Bericht Nr. 81, Dezember 1996

Arbeitspapiere des
Sonderforschungsbereichs 340

ISSN 0947-6954/96

Sprachtheoretische Grundlagen für die Computerlinguistik

Laura Kallmeyer:

Underspecification in Tree Description Grammars

Universität Stuttgart Universität Tübingen
IBM Deutschland Informationssysteme GmbH

Arbeitspapiere des SFB 340
Bericht Nr. 81, Dezember 1996

UNDERSPECIFICATION IN TREE DESCRIPTION GRAMMARS

LAURA KALLMEYER

Abstract. In this article Tree Description Grammars (TDG) are presented, a constraint based extension of Tree Adjoining Grammars (TAG). The idea to define TAGs or similar tree grammars in a constraint based way comes from Vijay-Shanker (1992).
One of the advantages of using tree descriptions instead of trees in a grammar is the possiblility of underspecification. In a TDG for certain ambiguities there can be underspecified descriptions in the style of the underspecified representations Muskens derives in Muskens (1995) for quantifier scope ambiguities.
In principle the idea of TDGs is the following: The grammar consists of a set of tree descriptions, i.e. formulas in some first order logic, and the grammar is derivation based. Every derivation step is a conjunction of two formulas and additional equivalences of node names.
The formalism of Tree Description Grammars was introduced in Kallmeyer (1996). Here the definitions of TDGs and related notions (e.g. the derivation mode, the language etc.) are repeated. Some definitions will be slightly different from those in Kallmeyer (1996). The main focus will be on the formal properties of TDGs. I will show the decidability of lexicalized TDGs and the fact that the languages of TDGs are a true superset of tree adjoining languages.

1. Introduction

A lot of attention has been paid to underspecification recently, mainly to underspecified semantic representations. There are two reasons for replacing several analyses of a single expression by one underspecified analysis. The more practical reason is the desire to avoid combinatorial explosion. This so-called "combinatorial explosion puzzle" really can be a problem. To get an idea of the number of readings available for a single

To appear in: Kolb, H.-P. & U. Mönnich (eds.), *The Mathematics of Syntactic Structure.*

ISSN 0947-6954/96
©1996 Die Autorin

expression consider the following two examples from Poesio (1996):

(1) *We should move the engine at Avon, engine E1, to Dansville to pick up the boxcar there, then move it from Dansville to Corning, load some oranges, and then move it on to Bath.*

(2) *A politician can fool most voters on most issues most of the time, but no politician can fool all voters on every single issue all of the time.*

Poesio points out that in (1) the alternative analyses probably number in the hundreds and that (2) has at least 14400 readings.
The second reason for exploring underspecification is psychological plausibility. From a cognitive point of view it does not seem plausible that we process sententences by first generating all readings and then testing them. It appears more cognitively adequate to allow for underspecified representations.
To give an idea of what underspecified tree descriptions look like, Fig. 1 shows a descriptions for the two syntactic analyses for minimal modifier ambiguities as in *ancienne maison bleue* in French. (All examples in this paper are French examples.)

Besides underspecification there are some other advantages of using grammars consisting of tree descriptions instead of the more traditional Tree Adjoining Grammars.
As pointed out by Rambow, Vijay-Shanker and Weir 1995, one of the defficencies of TAGs is the way complement clauses are treated. Only for technical reasons they must be analized differently than other arguments. Rambow, Vijay-Shanker and Weir propose D-Tree Grammars where the *substitution* operation of TAGs is replaced by *subsertion*. In D-Tree Grammars all arguments including complement clauses are added by subsertion, i.e. they are treated in a uniform way. In TDGs subsertion-like derivation steps are possible as well, so arguments also can be treated uniformly.

Other problems with TAGs arise for phenomena where it is necessary to add several trees in one derivation step. Becker, Joshi and Rambow 1991 show that this is the case for scrambling in German. To deal with this multicomponent TAGs (MCTAG, cf. Joshi (1987)) have been proposed. Shieber and Schabes 1990 use MCTAGs to treat quantifiers. As it is possible to describe several subtrees of one tree in one description without specifying linear precedence between these subtrees, TDGs also

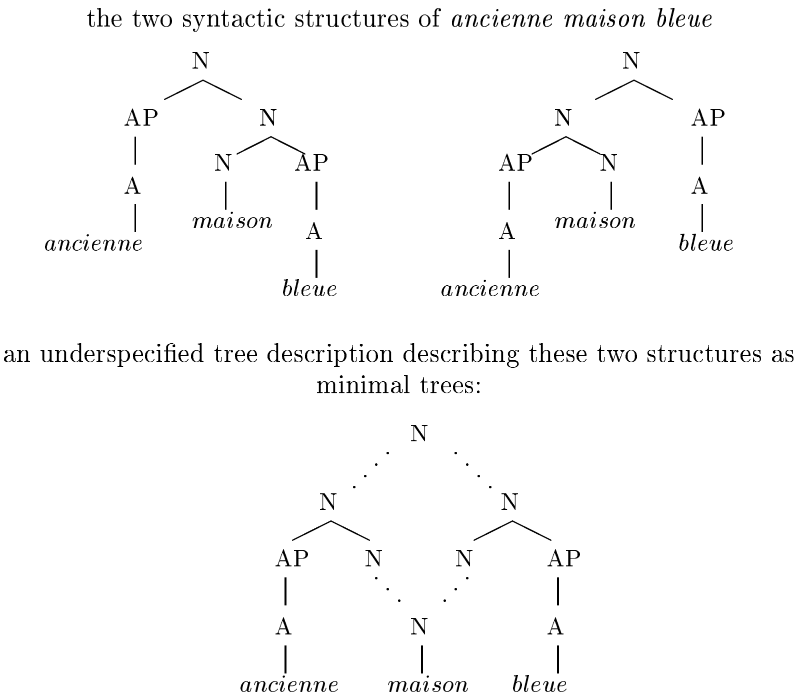


FIGURE 1. syntactic underspecification: *ancienne maison bleue*

offer the advantages of MCTAGs.

In this article I will first present the tree logic used in TDGs, a simple quantifier-free first order logic mainly based on Rogers (1994b). Using this logic the TDG formalism will be introduced and some formal properties of TDGs will be proved. In the end I will give some examples to show that underspecification and subsertion-like derivation steps are possible in TDGs.

2. Tree descriptions

2.1. **Logic.** A TDG consists of tree descriptions, i.e. constraints for finite labelled trees. In these trees internal nodes are labelled by finite feature structures and leaves by terminal symbols. The logic of tree descriptions used for TDGs is a quantifier-free first order logic. It is similar to the logic Rogers proposes in chapter 3 of his dissertation Rogers (1994b) for feature structure based TAGs (FTAGs, see Vijayashanker (1987), Vijay-Shanker (1992) and Vijay-Shanker and Joshi (1988)).

Definition 1. *(Syntax)*
Let A be a set of attribute names, V a set of attribute values and T a set of terminals such that A, V and T are pairwise disjoint finite sets. Formulas (for A, V and T) are then words over the alphabet of the following symbols:
- *the symbols of $V \cup A \cup T \cup \{\epsilon, \perp\}$*
- *a nonempty countable set K of variables called node names,*
- *constants $\lhd, \lhd^*, \prec$, $\approx$ (for binary relations) and δ (for the labeling function),*
- *logical connectives $\wedge, \vee, \neg$*
- *bracket symbols (,).*

Formulas are inductively defined:
1. *Terms:*
   (a) *Each $x \in T \cup \{\epsilon\}$ and each $\delta(k)$ for all $k \in K$ is a l-term (denoting labels).*
   (b) *Each $x \in V \cup \{\perp\}$ and each $a(\delta(k))$ for all $a \in A$ and $k \in K$ is a v-term (denoting attribute values).*
   (c) *All $k \in K$ are n-terms (denoting nodes).*
2. *Formulas:*
   (a) *For all τ_1, τ_2, that are either both l-terms or both v-terms or both n-terms, $\tau_1 \approx \tau_2$ is a formula.*
   (b) *For alle n-terms k_1, k_2: $k_1 \lhd k_2$, $k_1 \lhd^* k_2$ and $k_1 \prec k_2$ are formulas.*
   (c) *If ϕ is a formula, then $\neg\phi$ also is a formula.*
   (d) *If ϕ, ψ are formulas, then $(\phi \wedge \psi)$ and $(\phi \vee \psi)$ are also formulas.*

   Formulas containing at most one negation and no other logical connective are called elementary formulas.

   $node(\phi)$ is the set of all node names occuring in a formula ϕ.

If the order of logical connectives does not matter or if there is only one possible order, brackets will be sometimes omitted.

2.2. **Intended models.** The set of intended models is restricted to finite labelled trees. Such a tree consists of a set of nodes structured by binary relations *parent*, *dominance* and *linear precedence.* Each leaf is labelled by a terminal symbol or the empty word and each internal node by a finite atomic feature structure.

Definition 2. *(Finite labelled trees)*
A finite labelled tree is a tuple $(\mathcal{U}, u_0, \mathcal{P}, \mathcal{P}^*, \mathcal{L}, \mu, N, T)$ such that

1. *N, T are disjoint finite nonempty sets, the nonterminals and terminals.*
2. *$\mathcal{U}$ is a finite nonempty set, the set of nodes, with $\mathcal{U} \cap (N \cup T) = \emptyset$.*
3. *$u_0 \in \mathcal{U}$ is a special node, the root.*
4. *$\mathcal{P}, \mathcal{L}, \mathcal{P}^*, \in \mathcal{U} \times \mathcal{U}$, such that for all $w, x, y, z \in \mathcal{U}$:*
   - *(a) $(u_0, x) \in \mathcal{P}^*$*
   - *(b) $\mathcal{P}^*$ is the reflexive transitive closure of $\mathcal{P}$.*
   - *(c) $(x, y) \in \mathcal{P}^*$ or $(y, x) \in \mathcal{P}^*$ or $(x, y) \in \mathcal{L}$ or $(y, x) \in \mathcal{L}$*
   - *(d) $(x, y) \in \mathcal{L} \Rightarrow (x, y) \notin \mathcal{P}^*$ and $(y, x) \notin \mathcal{P}^*$*
   - *(e) $x \neq u_0 \Rightarrow$ there is exactly one $v \in \mathcal{U}$ mit $(v, x) \in \mathcal{P}$*
   - *(f) $(x, y) \in \mathcal{L}$ and $(x, w), (y, z) \in \mathcal{P}^* \Rightarrow (w, z) \in \mathcal{L}$*
   - *(g) $\mathcal{P}$ irreflexive, asymmetric and intransitive*
   - *(h) $\mathcal{L}$ irreflexive, asymmetric and transitive*
5. *$\mu : \mathcal{U} \to T \cup N \cup \{\epsilon\}$ is a function, such that for all $u \in \mathcal{U}$: $\mu(u) \in T \cup \{\epsilon\}$ iff there is no $u' \in \mathcal{U}$ with $(u, u') \in \mathcal{P}$ (u is a leaf).*

Definition 3. *(Finite atomic feature structures)*
A set of finite atomic feature structures $N_{\mathcal{A},V}$ consists of sets N, $\mathcal{A}$ (attriutes) and V (attribute values), such that
1. *N and V are finite sets, $N \cap V = \emptyset$.*
2. *$\mathcal{A}$ is a finite set of pairs $\langle a, \alpha_a \rangle$, where α_a is a partial function from N to V and a is the unique name of α_a.*

2.3. **Semantics.** The notion of satisfiability in this logic is the same as in ordinary predicate logic. Satisfiability of a formula in a certain model is considered with respect to a node assignment function assigning nodes to all node names.

Definition 4. *(Satisfiability)*
Let $B = (\mathcal{U}, u_0, \mathcal{P}, \mathcal{P}^*, \mathcal{L}, \mu, N_{A,V}, T)$ be a finite tree labelled with finite atomic feature structures $N_{A,V}$ as nonterminals and terminals T.
$g : K \to \mathcal{U}$ is a function, the node assignment function.
ϕ, ϕ_1 and ϕ_2 are formulas with attribute names $\{a$; there is a partial function α_a such that $\langle a, \alpha_a \rangle \in A\}$, attribute values V and terminals T.
B satisfies ϕ wrt g, $B \models_g \phi$ is inductively defined:
1. *For all $k_1, k_2 \in K$:*
   - $B \models_g k_1 \lhd k_2$ *iff* $(g(k_1), g(k_2)) \in \mathcal{P}$
   - $B \models_g k_1 \lhd^* k_2$ *iff* $(g(k_1), g(k_2)) \in \mathcal{P}^*$
   - $B \models_g k_1 \prec k_2$ *iff* $(g(k_1), g(k_2)) \in \mathcal{L}$
   - $B \models_g k_1 \approx k_2$ *iff* $g(k_1) = g(k_2)$

2. *For all l-terms* l_1, l_2*:*
   $B \models_g l_1 \approx l_2$ *iff* $[[l_1]]_g = [[l_2]]_g$ *where*
   - $[[\delta(k)]]_g = \mu(g(k))$ *for all* $k \in K$,
   - $[[x]]_g = x$ *for all* $x \in T \cup \{\epsilon\}$.
3. *For all v-terms* v_1, v_2*:*
   $B \models_g v_1 \approx v_2$ *iff* $[[v_1]]_g = [[v_2]]_g$ *where*
   - *for all* $k \in K$*:* $[[a(\delta(k))]]_g = \alpha_a(\mu(g(k)))$, *if* $\langle a, \alpha_a \rangle \in A$ *and if* $\alpha_a(\mu(g(k)))$ *is defined, else* $[[a(\delta(k))]]_g = \perp$.
   - $[[v]]_g = v$ *for all* $v \in V \cup \{\perp\}$.
4. $B \models_g (\phi_1 \wedge \phi_2)$ *iff* $B \models_g \phi_1$ *and* $B \models_g \phi_2$.
5. $B \models_g (\phi_1 \vee \phi_2)$ *iff* $B \models_g \phi_1$ *or* $B \models_g \phi_2$.
6. $B \models_g \neg\phi_1$ *iff not* $B \models_g \phi_1$.

$B \models \phi$ *iff there is an assignment function* g*, such that* $B \models_g \phi$.
$\phi_1 \models \phi_2$ *iff for all trees* B *and all assignments* g*:* $B \models_g \phi_1 \Rightarrow B \models_I \phi_2$.

