Obtaining Trees from Their Descriptions: An Application to Tree-Adjoining Grammars[1] [2]

James Rogers
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19716
jrogers@udel.edu

K. Vijay-Shanker
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19716
vijay@udel.edu

[1]This work was partially supported by NSF grant IR19016591
[2]The authors are indebted to the referees for considerable help in clarifying the presentation of this material.

Abstract

Partial descriptions are sets of constraints which do not necessarily determine a unique object. Two related issues arise when objects are specified by partial descriptions. The first asks whether a given description is satisfied by any object. The second seeks to produce, from a description, an appropriate representative of those objects which satisfy it. We explore these issues in the context of two recent proposals utilizing partial descriptions of trees in the area of Tree-Adjoining Grammars. In this context, what counts as an appropriate representative of the set of trees which satisfy a description is a tree in that set which is minimal in the sense that every other tree in the set can be derived from it by adjunction. We call a partial description for which there is a unique such minimal tree a quasi-tree.

We formalize the notions of partial descriptions of trees and of quasi-trees, and, using these, provide mechanisms which resolve any description of trees into an equivalent set of quasi-trees and which produce, from that set, the minimal trees satisfying the description. Such mechanisms are presupposed by the proposals we consider, but are provided in a complete form for the first time here.

Keywords: partial descriptions, tree-adjoining grammars, constraint-based formalisms, tableau methods

1 Introduction

Partial descriptions arise naturally whenever systems specify objects as the composition of their component parts. The properties of the component parts form a description of the object—a set of constraints which it must satisfy. If more than one object meets those constraints then the description is partial; the object is underspecified. While systems which employ these descriptions generally must produce some unique object, it is often undesirable or even impossible for the description to totally define the object. It is frequently the case, for example, that every description built by the system can be extended with additional constraints. In these cases, while the set of objects satisfying the description can always be further restricted, as long as it is non-empty it is infinite. If a system in which such cases can arise is to produce a unique object it must include some notion of how to select an appropriate representative from the set of objects which satisfy a description. A complementary case arises when that set is empty, that is, when a description of a hypothetical object is built which no actual object satisfies. Thus there are two related issues which these systems must resolve: determining if a given description is satisfiable and, if it is, producing an object representative of those which satisfy it. In this paper we are concerned with these issues in the context of two recent developments in Tree-Adjoining Grammars which employ partial descriptions of trees. The first of these is a reinterpretation of TAGs as a constraint-based formalism (Vijay-Shanker, 1992). The second is a means of reducing the size of the presentation of a Lexicalized TAG (LTAG) grammar through structure sharing (Vijay-Shanker & Schabes, 1992).

1.1 Descriptions of trees and TAG

Constraint-based parsers (Shieber, 1986; Shieber, 1989) operate by building, on the basis of the input, a collection of constraints on the structure interpreting that input, a phrase-structure tree for instance. One useful way to think of this process is that the parser starts with a completely general description—one that is satisfied by every phrase-structure tree—and proceeds by adding constraints to the description as the input is scanned, thus restricting the set of trees which satisfy it. Ultimately, only those trees which are acceptable readings of the input remain. A key aspect of this approach is that the set of trees which satisfy the description should decrease monotonically, i.e., trees which have been ruled out at one stage should not be reintroduced at a latter stage.

If TAGs are to be re-interpreted in this context it is clear that they cannot manipulate trees. For, while a tree partially describes any tree which extends it at the frontier (by substitution), it does not describe those trees which are derived from it by adjunction. In figure 1, for example, γ_1 is the result of adjunction of β_1 into α_1 at the node labeled VP. But α_1 cannot be regarded as a partial description of γ_1 since, among other things, the node labeled S is the grandparent of the node labeled v in α_1 but not in γ_1. In (Vijay-Shanker, 1992), TAGs are interpreted as manipulating partial descriptions of trees of a particular form, termed *quasi-trees*. In place of the elementary tree α_1 the grammar would include the quasi-tree α_1' of figure 2. Here, the dotted line denotes (reflexive) *domination*, a path of any length, rather than *immediate domination* (or, in the terminology we adopt here, *parent*). Thus α_1' is a partial description of α_1, lacking only the information that the nodes denoted by vp_1 and vp_2 are the same (and the consequences of this). But, since the path between vp_1 and vp_2 can be of any length, it is also a partial description of every tree which can be derived from α_1 by adjunction at VP. In this context, adjunction of β'_1 into α'_1 at vp_1/vp_2 is accomplished by conjoining the constraints represented by α'_1 with those represented by β'_1 along with assertions that vp_2 and vp_4 name the same nodes. That vp_1 then dominates vp_3 follows as a consequence of the properties of trees.

Ultimately, of course, the parser must produce a phrase-structure tree for the input. While under some circumstances any of the objects described by the final description might do, in this case only one of these is appropriate—that in which all of the remaining underspecified links (dotted lines) are taken to be paths of length zero. In (Vijay-Shanker, 1992) this is referred to as the *circumscriptive reading*. This is a well defined concept solely because these quasi-trees were not intended to describe arbitrary sets of trees. Rather they describe sets of trees which have a unique minimum with respect to that partial-order which relates each tree to those trees which can be derived from it by adjunction. That minimum tree is the circumscriptive reading of the quasi-tree. To emphasize the role of this partial order in determining the circumscriptive reading of a quasi-tree, we refer to the circumscriptive reading here as the unique *minimal referent* of the quasi-tree.

We adopt the definition of quasi-tree as any set of trees which has such a unique minimal referent or, equivalently, any partial description of trees for which the set of trees that satisfy it is such a set. Descriptions are couched in a simplified first-order logic for trees. In the language of this logic α'_1 could be expressed:

$$s_1 \triangleleft np_1 \wedge s_1 \triangleleft vp_1 \wedge np_1 \prec vp_1 \wedge$$
$$vp_1 \triangleleft^* vp_2 \wedge vp_2 \triangleleft v_2 \wedge vp_2 \triangleleft np_2 \wedge v_2 \prec np_2.$$

where $\triangleleft$, $\triangleleft^*$, and $\prec$ represent the parent, domination, and left-of (linear precedence) relations respectively. There is some tree meeting a description just in case that description is satisfiable over trees. Adjunction then is an operation on formulae which takes two formulae into the conjunction of those formulae along with additional formulae identifying the nodes at which the adjunction is applied. With this interpretation it is apparent that a number of restrictions on adjunction can be relaxed, such as the requirement that the structures used for adjunction have a single contiguous component or the requirement that both ends of the underspecified link be labeled identically. This additional flexibility is exploited in the compact representation of LTAG grammars proposed in (Vijay-Shanker & Schabes, 1992), in which combining operations on potentially arbitrary partial descriptions of trees are employed.

Vijay-Shanker and Shabes (1992) address the fact that there tends to be a great deal of redundancy in the information encoded by the elementary trees of a Lexicalized Tree-Adjoining Grammar. There is information, for instance, that appears in all of the trees for verbs—that represented by α'_3 of figure 3. This makes for grammars of unwieldy size which are difficult to maintain. To remedy this, a hierarchical representation of a grammar is proposed. Each class of the hierarchy is associated with a partial description which encodes the information that is common to the trees in that class. Thus α'_3 is associated with the class of all verbs, while α'_4 is associated with the class of all transitive verbs. To extract the elementary tree for a lexical item one takes the conjunction of the descriptions associated with each of the classes of which it is a member, along with the assertion that the anchors of those descriptions (the nodes representing the lexical item) are the same node. The elementary tree is the minimal referent of the resulting description. In this way α'_1, which describes elementary trees for transitive verbs, can be obtained from the descriptions associated with the class of all verbs, α'_3, and the class of all transitive verbs, α'_4.

It is important to recognize that α'_3 is not a quasi-tree. There is no unique tree from which every tree satisfying that description can be derived by adjunction. Further, there is no guarantee that the result of taking the conjunction of the partial-descriptions associated with the classes including a lexical item either is satisfiable or has a unique minimal referent. These are requirements, however, for a properly formed grammar.[1] This calls for a mechanism to establish that these requirements in fact do hold for all lexical items. In addition a mechanism is required to construct the elementary tree from its description. These were the issues which motivated this work initially.

1.2 Outline of the paper

We provide, first, formal definitions of quasi-trees as distinguished sets of trees (and the partial descriptions which define them) and of minimal referents. Formal definitions of these concepts were not included in either (Vijay-Shanker, 1992) or (Vijay-Shanker & Schabes, 1992).[2] Secondly, we provide a mechanism which takes arbitrary (finite) descriptions of trees—those that include disjunction and negation in addition to the conjunctive descriptions we have considered so far—and resolves them into an equivalent set of quasi-trees. Each of these quasi-trees is, by definition, satisfiable by at least its minimal referent. Thus this mechanism decides satisfiability of the given description as well; the description is satisfiable iff the set of quasi-trees derived from it is non-empty. Finally, we provide a mechanism for constructing, from a given quasi-tree, its minimal referent.

