mathcal{D}, R\rangle$ relates pairs of terms from $T(\mathcal{F}, \mathcal{N} \times \mathcal{D})$ and u-substitutions, such that $(s, e) \Rightarrow (t, e')$ iff there exist a context C, a rule $(A, d) \to a((B_1, d'_1), \ldots, (B_n, d'_n))$ in R with fresh variables in the feature structures, a structure

[2] The set of *terms* over the alphabet $\mathcal{F}$ and the set of variables $\mathcal{X}$ is denoted by $T(\mathcal{F}, \mathcal{X})$; $T(\mathcal{F}, \emptyset) = T(\mathcal{F})$ is the set of trees over $\mathcal{F}$.

[3] A *context* C is a term of $T(\mathcal{F}, \mathcal{X} \cup \{x\})$, $x \notin \mathcal{X}$, which contains a single occurrence of x. The term $C[t]$ for some term t of $T(\mathcal{F}, \mathcal{X})$ is obtained by replacing this occurrence by t.

[4] In order to differentiate TAG tree substitutions from term substitutions, we call the latter *u-substitutions*. Given two feature structures d and d' in $\mathcal{D}$, we denote by the u-substitution $\sigma = \mathsf{mgu}(d, d')$ their *most general unifier* if it exists. We denote by $\top$ the most general element of $\mathcal{D}$, and by id the identity.

d', and an u-substitution σ verifying

$$s = C[(A, d')],\ \sigma = \mathsf{mgu}(d, e(d')),\ e' = \sigma \circ e$$
$$\text{and } t = C[a((B_1, \sigma(d'_1)), \ldots, (B_n, \sigma(d'_n)))].$$

The *language* of G is

$$L(G) = \{t \in T(\mathcal{F}) \mid \exists e, ((S, \top), id) \Rightarrow^* (t, e)\}.$$

□

Features percolate hierarchically through the computation of the most general unifier mgu at each derivation step, while the global u-substitution e acts as an environment that communicates unification results between the branches of our terms.

Feature-based RTGs with a finite domain $\mathcal{D}$ are equivalent to regular tree grammars. Unrestricted feature-based RTGs can encode Turing machines just like unification grammars (Johnson, 1988), and thus we can reduce the halting problem on the empty input for Turing machines to the emptiness problem for feature-based RTGs, which is thereby undecidable.

3.2 Encoding Feature-based TAGs

For each tree γ with rank n, we now create a rule $P \to \gamma(P_1, \ldots, P_n)$. A right-hand side pair $P_i = (\mathsf{nt}(\gamma_i), d'_i)$ stands for an active node γ_i with feature structure $d'_i = \mathsf{feats}(\gamma_i) = \begin{bmatrix} top : \mathsf{top}(\gamma_i) \\ bot : \mathsf{bot}(\gamma_i) \end{bmatrix}$, where $\mathsf{top}(\gamma_i)$ and $\mathsf{bot}(\gamma_i)$ denote respectively the top and bottom feature structures of γ_i.

The left-hand side pair $P = (A, d)$ carries the *interface* $d = \mathsf{in}(\gamma)$ of γ with the rest of the grammar, such that d percolates the root *top* feature, and the foot *bot* feature for auxiliary trees. Formally, for each initial tree α in I and auxiliary tree β in A, using a fresh variable t, we define

$$\mathsf{in}(\alpha) = \begin{bmatrix} top : t \\ top : \mathsf{top}(\alpha_r) \end{bmatrix} \tag{2}$$

$$\mathsf{in}(\beta) = \begin{bmatrix} top : t \\ top : \mathsf{top}(\beta_r) \\ bot : \mathsf{bot}(\beta_f) \end{bmatrix} \tag{3}$$