3. Properties of the tree logic

In this section I will define some syntactic equivalent for the semantic consequence relation "$\models$". This can be done by using inference rules as proposed by Rogers. The idea is that starting from a formula ψ by applying inference rules it is possible to construct a conjunctive normal form of all the consequences of ψ. A syntactic notion of consequence can then be defined by comparing two such conjunctive normal forms of consequences (see Kallmeyer (1996)).
The syntactic consequence defined in this way is decidable, sound and even complete as will be shown below.

3.1. Syntactic consequence.

Instead of conjunctions of disjunctions of formulas we will deal with sets of sets of formulas and interprete the whole set (branch) conjunctively and the elements (clauses) disjunctively. This is only a different notation which is used in order to see more clearly how inference rules work.

Definition 5. *(clause and branch)*

1. *A clause is a set* Φ *of formulas .* $B \models_g \Phi$ *iff there is a* $\phi \in \Phi$ *such that* $B \models_g \phi$*. Notation:* $[\Phi]$
2. *A branch is a set* Γ *of clauses.* $B \models_g \Gamma$ *iff* $B \models_g \Phi$ *for all* $\Phi \in \Gamma$*. Notation:* $||\Gamma||$.

This definition does not exclude infinite disjunctions or conjunctions. But for our purposes we only need finite clauses and branches.

Before introducing the inference rules a general definition of the syntax and semantics of inference rules must be given.

Definition 6. *(inference rule)*

1. *Syntax: Let* $V_\Gamma := \{\Gamma_1, \Gamma_2, \cdots\}$, $V_\Phi := \{\Phi_1, \Phi_2, \cdots\}$ *and* $V_\phi := \{\phi_1, \phi_2, \cdots\}$ *be pairwise disjoint countable sets.*
   (i) A branch schema can be inductively defined:
      (a) All $\psi_1 \in V_\phi$ *are formula schemata, and for all formula schemata* ψ_1 *and* ψ_2*:* $\neg\psi_1$, $(\psi_1 \wedge \psi_2)$ *and* $(\psi_1 \vee \psi_2)$ *are formula schemata.*
      (b) Each formula schema and each $\Psi \in V_\Phi$ *is a clause schema, and for all clause schemata* Ψ_1, Ψ_2*:* Ψ_1, Ψ_2 *is a clause schema.*
      (c) For each clause schema Ψ, $[\Psi]$ *is a branch schema, each* $\Gamma \in V_\Phi$ *is a branch schema, and for all branch schemata* Γ_1, Γ_2*:* Γ_1, Γ_2 *is a branch schema.*
   (ii) An inference rule $\frac{\Gamma_{old}}{\Gamma_{new}}\langle name\rangle$ *consists of two branch schemata* Γ_{old} *and* Γ_{new} *such that all* $x \in V_\Gamma \cup V_\Phi \cup V_\phi$ *occuring in* Γ_{new} *occur as well in* Γ_{old}. $\langle name\rangle$ *is the name of the rule.*
2. *Semantics: Let* Γ *be a branch.*
   (i) Γ *satisfies a branch schema* Γ_s*, if there is a function* f *such that*
      - $f(\Gamma_i)$ *is a branch for all* $\Gamma_i \in V_\Gamma$, $f(\Psi_i)$ *is a clause for all* $\Psi_i \in V_\Psi$ *and* $f(\psi_i)$ *is a formula for all* $\psi_i \in V_\phi$.
      - $f(\neg a) = \neg f(a), f((a \wedge b)) = (f(a) \wedge f(b))$ *and* $f((a \vee b)) = (f(a) \vee f(b))$ *for all formula schemata* a *and* b.
      - $f([a]) = \{f(a)\}$ *for all clause schemata* a.
      - $f(a, b) = f(a) \cup f(b)$ *for all clause or branch schemata* a, b
      - $f(\Gamma_s) = \Gamma$.
   (ii) A rule $\frac{\Gamma_{old}}{\Gamma_{new}}\langle name\rangle$ *can be applied to* Γ*, if there is a function* f *such that* Γ *satisfies* Γ_{old} *wrt* f*. The result of the application is* $f(\Gamma_{new})$.

The following inference rules in principle correspond to those proposed by Rogers (1994b). The logical and structural rules and *res* are exactly as in Rogers (1994b) and the tree axiom rules are similar to those in Rogers (1994b). Only node symbol rules do not figure there because Rogers does not consider labelled trees.

Definition 7. *(inference rules for the tree logic)*
The inference rules for our tree logic are the following:

1. *Logical rules (elimination of logical connectives)*

$$\frac{\Gamma, [\Phi, \neg\neg\phi]}{\Gamma, [\Phi, \phi]}\neg\neg \qquad \frac{\Gamma, [\Phi, (\phi \vee \psi)]}{\Gamma, [\Phi, \phi, \psi]}\vee \qquad \frac{\Gamma, [\Phi, \neg(\phi \vee \psi)]}{\Gamma, [\Phi, \neg\phi], [\Phi, \neg\psi]}\neg\vee$$

$$\frac{\Gamma, [\Phi, (\phi \wedge \psi)]}{\Gamma, [\Phi, \phi], [\Phi, \psi]}\wedge \qquad \frac{\Gamma, [\Phi, \neg(\phi \wedge \psi)]}{\Gamma, [\Phi, \neg\phi, \neg\psi]}\neg\wedge$$

2. *Structural rules taut (elimination of tautologies) and abs (absorbtion of a clause by a second clause, that implies the first one)*

$$\frac{\Gamma, [\Phi, \phi, \neg\phi]}{\Gamma}taut \qquad \frac{\Gamma, [\Phi], [\Phi, \Psi]}{\Gamma, [\Phi]}abs$$

3. *Resolution res*

$$\frac{\Gamma, [\Phi, \phi], [\Psi, \neg\phi]}{\Gamma, [\Phi, \phi], [\Psi, \neg\phi], [\Phi, \Psi]}res$$

4. *Tree axiom rules: For all $k_1, k_2, k_3, k_4 \in K$:*
   Properties of $\mathcal{P}, \mathcal{P}^*, \mathcal{L}$ and $=$ $(\{(u,u); u \in \mathcal{U}\})$:

$$\frac{\Gamma}{\Gamma, [\neg k_1 \approx k_2, \neg k_2 \approx k_3, k_1 \approx k_3]}tr_= \qquad \frac{\Gamma}{\Gamma, [\neg k_1 \approx k_2, k_2 \approx k_1]}sy_=$$

$$\frac{\Gamma}{\Gamma, [k_1 \approx k_1]}rf_= \qquad \frac{\Gamma}{\Gamma, [\neg k_1 \lhd^* k_2, \neg k_2 \lhd^* k_3, k_1 \lhd^* k_3]}tr_{\mathcal{P}^*}$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \approx k_2, k_1 \lhd^* k_2]}rf_{\mathcal{P}^*} \qquad \frac{\Gamma}{\Gamma, [\neg k_1 \lhd^* k_2, \neg k_2 \lhd^* k_1, k_1 \approx k_2]}antis_{\mathcal{P}^*}$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, \neg k_2 \approx k_1]}irrf_{\mathcal{P}} \qquad \frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, \neg k_2 \lhd k_3, \neg k_1 \lhd k_3]}intr_{\mathcal{P}}$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \prec k_2, \neg k_1 \approx k_2]}irrf_{\mathcal{L}} \qquad \frac{\Gamma}{\Gamma, [\neg k_1 \prec k_2, \neg k_2 \prec k_3, k_1 \prec k_3]}tr_{\mathcal{L}}$$

Relations between $\mathcal{L}$, $\mathcal{P}$ and $\mathcal{P}^*$:

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, k_1 \lhd^* k_2]}T_1 \qquad \frac{\Gamma}{\Gamma, [k_1 \lhd^* k_2, k_2 \lhd^* k_1, k_1 \prec k_2, k_2 \prec k_1]}T_2$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd^* k_2, \neg k_1 \prec k_2], [\neg k_2 \lhd^* k_1, \neg k_1 \prec k_2]} T_3$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, \neg k_3 \lhd k_2, k_1 \approx k_3]} T_4$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \prec k_2, \neg k_1 \lhd^* k_3, \neg k_2 \lhd^* k_4, k_3 \prec k_4]} T_5$$

5. *Node symbol rules: Let $T = \{t_1, \cdots, t_n\}$ be the terminals, $A = \{a_1, \cdots a_m\}$ the attribute names and $V = \{v_1, \cdots, v_l\}$ the attribute values. For all $k, k_1, k_2 \in K$:*

   Nonterminal labels for all internal nodes:

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, \neg \delta(k_1) \approx t_1], \cdots, [\neg k_1 \lhd k_2, \neg \delta(k_1) \approx t_n]} \mu(u)$$

   Uniqueness of μ:

$$\frac{\Gamma}{\Gamma, [\neg k_1 \approx k_2, \delta(k_1) \approx \delta(k_2)]} unique_\mu$$

   Uniqueness of the attribute functions: for all attribute names a:

$$\frac{\Gamma}{\Gamma, [\neg k_1 \approx k_2, a(\delta(k_1)) \approx a(\delta(k_2))]} attr_1$$

   Attribute functions undefined for terminals: for all $t \in T$ and $a \in A$:

$$\frac{\Gamma}{\Gamma, [\neg \delta(k_1) \approx t, a(\delta(k_1)) \approx \bot]} attr_2$$

   Attribute values in V or undefined: for all $a \in A$:

$$\frac{\Gamma}{\Gamma, [a(\delta(k)) \approx v_1, \cdots, a(\delta(k)) \approx v_l, a(\delta(k)) \approx \bot]} attr_3$$

Proposition 1. *(Model invariance) If Γ_2 can be constructed from Γ_1 in finitely many steps by applying inference rules, then:*

$$\bigwedge_{\Phi_1 \in \Gamma_1} \bigvee_{\phi_1 \in \Phi_1} \models \bigwedge_{\Phi_2 \in \Gamma_2} \bigvee_{\phi_2 \in \Phi_2} \text{ and } \bigwedge_{\Phi_1 \in \Gamma_1} \bigvee_{\phi_1 \in \Phi_1} \models \bigwedge_{\Phi_2 \in \Gamma_2} \bigvee_{\phi_2 \in \Phi_2}$$

This can be shown by induction on the number of rule applications.

The following definition is again taken from Rogers (1994b):

Definition 8. *(Saturated, open, closed and satisfiable branches)*

1. *A branch Γ is saturated, if:*
   - *No logical rule can be applied to Γ.*
   - *Neither abs nor taut can be applied to Γ.*
   - *For all applications of res to Γ the new clause introduced by this application can be removed by abs or taut.*
   - *For all applications of tree axiom or node symbol rules with node names occuring in Γ the resulting new clauses can be removed by abs.*
2. *A branch is closed, if it contains an empty clause.*
3. *A branch is open, if it is not closed.*
4. *A branch Γ is satisfiable, if there is a tree B and a node assignment function g such that $B \models_g \Gamma$.*

Proposition 2. *Let ψ be a formula with attribute names A, values V and terminals T and $\Upsilon \subseteq K$ a finite set such that $node(\psi) \subseteq \Upsilon$. In finitely many steps a unique saturated branch Γ can be constructed for ψ and Υ, such that*

$$\psi \models \bigwedge_{\Phi \in \Gamma} \bigvee_{\phi \in \Phi} \text{ and } \bigwedge_{\Phi \in \Gamma} \bigvee_{\phi \in \Phi} \models \psi$$

Notation: $\Gamma(\psi)$ is the saturated branch for ψ (wrt to a given set Υ). If it does not matter which set Υ was used, we simply write "$\Gamma(\psi)$".

Prop. 2 can be proved by first giving an algorithm for constructing $\Gamma(\psi)$ for a given ψ and then showing that the result has the properties mentioned in the proposition. The construction is very similar to the one proposed in Prop. 5 in Rogers (1994b).

Proof: Construction: $\Gamma_1 := \|[\psi]\|$

1. Apply each tree axiom and each node symbol rule with symbols from A, V and T and node names from Υ once to Γ_1. The result is called Γ_2.

2. Apply the logical rules as often as possible to Γ_2. The result is called Γ_3.
3. Apply *res* repeatedly to Γ_3, as long as new clauses can be added by *res*. The result is called Γ_4.
4. Apply the structural rules as often as possible to Γ_4. The result is Γ.

(The order of the first two steps is of no importance because neither tree axiom rules nor node symbol rules add any logical connectives.)

To show: The construction terminates after finitely many steps and the result is a saturated branch Γ, such that $B \models_g \psi$ iff $B \models_g \Gamma$.

(i) Construction terminates after finitely many steps, i.e. each of the four parts 1. to 4. only consists of a finite number of rule applications:
   (a) 1. part: There are only finitely many rules with node names from Υ and symbols from A, V and T that are applied to Γ_1.
   (b) 2. part: There is only a finite number of applications of logical rules to Γ_2 because Γ_2 is finite and each logical rule removes some logical connective.
   (c) 3. part: Each application of *res* in 3. adds a new clause. But the number of possible clauses containing elementary formulas with node names from Υ and symbols from A, V and T is finite:
   There are only finitely many elementary formulas, that can occur as elements of a clause. If n is the number of possible elementary formulas, then there are 2^n possible clauses and consequently at most 2^n applications of *res* in 3.
   (d) 4. part: Each structural rule removes a clause. Consequently only finitely many applications of *taut* and *abs* are possible.

(ii) Γ is saturated:
   (a) No logical rule can be applied to Γ because no logical connectives are added in 2., 3. or 4.
   (b) Because of the 4. part no structural rules can be applied to Γ.
   (c) Because of $\Gamma \subseteq \Gamma_4$ all applications of *res*, that are possible in Γ have already been done in 3. Each additional application of *res* either does not change anything or adds a clause that can be removed by *taut* or *abs*.
   (d) In the beginning all tree axiom rules and all node symbol rules with names from Υ have been applied. The clauses added by these rules are no tautologies and contain only elementary formulas. Because of this they have not been removed by *taut* or by a logical rule. Perhaps they have been diminished by *abs*. Consequently

for each result $\|\Gamma, \Phi_1, \Phi_2 \cdots\|$ of an application of a tree axiom rule or a node symbol rule to $\|\Gamma\|$: For each new clause Φ_i there is a clause Ψ_i in Γ with $\Psi_i \subseteq \Phi_i$, i.e. Φ_i can be eliminated by *abs*.

(iii) Model invariance: Holds because of $\psi = \bigwedge_{\Phi \in \|[\psi]\|} \bigvee_{\phi \in \Phi}$ and Prop. 1. □

Closed branches are not satisfiable. This means that a necessary condition for the satisfiability of a formula ϕ is, that $\Gamma(\phi)$ must be open. We will see that this is even sufficient, i.e. that if $\Gamma(\phi)$ is open, then ϕ is satisfiable.