We begin, in the next section, by formalizing the notions of descriptions and trees. In section 3 we develop formal definitions of minimal referent and (consequently) quasi-trees, along with a syntactic characterization of those descriptions which are quasi-trees. In section 4 we introduce our mechanism for resolving arbitrary descriptions into an equivalent set of quasi-trees. We show, in sections 5 and 6, that this mechanism produces a set of quasi-trees from any satisfiable description. In the process we develop a mechanism for constructing minimal trees from these quasi-trees (in section 5). Finally, in section 7 we establish that the set of minimal referents of a description is precisely characterized by the set of quasi-trees derived from it by the mechanism. The paper closes with some brief concluding remarks.

[1] Although one can easily imagine circumstances where it might be desirable to associate a lexical item with a finite set of trees—lexical ambiguity for instance.

[2] While a definition of quasi-trees as partial models was introduced in (Rogers & Vijay-Shanker, 1992), the approach we take here provides a much more satisfactory foundation. A brief contrast of these approaches can be found in (Rogers & Vijay-Shanker, 1993).

2 Formal Preliminaries

2.1 Language

We begin by fixing formal definitions of the notions of trees, descriptions of trees, and what it means for a tree to meet some description. Our descriptions are couched in a simplified first-order language in which neither variables nor quantifiers occur.

Definition 1 *For K a countable set of constant symbols, let L_K be the language built up from K and the symbols:*

$\triangleleft, \triangleleft^*, \prec, \approx$	—	*two place predicates,* parent, domination, left-of *and equality respectively,*
$\wedge, \vee, \neg$	—	*usual logical connectives*
$(,), [,]$	—	*usual grouping symbols*

The language includes predicates for both parent and domination. This allows descriptions to specify domination independently of parent, making possible descriptions like α_1' of figure 2 in which individuals are related by domination without specifying a chain of parent links between them. Since variables do not occur in our language the only way to refer to an individual is with a constant; the only **terms** of the language are the constants. **Atomic formulae**, **literals** and **well-formed-formulae** are generated in the usual fashion. We use t, u, v, etc. to denote terms and ϕ, ψ, etc. to denote wffs. R denotes any of the predicates. A **description** is just an arbitrary, finite set of formulae.

While, in general, we work with a family of languages parameterized by K, the set of constant symbols from which the formulae are built, this set will usually be fixed by the context. In particular, since we are working with descriptions, we will assume the language including just the set of constant symbols occurring in the description.

Definition 2 *For any set of formulae Φ, let $\mathcal{T}(\Phi)$ denote the set of terms occurring in Φ. Let L_Φ denote the language $L_{\mathcal{T}(\Phi)}$.*

Thus given a description Φ, we will assume the language L_Φ.

2.2 Models

For simplicity in interpreting the language, we define trees to be a particular class of first order structures. A tree for a language L_K is a structure interpreting the constants and predicates of L_K in which the interpretation of the predicates reflects the properties of trees.

Definition 3 *For any language L_K, a* model for L_K *is a tuple $\langle \mathcal{U}, \mathcal{I}, \mathcal{P}, \mathcal{D}, \mathcal{L} \rangle$, where:*

$\mathcal{U}$ is a non-empty universe,

$\mathcal{I}$ is a function from K to $\mathcal{U}$

$\mathcal{P}$, $\mathcal{D}$, and $\mathcal{L}$ are binary relations over $\mathcal{U}$ (interpreting $\triangleleft$, $\triangleleft^*$, and $\prec$ respectively).

A tree *is a model satisfying the conditions:*

For some $\mathcal{R} \in \mathcal{U}$ and all $w, x, y, z \in \mathcal{U}$,

T1 $\quad \langle \mathcal{R}, x \rangle \in \mathcal{D}$,

T2 $\quad \langle x, x \rangle \in \mathcal{D}$,

T3 $\langle x,y\rangle, \langle y,x\rangle \in \mathcal{D} \Rightarrow x = y$,

T4 $\langle x,y\rangle, \langle y,z\rangle \in \mathcal{D} \Rightarrow \langle x,z\rangle \in \mathcal{D}$,

T5 $\langle x,y\rangle \in \mathcal{P} \Rightarrow \langle x,y\rangle \in \mathcal{D}$, *and* $\langle y,x\rangle \notin \mathcal{D}$

T6 $\langle x,z\rangle \in \mathcal{P}$ *and* $\langle x,y\rangle, \langle y,z\rangle \in \mathcal{D} \Rightarrow x = y$ *or* $y = z$,

T7 $\langle x,y\rangle \in \mathcal{D} \Rightarrow \langle y,x\rangle \in \mathcal{D}$ *or* $(\exists x', y')[\langle x,x'\rangle, \langle y',y\rangle \in \mathcal{P}$ *and* $\langle x,y'\rangle, \langle x',y\rangle \in \mathcal{D}]$,

T8 $\langle x,y\rangle \in \mathcal{D}$ *or* $\langle y,x\rangle \in \mathcal{D}$ *or* $\langle x,y\rangle \in \mathcal{L}$ *or* $\langle y,x\rangle \in \mathcal{L}$,

T9 $\langle x,y\rangle \in \mathcal{L} \Rightarrow \langle x,y\rangle \notin \mathcal{D}$, $\langle y,x\rangle \notin \mathcal{D}$, *and* $\langle y,x\rangle \notin \mathcal{L}$,

T10 $\langle x,y\rangle \in \mathcal{L}$ *and* $\langle x,w\rangle, \langle y,z\rangle \in \mathcal{D} \Rightarrow \langle w,z\rangle \in \mathcal{L}$,

T11 $\langle x,y\rangle, \langle y,z\rangle \in \mathcal{L} \Rightarrow \langle x,z\rangle \in \mathcal{L}$,

And meeting the additional condition:

for every $x \in \mathcal{U}$ *the sets* $\mathbf{B}_x = \{y \mid \langle y,x\rangle \in \mathcal{D}\}$ *and* $\mathbf{L}_x = \{y \mid (\exists z)[\langle z,y\rangle, \langle z,x\rangle \in \mathcal{P}$ *and* $\langle y,x\rangle \in \mathcal{L}]\}$ *are finite.*

Note that $\mathcal{I}$ is not required to be either one-to-one or onto. Thus trees for L_K may contain both individuals that are not named by constants and those that are named by more than one constant.

Conditions ***T2***–***T4*** require domination to be reflexive, transitive and anti-symmetric. Thus every node dominates itself. We sometimes use **proper domination** ($\triangleleft^+$) to mean a domination by a path of length greater than zero. Therefore, for $x, y \in \mathcal{U}$ we will say that x properly dominates y if $\langle x,y\rangle \in \mathcal{D}$ and $\langle y,x\rangle \notin \mathcal{D}$. ***T5*** through ***T7*** relate the parent and domination relations. ***T6***, for instance, requires that if x is the parent of y then there is no other node in the path from x to y. The domination and precedence relations are related via ***T8***–***T10***. For instance, ***T8*** requires every pair of nodes to be related either by domination or left-of. ***T9*** requires the relations to be mutually exclusive. Asymmetry and transitivity of left-of are required by ***T9*** and ***T11***. The condition on $\mathbf{B}_x$ and $\mathbf{L}_x$ requires that the length of path from the root to any node be finite, and that each node have finitely many left siblings.

At least one tree (that consisting of only a root node) satisfies all of these conditions simultaneously. Thus they are consistent. It is not hard to exhibit a model for each condition in which that condition fails while all of the others hold. Thus the conditions are independent. It is straightforward to relate this definition of trees to other, perhaps more familiar, ones. A translation between these structures and tree structures defined in terms of tree domains, for instance, is provided in (Rogers & Vijay-Shanker, 1993).

Satisfaction is simply classical first-order satisfaction for unquantified sentences. If T is a tree for a language L_K then $\mathbf{Th}(T)$ is the set of sentences satisfied by T. $\mathbf{Th}(T)$ is necessarily a subset of L_K. If $\mathcal{T}$ is a set of trees, then $\mathbf{Th}(\mathcal{T})$ is the set of sentences satisfied by all $T \in \mathcal{T}$.

If Φ is a set of sentences, $\mathbf{Mod}(\Phi)$ is the set of *trees* which satisfy Φ. Thus, given a description Φ, the set of trees which meet the description is $\mathbf{Mod}(\Phi)$. If Φ is satisfiable then $\mathbf{Mod}(\Phi)$ includes trees for every language extending L_Φ. If Φ is a set of sentences, then $\mathbf{Cn}(\Phi)$ is $\mathbf{Th}(\mathbf{Mod}(\Phi))$. Since $\mathbf{Mod}(\Phi)$ is restricted to trees and the language is restricted to unquantified sentences, $\mathbf{Cn}(\Phi)$ is the set of quantifier-free sentences which are consequences of Φ with respect to trees.

Our mechanisms are, for the most part, purely syntactic, which is to say that they are concerned with $\mathbf{Th}(T)$ rather than the tree T itself. Some care is required since our language is quantifier-free and thus individuals in T which are unnamed by constants are not reflected in $\mathbf{Th}(T)$.[3] This is appropriate, we want the relationships between individuals in a model of a description which are named in the description to be fixed by that description, but we would like to ignore all individuals which are not named in the description. This means that the class of models we are interested in depends on the description. The **referents** of a description Φ are those trees *for* L_Φ among $\mathbf{Mod}(\Phi)$, that is, trees which satisfy Φ and interpret just those constants which occur in Φ.