The interface thus uses the top features of the root node of an elementary tree, and we have to implement the fact that this top structure is the same as the top structure of the variable that embodies the root node in the rule right-hand side. With the same variable t, we define accordingly:

$$\mathsf{feat}(\gamma_i) = \begin{cases} \begin{bmatrix} top : t \\ bot : \mathsf{bot}(\gamma_r) \end{bmatrix} & \text{if } \gamma_i = \gamma_r \\ \begin{bmatrix} top : \mathsf{top}(\gamma_i) \\ bot : \mathsf{bot}(\gamma_i) \end{bmatrix} & \text{otherwise} \end{cases} \tag{4}$$

Finally, we add ε-rules $(X_A, \begin{bmatrix} top : v \\ bot : v \end{bmatrix}) \to \varepsilon_A$ for each symbol X_A in order to account for adjunction sites where no adjunction takes place. Let us denote by $\mathsf{tr}(\gamma_i)$ the pair $(\mathsf{nt}(\gamma_i), \mathsf{feats}(\gamma_i))$.

Definition 4. The feature-based RTG $G = \langle S_S, N_S \cup N_A, \mathcal{F}, \mathcal{D}, R\rangle$ of a TAG $\langle \Sigma, N, I, A, S\rangle$ with feature structures in $\mathcal{D}$ has terminal alphabet $\mathcal{F} = I \cup A \cup \{\varepsilon_A\}$ with respective ranks $\mathsf{rk}(\alpha)$, $\mathsf{rk}(\beta)$, and 0, and set of rules

$$\begin{aligned} R = \{&(X_S, \mathsf{in}(\alpha)) \to \alpha(\mathsf{tr}(\alpha_1), \ldots, \mathsf{tr}(\alpha_n)) \\ &\quad \mid \alpha \in I, n = \mathsf{rk}(\alpha), X = \mathsf{lab}(\alpha_r)\} \\ \cup\{&(X_A, \mathsf{in}(\beta)) \to \beta(\mathsf{tr}(\beta_1), \ldots, \mathsf{tr}(\beta_n)) \\ &\quad \mid \beta \in A, n = \mathsf{rk}(\beta), X = \mathsf{lab}(\beta_r)\} \\ \cup\{&X_A \begin{bmatrix} top : t \\ bot : t \end{bmatrix} \to \varepsilon_A \mid X_A \in N_A\} \end{aligned}$$

□

Example 5. With the grammar of Figure 2, we obtain the following ruleset:

$$\begin{gathered} S_S \top \to \text{caught} \\ \left(NP_S \begin{bmatrix} top : \begin{bmatrix} agr : x \end{bmatrix} \end{bmatrix}, VP_A \begin{bmatrix} top : \begin{bmatrix} agr : x \\ mode : \text{ind} \end{bmatrix} \\ bot : \begin{bmatrix} mode : \text{ppart} \end{bmatrix} \end{bmatrix}, NP_S \top\right) \\ NP_S \begin{bmatrix} top : t \end{bmatrix} \to \text{cats}\left(NP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : 3pl \end{bmatrix} \end{bmatrix}\right) \\ NP_S \begin{bmatrix} top : t \end{bmatrix} \to \text{fish}(NP_A \begin{bmatrix} top : t \end{bmatrix}) \\ NP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : x \\ const : - \end{bmatrix} \end{bmatrix} \to \text{the}\left(NP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : x \\ const : + \\ def : + \end{bmatrix} \end{bmatrix}\right) \\ NP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : \text{3sg} \\ const : - \end{bmatrix} \end{bmatrix} \to \text{a}\left(NP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : \text{3sg} \\ const : + \\ def : - \end{bmatrix} \end{bmatrix}\right) \\ NP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : \text{3pl} \\ def : + \end{bmatrix} \end{bmatrix} \to \text{one of}\left(NP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : \text{3sg} \\ const : + \end{bmatrix} \end{bmatrix}\right) \\ NP_A \begin{bmatrix} top : v \\ bot : v \end{bmatrix} \to \varepsilon_A \\ VP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} mode : \text{ppart} \end{bmatrix} \end{bmatrix} \to \text{has}\left(VP_A \begin{bmatrix} top : t \\ bot : \begin{bmatrix} agr : \text{3sg} \\ mode : \text{ind} \end{bmatrix} \end{bmatrix}\right) \\ VP_A \begin{bmatrix} top : v \\ bot : v \end{bmatrix} \to \varepsilon_A \end{gathered}$$