By comparing the saturated branches $\Gamma(\phi_1)$ and $\Gamma(\phi_2)$ of two formulas ϕ_1 and ϕ_2 a syntactic notion of consequence $\phi_1 \vdash \phi_2$ can be defined:

Definition 9. *(Syntactic consequence $\phi_1 \vdash \phi_2$)*
Let ϕ_1 and ϕ_2 be formulas, Γ_1 and Γ_2 the corresponding saturated branches with node names from $node(\phi_1) \cup node(\phi_2)$.
$\phi_1 \vdash \phi_2$ iff for all $\Psi_2 \in \Gamma_2$ there is a $\Psi_1 \in \Gamma_1$, such that $\Psi_1 \subseteq \Psi_2$.

$\vdash$ is transitive and because of Prop. 2 it is also decidable.

3.2. **Soundness and Completeness.** If $\phi_1 \vdash \phi_2$, then for all trees B and all node assignment functions g: If $B \models_g \phi_1$, then $B \models_g \phi_2$. This immediately follows from Def. 9 together with Prop. 2 and Def. 5. Consequently the following holds:

Proposition 3. *(Soundness)*
For all ϕ_1, ϕ_2: If $\phi_1 \vdash \phi_2$, then $\phi_1 \models \phi_2$.

In order to prove completeness we first need the following proposition:

Proposition 4. *(Satisfiability of open saturated branches)*
For all formulas ψ and all $\Gamma = \Gamma(\psi)$:
Γ is open iff Γ is satisfiable (iff ψ is satisfiable).

Proof: We only have to show, that: Γ open $\Rightarrow \psi$ satisfiable.
Let $\Gamma = \Gamma(\psi)$ be open, Γ is constructed wrt a set Υ of node names.

(i) First we will extend Γ to a branch Γ', such that for all $k_1, k_2 \in \Upsilon$:
$[k_1 \prec k_2] \in \Gamma'$ or $[\neg k_1 \prec k_2] \in \Gamma'$, $[k_1 \lhd k_2] \in \Gamma'$ or $[\neg k_1 \lhd k_2] \in \Gamma'$, $[k_1 \lhd^* k_2] \in \Gamma'$ or $[\neg k_1 \lhd^* k_2] \in \Gamma'$ and $[k_1 \approx k_2] \in \Gamma'$ or $[\neg k_1 \approx k_2] \in \Gamma'$. This means that Γ' must be completely specified wrt $\prec, \lhd, \lhd^*, \approx$ (in general Γ' is not unique).

Construction:
$\Gamma_0 := \Gamma \cup \{[k_1 \text{ R } k_2, \neg k_1 \text{ R } k_2] |\ \text{R} \in \{\prec, \lhd, \lhd^*, \approx\}, k_1, k_2 \in \Upsilon\}$.

$\Gamma' := \Gamma_0$. Apply first *res* (as long as no tautology is introduced) and then *abs* as often as possible to Γ' (only finitely many applications).

(*): If $|[\Phi]| = 1$ for all $[\Phi] \in \Gamma'$, then the construction is terminated. Else: Choose one $[\Phi] \in \Gamma'$, such that $|[\Phi]| > 1$, and then choose one $\phi \in \Phi$. $\Gamma'' := \Gamma' \cup \{[\phi]\}$.
Apply *res* as often as possible to Γ'', as long as *res* does not introduce any tautologies (these applications of *res* add only subsets of clauses that are already in Γ'').
Apply then *abs* as often as possible to Γ'' (this only removes clauses that are supersets of $[\phi]$ or of the clauses just added by *res*). Let Γ' be the resulting branch.
Continue with *(*)*.

The construction terminates after finitely many steps because no new clauses are introduced and in each step at least one $[\Phi]$ is diminished.

No clause is completely removed. Consequently for each tautology added to Γ in the beginning a subset containing one formula is in Γ'.

(ii) To show: Γ' is saturated.

- No logical rule can be applied to Γ.
- *res* and *abs* have been applied as often as possible.
- *taut* cannot be applied to Γ' because $|\Phi| = 1$ for all $\Phi \in \Gamma'$.
- $\Gamma \subseteq \Gamma_0$ and for all $\Psi \in \Gamma_0$ there is a $\Phi \in \Gamma'$, such that $\Phi \subseteq \Psi$.
  $\Rightarrow$ for all $\Psi \in \Gamma$ there is a $\Phi \in \Gamma''$, such that $\Phi \subseteq \Psi$. This means that all clauses, that can be added to Γ' by tree axiom rules or node symbol rules, can be removed by *abs*, and that for all trees B and all node assignments g: $B \models_g \Gamma' \Rightarrow B \models_g \Gamma$.

Rogers (1994b) shows that for each saturated branch that is completely specified wrt $\lhd, \lhd^*, \prec$ and $\approx$ in the way described above, there is a tree satisfying that branch. Consequently Γ' (and also Γ) is satisfiable.

□

With this proposition completeness can be shown:

Proposition 5. *(Completeness)*
For all ϕ_1, ϕ_2: If $\phi_1 \models \phi_2$, then $\phi_1 \vdash \phi_2$.

As a first part of the completeness proof we will show the following lemma:

Lemma. *For all $\phi = \psi_1 \vee \cdots \psi_{n-1}$ and ψ_n, such that ψ_i is an elementary formula for $1 \leq i \leq n$: $\phi \vdash \phi \vee \psi_n$.*

Proof of the lemma: Let $\Gamma = \Gamma(\phi)$ and $\Gamma' = \Gamma(\phi \vee \psi_n)$ be those saturated branches that can be constructed using node names from $node(\phi \vee \psi_n)$.
If ϕ is a tautology, then $\Gamma = \Gamma'$ holds. $\Rightarrow \phi \vdash \phi \vee \psi_n$ trivially holds.
Let ϕ be no tautology.
Construction of Γ and Γ' (here the logical rules will be applied first):
$\|[\psi_1, \cdots, \psi_{n-1}]\|$ and $\|[\psi_1, \cdots, \psi_{n-1}, \psi_n]\|$ are the results of all possible applications of logical rules to $\|[\phi]\|$ and $\|[\phi \vee \psi_n]\|$.
Let Γ_n be the result of applying n rules to $\|[\psi_1, \cdots, \psi_{n-1}, \psi_n]\|$.
Claim: For all $\Psi_n \in \Gamma_n$, that cannot be removed by *taut*, there is a $\Phi \in \Gamma$ such that $\Phi \subseteq \Psi_n$.
Induction on n:

(i) Induction start $n = 0$: As Γ is constructed from $\|[\psi_1, \cdots, \psi_{n-1}]\|$ and as *taut* cannot be applied to any subset of this branch, Γ still contains a subset of $[\psi_1, \cdots, \psi_{n-1}]$ (and also of $[\psi_1, \cdots, \psi_{n-1}, \psi_n]$).

(ii) Induction step $n \Rightarrow n + 1$: Possible rules that can be applied to Γ_n to construct Γ_{n+1}:
1. Tree axiom or node symbol rules: Because of saturatedness of Γ the claim still holds.
2. *res*: Let $[\Phi, \phi], [\Psi, \neg\phi] \in \Gamma_n$, *res* adds the clause $[\Phi, \Psi]$.
   There are $[\Phi'] \subseteq [\Phi, \phi]$ and $[\Psi'] \subseteq [\Psi, \neg\phi]$, such that $[\Phi'], [\Psi'] \in \Gamma$ (induction claim).
   - 1. case: $[\Phi'] \subseteq [\Phi]$ or $[\Psi'] \subseteq [\Psi] \Rightarrow [\Phi'] \subseteq [\Phi, \Psi]$ or $[\Psi'] \subseteq [\Phi, \Psi]$.
   - 2. case: $[\Phi'] = [\Phi'', \phi]$ and $[\Psi'] = [\Psi'', \neg\phi]$
     $\Rightarrow$ (Γ saturated) there is a subset of $[\Phi'', \Psi'']$ (and of $[\Phi, \Psi]$) in Γ, or $[\Phi'', \Psi'']$ (and consequently also $[\Phi, \Psi]$) is a tautology and can be removed by *taut*.
3. *taut* and *abs* only remove clauses, i.e. claim still holds for the result of applying *taut* or *abs*.

$\square$

Proof of Prop. 5: Let ϕ_1, ϕ_2 be formulas, $\phi_1 \models \phi_2$, and let $\Gamma_1 = \Gamma(\phi_1), \Gamma_2 = \Gamma(\phi_2)$ be constructed with $\Upsilon = node(\phi_1) \cup node(\phi_2)$. Without loss of generality ϕ_2 is in conjunctive normal form without double negations.
If ϕ_1 is not satisfiable, then $\Gamma_1 = \|[\quad]\|$ and $\phi_1 \vdash \phi_2$ trivially holds.
Let ϕ_1 be satisfiable. Because of $\phi_1 \models \phi_2$ ϕ_2 is then satisfiable as well.
To show: $\phi_1 \vdash \phi_2$.
Induction on the structure of ϕ_2:

(i) Induction start: Let ϕ_2 be an elementary formula. Then $[\phi_2] \in \Gamma_2$.

Suppose that $[\phi_2] \notin \Gamma_1$. $\phi_2^{neg} := \neg\phi_2$ if ϕ_2 does not contain negations, $\phi_2^{neg} := \phi_2'$, if $\phi_2 = \neg\phi_2'$.

$\Rightarrow$ starting from $\Gamma_1 \cup \{[\phi_2^{neg}]\}$ no closed branch can be constructed by applying *res*.

$\Rightarrow$ (Prop. 4) $\phi_1 \wedge \neg\phi_2$ is satisfiable.

Contradiction to $\phi_1 \models \phi_2$.

$\Rightarrow [\phi_2] \in \Gamma_1$.

Γ_1 and Γ_2 are saturated, and all rule applications, that were done to construct Γ_2, were done as well (with the same instantiation f of the V_Γ, V_Φ and V_ϕ) to construct Γ_1.

$\Rightarrow \phi_1 \vdash \phi_2$.

(ii) Induction step for disjunctions: $\phi_2 = \psi_1 \vee \psi_2 \cdots \vee \psi_n$ with elementary formulas ψ_i, $1 \leq i \leq n$.

Because of $\phi_1 \models \phi_2$ either $\phi_1 \models \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1}$ or $\phi_1 \models \psi_n$ hold.

$\Rightarrow$ (induction claim) $\phi_1 \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1}$ or $\phi_1 \vdash \psi_n$.

$\psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vee \psi_n$ (lemma) and $\psi_n \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vee \psi_n$ (transitivity of $\vdash$).

$\Rightarrow$ (transitivity of $\vdash$) $\phi_1 \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vee \psi_n$.

(iii) Induction step for conjunctions: $\phi_2 = \psi_1 \wedge \psi_2 \cdots \wedge \psi_n$ and $\phi_1 \models \phi_2$.

$\Rightarrow \phi_1 \models \psi_1 \wedge \cdots \psi_{n-1}$ and $\phi_1 \models \psi_n$.

$\Rightarrow$ (induction claim) $\phi_1 \vdash \psi_1 \wedge \cdots \psi_{n-1}$ and $\phi_1 \vdash \psi_n$.

$\Gamma_{n-1} := \Gamma(\psi_1 \wedge \cdots \psi_{n-1}), \Gamma' := \Gamma(\psi_n)$ with node names from $node(\phi_1) \cup node(\phi_2)$. For all $\Psi \in \Gamma_{n-1} \cup \Gamma'$ there is a $\Phi \in \Gamma_1$, such that $\Phi \subseteq \Psi$ (induction claim).

$\Gamma_2 = \Gamma(\phi_2)$ with names from $node(\phi_1) \cup node(\phi_2)$ can be constructed from $\Gamma_{n-1} \cup \Gamma'$ by repeatedly applying *res*, *taut* and *abs*. Let $\Gamma_{n,k}$ be the branch we get after k applications of *res*, *taut* and *abs* to $\Gamma_{n-1} \cup \Gamma'$.

To show: For all $\Psi \in \Gamma_{n,k}$ there is a $\Phi \in \Gamma_1$, such that $\Phi \subseteq \Psi$.

Induction on k:

- Induction start $k = 0$: Claim holds because of $\phi_1 \vdash \psi_1 \wedge \cdots \psi_{n-1}$ and $\phi_1 \vdash \psi_n$.
- Induction step $k \Rightarrow k+1$: Possible rule applications:

res: Let *res* add a clause to $\Gamma_{n,k}$, that is no tautology.

$\Rightarrow$ There are $[\Psi_1, \psi], [\Psi_2, \neg\psi] \in \Gamma_{n,k}$, such that $[\Psi_1, \Psi_2] \in \Gamma_{n,k+1}$.

There are subsets Ξ_1 of $[\Psi_1, \psi]$ and Ξ_2 of $[\Psi_2, \neg\Psi]$ in Γ_1.

1. case: $\psi \notin \Xi_1$ or $\neg\psi \notin \Xi_2 \Rightarrow \Xi_1 \subseteq [\Psi_1, \Psi_2]$ or $\Xi_2 \subseteq [\Psi_1, \Psi_2]$.
3. case: $\Xi_1 = [\Psi_1', \psi], \Xi_2 = [\Psi_2', \neg\psi] \Rightarrow$ there is a subset of $[\Psi_1', \Psi_2']$ in Γ_1 and $[\Psi_1', \Psi_2'] \subseteq [\Psi_1, \Psi_2]$ holds.
Structural rules: *taut* and *abs* only remove clauses.
Consequently $\phi_1 \vdash \psi_1 \wedge \cdots \psi_{n-1} \wedge \psi_n$ also holds.

□

4. TDG-Formalism

4.1. **Grammar.** A Tree Description Grammar consists of a set of formulas (more precisely equivalence classes of formulas), that have a certain specific form (see Fig. 2). The idea is that certain subtrees are completely described whereas between these subtrees only dominance relations are known. This is formalized by the notion of "description". The definition of description given here is more restricted than in Kallmeyer (1996). In contrast to Kallmeyer (1996) negation is not allowed here and disjunction is only allowed for equivalences of l-terms or v-terms.