3 Minimal Referents

We are looking for a syntactic mechanism which will resolve arbitrary descriptions into an equivalent set of quasi-trees. The foundation of our definition of quasi-trees is the relation between a tree and the trees which can be derived from it by adjunction. This gives the partial-order on trees on which we based the notion of minimal referent. As our mechanism is syntactic, operating on sets of formulae, it is founded on a syntactic equivalent to that relation, that is, a relation between theories of trees which captures the "derivable-by-adjunction" relationship between those trees. We define such a relation here and use it to develop a syntactic condition on the consequences of a description which is true iff the description has a unique model which is minimal with respect to that relation—a unique minimal referent. The mechanism, then, operates by splitting a given description into an equivalent set of descriptions, each of which satisfies that condition.

Just as we are interested in operations on descriptions of trees which generalize the adjunction operation, the syntactic relation we employ captures a generalization of the derivable-by-adjunction relation. We consider two trees related if one can be derived from the other by insertion of new nodes into its branches. We ignore all issues having to do with restrictions on the nodes which can be inserted or where they can be inserted. These are issues for the applications and are beyond the scope of this work.

Consider a tree T_1 (e.g., α_1 of Fig. 1) and a tree T_2 obtained by adjoining in T_1 (e.g., γ_1). Each node in T_1 has a distinct corresponding node in T_2 with the exception of the node at which the adjunction occurs. This node might be regarded as corresponding to any or all the nodes in T_2 on the path between that portion which corresponds to the part of T_1 which is "above" the node of adjunction and that portion that corresponds to the part of T_1 which is dominated by that node. One expects the interpretation of constants in T_2 to respect this natural correspondence. It is clear that any precedence relation which holds between two nodes in T_1 will hold between the corresponding nodes in T_2. Further, if two nodes are related by proper domination in T_1 their corresponding nodes will be similarly related in T_2, albeit possibly by a longer path. Thus any constants which name distinct nodes in T_1 must name distinct nodes in T_2 as well, since these

[3] One consequence of this is that until we actually need to construct a tree our mechanisms are oblivious to the conditions $T1$ and $T7$. Although this complicates that construction, it has no other effect.

nodes must be related by either left-of or proper domination. But constants that name the node in T_1 at which the adjunction occurs might name any of the corresponding nodes in T_2. In the example, for instance, vp_1 and vp_2 name the same node in α_1 but distinct nodes in γ_1.

Thus, the relation of being derivable by adjunction incorporates two principles:

1 Adjunction does not affect precedence relations between the nodes in T_1,

2 Constants that name the same node in T_1 may name different nodes in T_2 but those which name distinct nodes in T_1 never name the same node in T_2.

Formally, we have that T_1 and T_2 are trees for L_Φ among $\mathbf{Mod}(\Phi)$. The first principle simply requires that the trees must agree on their interpretation of $\prec$ for all nodes named by constants which occur in Φ. The second comes down to a requirement that proper-domination must be preserved for those nodes, while domination need not be. The relation depends on Φ in the same way that the class of models which count as referents does.

Definition 4 *Let* $T_1 \leq_\Phi T_2$ *iff* $(\forall t, u \in \mathcal{T}(\Phi))$:

1 $T_1 \models t \prec u \Leftrightarrow T_2 \models t \prec u$, *and*

2 $T_1 \models \neg t \triangleleft^* u \Rightarrow T_2 \models \neg t \triangleleft^* u$.

Let $T_1 \doteq_\Phi T_2$ *if* $T_1 \leq_\Phi T_2$ *and* $T_2 \leq_\Phi T_1$.

To see that condition 2 assures that proper domination is preserved for these nodes without requiring that domination be preserved, suppose that $T_1 \models t \triangleleft^* u$. Then, by condition 1 and property ***T*8**, either $T_2 \models t \triangleleft^* u$ or $T_2 \models u \triangleleft^* t$. If the relationship between t and u in T_1 is proper domination, then $T_1 \models \neg u \triangleleft^* t$ and, by 2, $T_2 \models \neg u \triangleleft^* t$ as well. On the other hand, if the domination relationship between t and u is not proper, then it must be reflexive, i.e. $T_1 \models t \triangleleft^* u, u \triangleleft^* t$, but $T_1 \models u \triangleleft^* t$ only implies $T_2 \models t \triangleleft^* u$ or $T_2 \models u \triangleleft^* t$ just as $T_1 \models t \triangleleft^* u$ does. Thus T_2 is free to model $\neg t \triangleleft^* u$ or $\neg u \triangleleft^* t$ as long as it does not model both.

From this definition we have:

Proposition 1 *Let* $\alpha \in \{t \prec u, t \not\prec u, \neg t \triangleleft^* u \mid t, u \in \mathcal{T}(\Phi)\}$. *For all* $T_1, T_2 \in \mathbf{Mod}(\Phi)$ *such that* $T_1 \leq_\Phi T_2$, *if* $T_1 \models \alpha$ *then* $T_2 \models \alpha$.

The relation $\leq_\Phi$ is a preorder; to get a partial order we take the quotient with respect to $\doteq_\Phi$. A quasi-tree in this context, is a description Φ for which there is a unique minimum referent modulo $\doteq_\Phi$. Henceforth, when we refer to a minimal referent of a description Φ, minimality should be understood to be modulo $\doteq_\Phi$.

Definition 5 *Let the set of* minimal referents *of a set of formulae* Φ *be the set of minima wrt* $\leq_\Phi$ *among trees for* L_Φ *in* $\mathbf{Mod}(\Phi)$. *We refer to this set as* $\boldsymbol{\mu}\Phi$*:*

$$\boldsymbol{\mu}\Phi = \{T \models \Phi \mid T \textit{ is a tree for } L_\Phi \textit{ and } (\forall T' \in \mathbf{Mod}(\Phi))\,[T' \leq_\Phi T \Rightarrow T \leq_\Phi T']\}.$$

Let $[T]_\Phi = \{T' \mid T \leq_\Phi T', T' \leq_\Phi T\}$.

Quasi-trees, then, are descriptions Φ which have unique minimum referents up to equivalence wrt $\leq_\Phi$.

Definition 6 *A* quasi-tree *is any description* Φ *for which* $\mathbf{card}(\{[T]_\Phi \mid T \in \boldsymbol{\mu}\Phi\}) = 1$.

If a consistent description Φ is not a quasi-tree then there will be at least two trees in $\boldsymbol{\mu}\Phi$ which are incomparable wrt $\leq_{\Phi}$. Since $\leq_{\Phi}$ is defined in terms of the formulae satisfied by the trees, for every pair of incomparable trees in $\boldsymbol{\mu}\Phi$ there will be a pair of formulae which witness that fact. These formulae will be of the form $t \prec u$ or $t \triangleleft^{*} u$ for some $t, u \in \mathcal{T}(\Phi)$. Say $T_1 \not\leq_{\Phi} T_2$. Then, by Definition 4, T_1 satisfies one of $t \prec u$, $t \not\prec u$, or $\neg t \triangleleft^{*} u$, for some t, u, while T_2 does not. A similar formula will witness the fact that $T_2 \not\leq T_1$.

These witnessing formulae will be reflected in the consequences of Φ. By Proposition 1 every tree in $\mathbf{Mod}(\Phi)$ will satisfy at least one of them (since every tree is greater than or equal to at least one of the minimal trees), but none of them will be satisfied by all of the trees in $\mathbf{Mod}(\Phi)$ (since there are at least two incomparable trees). Thus none of the witnessing formulae will be in $\mathbf{Cn}(\Phi)$ but their disjunction will be. This lets us characterize when Φ is a quasi-tree purely on the syntactic structure of $\mathbf{Cn}(\Phi)$. We get this via two propositions.

Proposition 2 *Suppose* $\mathbf{card}(\{[T]_{\Phi} \mid T \in \boldsymbol{\mu}\Phi\}) = n > 1$. *Then there exists some* $A_{\Phi} = \{\alpha_1, \alpha_2, \ldots\}$, *where* $\alpha_i \in \{t \not\prec u, \neg t \triangleleft^{*} u \mid t, u \in \mathcal{T}(\Phi)\}$ *such that* $\mathbf{card}(A_{\Phi}) \geq 2$, $\bigvee A_{\Phi} \in \mathbf{Cn}(\Phi)$, *and for all* $\alpha_i \in A_{\Phi}$, $\alpha_i \notin \mathbf{Cn}(\Phi)$.

The proof just formalizes the preceding discussion, with A_{Φ} being the set of witnessing formulae. A little added work is needed to show that witnessing formulae of the form $t \not\prec u$ or $\neg t \triangleleft^{*} u$ can be found in all cases. The details of this and other proofs omitted here can be found in the technical report (Rogers & Vijay-Shanker, 1993).