□

With the grammar of Example 5, one can generate the derivation tree for "One of the cats has caught a fish." This derivation is presented in Figure 4. Each node of the tree consists of a label and of a pair (t, e) where t is a term from $T(\mathcal{F}, \mathcal{N} \times \mathcal{D})$ and e is an environment.[5] In order to obtain fresh variables, we rename variables from the RTG: we reuse the name of the variable in the grammar, prefixed by the Gorn address of the node where the rewrite step takes place. Labels indicate the chronological order of the narrowings in the derivation.

Labels in Figure 4 suggest that this derivation has been computed with a left to right strategy. Of course, other strategies would have led to the same

[5] Actually, we only write the change in the environment at each point of the derivation.

result. The important thing to notice here is that the crux of the derivation lies in the fifth rewrite step, where the agreement between the subject and the verb is realized. Substitutions sites are completely defined when all adjunctions in the subtree have been performed. In the next section we propose a different translation that overcomes this drawback.

4 Left Corner Transformation

Derivations in the previous feature-based RTG are not very predictive: the substitution of "cats" into "caught" in the derivation of Figure 1b does not constrain the agreement feature of "caught". This feature is only set at the final ε-rewrite step after the adjunction of "one of", when the top and bottom features are unified. More generally, given a substitution site, we cannot *a priori* rule out the substitution of most initial trees, because their root does usually not carry a top feature.

A solution to this issue is to compute the derivations in a transformed grammar, where we start with the ε-rewrite, apply the root adjunctions in reverse order, and end with the initial tree substitution. Since our encoding sets the root adjunct as the leftmost child, this amounts to a selective left corner transformation (Rosenkrantz and Lewis II, 1970) of our RTG—an arguably simpler intuition than what we could write for the corresponding transformation on derived trees.

4.1 Transformed Regular Tree Grammars

The transformation involves regular tree grammar rules of form $X_S \to \alpha(X_A, ...)$ for substitutions, and $X_A \to \beta(X_A, ...)$ and $X_A \to \varepsilon_A$ for root adjunctions. After a reversal of the recursion of root adjunctions, we will first apply the ε rewrite using a rule $X_S \to \varepsilon_S(X)$ with rank 1 for ε_S, followed by the root adjunctions $X \to \beta(X, ...)$, and finally the substitution itself $X \to \alpha(...)$, with a decremented rank for initial trees.

Example 6. On the grammar of Figure 2, we obtain the rules:

$$
\begin{aligned}
S_S &\to \text{caught}(NP_S, VP_A, NP_S)\\
NP_S &\to \varepsilon_S(NP)\\
NP &\to \text{cats}\\
NP &\to \text{fish}\\
NP &\to \text{the}(NP)\\
NP &\to \text{one of}(NP)\\
VP_A &\to \text{has}(\text{VP}_A)\\
VP_A &\to \varepsilon_A
\end{aligned}
$$

□

Adjunctions that do not occur on the root of an initial tree, like the adjunction of "has" in our example, keep their original translation using $X_A \to \beta(X_A, ...)$ and $X_A \to \varepsilon_A$ rules. We use the nonterminal symbols X of the grammar for root adjunctions and initial trees, and we retain X_S for the initial ε_S rewrite on substitution nodes.