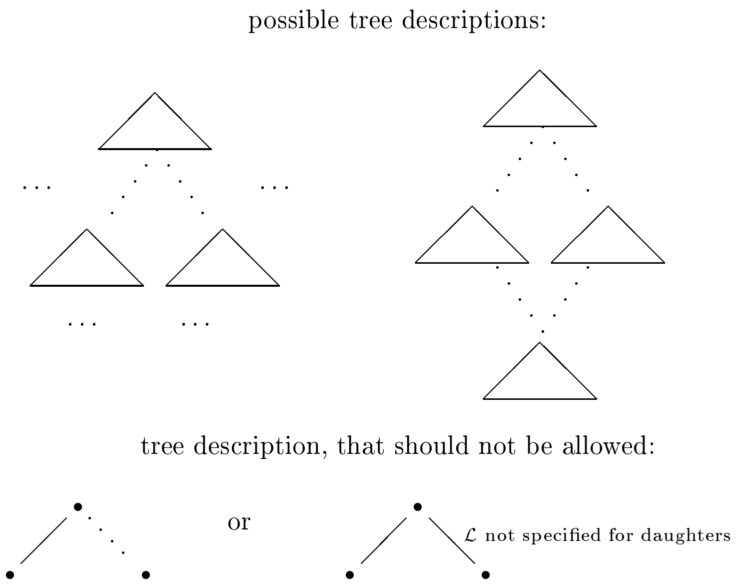


Figure 2. tree descriptions in TDGs

In the following definition, a few notions will be introduced, that are frequently used in course of the article.

Definition 10. *(minimal name, strong dominance, leaf name)*
Let ϕ be a formula, $k, k_1, k_2 \in node(\phi)$

1. *k is a minimal name in ϕ iff $\phi \vdash k \lhd^* k'$ for all $k' \in node(\phi)$.*
2. *$k_1 \lhd^* k_2$ is a strong dominance in ϕ, ($\phi \vdash_s k_1 \lhd^* k_2$), if (apart from the order of the conjuncts in ϕ) $\phi = \phi_1 \wedge k_1 \lhd^* k_2$ and $\phi_1 \not\vdash k_1 \lhd^* k_2$ holds.*
3. *k is a leaf name in ϕ, if for all $k' \in node(\phi)$: $\phi \vdash k \lhd^* k' \Rightarrow \phi \vdash k \approx k'$.*

Definition 11. *(Description)*
A formula ϕ is called a description, if:

1. *ϕ is satisfiable.*
2. *There is at least one minimal name in ϕ.*
3. *ϕ does not contain any negations.*
4. *If ϕ_d is a disjunction $(\phi_1 \vee \phi_2)$ occuring in ϕ, then each of the formulas ϕ_1 and ϕ_2 is either a disjunction itself or it is of the form $t_1 \approx t_2$ where t_1 and t_2 are either l-terms or v-terms.*
5. *For all $k_1, k_2, k_3 \in K$: If $\phi \vdash k_1 \lhd k_2 \wedge k_1 \lhd^* k_3$, then either $\phi \vdash k_1 \approx k_3$ holds or there is a k_4, such that $\phi \vdash k_1 \lhd k_4 \wedge k_4 \lhd^* k_3$.*
6. *For all $k_1, k_2, k_3 \in K$: If $\phi \vdash k_1 \lhd k_2 \wedge k_1 \lhd k_3$, then either $\phi \vdash k_2 \approx k_3$ or $\phi \vdash k_2 \prec k_3$ or $\phi \vdash k_3 \prec k_2$ holds.*

The first condition holds iff $\Gamma(\phi)$ is open (see Prop. 4), i.e. this condition also can be formulated in a syntactic way. This means especially that the satisfiability of a formula is decidable. Then the question whether a formula is a description or not, is also decidable because of the finiteness of $node(\phi)$.

It is easy to see that for each description ϕ and each $k \in node(\phi)$:

- k minimal in ϕ or there is a k' such that $\phi \vdash k' \lhd k$ or $\phi \vdash_s k' \lhd^* k$.
- k leaf name in ϕ or there is a k' such that $\phi \vdash k \lhd k'$ or $\phi \vdash_s k \lhd^* k'$.

The following equivalence relation is necessary to make sure that in each derivation step descriptions with disjoint sets of node names can be chosen.

Definition 12. *Let ϕ, ϕ' be formulas. $\phi \approx_K \phi'$ iff ϕ and ϕ' differ only in a bijection of K.*

$\approx_K$ is an equivalence relation. Let $\overline{\phi}$ be the equivalence class containing ϕ.

For all trees B and formulas ϕ and ϕ' with $\phi \approx_K \phi'$: $B \models \phi$ iff $B \models \phi'$.

Definition 13. *(Tree Description Grammar) A Tree Description Grammar (TDG) is a tuple $G = (A, V, T, D, \phi_S)$, such that:*

1. *A, V and T are pairwise disjoint finite sets, the attribute names, attribute values and the terminal symbols.*
2. *D is a finite set of equivalence classes $\overline{(\psi, K_\psi)}$, such that for all $(\psi, K_\psi) \in \overline{(\psi, K_\psi)}$ ψ is a description (with A, V and T) and $K_\psi \subseteq node(\psi)$.*

   $(\psi_2, K_{\psi_2}) \in \overline{(\psi_1, K_{\psi_2})}$ iff $\psi_1 \approx_K \psi_2$ with a bijection $f_K : K \to K$ and $K_{\psi_2} = \{k \in K \mid$ there is a $k' \in K_{\psi_1}$ such that $k = f_K(k')\}$.

   Each representative of a class in D is called an elementary description.
3. *ϕ_S is a description (with A, V and T), the start description.*

An elementary description ψ is a description with certain marked node names (those in K_ψ). These are important for the derivation process.

4.2. **Derivation.** A derivation starts with the start description ϕ_S. In each derivation step the description that is already derived is extended by adding a new elementary description and some additional node name equivalences.

Definition 14. *(Derivation mode)*
Let $G = (A, V, T, D, \phi_S)$ be a TDG, ϕ_1, ϕ_2 descriptions, and (ψ, K_ψ) an elementary description in G. Without loss of generality $node(\phi_1)$ and $node(\psi)$ are disjoint (if not, choose an appropriate representative of $\overline{(\psi, K_\psi)}$).

1. *ϕ_2 can be derived in one step from ϕ_1 using (ψ, K_ψ) ($\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ or $\phi_1 \overset{(\psi,K_\psi)}{\Rightarrow} \phi_2$), if:*
   - *(A1) $\phi_2 \vdash \phi_1 \wedge \psi$.*
   - *(A2) For all $k_\psi \in node(\psi)$: There is a $k \in node(\phi_1)$ such that $\phi_2 \vdash k_\psi \approx k$ iff $k_\psi \in K_\psi$ or k_ψ minimal in ψ.*
   - *(A3) For all $k_1 \in node(\phi_1), k_2 \in node(\psi)$, such that $\phi_2 \vdash k_1 \approx k_2$: There are no $k_1' \in node(\phi_1), k_2' \in node(\psi)$, such that $\phi_1 \vdash k_1 \lhd k_1'$ and $\psi \vdash k_2 \lhd k_2'$.*
   - *(A4) For all k_1, k_2, such that $\phi_2 \vdash k_1 \lhd k_2$: there are k_3, k_4, such that $\phi_2 \vdash k_1 \approx k_3 \wedge k_2 \approx k_4$ and $\phi_1 \wedge \psi \vdash k_3 \lhd k_4$.*
   - *(A5) For all ϕ_3 fulfilling (A1) - (A4) the following holds: If $\phi_2 \vdash \phi_3$, then $\phi_3 \vdash \phi_2$.*

2. *ϕ_2 can be derived in one step from ϕ_1, $\phi_1 \Rightarrow \phi_2$, if there is an elementary description ψ such that $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$.*
3. *ϕ' can be derived from ϕ (in finitely many steps), $\phi \overset{*}{\Rightarrow} \phi'$, if there are descriptions $\phi_1, \cdots \phi_n$, such that $\phi = \phi_1 \Rightarrow \phi_2 \cdots \Rightarrow \phi_n = \phi'$.*

The definition of the derivation mode is slightly different from the one in Kallmeyer (1996). There $(A4)$ does not occur in the definition of derivation, and for names in ψ, that are neither minimal nor marked, equivalent names in the old description are allowed.

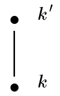
A derivation step in TDGs is a conjunction of two descriptions and additional equivalences of node names.
According to $(A2)$ exactly for those names in the new elementary description ψ, that are minimal or marked names, there must be names in the old description describing the same nodes (i.e. the two names are equivalent). The marking of names somehow says how to put the two descriptions together.
$(A3)$ states that two node names cannot be identified if they both describe internal nodes in completely described subtrees (k_2 and k_5 in fig. 4 for example cannot be identified). At least one of them must be a leaf name or on the left side of a strong dominance in the old descriptions.
$(A4)$ simply says that in a derivation step no new $\lhd$ relations can be added.
The last condition means that ϕ_2 must be "maximally underspecified".

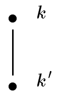
Proposition 6. *If $\phi_1 \overset{(\psi,K_\psi)}{\Rightarrow} \phi_2$, then $node(\phi_2) = node(\phi_1) \cup node(\psi)$.*

Proof:
1. To show: $node(\phi_1) \cup node(\psi) \subseteq node(\phi_2)$.
   ϕ_1 and ψ are descriptions
   $\Rightarrow$ there is a $k_r^{\phi_1}$ such that $\phi_1 \vdash k_r^{\phi_1} \lhd^* k_{\phi_1}$ for all $k_{\phi_1} \in node(\phi_1)$, and there is a k_r^{ψ} such that $\psi \vdash k_r^{\psi} \lhd^* k_\psi$ for all $k_\psi \in node(\psi)$.
   $\Rightarrow$ (transitivity of $\vdash$) $\phi_2 \vdash k_r^{\phi_1} \lhd^* k_{\phi_1}$ for all $k_{\phi_1} \in node(\phi_1)$ and $\phi_2 \vdash k_r^{\psi} \lhd^* k_\psi$ for all $k_\psi \in node(\psi)$.
   $\Rightarrow node(\phi_1) \cup node(\psi) \subseteq node(\phi_2)$.
2. To show: $node(\phi_2) \subseteq node(\phi_1) \cup node(\psi)$.
   Suppose there is a $k \in node(\phi_2) \setminus (node(\phi_1) \cup node(\psi))$.
   (i) 1. case: there is a $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$.
   Let ϕ_2' be the description resulting from replacing all occurences of k by k'. $(A1)$ - $(A4)$ hold for ϕ_2', and $\phi_2 \vdash \phi_2'$ and $\phi_2' \not\vdash \phi_2$ hold.

Contradiction to $(A5)$.

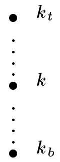
(ii) 2. case: There is no $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$, and there is a $k' \in node(\phi_2)$ such that $\phi_2 \vdash k' \lhd k$ or $\phi_2 \vdash k \lhd k'$.

ϕ_2: 

or 

Contradiction to $(A4)$.

(iii) 3. case: There is no $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$. There is a k_t such that $\phi_2 \vdash_s k_t \lhd^* k$, and k either is a leaf name in ϕ_2 or there is a k_b such that $\phi_2 \vdash_s k \lhd^* k_b$.

ϕ_2: 

ϕ_2' is the description resultung from adding all conjuncts $k_1 \text{ R } k_2$ for $\text{R} \in \{\lhd^*, \prec\}$ such that $\phi_2 \vdash k_1 \text{ R } k_2$ and $k_1, k_2 \neq k$ to ϕ_2 and by then removing all conjuncts containing an occurence of k.
ϕ_2' is a description, $(A1)$ - $(A4)$ hold for ϕ_2', and $\phi_2 \vdash \phi_2'$ and $\phi_2' \not\vdash \phi_2$ hold.
Contradiction to $(A5)$.

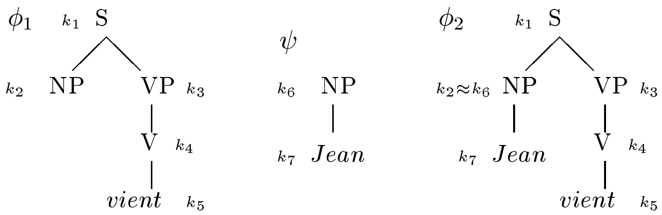
(iv) 4. case: There is no $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$, and k is minimal in ϕ_2: Because of (i)-(iii), for all $k' \in node(\phi_2) \setminus (node(\phi_1) \cup node(\psi))$, $\phi_2 \vdash k \approx k'$ holds. ϕ_2' is the result of removing all conjuncts with occurences of any element of $node(\phi_2) \setminus (node(\phi_1) \cup node(\psi))$ from ϕ_2. ϕ_2' is a description. $(A1)$ - $(A4)$ hold for ϕ_2', and $\phi_2 \vdash \phi_2'$ and $\phi_2' \not\vdash \phi_2$ hold. Contradiction to $(A5)$.

□

The operations substitution and adjunction defined for TAGs can be imitated in TDGs:
For a derivation step similar to substitution (see Fig. 3) an elementary description (ψ, K_ψ) with $K_\psi = \emptyset$ must be added. Then only for the minimal names in ψ there are equivalent node names in the old description.
For an adjunction-like derivation step (ψ, K_ψ) must be such that all marked names are equivalent and they are all leaf names. In the old description ϕ_1 there must be a strong dominance $k_1 \lhd^* k_2$. Then a derivation step adding ψ can consist of putting ψ between k_1 and k_2, which is

Additional to the descriptions a graphical representation is given. $\mathcal{P}$ (normal edges) and $\mathcal{P}^*$ (dotted edges as in fig. 4) are represented.

$$\phi_1 \overset{(\psi, K_\psi)}{\Rightarrow} \phi_2 \text{ with } \phi_2 = \phi_1 \wedge \psi \wedge k_2 \approx k_6$$



$$\begin{array}{rcl}
\phi_1 & = & k_1 \lhd k_2 \wedge k_1 \lhd k_3 \wedge k_2 \prec k_3 \wedge k_3 \lhd k_4 \wedge k_4 \lhd k_5 \\
& & \wedge cat(\delta(k_1)) \approx \mathrm{S} \wedge cat(\delta(k_2)) \approx \mathrm{NP} \wedge cat(\delta(k_3)) \approx \mathrm{VP} \\
& & \wedge cat(\delta(k_4)) \approx \mathrm{V} \wedge \delta(k_5) \approx vient \\
\psi & = & k_6 \lhd k_7 \wedge cat(\delta(k_6)) \approx \mathrm{NP} \wedge \delta(k_7) \approx Jean \\
K_\psi & = & \emptyset
\end{array}$$

FIGURE 3. *Jean vient*: derivation step similar to substitution in TAGs

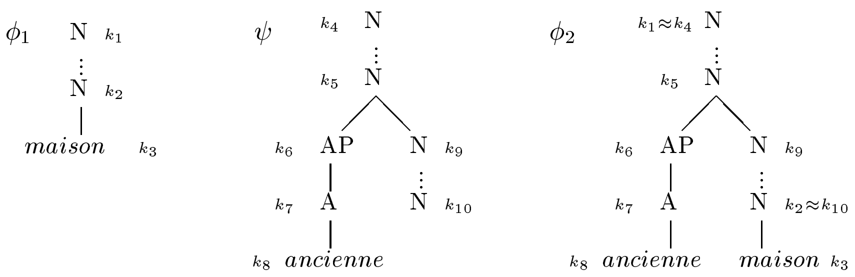
like an adjunction in a TAG (see Fig. 4). If $\phi_1 \vdash \neg\delta(k_1) \approx \delta(k_2)$, then this is comparable to an OA-condition (obligatory adjunction) in a TAG. This idea of two node names k_1, k_2 with a strong dominance such that other descriptions can be put in between, is first mentioned by Vijay-Shanker (1992).