Proposition 3 *Suppose* Φ *is satisfiable over trees and there exists some* $A_{\Phi} = \{\alpha_1, \alpha_2, \ldots\}$, *where* $\alpha_i \in \{t \not\prec u, \neg t \triangleleft^{*} u \mid t, u \in \mathcal{T}(\Phi)\}$ *such that* $\mathbf{card}(A_{\Phi}) = n \geq 2$, $\bigvee A_{\Phi} \in \mathbf{Cn}(\Phi)$, *and for all* $\alpha_i \in A_{\Phi}$, $\alpha_i \notin \mathbf{Cn}(\Phi)$. *Then* $\mathbf{card}(\{[T]_{\Phi} \mid T \in \boldsymbol{\mu}\Phi\}) > 1$.

Here the hypothesis implies that for all $i \leq n$ there is some tree T_i such that $T_i \models \Phi \cup \{\neg\alpha_i\}$. Then if T is a lower bound of $\mathbf{Mod}(\Phi)$ wrt $\leq_{\Phi}$, by Proposition 1 it necessarily fails to satisfy any of the α_i. But the disjunction of the α_i is a consequence of Φ. It follows, then, that $T \notin \mathbf{Mod}(\Phi)$. Since no lower bound wrt $\leq_{\Phi}$ of $\mathbf{Mod}(\Phi)$ exists in $\mathbf{Mod}(\Phi)$, there must be more than one class of trees among $\mathbf{Mod}(\Phi)$ which are minimal wrt $\leq_{\Phi}$.

Thus we can characterize quasi-trees by the presence of a clause with the specific form A_{Φ} in their consequences:

Corollary 1 *A set of formulae* Φ *is a quasi-tree iff it is satisfiable over trees and* $\mathbf{Cn}(\Phi)$ *contains no such* $\bigvee A_{\Phi}$.

This characterization is useful because we have, at the heart of our mechanism, a syntactic method of constructing a finite set of formulae from any Φ which determines $\mathbf{Cn}(\Phi)$ in such a way that a formula of form $\bigvee A_{\Phi}$ occurs in $\mathbf{Cn}(\Phi)$ iff such a formula occurs in that finite set. The presence of such a formula in that set, then, decides whether a given description has multiple minimal referents and thus whether, in the scheme of (Vijay-Shanker & Schabes, 1992), a given description of the elementary structures to be associated with lexical items in the grammar is an acceptable description.

4 Mechanism

The primary mechanism we provide resolves arbitrary (finite) descriptions into a set of quasi-trees whose minimum referents coincide with the minimal referents of the description. The description is unsatisfiable if and only if this set is empty. In many applications, particularly if the description is to be further refined, resolution into this set of quasi-trees suffices. Applications which must produce trees require an additional mechanism for extracting the minimum referent of the quasi-trees. This is provided in the next section.

The mechanism to resolve a given description into an equivalent set of quasi-trees is an extension of an ordinary propositional resolution proof system. It is useful here to regard resolution as a generalization of Modus Ponens which we can depict as:

$$\phi \rightarrow \Phi, \phi \vee \Psi \vdash \Phi \vee \Psi.$$

Regarded in this way, a resolution system extends a set of clauses—sets of literals, interpreted disjunctively—by adding clauses which are consequences of that set. Refutational completeness requires that whenever the set is unsatisfiable, and thus has the empty clause as a consequence, the empty clause will eventually be added. We show below that, with the addition of some structural rules to eliminate tautologies and handle absorption, the resolution mechanism in fact produces the set of *minimal clausal consequences* of the original set of clauses. The set of minimal clausal consequences is defined precisely in section 6. In essence it is the set of clauses which are consequences of the original set and which are minimal in the sense that no clause in the set is a consequence of any combination of the others. This set is finite whenever the set it is derived from is finite.

In order to handle *arbitrary* finite descriptions we extend the typical resolution mechanism to handle sets of *generalized clauses*—sets of arbitrary formulae, interpreted disjunctively. This is accomplished with a set of inference rules which expand generalized clauses to an equivalent set of *proper clauses*—literals only. Since the set of clauses is interpreted conjunctively, this has the effect of converting the set of arbitrary formulae to Conjunctive Normal Form. This extension is not uncommon, see for instance (Fitting, 1990).

In the previous section we established that a set of formulae Φ is a quasi-tree iff its consequences do not contain any disjunction of the form $\bigvee A_\Phi$, where $A_\Phi \subseteq \{t \not\approx u, \neg t \lhd^* u \mid t, u \in \mathcal{T}(\Phi)\}$ and no member of A_Φ is itself a consequence of Φ. If any such disjunction occurs in the consequences of Φ some subset of A_Φ will occur in the minimal clausal consequences of Φ. Thus we can assure that a set of formulae we derive is a quasi-tree by eliminating such A_Φ. We do this by adding a layer of disjunction to the configurations on which the mechanism operates. A typical resolution theorem prover operates on a single set of clauses, interpreted as the conjunction of those clauses. Here, when a clause of the form A_Φ is encountered, the configuration is split into a set of branches, one for each disjunct in the A_Φ. Each branch duplicates the set of clauses in which the A_Φ occurred with the exception that the A_Φ is replaced by one of its disjuncts. Thus, in the limit, no branch will contain any such clause. Our mechanism, then, operates on a configuration which is a set of sets of generalized clauses:

Definition 7

- *A* **(generalized) clause** $[\Phi]$ *is a set of formulae that is interpreted disjunctively:*

$$T \models [\Phi] \Leftrightarrow T \models \phi \textit{ for some formula } \phi \in \Phi$$

Generally we use ϕ and ψ to denote formulae and Φ and Ψ for sets of formulae. Clauses will be denoted $[\Phi, \phi_1, \ldots]$, where, as usual, $\Phi, \phi_1, \ldots$ is to be read as $\Phi \cup \{\phi_1, \ldots\}$. When the context is clear we will omit the brackets when referring to clauses.

- *A* **branch** $[\![\Gamma]\!]$ *is a set of (generalized) clauses that is interpreted conjunctively:*

$$T \models [\![\Gamma]\!] \Leftrightarrow T \models \Phi \textit{ for all clauses } \Phi \in \Gamma$$

We use Γ and Δ to denote sets of clauses. Branches will be denoted $[\![\Gamma, \Phi_1, \ldots]\!]$, where, again, the brackets will be omitted if the context is clear.

- *A* **configuration** Σ *is a set of branches that is interpreted disjunctively:*

$$T \models \Sigma \Leftrightarrow T \models \Gamma \textit{ for some branch } \Gamma \in \Sigma$$

We use Σ and Ξ to denote sets of branches.

We extend the domain of **Mod** to clauses, branches, and configurations on the basis of the definition of satisfaction for these sets.

The mechanism consists of a set of inference rules of the form: $\frac{\Sigma}{\Xi}$ where Σ and Ξ are schemata of configurations. Such a rule **applies** to a configuration if it is an instance of Σ. The result of its application is the corresponding instance of Ξ. A **derivation** from a configuration Σ_0 is a sequence of configurations $\langle \Sigma_0, \Sigma_1, \ldots \rangle$ where each Σ_{i+1} is the result of application of a rule to Σ_i. It is **acyclic** if no configuration occurs in it more than once. Clearly, every derivation has as a subsequence an acyclic derivation which has the same final configuration. A **derivation from a set of formulae** Φ is a derivation from $\{[\![[\Phi]]\!]\}$.

We will frequently be somewhat informal in referring to instances of rules and results of derivations. When we say an instance of a rule occurs in a configuration or branch we mean that the configuration or the configuration containing just the branch is an instance of the antecedent of the rule. When we say that a clause or branch derives from a configuration, we mean that there is a derivation from that configuration of a configuration containing the clause or branch.

4.1 Inference Rules

4.1.1 Logical Rules

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

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

$$\frac{\Sigma, \llbracket \Gamma, [\alpha_1, \Phi] \rrbracket}{\Sigma, \llbracket \Gamma, [\alpha_1] \rrbracket, \llbracket \Gamma, [\Phi] \rrbracket} \; \boldsymbol{exp}$$
$$[\alpha_1, \Phi] \subseteq \{\neg t \triangleleft^* u, t \not\approx u \mid t, u \text{ any constants}\}$$
$$\Phi \neq \emptyset$$

The first five logical rules expand improper clauses into CNF. The ***exp*** (expand) rule introduces a new branch for each of the disjuncts of a clause of the form A_Φ. It is this rule that causes us to deal with configurations rather than just a single branch as a standard resolution system would.

4.1.2 Structural Rules

$$\frac{\Sigma, \llbracket \Gamma, [\Phi, \phi, \neg\phi] \rrbracket}{\Sigma, \llbracket \Gamma \rrbracket} \; \boldsymbol{taut} \qquad \frac{\Sigma, \llbracket \Gamma, [\Phi], [\Phi, \Psi] \rrbracket}{\Sigma, \llbracket \Gamma, [\Phi] \rrbracket} \; \boldsymbol{abs} \qquad \frac{\Sigma, \llbracket \Gamma \rrbracket, \llbracket \Delta \rrbracket}{\Sigma, \llbracket \Delta \rrbracket} \; \boldsymbol{sub}$$
$$(\forall \Phi \in \Delta)(\exists \Psi \in \Gamma)[\Psi \subseteq \Phi]$$

The rules ***taut*** and ***abs*** eliminate tautological clauses and those which are absorbed by another clause in the same branch. The ***sub*** rule eliminates any branch which is subsumed by another, that is, it eliminates any branch Γ for which there is a branch Δ such that $\mathbf{Mod}(\Gamma) \subseteq \mathbf{Mod}(\Delta)$. This is a semantic interpretation of the syntactic rule. While it's easy to see that $\mathbf{Mod}(\Gamma) \subseteq \mathbf{Mod}(\Delta)$ whenever ***sub*** applies to Γ and Δ, we do not actually establish the fact that ***sub*** will eventually apply whenever $\mathbf{Mod}(\Gamma) \subseteq \mathbf{Mod}(\Delta)$ until after we have shown how the branches generated by the mechanism determine the consequences of Φ.