Definition 7. The *left-corner transformed* RTG $G_{\text{lc}} = \langle S_S, N \cup N_S \cup N_A, \mathcal{F}_{\text{lc}}, R_{\text{lc}} \rangle$ of a TAG $\langle \Sigma, N, I, A, S \rangle$ has terminal alphabet $\mathcal{F}_{\text{lc}} = I \cup A \cup \{\varepsilon_A, \varepsilon_S\}$ with respective ranks $\mathsf{rk}(\alpha) - 1$, $\mathsf{rk}(\beta)$, 0, and 1, and set of rules

$$
\begin{aligned}
R_{\text{lc}} = \;& \{X_S \to \varepsilon_S(X) \mid X_S \in N_S\}\\
\cup\;& \{X \to \alpha(\mathsf{nt}(\alpha_2), \ldots, \mathsf{nt}(\alpha_n))\\
&\quad \mid \alpha \in I, n = \mathsf{rk}(\alpha), X = \mathsf{lab}(\alpha_r)\}\\
\cup\;& \{X \to \beta(X, \mathsf{nt}(\beta_2) \ldots, \mathsf{nt}(\beta_n))\\
&\quad \mid \beta \in A, n = \mathsf{rk}(\beta), X = \mathsf{lab}(\beta_r)\}\\
\cup\;& \{X_A \to \beta(\mathsf{nt}(\beta_1), \ldots, \mathsf{nt}(\beta_n))\\
&\quad \mid \beta \in A, n = \mathsf{rk}(\beta), X = \mathsf{lab}(\beta_r)\}\\
\cup\;& \{X_A \to \varepsilon_A \mid X_A \in N_A\}
\end{aligned}
$$

□

Due to the duplicated rules for auxiliary trees, the size of the left-corner transformed RTG of a TAG is doubled at worst. In practice, the reduced grammar witnesses a reasonable growth (10% on the French TAG grammar of Gardent (2006)).

The transformation is easily reversed. We define accordingly the function lc^{-1} from $T(\mathcal{F}_{\text{lc}})$ to $T(\mathcal{F})$:

$$
\begin{aligned}
\mathsf{lc}^{-1}(\varepsilon_S(t)) &= \mathsf{s}(t, \varepsilon_A)\\
\mathsf{s}(\beta(t_1, t_2, ..., t_n), t) &\\
&= \mathsf{s}(t_1, \beta(t, f_{\beta_2}(t_2), ..., f_{\beta_n}(t_n)))\\
\mathsf{s}(\alpha(t_1, ..., t_n), t) &= \alpha(t, f_{\alpha_2}(t_1), ..., f_{\alpha_{n+1}}(t_n))\\
\mathsf{a}(\gamma(t_1, ..., t_n)) &= \gamma(f_{\gamma_1}(t_1), ..., f_{\gamma_n}(t_n))\\
f_{\gamma_i}(t) &= \begin{cases} \mathsf{a}(t) & \gamma_i \text{ adjunction site}\\ \mathsf{lc}^{-1}(t) & \gamma_i \text{ substitution site}\end{cases}
\end{aligned}
$$

We can therefore generate a derivation tree in $L(G_{\text{lc}})$ and recover the derivation tree in $L(G)$ through lc^{-1}.

4.2 Features in the Transformed Grammar

Example 8. Applying the same transformation on the feature-based regular tree grammar, we obtain the following rules for the grammar of Figure 2:

$$
S_S\top \to \text{caught}
$$
$$
\left(NP_S \left[top : [\, agr : x \,] \right], VP_A \begin{bmatrix} top : \begin{bmatrix} agr : x \\ mode : \text{ind} \end{bmatrix} \\ bot : [\, mode : \text{ppart} \,] \end{bmatrix}, NP_S\top\right)
$$