4.3. **Language.** In contrast to TAGs for TDGs three different languages can be distinguished: First the set of descriptions derived by a grammar (the description language), second the set of corresponding "minimal" trees (tree language) and then as in TAGs the set of strings (string language) for these trees.

A possible start description for example is $\phi_S = cat(\delta(k_1)) \approx S, cat \in A, S \in V$ in order to make sure, that the root has the category S.

Definition 15. *(Description language) Let $G = (A, V, T, D, \phi_S)$ be a TDG. The description language of G is $L_D(G) := \{\phi | \ \phi_S \overset{*}{\Rightarrow} \phi\}$.*

$$\phi_1 \overset{(\psi,K_\psi)}{\Rightarrow} \phi_2 \text{ with } \phi_2 = \phi_1 \wedge \psi \wedge k_1 \approx k_4 \wedge k_2 \approx k_{10}$$



$$\begin{aligned}
\phi_1 &= k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge cat(\delta(k_1)) \approx \text{N} \wedge cat(\delta(k_2)) \approx \text{N} \\
&\quad \wedge \delta(k_3)) \approx maison \\
\psi &= k_4 \lhd^* k_5 \wedge k_5 \lhd k_6 \wedge k_5 \lhd k_9 \wedge k_6 \prec k_9 \wedge k_6 \lhd k_7 \wedge k_7 \lhd k_8 \\
&\quad \wedge k_9 \lhd^* k_{10} \wedge cat(\delta(k_4)) \approx \text{N} \wedge cat(\delta(k_5)) \approx \text{N} \wedge cat(\delta(k_6)) \approx \text{AP} \\
&\quad \wedge cat(\delta(k_7)) \approx \text{A} \wedge \delta(k_8) \approx ancienne \wedge cat(\delta(k_9)) \approx \text{N} \\
&\quad \wedge cat(\delta(k_{10})) \approx \text{N} \\
K_\psi &= \{k_{10}\}
\end{aligned}$$

FIGURE 4. *ancienne maison*: adjunction-like derivation step

Now the minimal trees of a description ϕ must be defined. Such a tree must satisfy ϕ, and it has to be minimal in the sense that its $\mathcal{P}$ relation is already described in ϕ. This means that it is sufficient to add term equivalences to ϕ in order to get a complete description of one of its minimal trees.
To avoid some undesirable special cases a third condition for minimal trees is necessary. This condition says that if two node names are not equivalent in the description but describe the same node in a minimal tree, then they both have to be part of some strong dominance in the description.

Definition 16. *(Minimal tree) Let $G = (A, V, T, D, \phi_S)$ be a TDG, $\phi \in L_D(G)$. A tree B is a minimal tree of ϕ, if there is a function g such that*
1. *$B \models_g \phi$.*
2. *For all u_1, u_2 in $\mathcal{U}$: If $\langle u_1, u_2 \rangle \in \mathcal{P}$, then there are $k_1, k_2 \in K$ such that $B \models_g k_1 \lhd k_2$ and $g(k_1) = u_1, g(k_2) = u_2$.*

3. *For all k_1, k_2 with $g(k_1) = g(k_2)$: If $\phi \not\vdash k_1 \approx k_2$, then there are k_1', k_2' such that first either $\phi \vdash_s k_1 \lhd^* k_1'$ or $\phi \vdash_s k_1' \lhd^* k_1$ holds and second either $\phi \vdash_s k_2 \lhd^* k_2'$ or $\phi \vdash_s k_2' \lhd^* k_2$ holds.*

A description $\phi \in L_D(G)$ can have more than one minimal tree (see ϕ_2 in fig. 8) or even no minimal tree. The set of minimal trees of a given description is decidable:

Proposition 7. *Let G be a TDG. For each $\phi \in L_D(G)$ the set of minimal trees of ϕ can be constructed in finitely many steps.*

More precisely we will construct representatives of classes of isomorphic minimal trees. Two trees $B_1 = (\mathcal{U}_1, u_1, \mathcal{P}_1, \mathcal{P}_1^*, \mathcal{L}_1, \mu_1, N_1, T_1)$ and $B_2 = (\mathcal{U}_2, u_2, \mathcal{P}_2, \mathcal{P}_2^*, \mathcal{L}_2, \mu_2, N_2, T_2)$ are isomorphic if there is a bijection $f : \mathcal{U}_1 \to \mathcal{U}_2$ such that for all $u_1, u_2 \in \mathcal{U}_1$:

1. For all $\mathcal{R} \in \{\mathcal{P}, \mathcal{P}^*, \mathcal{L}\} : \langle u_1, u_2 \rangle \in \mathcal{R}_1$ iff $\langle f(u_1), f(u_2) \rangle \in \mathcal{R}_2$, and
2. $\mu_1(u_1) = \mu_2(f(u_1))$.

Henceforward we will not distinguish between isomorphic trees. A tree will be identified with the equivalent class it presents.

Proof: $\phi \in L_D(G)$. To find a minimal tree of ϕ a description ϕ' must be found such that $\phi' \vdash \phi$, $node(\phi) = node(\phi')$ and for all $k_1, k_2 \in node(\phi)$:

(a) $\phi' \vdash_s k_1 \lhd^* k_2$ does not hold.

(b) $\phi' \vdash k_1 \approx k_2$ or $\phi' \vdash \neg k_1 \approx k_2$.

(c) $\phi' \vdash k_1 \lhd k_2 \Rightarrow$ there are k_3, k_4 such that $\phi' \vdash k_1 \approx k_3 \wedge k_2 \approx k_4$ and $\phi \vdash k_3 \lhd k_4$ (no new parent relations).

(d) If $\phi' \vdash k_1 \approx k_2$ and $\phi \not\vdash k_1 \approx k_2$, then there are k_1', k_2' such that first either $\phi \vdash_s k_1 \lhd^* k_1'$ or $\phi \vdash_s k_1' \lhd^* k_1$ holds and second either $\phi \vdash_s k_2 \lhd^* k_2'$ or $\phi \vdash_s k_2' \lhd^* k_2$ holds.

(e) For all $k \in node(\phi')$: If k is a leaf name in ϕ, then there is a $t \in T \cup \{\epsilon\}$ such that $\phi' \vdash \delta(k) \approx t$ (leaves labelled by terminal symbols).

(f) For each $k \in node(\phi')$, that is no leaf name in ϕ' and for each attribute name a, there is a $v \in V \cup \{\bot\}$, such that $\phi' \vdash a(\delta(k)) \approx v$.

To find such a ϕ' term equivalences are added as conjuncts to ϕ in the following way:

$\phi' := \phi$

Add arbitrary conjuncts $k_1 \approx k_2$ to ϕ' with $k_1 \neq k_2$ and $k_1, k_2 \in node(\phi)$.

Add arbitrary conjuncts $\delta(k) \approx t$ to ϕ' with $k \in node(\phi)$ and $t \in T \cup \{\epsilon\}$.

For all $k \in node(\phi)$ where no formula $\delta(k) \approx t$ was added and for all $a \in A$, choose a $v \in V \cup \{\bot\}$ and add the conjunct $a(\delta(k)) \approx v$ to ϕ'.

If the result ϕ' is a description such that conditions (a) - (f) hold (decidable), then it is one of the descriptions we wanted to construct. The construction of course is non-deterministic. But as $node(\phi)$, A, V and T are finite, there are only finitely many possibilities to construct a ϕ' in this way. Consequently all possible ϕ' can be found in finitely many steps.

Now we must show that the resulting ϕ' really correspond to minimal trees:

(i) Each of the ϕ' uniquely describes a minimal tree $B_{\phi'}$ of ϕ:
$B_{\phi'} = (\mathcal{U}_{\phi'}, u_{0\phi'}, \mathcal{P}_{\phi'}, \mathcal{P}^*_{\phi'}, \mathcal{L}_{\phi'}, \mu, N_{\mathcal{A},V}, T)$ such that

(a) $\mathcal{U}_{\phi'} = \{\overline{k} \mid \overline{k}$ is equivalence class wrt $\{\langle k_1, k_2\rangle | \phi' \vdash k_1 \approx k_2\}\}$

(b) $u_{0\phi'} = \{\overline{k} \mid$ all $k \in \overline{k}$ are minimal in $\phi'\}$

(c) $\mathcal{P}_{\phi'} = \{\langle\overline{k_1}, \overline{k_2}\rangle \mid \phi' \vdash k_1 \lhd k_2$ for all $k_1 \in \overline{k_1}, k_2 \in \overline{k_2}\}$

(d) $\mathcal{P}^*_{\phi'} = \{\langle\overline{k_1}, \overline{k_2}\rangle \mid \phi' \vdash k_1 \lhd^* k_2$ for all $k_1 \in \overline{k_1}, k_2 \in \overline{k_2}\}$

(e) $\mathcal{L}_{\phi'} = \{\langle\overline{k_1}, \overline{k_2}\rangle \mid \phi' \vdash k_1 \prec k_2$ for all $k_1 \in \overline{k_1}, k_2 \in \overline{k_2}\}$

(f) A are the attribute names, V the attribute values for $N_{\mathcal{A},V}$.
For all $k_1, k_2 \in K$, that are no leaf names in ϕ', we define: $k_1 \approx_\mu k_2$:iff $\phi' \vdash a(\delta(k_1)) \approx a(\delta(k_2))$ for all $a \in A$.
$N_{\mathcal{A},V} := \{\overline{k}_\mu; \overline{k}_\mu$ is equivalence class wrt $\approx_\mu\}$.
For all $a \in A$: $\langle a, \alpha_a\rangle \in \mathcal{A}$ such that for all $\overline{k}_\mu \in N_{\mathcal{A},V}$ and $v \in V$: $\alpha_a(\overline{k}_\mu) = v$ iff $\phi' \vdash a(\delta(k)) \approx v$ for all $k \in \overline{k}_\mu$.

(g) T are the terminals of the TDG.

(h) $\mu : \mathcal{U}_{\phi'} \to N_{\mathcal{A},V} \cup T \cup \{\epsilon\}$ such that for all $\overline{k} \in \mathcal{U}_{\phi'}$:
1. For all $t \in T \cup \{\epsilon\}$: $\mu(\overline{k}) = t$ iff $\phi' \vdash \delta(k) \approx t$ for all $k \in \overline{k}$.
2. For all $N \in N_{\mathcal{A},V}$: $\mu(\overline{k}) = N$ iff $\overline{k} \subseteq N$.

Because of the inference rules, it is easy to see that $B_{\phi'}$ is a tree.
To show: $B_{\phi'}$ is a minimal tree of ϕ:

1. $B \models_g \phi$ holds if $g(k) = \overline{k}$ for all $k \in node(\phi')$ and $g(k) = u_{0\phi'}$ for all $k \notin node(\phi')$
2. For all $\langle u_1, u_2\rangle \in \mathcal{P}_{\phi'}$ (definition of $\mathcal{P}_{\phi'}$) there are $k_1, k_2 \in node(\phi)$ such that $\phi' \vdash k_1 \lhd k_2$ and $g(k_1) = u_1, g(k_2) = u_2$.
$\Rightarrow$ (condition (c) for ϕ') there are $k_3, k_4 \in node(\phi)$ such that $\phi \vdash k_3 \lhd k_4$ and $\phi' \vdash k_1 \approx k_3 \wedge k_2 \approx k_4$
$\Rightarrow$ (definition of $\mathcal{U}_{\phi'}$) there are $k_3, k_4 \in node(\phi)$ such that $\phi \vdash k_3 \lhd k_4$ and $g(k_3) = u_1, g(k_4) = u_2$.
3. Let $g(k_1) = g(k_2)$ and $\phi \not\vdash k_1 \approx k_2$ hold.
$\Rightarrow \phi' \vdash k_1 \approx k_2$ and $\phi \not\vdash_1 \approx k_2$

$\Rightarrow$ (cond. (d)) there are k_1', k_2' such that first either $\phi \vdash_s k_1 \lhd^* k_1'$ or $\phi \vdash_s k_1' \lhd^* k_1$ holds and second either $\phi \vdash_s k_2 \lhd^* k_2'$ or $\phi \vdash_s k_2' \lhd^* k_2$ holds.

(ii) For each minimal tree B of ϕ there is a ϕ' that can be constructed from ϕ such that B is isomorphic to $B_{\phi'}$:

Let B be a minimal tree of $\phi \in L_D(G)$ wrt the node assignment g.

For all $k, k_1, k_2 \in node(\phi)$: If $k_1 \neq k_2$ and $g(k_1) = g(k_2)$, then add $k_1 \approx k_2$ as a conjunct to ϕ. If there is a $t \in T \cup \{\epsilon\}$ such that $\mu(g(k)) = t$, then add $\delta(k) \approx t$ as a conjunct to ϕ. If $g(k)$ is no leaf, then for all $\langle a, \alpha_a \rangle \in \mathcal{A}$: If there is a $v \in V$ such that $\alpha_a(g(k)) = v$, then add the conjunct $a(\delta(k)) \approx v$, else add the conjunct $a(\delta(k)) \approx \perp$. The resulting description ϕ' is one of the descriptions, that can be constructed from ϕ in the way described above.

Obviously the corresponding tree $B_{\phi'}$ is isomorphic to B.