4.1.3 Resolution

$$\frac{\Sigma, \llbracket \Gamma, [\Phi, \phi], [\Psi, \neg\phi] \rrbracket}{\Sigma, \llbracket \Gamma, [\Phi, \phi], [\Psi, \neg\phi], [\Phi, \Psi] \rrbracket} \; \boldsymbol{res}$$

This is just ordinary resolution, applied to some branch of the configuration.

4.1.4 Tree Axiom Rules

The tree axiom rules essentially introduce instances of a (partial) set of the conditions of the definition of trees in section 2.2. The numbering is intended to associate them with the corresponding conditions. In some cases the correspondence is rather loose. As a result of these rules the mechanism derives consequences with respect to trees. There are no axiom rules for conditions $\boldsymbol{T1}$ or $\boldsymbol{T7}$, since these have no quantifier-free consequences.

For $t, u, v, w \in \mathcal{T}(\Gamma)$:

$$\frac{\Sigma, \llbracket \Gamma \rrbracket}{\Sigma, \llbracket \Gamma, [t \triangleleft^* t] \rrbracket} \; \boldsymbol{2a} \qquad \frac{\Sigma, \llbracket \Gamma \rrbracket}{\Sigma, \llbracket \Gamma, [t \not\approx u, t \triangleleft^* u], [t \not\approx u, u \triangleleft^* t] \rrbracket} \; \boldsymbol{2b}$$

$$\frac{\Sigma, \llbracket \Gamma \rrbracket}{\Sigma, \llbracket \Gamma, [\neg t \triangleleft^* u, \neg u \triangleleft^* t, t \approx u] \rrbracket} \; \boldsymbol{3} \qquad \frac{\Sigma, \llbracket \Gamma \rrbracket}{\Sigma, \llbracket \Gamma, [\neg t \triangleleft^* u, \neg u \triangleleft^* v, t \triangleleft^* v] \rrbracket} \; \boldsymbol{4}$$

$$\frac{\Sigma, [\![\Gamma]\!]}{\Sigma, [\![\Gamma, [\neg t \lhd u, t \lhd^* u], [\neg t \lhd u, \neg u \lhd^* t]]\!]} \; \mathbf{5a}$$

$$\frac{\Sigma, [\![\Gamma]\!]}{\Sigma, [\![\Gamma, [\neg t \lhd u, \neg t \lhd^* w, \neg w \lhd^* t, \neg u \lhd^* v, \neg v \lhd^* u, w \lhd v]]\!]} \; \mathbf{5b}$$

$$\frac{\Sigma, [\![\Gamma]\!]}{\Sigma, [\![\Gamma, [\neg t \lhd u, \neg t \lhd^* v, v \lhd^* t, \neg v \lhd^* u, u \lhd^* v]]\!]} \; \mathbf{6} \qquad \frac{\Sigma, [\![\Gamma]\!]}{\Sigma, [\![\Gamma, [t \lhd^* u, u \lhd^* t, t \prec u, u \prec t]]\!]} \; \mathbf{8}$$

$$\frac{\Sigma, [\![\Gamma]\!]}{\Sigma, [\![\Gamma, [t \not\prec u, \neg t \lhd^* u], [t \not\prec u, \neg u \lhd^* t], [t \not\prec u, u \not\prec t]]\!]} \; \mathbf{9}$$

$$\frac{\Sigma, [\![\Gamma]\!]}{\Sigma, [\![\Gamma, [t \not\prec u, \neg t \lhd^* w, \neg u \lhd^* v, w \prec v]]\!]} \; \mathbf{10} \qquad \frac{\Sigma, [\![\Gamma]\!]}{\Sigma, [\![\Gamma, [t \not\prec u, u \not\prec v, t \prec v]]\!]} \; \mathbf{11}$$

4.2 Derivations

The fundamental property of the inference system is that **Mod** is invariant across all configurations of a derivation. (Which is easily verified.)

Proposition 4 (Model Invariance) *For all derivations* $\langle \Sigma_0, \Sigma_1, \ldots \rangle$*, for all* i*,* $\mathbf{Mod}(\Sigma_i) = \mathbf{Mod}(\Sigma_0)$*.*

A branch is **closed** if it contains an empty clause. A configuration is closed if each of its branches is closed. A branch or configuration which is not closed is **open**. By ***abs*** and ***sub*** every closed branch or configuration derives a branch or configuration consisting of only $[\ ]$ or $[\![[\ ]]\!]$, respectively.

Proposition 5 (Refutational Soundness) *If* Φ *is a set of formula and there is a closed derivation from* Φ*, then* Φ *is unsatisfiable.*

Clearly, a closed configuration has no model. A closed derivation from a set of formulae constitutes a refutation of the satisfiability of that set. Refutational soundness requires that such a derivation can be obtained only in the case that the set of formulae has no model. This follows directly from model invariance.

The mechanism terminates when no rule applies productively, i.e., when no logical or structural rule applies and:

1. For all instances of ***res*** that apply, either the resolvent is an instance of ***taut*** or some subset of the resolvent already occurs in the branch to which it applies, i.e., if the instance is $\Sigma, [\![\Gamma, [\Phi, \phi], [\Psi, \neg\phi]]\!]$ then there is some clause $\Upsilon \in \Gamma$ such that $\Upsilon \subseteq [\Phi, \Psi]$;
2. For every branch Γ, if $\Sigma, [\![\Gamma, \Phi_1, \Phi_2, \ldots]\!]$ is an instance of a tree axiom rule in L_Γ, then for each of the Φ_i there is some $\Psi_i \in \Gamma$ such that $\Psi_i \subseteq \Phi_i$.

These conditions require that any clause which may be introduced by ***res*** or by one of the tree axiom rules can be immediately eliminated by either ***taut*** or ***abs***.

Definition 8 *A configuration that meets the above conditions is* **saturated***. A branch is saturated if the configuration consisting of just that branch is saturated. A derivation is saturated if its final configuration is saturated. A configuration* Σ *is a* **saturated expansion** *of a set of formulae* Φ *iff* Σ *is open, saturated and the final configuration of a derivation from* Φ*.*

Note that the canonical closed configuration $[\![[\ ]]\!]$ is saturated.

In the next section we will establish that whenever a saturated expansion of a set of formulae can be derived then it is satisfiable. We show further, in section 6, that each branch of that saturated expansion is a quasi-tree and, in section 7, that the minimum referents of those branches correspond to the minimal referents of the original set of formulae. Thus the goal of our primary mechanism is to derive a saturated configuration. The following lemma establishes that such a saturated configuration can always be derived. Coupling that with the fact that any configuration which can be derived can be derived acyclically we get weak termination.

Lemma 1 *Every derivation can be extended to one in which the final configuration is saturated.*

This can be proved by exhibiting an ordering of the inference rules which will produce a saturated configuration from any configuration in finitely many steps. Such an ordering is not hard to obtain. The only complication is that the rules ***res*** and ***exp*** can each introduce instances of the other. Thus their application must be alternated, but it can be shown that finitely many alternations suffice. Details are in (Rogers & Vijay-Shanker, 1993).

Proposition 6 (Weak Termination) *For every finite set of formulae Φ there is a finite saturated derivation from Φ.*

Proof: By the previous lemma, every configuration can be extended to one that is saturated. Since for every derivation there is an acyclic derivation with the same final configuration, for every initial configuration there is an acyclic saturated derivation. Configurations are just sets of sets of sets of formulae where every formula in any clause of any branch of a derived configuration is either a subformula of one occurring in the initial configuration or a literal formula over constants occurring in the initial configuration. (These latter are introduced by the tree axiom rules while the former are introduced by the other rules.) Thus for every finite initial configuration there can only be finitely many distinct configurations occurring in any derivation and, as a result, every acyclic derivation from a finite configuration has finite length. □

By Propositions 6 and 5 every satisfiable set of formulae Φ will have some saturated expansion $\{\Gamma_1, \Gamma_2, \ldots\}$ and, by Proposition 4, $\mathbf{Mod}(\Phi) = \bigcup_i \mathbf{Mod}(\Gamma_i)$. Each branch of the expansion is ultimately introduced by the ***exp*** rule. Thus, for each Γ_i there is a unique set $\Lambda_{\Gamma_i} = \{\alpha_i \mid \alpha_i \text{ is introduced by } \boldsymbol{exp} \text{ into branch } \Gamma_i\}$, although in the case of a single Γ it may well be empty. The set Λ_{Γ_i} determines $\mathbf{Mod}(\Gamma_i)$ in the following way:

Proposition 7 *Suppose Γ_i is a branch of a saturated expansion of Φ. Then $\mathbf{Mod}(\Gamma_i) = \mathbf{Mod}(\Phi \cup \Lambda_{\Gamma_i})$.*

Proof: Suppose a branch $[\![[\alpha_i], \Gamma]\!]$ is derived from a branch $[\![[\alpha_i, \Phi], \Gamma]\!]$ by a single application of ***exp***. $\mathbf{Mod}([\![[\alpha_i], \Gamma]\!]) = \mathbf{Mod}([\![[\alpha_i], [\alpha_i, \Phi], \Gamma]\!])$ by absorption. Since branches are introduced only by ***exp***, the result follows by induction on the length of the derivation. □

If there is but a single Γ (and thus $\Lambda_\Gamma = \emptyset$) then this just restates model invariance. The sets $\mathbf{Mod}(\Gamma_i)$ are not in general disjoint, but by the semantic interpretation of ***sub*** none is a subset of any other.

