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

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

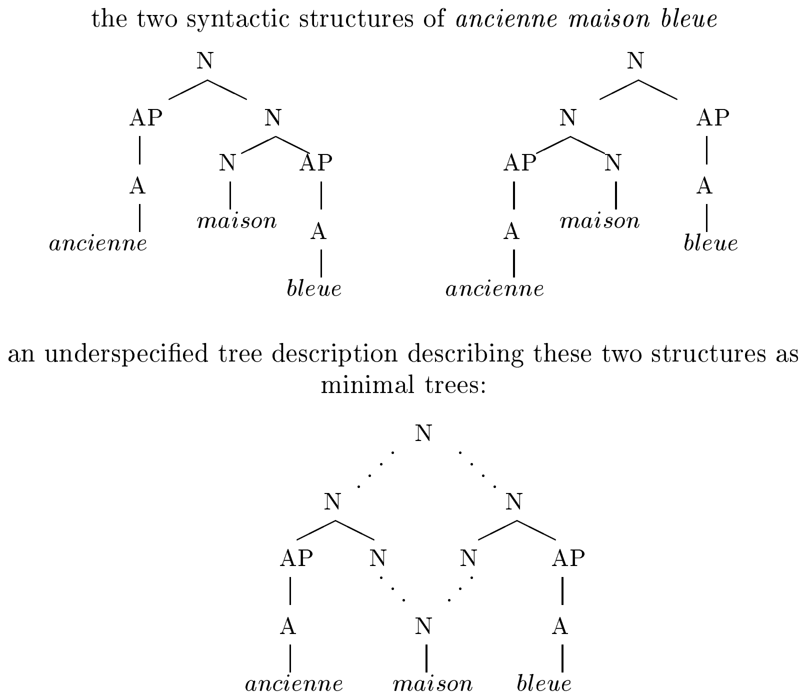

FIGURE 1. syntactic underspecification: *ancienne maison bleue*

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 $V \cup A \cup T \cup \{\epsilon, \bot\}$*
- *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 $x \in V \cup \{\bot\}$ 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* $[[a(\delta(k))]]_g = \bot$.
   - $[[v]]_g = v$ *for all* $v \in V \cup \{\bot\}$.
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