□

Definition 17. *(Tree language) Let $G = (A, V, T, D, \phi_S)$ be a TDG. The tree language of G is $L_T(G) := \{B;\ B$ is minimal tree of a $\phi \in L_D(G)\}$.*

Definition 18. *(String language)*
Let $G = (A, V, T, D, \phi_S)$ be a TDG, $B \in L_T(G)$.

1. *The string of B, $word(B)$ is the string $w = t_1 t_2 ... t_n \in T^*$ such that:*
   If $b_1, b_2, ..., b_n$ are all leaves in B, that are not labelled by the empty word and if $\langle b_i, b_{i+1} \rangle \in \mathcal{L}_B, 1 \leq i \leq n-1$, then $\mu(b_j) = t_j, 1 \leq j \leq n$.
2. *The string language of G is*
   $L_S(G) := \{w;$ there is a $B \in L_T(G)$ such that $w = word(B)\}$.
3. *A language $L \subseteq T^*$ is a Tree Description Language (TDL), if there is a TDG G such that $L_S(G) = L$.*

5. Formal properties of TDGs

In this section I will first show that TDLs are a true superset of TALs. Then I will define lexicalized TDGs and prove their decidability.

5.1. **Relation to TAGs.** In order to prove that every TAL is a TDL, we have to construct an equivalent TDG for a given TAG. The idea is that a TAG is a special kind of TDG, where for each node name either daughters are described or there is exactly one strong dominance with this name on the left side. Before showing that TALs are a subset of TDLs I will define which descriptions correspond to a tree derived in a TAG:

Definition 19. *(Description of a tree)*
Let γ be a tree derived wrt to a TAG from an elementary tree. The description of γ, $d(\gamma)$, is one unique description ϕ, that can be inductively constructed in the following way:

(i) Choose an arbitrary node name k_r. $\phi := k_r \approx k_r$

(ii) Let u_r be the root node of γ and $\beta_1, \cdots \beta_k$ those auxiliary trees that cannot be adjoined to u_r.

If u_r is labelled by $A \in N$, then $\phi := \phi \wedge cat(\delta(k_r)) \approx A$.

If u_r is labelled by $t \in T \cup \{\epsilon\}$, then $\phi := \phi \wedge \delta(k_r) \approx t$.

Choose a new name k_f. k_r is called the top-name, and k_f the bottom-name of u_r.

If u_r has a OA-constraint, then $\phi := \phi \wedge k_r \lhd^* k_f \wedge adj(\delta(k_r)) \approx root \wedge aux(\delta(k_r)) \approx yes \wedge adj(\delta(k_f)) \approx foot \wedge \beta_1(\delta(k_r)) \approx no \wedge \cdots \wedge \beta_k(\delta(k_r)) \approx no$.

If u_r has a SA-constraint, then $\phi := \phi \wedge k_r \lhd^* k_f \wedge aux(\delta(k_r)) \approx yes \wedge adj(\delta(k_f)) \approx foot \wedge \beta_1(\delta(k_r)) \approx no \wedge \cdots \wedge \beta_k(\delta(k_r)) \approx no$.

$$OA: \quad \begin{array}{l} \bullet\ {\scriptstyle adj \approx root, aux \approx yes} \\ \vdots \\ \bullet\ {\scriptstyle adj \approx foot} \end{array} \qquad\qquad SA: \quad \begin{array}{l} \bullet\ {\scriptstyle aux \approx yes} \\ \vdots \\ \bullet\ {\scriptstyle adj \approx foot} \end{array}$$

If W has a NA-constraint, then $\phi := \phi \wedge k_r \approx k_f$.

If $\gamma_1, \cdots \gamma_n$ are from left to right the trees immediately dominated by W in γ, then choose new names $k_1, \cdots k_n$. $\phi := \phi \wedge k_f \lhd k_1 \wedge \cdots k_f \lhd k_n \wedge k_1 \prec k_2 \wedge \cdots k_{n-1} \prec k_n$. Then from $i = 1$ up to $i = n$: $\gamma := \gamma_i$, $k_r := k_i$, $i := i + 1$ and go to (ii).

Proposition 8. *For every TAG G_A a TDG G_D can be constructed, such that $L_S(G_A) = L_S(G_D)$. The tree languages of G_A and G_D only differ wrt to the nonterminal symbols.*

Before showing the proposition we will prove the following lemma:

Lemma. *For a TAG G there is an equivalent TAG G' such that $L_S(G) = L_S(G')$, the tree languages differ only in nonterminal symbols, and in G' for all nodes u_1, u_2: If u_1 and u_2 are both on a spine of some auxiliary tree (not necessarily the same), if both are neither root nor foot and if for both adjunction is possible, then they are labelled by different nonterminals.*

Proof of the lemma: Let G_A be a TAG. Without loss of generality let G_A be without feature structures and without substitution. The nodes can have adjunction constraints NA, $SA(\{\beta_1, \cdots \beta_n\})$ or $OA(\{\beta_1, \cdots \beta_n\})$.

For all auxiliary trees: All labels of nodes on the spine other than the foot are different from the label of the root (cf. Lemma 1 in Rogers (1994a)).
Construction of an equivalent TAG G'_A:

If there are two nodes with the same label X, both on a spine of some auxiliary tree and both neither root nor foot, such that adjunction is possible for both, then replace one of the X by a new label X_i. Continue until there is no such pair of nodes.

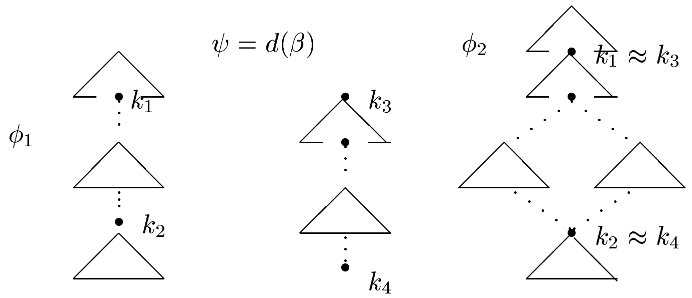
Then for each new label X_i all auxiliary trees β with root label X are copied, and in the new tree β_i the labels X of foot and root are replaced by X_i. In the adjunction constraint for a node labelled by X_i, β is replaced by β_i.

Let G'_A be the new TAG constructed in this way, N' the new nonterminals.

Clearly $L_S(G_A) = L_S(G'_A)$ holds, and the trees of $L_T(G_A)$ and $L_T(G'_A)$ only differ wrt nonterminal symbols.

□

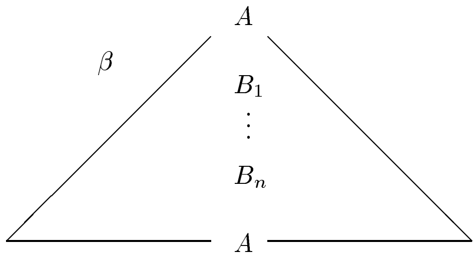
To prove Prop. 8 it is not enough to take the descriptions $d(\gamma)$ for all elementary trees γ in G_A (or G'_A) as elementary descriptions in G_D. The construction of G_D is more complicated because derivation steps $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ as shown in the following figure must be avoided:



Proof of Prop. 8: Let $G'_A = \langle I', A', N', T' \rangle$ be the TAG constructed from G_A as described in the lemma. Without loss of generality let the root nodes of auxiliary trees have a NA-constraint.
Construction of a TDG $G_D = (A_D, V, T_D, D, \phi_S)$ equivalent to G'_A:
$\phi_S := k_1 \approx k_1$, $T_D := T'$.
$A_D := \{cat, adj, aux\} \cup \{\beta_i;\, \beta_i \in A'\}$, $V := N' \cup \{root, foot, yes, no\}$.
Elementary descriptions:
For all $\alpha \in I'$: $\overline{(d(\alpha), \emptyset)} \in D$.
For all $\beta \in A'$:

(i) If there is no node different from the foot on the spine of β where adjunction is possible, then:
$\overline{(d(\beta) \wedge \beta(\delta(k_r)) \approx yes \wedge adj(\delta(k_r)) \approx root \wedge aux(\delta(k_r)) \approx yes, \{k_b\})} \in D$ with k_r minimal and k_b bottom-name of the foot in $d(\beta)$.

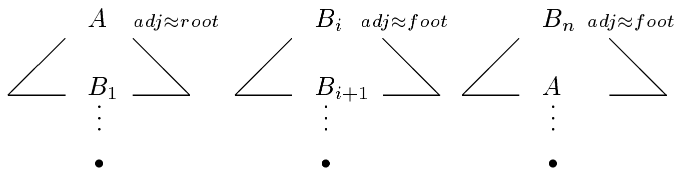
(ii) If $K_1, \cdots, K_n$ are all nodes (different from the foot) on the spine of β, where adjunction is possible, K_i is labelled B_i and K_i dominates K_{k+1} for $1 \leq i \leq n-1$, then:



Let γ be β without the subtree with root K_1. Then:
$\overline{(d(\gamma)' \wedge adj(\delta(k_r)) \approx root \wedge aux(\delta(k_r)) \approx yes \wedge \beta(\delta(k_r)) \approx yes, \{k_b\})} \in D$ with k_r minimal and k_t, k_b is the top and bottom pair of K_1 in $d(\gamma)$, and $d(\gamma)'$ results from removing the conjunct $aux(\delta(k_t)) \approx yes$ from $d(\gamma)$.

For all $i, 1 \leq i \leq n$: Let γ_i be the subtree with root K_i without the one with root K_{i+1}, $d(\gamma_i)$ the description of γ_i, such that k_{min} minimal in $d(\gamma_i)$ and k_t, k_b the top and bottom pair of K_{i+1}. $d(\gamma_i)'$ results from removing the conjunct $aux(\delta(k_t)) \approx yes$ from $d(\gamma_i)$. Then:
$\overline{(d(\gamma_i)' \wedge adj(\delta(k_{min})) \approx foot \wedge aux(\delta(k_{min})) \approx no, \{k_b\})} \in D$.

Let γ' be the subtree dominated by K_n, k_{min} minimal in $d(\gamma')$ and k_t, k_b the top and bottom pair of the foot node. Then:
$\overline{(d(\gamma') \wedge adj(\delta(k_{min})) \approx foot \wedge aux(\delta(k_{min})) \approx no, \{k_b\})} \in D$.

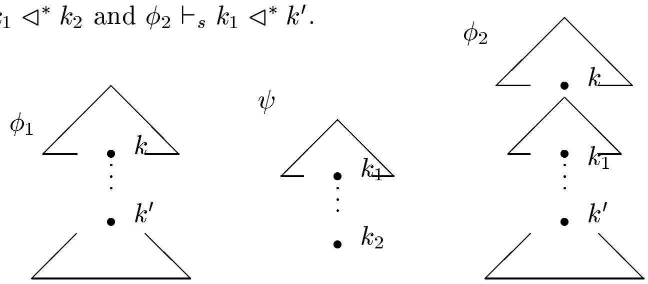


These are all elements in D.

To show: $L_S(G'_A) = L_S(G_D)$.
First we show two lemmata for G_D:

Lemma 1: For all ϕ such that $\phi_S \stackrel{*}{\Rightarrow} \phi$: If $\phi \vdash_s k_1 \lhd^* k_2$ and $\phi \vdash_s k_3 \lhd^* k_4$ and $\phi \vdash k_1 \approx k_3 \vee k_2 \approx k_4$, then $\phi \vdash k_1 \approx k_3 \wedge k_2 \approx k_4$ holds.

For each derivation step $\phi_1 \stackrel{\overline{(\psi,\{k_2\})}}{\Rightarrow} \phi_2$ apart from the first one: ψ is a description constructed for an auxiliary tree. There are k, k' such that $\phi_1 \vdash_s k \lhd^* k'$, and there is a k_{min} minimal in ψ such that $\phi_2 \vdash k \approx k_{min} \wedge k' \lhd^* k_2$. Furthermore there is a k_1 such that $\psi \vdash_s k_1 \lhd^* k_2$ and $\phi_2 \vdash_s k_1 \lhd^* k'$.



(k_2 is identified with a name dominated by k'.)

Proof of Lemma 1: Induction on length n of the derivation

Induction start $n = 1$: For all $\overline{(\psi, K_\psi)} \in D$: There is a ϕ such that $\phi_S \stackrel{\overline{(\psi,K_\psi)}}{\Rightarrow} \phi$ iff $\psi = d(\alpha)$ for an initial tree α and $K_\psi = \emptyset$. This follows from $|node(\phi_S)| = 1$.

Because of this and because of the construction of $d(\alpha)$ for initial trees α, the induction claim holds for $n = 1$.

Induction step: $\phi_S \Rightarrow \phi \stackrel{*}{\Rightarrow} \phi_1 \stackrel{\overline{(\psi,K_\psi)}}{\Rightarrow} \phi_2$, induction claim holds for ϕ_1.

Suppose: $\psi = d(\gamma)$ where all leaves in γ are labelled by terminals, $K_\psi = \emptyset$, and γ has more than one node.

Let k_{min} be minimal in ψ and $\psi \vdash cat(\delta(k_{min})) \approx X$. There are (def. 14) k_1, k_2 such that $\phi_1 \vdash_s k_1 \lhd^* k_2$ and $\phi_2 \vdash k_1 \approx k_{min}$.

(i) 1. case: There is no pair $k'_1, k'_2 \in node(\psi)$ of a top- and a bottom-name such that there is a $k' \in node(\phi_1)$ with $\phi_2 \vdash k'_1 \lhd^* k' \wedge k' \lhd^* k'_2$.
$\Rightarrow$ (def. 11 and 14) There is a leaf name k in ψ and a $t \in T \cup \{\epsilon\}$, such that $\phi_2 \vdash k_1 \approx k_{min} \wedge k \lhd^* k_2 \wedge \delta(k) \approx t \wedge \neg\delta(k_2) \approx t$.
$\Rightarrow$ ϕ_2 is not satisfiable. Contradiction.

(ii) 2. case: There are k'_1, k'_2 in ψ as top and bottom of a node with adjunction possible, and there is a $k' \in node(\phi_1)$ with $\phi_2 \vdash k'_1 \lhd^* k' \wedge k' \lhd^* k'_2$. k'_1, k'_2 are not dominated by any other pair k''_1, k''_2 with the same properties.
Then $\phi_2 \vdash_s k'_1 \lhd^* k_2$ and there is a $k_3 \in node(\phi_1)$ such that $\phi_2 \vdash_s k_3 \lhd^* k'_2$. If k_3 is a leaf name, then we immediately have a contradiction as in (i). If not, then inductively we get a contradiction as in (i).