5 Satisfiability of Open Saturated Branches

Weak termination assures that every description can be expanded to either a closed configuration or a saturated expansion. Refutational soundness assures that if a closed configuration can be derived then the description is unsatisfiable. By the contrapositive, if a description is satisfiable it will derive no closed configuration, and thus must have a saturated expansion. Ultimately we claim that the branches of a saturated expansion of a description form a set of quasi-trees which characterize the set of models of the description in a strong way: the union of the sets of models of the quasi-trees coincides with the set of models of the description and, furthermore, the set of minimum referents of the quasi-trees coincides with the set of minimal referents of the description. The first of these criteria requires that $\mathbf{Mod}(\Sigma) = \mathbf{Mod}(\Phi)$ whenever Σ is a saturated expansion of Φ. This is just model invariance. The second criterion has two components: the branches of Σ must be quasi-trees and the minimum referents of those branches must be the minimal referents of Φ. We will show that each branch of a saturated expansion is a quasi-tree in this section. In section 7 we will show that the minimum referents of these branches together give the minimal referents of the given description.

Employing our characterization of quasi-trees from section 3, Corollary 1, a branch Γ of a saturated expansion is a quasi-tree iff it is satisfiable and the consequences of Γ contain no clause of the form A_Γ. In this section we first establish that every branch of a saturated expansion is satisfiable. As a corollary we get *refutational completeness*, that is, whenever a description has no closed expansion it is satisfiable. This follows by model invariance—any tree which satisfies some branch of a saturated expansion of Φ must satisfy Φ as well.

The approach of this section follows a strategy which is frequently encountered in completeness proofs: given a branch Γ of a saturated expansion of Φ, we find a *completion* $\overline{\Gamma}$ such that $\overline{\Gamma} \models \Gamma$ and $\overline{\Gamma}$ is atomically complete wrt $\mathcal{T}(\Phi)$, that is, it contains either $[t \mathrm{~R~} u]$ or $[\neg t \mathrm{~R~} u]$ for all predicates R and $t, u \in \mathcal{T}(\Gamma)$. Then Γ is satisfiable if $\mathbf{Cn}(\overline{\Gamma})$ is the theory of some tree. We proceed by constructing such a tree T_Γ. Subsequently, in section 7, we show that the tree T_Γ is minimum wrt $\leq_\Gamma$ among $\mathbf{Mod}(\Gamma)$, in other words, it is the minimum referent (modulo $\doteq_\Gamma$) of Γ. Thus the constructions of this section provide the mechanism for constructing the minimum trees from the quasi-trees derived by our primary mechanism.

We begin by noting that the branches of a saturated expansion are already complete for $\prec$.

Lemma 2 *Suppose* Γ *is a branch of a saturated expansion of some set of formulae* Φ*. Then for all* $t, u \in \mathcal{T}(\Phi)$ *either* $[t \prec u] \in \Gamma$ *or* $[t \not\prec u] \in \Gamma$.

Proof: Since Γ is saturated it must include some subset of every instance of the rule **9**. Since it is open none of these subsets is empty. Thus, for every $T, u \in \mathcal{T}(\Phi)$ some non-empty subset of each of

$$[t \not\prec u, \neg t \triangleleft^* u], [t \not\prec u, \neg u \triangleleft^* t], [t \not\prec u, u \not\prec t]$$

occurs in Γ. These are each instances of ***exp***, and since Γ is saturated, only singleton subsets of these can occur in Γ. Thus

$$\begin{array}{rlcl} & [t \not\prec u] \in \Gamma & \text{or} & [\neg t \triangleleft^* u] \in \Gamma \\ \text{and} & [t \not\prec u] \in \Gamma & \text{or} & [\neg u \triangleleft^* t] \in \Gamma \\ \text{and} & [t \not\prec u] \in \Gamma & \text{or} & [u \not\prec t] \in \Gamma \end{array}$$

Suppose $[t \not\prec u] \notin \Gamma$. Then $[\neg t \triangleleft^* u], [\neg u \triangleleft^* t], [u \not\prec t] \in \Gamma$. By **8** and saturation of Γ some subset of $[t \triangleleft^* u, u \triangleleft^* t, t \prec u, u \prec t]$ is in Γ. But each member of this clause except $t \prec u$ can be eliminated by ***res*** and ***abs***, and thus, by saturation, $[t \prec u] \in \Gamma$. □

An important consequence of this lemma is that $\mathcal{T}(\Gamma) = \mathcal{T}(\Phi)$ for all branches Γ of a saturated expansion of Φ. This follows from the completeness of the Γ for left-of with respect to $\mathcal{T}(\Phi)$ and the fact that the mechanism never introduces new constants. Thus we can consider $\leq_\Gamma$ and $\leq_\Phi$ interchangeably.

5.1 Constructing completions of open saturated branches

The existence of an atomically complete extension of an open saturated branch is closely related to Lindenbaum's lemma. Following the usual approach to the proof of that lemma, we construct the completion of the branch by adding, one at a time, literal clauses to the branch in a way that preserves its consistency.

For any branch Γ, let $\mathrm{res}([\phi], \Gamma)$ be the result of first applying ***res*** to each instance $[\phi], [\neg\phi, \Phi]$ in $\Gamma \cup \{[\phi]\}$, and then applying ***abs*** to every instance in the result.

Suppose Γ is a branch of a saturated expansion of some set of formulae Φ. We wish to complete Γ wrt $\mathcal{T}(\Phi)(= \mathcal{T}(\Gamma))$ for the two remaining relation symbols $\triangleleft^*$ and $\triangleleft$. To do this we construct $\overline{\Gamma}$, a *completion* of Γ, by considering every pair of constants t, u in $\mathcal{T}(\Gamma)$ in turn and adding $[t \mathrel{\mathrm{R}} u]$, where $\mathrm{R} \in \{\triangleleft^*, \triangleleft\}$, iff it can be added without losing consistency:

Let $k = \mathbf{card}(\mathcal{T}(\Gamma) \times \mathcal{T}(\Gamma))$.
Let $\langle t_i, u_i \rangle$, for $1 \leq i \leq k$, enumerate pairs of terms in $\mathcal{T}(\Gamma)$.
Let $S_0 = \Gamma$.
For $1 \leq i \leq k$:

$$\text{Let } S_i = \begin{cases} \mathrm{res}([t_i \triangleleft^* u_i], S_{i-1}) & \text{if } [\neg t_i \triangleleft^* u_i] \notin S_{i-1}, \\ S_{i-1} & \text{otherwise .} \end{cases}$$

For $k+1 \leq i \leq 2k$:

$$\text{Let } S_i = \begin{cases} \mathrm{res}([t_{i-k} \triangleleft u_{i-k}], S_{i-1}) & \text{if } [\neg t_{i-k} \triangleleft u_{i-k}] \notin S_{i-1}, \\ S_{i-1} & \text{otherwise .} \end{cases}$$

Let $\overline{\Gamma} = S_{2k}$. $\overline{\Gamma}$ is a completion of Γ. It depends on the enumeration $\langle t_i, u_i \rangle$ as well as Γ, and is not in general unique.

We claim that $\overline{\Gamma}$ is saturated, open, atomically complete and $\overline{\Gamma} \models \Gamma$.

Lemma 3 *If $\overline{\Gamma}$ is a completion of a branch Γ of a saturated expansion of Φ, then $\overline{\Gamma}$ is saturated, open, atomically complete wrt $\mathcal{T}(\Phi)$, and consists solely of literals.*

This follows from a sequence of lemmas which we do not include here. The full proof is in our technical report (Rogers & Vijay-Shanker, 1993). The basic idea is to show that the result of each step is saturated and open. The first k steps then complete the branch for $\triangleleft^*$, and the second k steps complete it for $\triangleleft$. Atomic completeness of $\overline{\Gamma}$ then follows from completeness of Γ for $\prec$.