$$
\begin{aligned}
NP_S \left[\textit{top} : t \right] &\rightarrow \varepsilon_S \left(NP \begin{bmatrix} \textit{top} : t \\ \textit{bot} : t \end{bmatrix} \right) \\
NP \left[\textit{bot} : \left[\textit{agr} : \text{3pl} \right] \right] &\rightarrow \text{cats} \\
NP \top &\rightarrow \text{fish} \\
NP \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \begin{bmatrix} \textit{agr} : x \\ \textit{const} : + \\ \textit{def} : + \end{bmatrix} \end{bmatrix} &\rightarrow \text{the} \left(NP \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \begin{bmatrix} \textit{agr} : x \\ \textit{const} : - \end{bmatrix} \end{bmatrix} \right) \\
NP \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \begin{bmatrix} \textit{agr} : \text{3sg} \\ \textit{const} : + \\ \textit{def} : - \end{bmatrix} \end{bmatrix} &\rightarrow \text{a} \left(NP \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \begin{bmatrix} \textit{agr} : \text{3sg} \\ \textit{const} : - \end{bmatrix} \end{bmatrix} \right) \\
NP \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \begin{bmatrix} \textit{agr} : \text{3sg} \\ \textit{const} : + \end{bmatrix} \end{bmatrix} &\rightarrow \text{one of} \left(NP \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \begin{bmatrix} \textit{agr} : \text{3pl} \\ \textit{def} : + \end{bmatrix} \end{bmatrix} \right) \\
VP_A \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \left[\textit{mode} : \text{ppart} \right] \end{bmatrix} &\rightarrow \text{has} \left(VP_A \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \begin{bmatrix} \textit{agr} : \text{3sg} \\ \textit{mode} : \text{ind} \end{bmatrix} \end{bmatrix} \right) \\
VP_A \begin{bmatrix} \textit{top} : v \\ \textit{bot} : v \end{bmatrix} &\rightarrow \varepsilon_A
\end{aligned}
$$

□

Since we reversed the recursion of root adjunctions, the feature structures on the left-hand side and on the root node of the right-hand side of auxiliary rules are swapped in their transformed counterparts (e.g. in the rule for "one of").

This version of a RTG for our example grammar is arguably much easier to read than the one described in Example 5: a derivation has to go through "one of" and "the" before adding "cats" as subject of "caught".

The formal translation of a TAG into a transformed feature-based RTG requires the following variant tr_{lc} of the tr function: for any auxiliary tree β in A and any node γ_i of an elementary tree γ in $I \cup A$, and with t a fresh variable of $\mathcal{D}$:

$$
\mathsf{in}_{\text{lc}}(\beta) = \begin{bmatrix} \textit{top} : t \\ \textit{bot} : \mathsf{bot}(\beta_f) \end{bmatrix} \tag{5}
$$

$$
\mathsf{feats}_{\text{lc}}(\gamma_i) = \begin{cases} \begin{bmatrix} \textit{top} : t \\ \textit{top} : \mathsf{top}(\gamma_r) \\ \textit{bot} : \mathsf{bot}(\gamma_r) \end{bmatrix} & \text{if } \gamma_i = \gamma_r \\ \mathsf{feats}(\gamma_i) & \text{otherwise} \end{cases} \tag{6}
$$

$$
\mathsf{tr}_{\text{lc}}(\gamma_i) = (\mathsf{nt}(\gamma_i), \mathsf{feats}_{\text{lc}}(\gamma_i)) \tag{7}
$$

Definition 9. The *left-corner transformed* feature-based RTG $G_{\text{lc}} = \langle S_S, N \cup N_S \cup N_A, \mathcal{F}_{\text{lc}}, \mathcal{D}, R_{\text{lc}} \rangle$ of a TAG $\langle \Sigma, N, I, A, S \rangle$ with feature structures in $\mathcal{D}$ has terminal alphabet $\mathcal{F}_{\text{lc}} = I \cup A \cup \{\varepsilon_A, \varepsilon_S\}$ with respective ranks $\mathsf{rk}(\alpha) - 1$, $\mathsf{rk}(\beta)$, 0, and 1, and set of rules