This means that $\psi = d(\gamma)$ was constructed for an auxiliary tree. Then clearly the derivation step is of the form shown above.

Lemma 2: If $\psi_1, \cdots \psi_n$ with minimal names $k_1, \cdots, k_n$ are the descriptions constructed (in this order) from an auxiliary tree β, then:

If k_r, k_f in ϕ_1 are top and bottom of a node with adjunction possible and if $\phi_1 \overset{\psi_1}{\Rightarrow} \phi'$ with $\phi' \vdash k_r \approx k_1$ holds, then for all ϕ_2, such that $\phi' \overset{*}{\Rightarrow} \phi_2$ and ϕ_2 has minimal trees:

$\phi' \overset{*}{\Rightarrow} \cdots \overset{\psi_1}{\Rightarrow} \cdots \overset{*}{\Rightarrow} \cdots \overset{\psi_n}{\Rightarrow} \cdots \overset{*}{\Rightarrow} \phi_2$

with $\phi_2 \vdash k_r \approx k_1 \wedge k_1 \lhd^* k_2 \wedge \cdots k_n \lhd^* k_f$.

None of the ψ_i is used more often than ψ_1 in a derivation.

Proof of Lemma2: Induction on the maximal number of top and bottom pairs k_t, k_b, such that there is a single $k \in K$ with $\phi_2 \vdash k_t \lhd^* k \wedge k \lhd^* k_b$ for all these pairs k_t, k_b.

Induction start: $n = 0$ trivial (no "adjunction").

Induction step: Let k_r, k_f in ϕ_1 be top and bottm of a node where adjunction is possible, $\phi_1 \vdash cat(\delta(k_r)) \approx A \wedge cat(\delta(k_f)) \approx A$.

Because of the attribute *aux* the only possibility of adding an elementary description by identifying its minimal name with k_r is to add the first of the descriptions of an auxiliary tree with root and foot label A.

Let $\psi_1, \cdots \psi_n$ (as in Lemma 2) be the descriptions of such a tree.

$\phi_1 \overset{\psi_1}{\Rightarrow} \phi'$ where $\phi' \vdash k_r \approx k_1 \wedge k'_1 \lhd^* k_f \wedge cat(\delta(k'_1)) \approx B_1 \wedge cat(\delta(k_f)) \approx A$, where k'_1 is top of the adjunction node on the spine. $B_1 \neq A$ (construction). $\Rightarrow \phi' \vdash_s k'_1 \lhd^* k_f$ and $\phi' \vdash \neg k'_1 \approx k_f \Rightarrow \phi'$ has no minimal tree.

We have to add something between k'_1 and k_f. This can be a description from an auxiliary tree with root and foot labelled by B_1. Then (induction claim) after having added all descriptions for this auxiliary tree, we arrive at a description ϕ'', such that for a new name k: $\phi'' \vdash_s k \lhd^* k_f$ and $\phi'' \vdash k'_1 \lhd^* k \wedge cat(\delta(k)) \approx B_1 \wedge cat(\delta(k_f)) \approx A$, $\Rightarrow \phi''$ still has no minimal tree.

$\Rightarrow$ there must be a description added with a minimal name labelled by B_1, that was not constructed for an auxiliary tree with root and foot label B_1.

$\Rightarrow$ (construction of G'_A) ψ_2 must be added such that k_2 must be equivalent to k'_1. Inductively this means that all $\psi_1, \cdots \psi_n$ must be added in the correct order between k_r and k_f.

We now define a new derivation mode for G_D where all elementary descriptions for one elementary tree are added at once:
First we remove all conjuncts $aux(\delta(k)) \approx no$ for any k from the descriptions of G_D. Then we define:

$\phi_1 \mapsto \phi_2$:if
either $\phi_1 = \phi_S$ and $\phi_1 \Rightarrow \phi_2$
or there are descriptions $\psi_1, \cdots \psi_n$ corresponding (as above) to an auxiliary tree with k_i minimal and k_i^b marked in ψ_i and $\psi_i \vdash_s k_i^t \lhd^* k_i^b$ (for $1 \leq i \leq n$) and there is a top and bottom pair k_r, k_f in ϕ_1, such that $\phi_2 = \phi_1 \wedge \psi_1 \wedge \cdots \wedge \psi_n \wedge k_r \approx k_1 \wedge k_1^b \approx k_2 \wedge \cdots \wedge k_n^b \approx k_f \wedge aux(\delta(k_1^t)) \approx yes \wedge \cdots aux(\delta(k_n^t)) \approx yes$.

As already shown (Lemma 2) $\phi_1 \overset{*}{\Rightarrow} \phi_2$ holds and ϕ_2 has minimal trees iff $\phi_1 \overset{*}{\mapsto} \phi_2$ and there is no strong dominance $\phi_2 \vdash_s k_1 \lhd^* k_2$, such that $\phi_2 \vdash adj(\delta(k_1)) \approx root \wedge adj(\delta(k_2)) \approx foot$.

Furthermore the following can be shown:
$\gamma_1 \overset{*}{\Rightarrow} \gamma_2$ in G'_A iff $\phi_1 \overset{*}{\mapsto} \phi_2$ in G_D with $\phi_1 = d(\gamma_1)$ and $\phi_2 = d(\gamma_2)$ (up to a bijection of K).
If there is no OA-constraint in γ_i, then γ_i is the only minimal tree of ϕ_i.
Proof by induction on the length n of the derivation:

(i) Induction start $n = 0$: Induction claim trivially holds.

(ii) Induction step $n \to n+1$: $\gamma_1 = \gamma_{k_1} \Rightarrow \gamma_{k_2} \cdots \Rightarrow \gamma_{k_n} \Rightarrow \gamma_2$.

Consequently (induction claim) $\phi_1 \mapsto \phi_{k_2} \cdots \mapsto \phi_{k_n}$ with $\phi_i = d(\gamma_i)$, $1 \leq i \leq n$.

Let K be a node in γ_{k_n}, where adjunction is possible, k_t, k_b are top and bottom of K in ϕ_{k_n}.

An auxiliary tree β can be adjoined to K iff the descriptions corresponding to β can be added between k_t and k_b. This follows immediately from the construction of G_D.

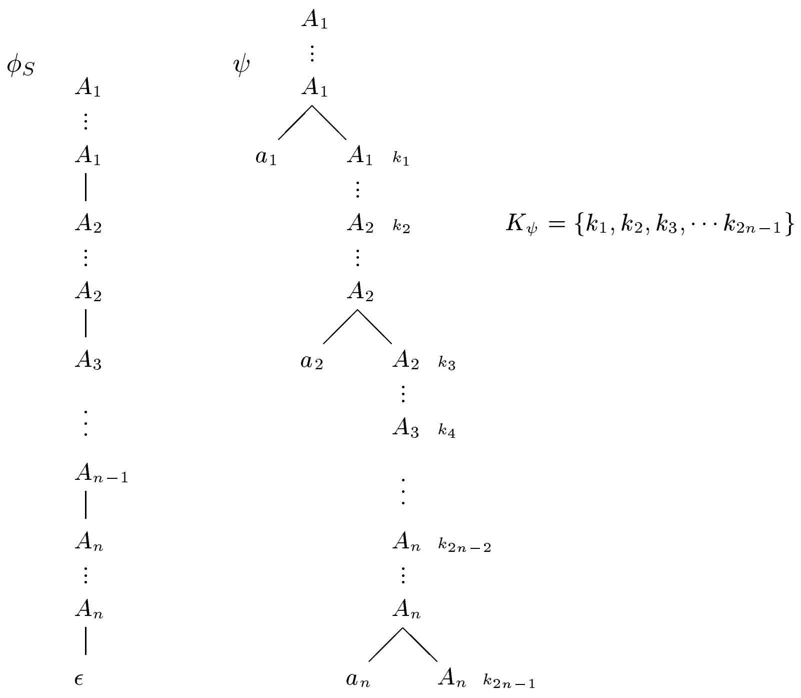


FIGURE 5. TDG for $\{a_1^k \cdots a_n^k\}$

Consequently $\gamma_{k_n} \Rightarrow \gamma_2$ iff $\phi_{k_n} \mapsto \phi_2$.

□

Proposition 9. *There are TDLs, that are no TALs.*

For example $\{a_1^k a_2^k ... a_n^k; 0 \leq k\}$ for $n \geq 5$ and $\{w^n; w \in T^*\}$ for $n \geq 3$ are TDLs. Fig. 5 and 6 show corresponding TDGs. According to the pumping lemma for TALs (see Vijayashanker (1987)) these languages are no TALs.

5.2. **Decidability.** In order to deal with natural languages in a way similar to TAGs, we will now define lexicalized TDGs (for a definition of lexicalized grammars in general see Schabes (1990)).

Definition 20. *(Lexicalized TDG) A TDG G is a lexicalized TDG (LTDG), if for every elementary description ψ in G there are two names k and k_t and there is a $t \in T$, such that $\psi \vdash k \lhd k_t \wedge \delta(k_t) \approx t$.*

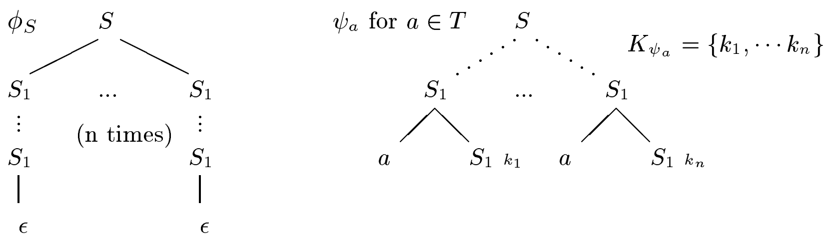


FIGURE 6. TDG for $\{w^n; w \in T^*\}$

With Prop. 6 and Prop. 7 the following proposition holds for LTDGs:

Proposition 10. *(Decidability) For LTDGs the word problem is decidable.*

Proof: First we will show that for a given description ϕ and an elementary ψ the set of equivalence classes of descriptions ϕ' with $\phi \overset{\psi}{\Rightarrow} \phi'$ is finite and can be constructed in finitely many steps (i.e. is decidable).
Here two descriptions ϕ'_1 and ϕ'_2 are called equivalent iff $\phi'_1 \vdash \phi'_2$ and $\phi'_2 \vdash \phi'_1$.
Let ϕ be a description and ψ an elementary decription. Construction of the descriptions ϕ' with $\phi \overset{\psi}{\Rightarrow} \phi'$:

1. Let Φ' be the set of those descriptions ϕ' where $node(\phi') = node(\phi) \cup node(\psi)$ and $\phi' = \phi \wedge \psi \wedge \xi$ for a conjunction ξ. Each conjunct in ξ is either an elementary formula without the symbol $\lhd$ or it is a disjunction of equivalences of l-terms or v-terms. No conjunct and no disjunct occurs more than once in ξ. Furthermore $(A1) - (A4)$ for derivation steps must hold for ϕ'. Φ' is finite and decidable and it is clearly a superset of the descriptions derivable from ϕ by adding ψ.
2. For all $\phi_i \in \Phi'$: if there is a $\phi_j \in \Phi'$ such that $\phi_j \vdash \phi_i$ and $\phi_i \not\vdash \phi_j$, then remove ϕ_i from Φ'.

Because of prop. 6 the set Φ' contains all descriptions ϕ' such that $\phi \overset{\psi}{\Rightarrow} \phi'$ (more precisely: one element of each equivalence class $\overline{\phi'}$).
To decide for a given $w = t_1 \cdots t_n$, $t_i \in T$, whether $w \in L_S(G)$ holds, all derivations $\phi_S \overset{*}{\Rightarrow} \phi$ must be done where for each of the t_i exactly one elementary description ψ is used in deriving ϕ with $\psi \vdash \delta(k) \approx t_i$ for one node name k. In this way in finitely many steps we get a finite set of descriptions. After that we only have to test whether $w = word(B)$

holds for a minimal tree B of one of the derived descriptions. According to Prop. 7 this is decidable.

$\square$

6. Advantages of TDGs for natural language

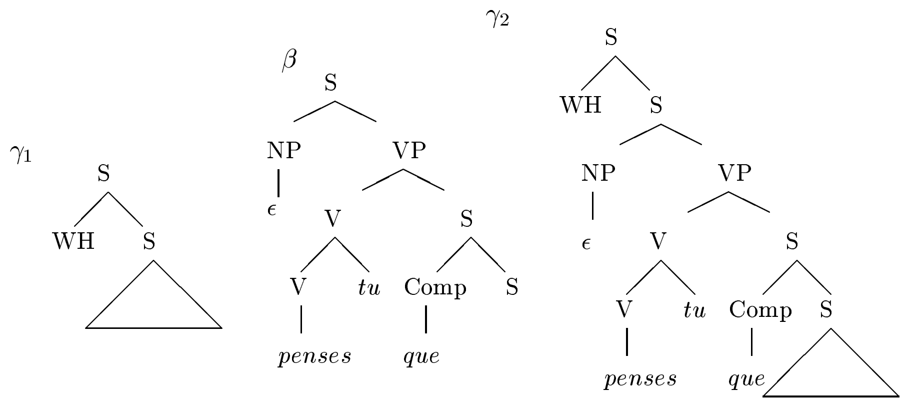
TDGs offer all advantages of multicomponent TAGs (MC-TAGs) because one elementary description can describe several subtrees that have to be added at the same time. MC-TAGs are an extension of TAGs necessary for linguistic purposes, e.g. to deal with long-distance scrambling in German (see Becker et al. (1991)). With TDGs scrambling phenomena can be analyzed in the way proposed by Kroch and Rambow (1994).

6.1. **Complement clauses.** As already pointed out in the introduction, one of the deficiencies of TAGs is, that complement clauses must be treated in a different way than other arguments (see Abeillé (1988) and Rambow et al. (1995)). One of the main ideas in TAG is, that arguments should be added by substitution to the argument structure wheras adjuncts are added by adjunction. The only exception are complement clauses. The reason for this difference is the desire to describe argument structures locally. In TAGs all argument nodes for one lexical head are in one elementary tree. Even for unbounded dependencies the argument that has been moved has to be in the same elementary tree as the head of the argument structure. If one wants to preserve this in a TAG, it is not possible to add complement clauses by substitution but the argument structure must be adjoined to the complement clause.

With TDGs it is possible to have this local analysis of unbounded dependencies without treating complement clauses in a way different from other arguments (see fig. 7). Even for unbounded dependencies inside of complement clauses the derivation can start from the argument structure and the complement clause is added in one derivation step. The crucial point here is that there can be arbitrarily many marked names in elementary descriptions.
This treatment of complement clauses corresponds to the *subsertion* operation proposed in Rambow et al. (1995).