That $\overline{\Gamma}$ consists solely of literals follows from its atomic completeness and the fact that it is saturated. Suppose that some non-singleton clause, $[t \triangleleft^* u, \Phi]$ for instance, did occur in $\overline{\Gamma}$. Since $\overline{\Gamma}$ is complete for $\triangleleft^*$ either $[t \triangleleft^* u] \in \overline{\Gamma}$ or $[\neg t \triangleleft^* u] \in \overline{\Gamma}$. In the first case, by ***abs***, $[t \triangleleft^* u, \Phi] \notin \overline{\Gamma}$. In the second, ***res*** will introduce $[\Phi]$ which, again by ***abs***, implies that $[t \triangleleft^* u, \Phi] \notin \overline{\Gamma}$.

Lemma 4 *Suppose $\overline{\Gamma}$ is a completion of Γ a branch of a saturated expansion. Then for all trees $T \models \overline{\Gamma} \Rightarrow T \models \Gamma$.*

This is easily shown by induction on i, the index of the stages of the construction.

5.2 Constructing trees from completed branches

Our objective now is to construct a tree which satisfies $\overline{\Gamma}$. The atomic completeness of $\overline{\Gamma}$ assures that such a tree can be built over the terms of $\overline{\Gamma}$, or rather, since we may have $[t \approx u] \in \overline{\Gamma}$ for distinct $t, u \in \mathcal{T}(\overline{\Gamma})$, over the set of equivalence classes in $\mathcal{T}(\overline{\Gamma})$ induced by the relation $\{\langle t, u\rangle \mid [t \approx u] \in \overline{\Gamma}\}$. This relation is, in fact, an equivalence relation, and is an appropriate interpretation of equality (in the sense that substitution holds) as a consequence of the fact that, since $\overline{\Gamma}$ is both saturated and open, it must include some non-empty subset of each instance of the tree axiom rules in $L_{\overline{\Gamma}}$. The same fact assures that the model we construct satisfies each of ***T2***...***T6*** and ***T8***...***T11***. Our only complication is that the tree may require individuals which are not referred to by any constant in order to satisfy ***T1*** and ***T7***.

Let

$$
\begin{aligned}
\mathcal{R}_\Gamma &= \begin{cases} \{\boldsymbol{r}\} & \text{if } \neg(\exists t)(\forall u \in \mathcal{T}(\Gamma))\,\big[[t \triangleleft^* u] \in \overline{\Gamma}\big] \\ \emptyset & \text{otherwise} \end{cases} \\
\mathcal{S}_\Gamma &= \{\boldsymbol{s}([t]_\approx, [u]_\approx) \mid [t \triangleleft^* u], [\neg u \triangleleft^* t], [\neg t \triangleleft u] \in \overline{\Gamma} \\
&\qquad\qquad \text{and } \neg(\exists v)\,\big[[t \triangleleft^* v], [\neg v \triangleleft^* t], [v \triangleleft^* u], [\neg u \triangleleft^* v] \in \overline{\Gamma}\big]\}
\end{aligned}
$$

$\mathcal{R}_\Gamma$ and $\mathcal{S}_\Gamma$ in essence serve as Skolem functions for the existential conditions ***T1*** and ***T7***. If there is no root among the nodes named in $\overline{\Gamma}$, $\mathcal{R}_\Gamma$ provides one. Similarly if a pair of individuals are related by proper domination and not by parent (e.g., $[t \triangleleft^* u], [\neg u \triangleleft^* t], [\neg t \triangleleft u] \in \overline{\Gamma}$) but no individual named in $\overline{\Gamma}$ separates them, $\mathcal{S}_\Gamma$ provides such an individual. With these we can construct a model of $\overline{\Gamma}$:

Let $T_\Gamma = \langle \mathcal{U}_\Gamma, \mathcal{I}_\Gamma, \mathcal{P}_\Gamma, \mathcal{D}_\Gamma, \mathcal{L}_\Gamma\rangle$ where:

$$
\begin{aligned}
\mathcal{U}_\Gamma &= \mathcal{T}(\Gamma)/\!\approx \cup\, \mathcal{R}_\Gamma \cup \mathcal{S}_\Gamma \\
\mathcal{I}_\Gamma(t) &= [t]_\approx \\
\mathcal{P}_\Gamma &= \{\langle [t]_\approx, [u]_\approx\rangle \mid [t \triangleleft u] \in \overline{\Gamma}\} \\
&\quad \cup \{\langle [t]_\approx, \boldsymbol{s}([t]_\approx, [u]_\approx)\rangle, \langle \boldsymbol{s}([t]_\approx, [u]_\approx), [u]_\approx\rangle \mid \boldsymbol{s}([t]_\approx, [u]_\approx) \in \mathcal{S}_\Gamma\} \\
&\quad \cup \{\langle \boldsymbol{r}, [t]_\approx\rangle \mid \boldsymbol{r} \in \mathcal{R}_\Gamma, t \in \mathcal{T}(\Gamma), \\
&\qquad\qquad \text{and } (\forall u \in \mathcal{T}(\Gamma))\,\big[[u \triangleleft^* t] \in \overline{\Gamma} \Rightarrow [t \triangleleft^* u] \in \overline{\Gamma}\big]\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{D}_\Gamma \;=\;& \{\langle [t]_\approx, [u]_\approx\rangle \mid [t \triangleleft^* u] \in \overline{\Gamma}\} \\
&\cup \{\langle [t]_\approx, \boldsymbol{s}([u]_\approx, [v]_\approx)\rangle \mid \boldsymbol{s}([u]_\approx, [v]_\approx) \in \mathcal{S}_\Gamma, [t \triangleleft^* u] \in \overline{\Gamma}\} \\
&\cup \{\langle \boldsymbol{s}([t]_\approx, [u]_\approx), [v]_\approx\rangle \mid \boldsymbol{s}([t]_\approx, [u]_\approx) \in \mathcal{S}_\Gamma, [u \triangleleft^* v] \in \overline{\Gamma}\} \\
&\cup \{\langle \boldsymbol{s}([t]_\approx, [u]_\approx), \boldsymbol{s}([v]_\approx, [w]_\approx)\rangle \mid \\
&\qquad\qquad \boldsymbol{s}([t]_\approx, [u]_\approx), \boldsymbol{s}([v]_\approx, [w]_\approx) \in \mathcal{S}_\Gamma, [u \triangleleft^* v] \in \overline{\Gamma}\} \\
&\cup \{\langle \boldsymbol{s}([t]_\approx, [u]_\approx), \boldsymbol{s}([t]_\approx, [u]_\approx)\rangle \mid \boldsymbol{s}([t]_\approx, [u]_\approx) \in \mathcal{S}_\Gamma\} \\
&\cup \{\langle \boldsymbol{r}, x\rangle \mid \boldsymbol{r} \in \mathcal{R}_\Gamma, x \in \mathcal{U}_\Gamma\} \\
\mathcal{L}_\Gamma \;=\;& \{\langle [t]_\approx, [u]_\approx\rangle \mid [t \prec u] \in \overline{\Gamma}\} \\
&\cup \{\langle [v]_\approx, \boldsymbol{s}([t]_\approx, [u]_\approx)\rangle \mid \boldsymbol{s}([t]_\approx, [u]_\approx) \in \mathcal{S}_\Gamma, [v \prec u] \in \overline{\Gamma}\} \\
&\cup \{\langle \boldsymbol{s}([t]_\approx, [u]_\approx), [v]_\approx\rangle \mid \boldsymbol{s}([t]_\approx, [u]_\approx) \in \mathcal{S}_\Gamma, [u \prec v] \in \overline{\Gamma}\} \\
&\cup \{\langle \boldsymbol{s}([t]_\approx, [u]_\approx), \boldsymbol{s}([v]_\approx, [w]_\approx)\rangle \mid \\
&\qquad\qquad \boldsymbol{s}([t]_\approx, [u]_\approx), \boldsymbol{s}([v]_\approx, [w]_\approx) \in \mathcal{S}_\Gamma, [u \prec w] \in \overline{\Gamma}\}
\end{aligned}
$$

The presence of $\mathcal{R}_\Gamma$ and $\mathcal{S}_\Gamma$ complicates the definition of T_Γ significantly. Without them $\mathcal{P}_\Gamma$, $\mathcal{D}_\Gamma$, and $\mathcal{L}_\Gamma$ need only the pairs corresponding to literals in $\overline{\Gamma}$. With $\mathcal{S}_\Gamma$ non-empty $\mathcal{P}_\Gamma$ must include pairs for each member of $\mathcal{S}_\Gamma$ and the pair of individuals it separates and with $\mathcal{R}_\Gamma$ non-empty $\mathcal{P}_\Gamma$ must include $\langle \boldsymbol{r}, t\rangle$ for each minimal individual t in $\mathcal{T}(\Gamma)/\!\approx$. $\mathcal{D}_\Gamma$ must then be expanded to the reflexive and transitive closure of the expanded $\mathcal{P}_\Gamma$ and $\mathcal{L}_\Gamma$ must be expanded to satisfy ***T*10** and ***T*11**.

Proposition 8 *Suppose Γ is a branch of a saturated expansion Σ of a set of formulae Φ, and that $\overline{\Gamma}$ is a completion of Γ. Then:*

- *T_Γ is a tree,*
- *$T_\Gamma \models \overline{\Gamma}$,*
- *$T_\Gamma \models \Phi$.*

The second part is clear from the construction. The third part then follows by Lemma 4 and model invariance:

$$T_\Gamma \models \overline{\Gamma} \Rightarrow T_\Gamma \models \Gamma \Rightarrow T_\Gamma \models \Sigma \Rightarrow T_\Gamma \models \Phi.$$

The first part is proven by checking that each of the conditions for a tree are met by T_Γ. For details see (Rogers & Vijay-Shanker, 1993).