$$
\begin{aligned}
R_{\text{lc}} = \{ & X_S \left[\textit{top} : t \right] \rightarrow \varepsilon_S (X \begin{bmatrix} \textit{top} : t \\ \textit{bot} : t \end{bmatrix}) \mid X_S \in N_S \} \\
\cup \{ & (X, \mathsf{feats}(\alpha_1)) \rightarrow \\
& \qquad \alpha(\mathsf{tr}_{\text{lc}}(\alpha_2), \ldots, \mathsf{tr}_{\text{lc}}(\alpha_n)) \\
& \qquad \mid \alpha \in I, n = \mathsf{rk}(\alpha), X = \mathsf{lab}(\alpha_r) \} \\
\cup \{ & (X, \mathsf{feats}_{\text{lc}}(\beta_1)) \rightarrow \\
& \qquad \beta((X, \mathsf{in}_{\text{lc}}(\beta)), \mathsf{tr}_{\text{lc}}(\beta_2), \ldots, \mathsf{tr}_{\text{lc}}(\beta_n)) \\
& \qquad \mid \beta \in A, n = \mathsf{rk}(\beta), X = \mathsf{lab}(\beta_r) \} \\
\cup \{ & (X_A, \mathsf{in}(\beta)) \rightarrow \\
& \qquad \beta(\mathsf{tr}(\beta_1), \mathsf{tr}_{\text{lc}}(\beta_2), \ldots, \mathsf{tr}_{\text{lc}}(\beta_n)) \\
& \qquad \mid \beta \in A, n = \mathsf{rk}(\beta), X = \mathsf{lab}(\beta_r) \} \\
\cup \{ & X_A \begin{bmatrix} \textit{top} : t \\ \textit{bot} : t \end{bmatrix} \rightarrow \varepsilon_A \mid X_A \in N_A \}
\end{aligned}
$$

□

Again, the translation can be computed in linear time, and results in a grammar with at worst twice the size of the original TAG.

5 Conclusion

We have introduced in this paper feature-based regular tree grammars as an adequate representation for the derivation language of large coverage TAG grammars. Unlike the restricted unification computations on the derivation tree considered before by Kallmeyer and Romero (2004), feature-based RTGs accurately translate the full range of unification mechanisms employed in TAGs. Moreover, left-corner transformed grammars make derivations more predictable, thus avoiding some backtracking in top-down generation.

Among the potential applications of our results, let us further mention more accurate reachability computations between elementary trees, needed for instance in order to check whether a TAG complies with the tree insertion grammar (Schabes and Waters, 1995, TIG) or regular form (Rogers, 1994, RFTAG) conditions. In fact, among the formal checks one might wish to perform on grammars, many rely on the availability of reachability relations.

Let us finally note that we could consider the string language of a TAG encoded as a feature-based RTG—in a parser for instance—, if we extended the model with topological information, in the line of Kuhlmann (2007).

References

Candito, Marie-Hélène and Sylvain Kahane. 1998. Can the TAG derivation tree represent a semantic graph? An answer in the light of Meaning-Text Theory. In *TAG+4*, pages 25–28.

Comon, Hubert, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 2007. *Tree Automata Techniques and Applications*.

de Groote, Philippe. 2002. Tree-adjoining grammars as abstract categorial grammars. In Frank, Robert, editor, *TAG+6*, pages 145–150.

Engelfriet, Joost and Heiko Vogler. 1985. Macro tree transducers. *Journal of Computer and System Sciences*, 31(1):71–146.

Gardent, Claire and Laura Kallmeyer. 2003. Semantic construction in feature-based TAG. In *EACL'03*, pages 123–130. ACL Press.

Gardent, Claire. 2006. Intégration d'une dimension sémantique dans les grammaires d'arbres adjoints. In Mertens, Piet, Cédrick Fairon, Anne Dister, and Patrick Watrin, editors, *TALN'06*, pages 149–158. Presses universitaires de Louvain.

Hanus, Michael. 1994. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19–20:583–628.

Johnson, Mark. 1988. *Attribute-Value Logic and the Theory of Grammar*, volume 16 of *CSLI Lecture Notes Series*. University of Chicago Press.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In Rozenberg, Grzegorz and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3: Beyond Words, chapter 2, pages 69–124. Springer.

Kallmeyer, Laura and Maribel Romero. 2004. LTAG semantics with semantic unification. In Rambow, Owen and Matthew Stone, editors, *TAG+7*, pages 155–162.

Kanazawa, Makoto. 2007. Parsing and generation as Datalog queries. In *ACL'07*, pages 176–183. ACL Press.

Koller, Alexander and Matthew Stone. 2007. Sentence generation as a planning problem. In *ACL'07*, pages 336–343. ACL Press.