In TDGs the elementary description for arguments and those for modifiers differ in the following way: Arguments are described by descriptions ψ

TAG-analysis: $\gamma_1 \overset{\beta}{\Rightarrow} \gamma_2$



TDG-analysis: $\phi_1 \overset{(\psi, K_\psi)}{\Rightarrow} \phi_2$ with $\phi_2 = \phi_1 \wedge \psi \wedge k_1 \approx k_{10} \wedge k_2 \approx k_{13} \wedge k_9 \approx k_{14}$

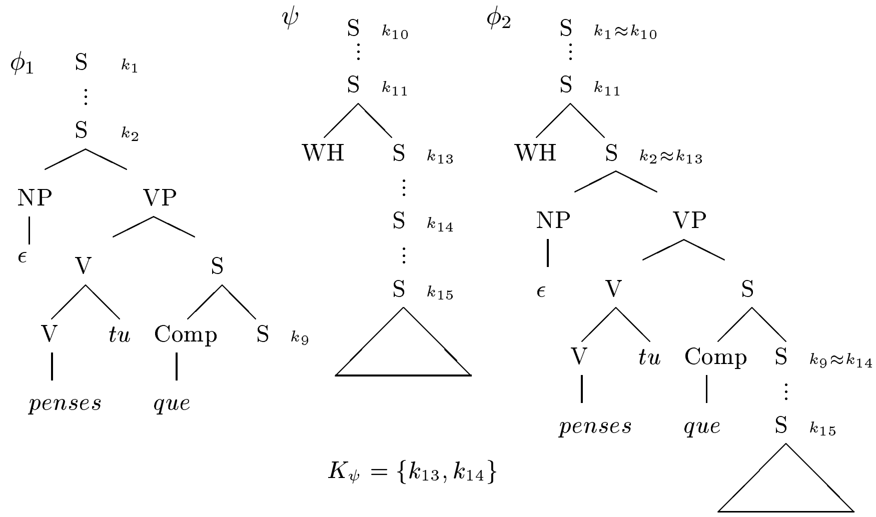
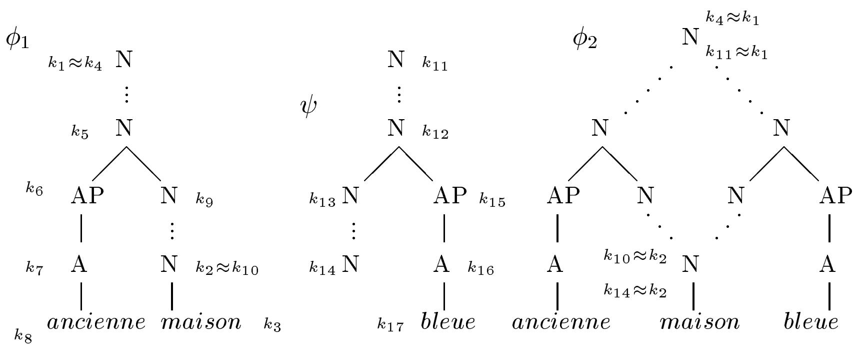


FIGURE 7. complement clauses with unbounded dependencies
qui penses-tu que Marie aime ϵ (see Abeillé (1988))

$$\phi_1 \overset{(\psi, K_\psi)}{\Rightarrow} \phi_2 \text{ with } \phi_2 = \phi_1 \wedge \psi \wedge k_2 \approx k_{14} \wedge k_1 \approx k_{11}$$



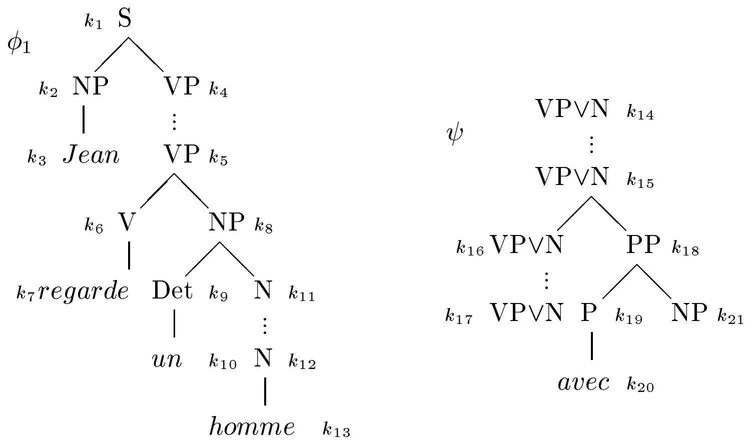
$$\begin{array}{rcl} \phi_1 & = & \text{see } \phi_2 \text{ in fig. 4} \\ \psi & = & k_{11} \lhd^* k_{12} \wedge k_{12} \lhd k_{13} \wedge k_{12} \lhd k_{15} \wedge k_{13} \prec k_{15} \wedge k_{13} \lhd^* k_{14} \\ & & \wedge k_{15} \lhd k_{16} \wedge k_{16} \lhd k_{17} \wedge cat(\delta(k_{11})) \approx \mathrm{N} \wedge cat(\delta(k_{12})) \approx \mathrm{N} \\ & & \wedge cat(\delta(k_{13})) \approx \mathrm{N} \wedge cat(\delta(k_{14})) \approx \mathrm{N} \wedge cat(\delta(k_{15})) \approx \mathrm{AP} \\ & & \wedge cat(\delta(k_{16})) \approx \mathrm{A} \wedge \delta(k_{17}) \approx bleue \\ K_\psi & = & \{k_{14}\} \end{array}$$

The minimal trees of ϕ_2 are those in fig. 1.

FIGURE 8. Nominal modifiers: *ancienne maison bleue*

such that no leaf name is marked. In contrast to this the elementary description of a modifier contains a marked leaf name, that can be compared to a foot node of an auxiliary tree in a TAG.

6.2. **Underspecification.** As already mentioned in the introduction one of the main reasons for defining TAG-like grammars in a constraint based way was the desire to have underspecified representations of trees. With TDGs it is possible to describe several readings in one tree description. Syntactic ambiguities that can be analyzed in this way are nominal modifier ambiguities as in Fig. 8 and also PP-attachment ambiguities as in Fig. 9. For scope ambiguities, for example quantifier ambiguities, it is also possible to derive underspecified representations similar to those in Muskens (1995).



$$
\begin{array}{rcl}
\phi_1 & = & k_1 \lhd k_2 \wedge k_1 \lhd k_4 \wedge k_2 \prec k_4 \wedge k_2 \lhd k_3 \wedge k_4 \lhd^* k_5 \wedge k_5 \lhd k_6 \\
 & & \wedge k_5 \lhd k_8 \wedge k_6 \prec k_8 \wedge k_6 \lhd k_7 \wedge k_8 \lhd k_9 \wedge k_8 \lhd k_{11} \wedge k_9 \prec k_{11} \\
 & & \wedge k_9 \lhd k_{10} \wedge k_{11} \lhd^* k_{12} \wedge k_{12} \lhd k_{13} \wedge \cdots \\
\psi & = & k_{14} \lhd^* k_{15} \wedge k_{15} \lhd k_{16} \wedge k_{15} \lhd k_{18} \wedge k_{16} \prec k_{18} \wedge k_{16} \lhd^* k_{17} \\
 & & \wedge k_{18} \lhd k_{19} \wedge k_{18} \lhd k_{21} \wedge k_{19} \prec k_{21} \wedge k_{19} \lhd k_{20} \\
 & & \wedge [cat(\delta(k_{14})) \approx \mathrm{VP} \vee cat(\delta(k_{14})) \approx \mathrm{N}] \\
 & & \wedge [cat(\delta(k_{15})) \approx \mathrm{VP} \vee cat(\delta(k_{15})) \approx \mathrm{N}] \wedge cat(\delta(k_{15})) \approx cat(\delta(k_{16})) \\
 & & \wedge [cat(\delta(k_{17})) \approx \mathrm{VP} \vee cat(\delta(k_{17})) \approx \mathrm{N}] \wedge cat(\delta(k_{18})) \approx \mathrm{PP} \\
 & & \wedge cat(\delta(k_{19})) \approx \mathrm{P} \wedge \delta(k_{20}) \approx avec \wedge cat(\delta(k_{21})) \approx \mathrm{NP} \\
K_\psi & = & \{k_{17}\} \\
\phi_2 & = & \phi_1 \wedge \psi_1 \wedge k_4 \approx k_{14} \wedge k_{12} \approx k_{17}
\end{array}
$$

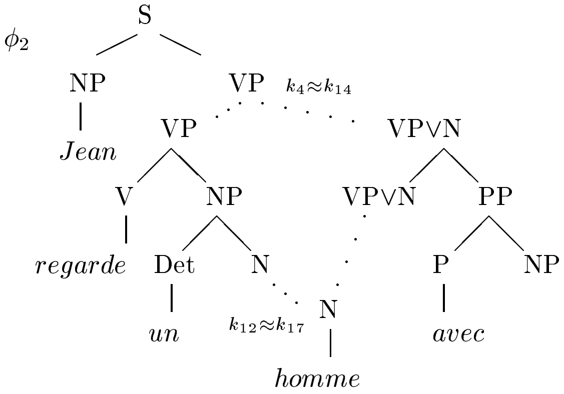


FIGURE 9. PP-attachment: *Jean regarde un homme avec un télescope*

7. Conclusion

In this article I have presented a constraint-based extension of Tree Adjoining Grammars. For the grammar formalism I am using a quantifier-free first order logic similar to the one proposed by Rogers (1994b). Based on inference rules in the style of those introduced by Rogers I have defined a notion of syntactic consequence that was proved to be decidable, sound and even complete.

A Tree Description Grammar consists of a set of formulas of this logic. These formulas are called descriptions and have a certain form, they describe certain subtrees and dominance relations between these subtrees. The grammar formalism is derivation based where in each derivation step a description is extended by adding another description belonging to the grammar.

Concerning their formal properties I have shown that Tree Description Languages are a true superset of Tree Adjoining Languages and that they contain the languages $\{a_1^k a_2^k ... a_n^k; 0 \leq k\}$ for $n \geq 5$ and $\{w^n; w \in T^*\}$ for $n \geq 3$. Tree Description Grammars seem to be powerful enough to deal with natural languages. But it is possible that they are even too powerful. I am currently working on restricted versions of TDGs that still have all the advantages of TDGs for natural language processing but that can be shown to describe semilinear languages, i.e. languages for which the constant growth property holds.

As we have seen TDGs not only provide a constraint-based definition of TAGs. They also offer the advantages of MC-TAGs and of D-Tree Grammars because they enable us to add several subtree descriptions at the same time and to treat arguments in a uniform way.
Furthermore TDGs allow the introduction of underspecification for TAGs. It seems to be quite natural to use constraint-based grammar formalisms to allow underspecification. Using TDGs, underspecified representations can be derived for structural ambiguities and for scope ambiguities.
Thus with respect to natural language processing TDGs are a reasonable extension of TAGs.

References

Abeillé, A.: 1988, Parsing French with tree adjoining grammar: some linguistic accounts, *Proceedings of COLING*, Budapest.

Backofen, R., Rogers, J. and Vijay-Shanker, K.: 1995, A first-order axiomatization of the theory of finite trees, *Jounal of Logic, Language and Information* **4**(1), 5–39.

Becker, T., Joshi, A. K. and Rambow, O.: 1991, Long-distance scrambling and tree adjoining grammars, *Proceedings of ACL-Europe*.

Ebbinghaus, H.-D., Flum, J. and Thomas, W.: 1992, *Einführung in die mathematische Logik*, BI-Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich.

Joshi, A. K.: 1987, An introduction to Tree Adjoining Grammars, *in* A. Manaster-Ramer (ed.), *Mathematics of Language*, John Benjamins, Amsterdam.

Kallmeyer, L.: 1996, Tree Description Grammars, *in* D. Gibbon (ed.), *Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference*, Mouton de Gruyter, Berlin.

Kroch, T. and Rambow, O.: 1994, The linguistic relevance of quasi-trees, *3e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3)*, Paris.

Marcus, M. P., Hindle, D. and Fleck, M. M.: 1983, D-theory: Talking about talking about trees, *Proceedings of ACL*.

Muskens, R.: 1995, Order-independence and underspecification, *in* J. Groenendijk (ed.), *Ellipsis, Underspecification, Events and More in Dynamic Semantics*, DYANA Report R2.2.C.

Poesio, M.: 1996, Semantic ambiguity and perceived ambiguity, *in* K. van Deemter and S. Peters (eds), *Semantic Ambiguity and Underspecification*, CLSI Publications Stanford.

Rambow, O.: 1994, Multiset-Valued Linear Index Grammars: Imposing dominance constraints on derivations, *Proceedings of ACL*.

Rambow, O., Vijay-Shanker, K. and Weir, D.: 1995, D-Tree Grammars, *Proceedings of ACL*.

Rogers, J.: 1994a, Capturing CFLs with Tree Adjoining Grammars, *Proceedings of ACL*.

Rogers, J.: 1994b, *Studies in the Logic of Trees with Applications to Grammar Formalisms*, PhD thesis, University of Delaware.

Rogers, J. and Vijay-Shanker, K.: 1992, Reasoning with descriptions of trees, *Proceedings of ACL*.

Schabes, Y.: 1990, *Mathematical and Computational Aspects of Lexicalized Grammars*, PhD thesis, University of Pennsylvania.

Shieber, S. M. and Schabes, Y.: 1990, Synchronous Tree-Adjoining Grammars, *Proceedings of COLING*.

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

Vijay-Shanker, K. and Joshi, A. K.: 1988, Feature structures based tree adjoining grammar, *Proceedings of COLING*, Budapest.

Vijay-Shanker, K. and Schabes, Y.: 1992, Structure sharing in lexicalized tree adjoining grammars, *Proceedings of COLING*, Nantes.

Vijay-Shanker, K., Weir, D. J. and Joshi, A. K.: 1987, Characterizing structural descriptions produced by various grammatical formalisms, *Proceedings of ACL*, Stanford.

Vijayashanker, K.: 1987, *A Study of Tree Adjoining Grammars*, PhD thesis, University of Pennsylvania.

SfS, Universität Tübingen, Wilhelmstr. 113, D-72074 Tübingen, Germany
E-mail address: `lk@sfs.nphil.uni-tuebingen.de`