Koller, Alexander and Kristina Striegnitz. 2002. Generation as dependency parsing. In *ACL'02*, pages 17–24. ACL Press.

Kuhlmann, Marco. 2007. *Dependency Structures and Lexicalized Grammars*. Doctoral dissertation, Saarland University, Saarbrücken, Germany.

Pogodalla, Sylvain. 2004. Computing semantic representation: Towards ACG abstract terms as derivation trees. In Rambow, Owen and Matthew Stone, editors, *TAG+7*, pages 64–71.

Robinson, J. Alan. 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41.

Rogers, James. 1994. Capturing CFLs with tree adjoining grammars. In *ACL'94*, pages 155–162. ACL Press.

Rosenkrantz, Daniel J. and Philip M. Lewis II. 1970. Deterministic left corner parsing. In *11th Annual Symposium on Switching and Automata Theory*, pages 139–152. IEEE Computer Society.

Schabes, Yves and Richard C. Waters. 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4):479–513.

Shieber, Stuart M. 2006. Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *EACL'06*. ACL Press.

Vijay-Shanker, K. 1992. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517.

XTAG Research Group. 2001. A lexicalized tree adjoining grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

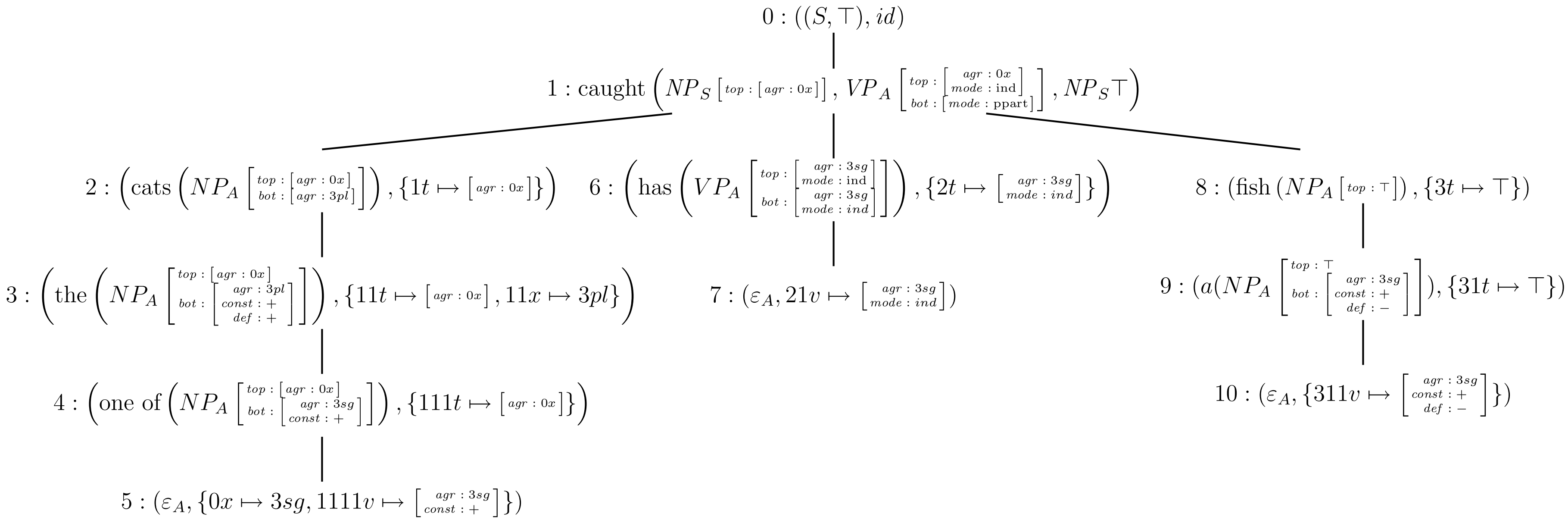


Figure 4: A rewrite sequence in the feature-based RTG for the sentence "One of the cats has caught a fish."