Thus every branch of a saturated expansion is satisfiable over trees.

Corollary 2 (Refutational Completeness) *If Φ has a saturated expansion then Φ is satisfiable over trees.*

Proof: If Γ is a branch in Σ, a saturated expansion of Φ, then $T_\Gamma \in \mathbf{Mod}(\Gamma) \subseteq \mathbf{Mod}(\Sigma) = \mathbf{Mod}(\Phi)$. □

More importantly, the proof is constructive and we have a particular model T_Γ, albeit one which depends on $\overline{\Gamma}$, for each branch of the saturated expansion. In section 7 we will establish that T_Γ is independent of $\overline{\Gamma}$ modulo $\doteq_\Gamma$ thus justifying the term T_Γ rather than $T_{\overline{\Gamma}}$.

6 Minimal Clausal Consequences

We have shown that, given any satisfiable set of formulae Φ, our mechanism can derive a saturated expansion $\{\Gamma_1, \Gamma_2, \ldots\}$. Further, for each branch Γ_i of this saturated expansion, $\mathbf{Cn}(\Gamma_i) = \mathbf{Cn}(\Phi \cup \Lambda_{\Gamma_i})$ since $\mathbf{Mod}(\Gamma_i) = \mathbf{Mod}(\Phi \cup \Lambda_{\Gamma_i})$. (Recall that Λ_{Γ_i} is the set of literals added to the branch Γ_i by the rule ***exp*** in the course of its derivation.) In this section we strengthen this by showing that every proper clause which is a consequence of $\Phi \cup \Lambda_{\Gamma_i}$ is either a tautology or is a superset of a clause in Γ_i. Thus, since the Γ_i contain no clause of form A_{Γ_i} by construction, the consequences of Γ_i can contain no such clause either. This establishes the second condition of Corollary 1 for each of the Γ_i. A second consequence of this is that no branch of a saturated expansion subsumes any other, that is, there is no pair of branches for which every tree modeling the first branch also models the second. This result is a necessary condition for our final claim—that the saturated expansion corresponds to the set of minimal referents of Φ. It also completes the connection between ***sub*** and its semantic interpretation.

To make this precise we will say that, by definition: a **proper** clause contains only literals; a clause is **trivial** if it contains both ϕ and $\neg\phi$ for some formula ϕ; a clause occurring in a set of clauses Φ is **minimal in** Φ if it is not a proper superset of any clause in Φ; and the **minimal clausal consequences** of a set of formulae Φ is the set of non-trivial, proper clauses that are minimal in $\{\Psi \mid \bigvee\Psi \in \mathbf{Cn}(\Phi)\}$. We claim that each branch Γ of a saturated expansion of Φ is the set of minimal clausal consequences of $\Phi \cup \Lambda_\Gamma$. This follows from two lemmas. First we establish that Γ has the correct form:

Lemma 5 *If Γ is a branch occurring in a saturated expansion then Γ consists of non-trivial, proper clauses that are minimal in Γ.*

Proof: Clauses in Γ are proper because no logical rule applies, they are non-trivial because ***taut*** does not apply, and they are minimal because ***abs*** does not apply. □

Next we show that every non-trivial clausal consequence of $\Phi \cup \Lambda_\Gamma$ is a superset of some clause in Γ.

Lemma 6 *Suppose Ψ is a proper clause, $\bigvee\Psi \in \mathbf{Cn}(\Phi \cup \Lambda_\Gamma)$, and Γ is a branch in Σ, a saturated expansion of Φ. Then either $\{\phi, \neg\phi\} \subseteq \Psi$ for some formula ϕ or there is some clause $\Upsilon \in \Gamma$ such that $\Upsilon \subseteq \Psi$.*

The idea behind the proof is that whenever $\bigvee\Psi \in \mathbf{Cn}(\Phi \cup \Lambda_\Gamma)$ then there is a derivation of [ ] from $\Gamma \cup \{\neg\psi \mid \psi \in \Psi\}$, while, of course, there is no such derivation from Γ by itself. The only way to introduce [ ] is by ***res*** and then only if there is some clause Υ for which $\{\Upsilon\} \cup \{\neg\upsilon \mid \upsilon \in \Upsilon\}$ is a subset of $\Gamma \cup \{\neg\psi \mid \psi \in \Psi\}$. It follows then that either Ψ contains both some ψ and $\neg\psi$ and is thus trivial, or Γ contains some $\Upsilon \subseteq \Psi$. The actual proof establishes a normal form for derivations for which this argument holds. The details are in the technical report.

With this we can prove the claim.

Proposition 9 *If Γ is a branch of a saturated expansion of Φ then Γ is the set of minimal clausal consequences of $\Phi \cup \Lambda_\Gamma$.*

Proof: (To show that Γ is a subset of the minimal clausal consequences of $\Phi \cup \Lambda_\Gamma$) By Proposition 7, $\mathbf{Mod}(\Gamma) = \mathbf{Mod}(\Phi \cup \Lambda_\Gamma)$. Thus $\Upsilon \in \Gamma$ implies $\bigvee\Upsilon \in \mathbf{Cn}(\Phi \cup \Lambda_\Gamma)$. Since Γ is saturated, Υ is not a tautology. Thus $\Upsilon \supseteq \Psi$ for some Ψ in the minimal clausal consequences of $\Phi \cup \Lambda_\Gamma$.

By definition of minimal clausal consequences, $\bigvee \Psi \in \mathbf{Cn}(\Phi \cup \Lambda_\Gamma)$. Then, by Lemma 6, $\Psi \supseteq \Upsilon'$ for some $\Upsilon' \in \Gamma$. Thus $\Upsilon' \subseteq \Psi \subseteq \Upsilon$. But, since Γ is saturated, Υ is minimal in Γ and thus $\Upsilon' \subseteq \Upsilon$ implies $\Upsilon' = \Upsilon$. Thus $\Psi = \Upsilon$ and Γ is a subset of the minimal clausal consequences of $\Phi \cup \Lambda_\Gamma$.

(For the other direction) Suppose that Ψ is a minimal clausal consequence of $\Phi \cup \Lambda_\Gamma$. Then again, Ψ is not tautological and $\bigvee \Psi \in \mathbf{Cn}(\Phi \cup \Lambda_\Gamma)$ and, by Lemma 6, $\Upsilon \subseteq \Psi$ for some $\Upsilon \in \Gamma$. But, as above, $\Upsilon \in \Gamma$ implies that $\bigvee \Upsilon \in \mathbf{Cn}(\Phi \cup \Lambda_\Gamma)$, and, by minimality of Ψ, $\Upsilon \subseteq \Psi$ implies $\Upsilon = \Psi$. □

Corollary 3 *Every branch of a saturated expansion is a quasi-tree.*

A second consequence of the proposition is incomparability of the branches of a saturated expansion with respect to subsumption.

Proposition 10 *If* Γ, Δ *are distinct branches in a saturated expansion then* $\mathbf{Mod}(\Gamma) \not\subseteq \mathbf{Mod}(\Delta)$.

Proof: Suppose, for contradiction, that $\mathbf{Mod}(\Gamma) \subseteq \mathbf{Mod}(\Delta)$. Then $\mathbf{Cn}(\Gamma) \supseteq \mathbf{Cn}(\Delta)$, and thus for all non-trivial, proper clauses $\Phi \in \mathbf{Cn}(\Delta)$ there is some $\Psi \in \Gamma$ such that $\Psi \subseteq \Phi$. In particular, for every $\Phi \in \Delta$ there is some $\Psi \in \Gamma$ such that $\Psi \subseteq \Phi$ and therefore ***sub*** applies to Δ and Γ contradicting hypothesis. □

Note that the proof completes the connection between the syntactic definition of the rule ***sub*** and its semantic interpretation. What it says, more or less, is that if one branch of a derivation is subsumed by another, eventually ***sub*** will eliminate it.

7 Saturated Expansions and Minimal Referents

So far, we have established that our primary mechanism can always derive a saturated configuration from a given set of formulae Φ, that this configuration will be closed whenever Φ is unsatisfiable, and that it will otherwise consist of a set of branches $\{\Gamma_1, \ldots\}$ where each branch is a quasi-tree. Further, we have demonstrated a mechanism for constructing, from each branch Γ_i, its minimal referent T_{Γ_i}. It remains only to show that these minimal referents are in fact, modulo $\leq_\Phi$, the minimal referents of the initial set of formulae Φ.

We begin by showing that T_Γ is in $\boldsymbol{\mu}\Gamma$.

Lemma 7 *If* Γ *is a branch of a saturated expansion, then* $T_\Gamma \leq_\Gamma T$ *for all trees* $T \in \mathbf{Mod}(\Gamma)$.

Proof: Recalling Definition 4, to show that $T_\Gamma \leq_\Gamma T$ we must show, for all $t, u \in \mathcal{T}(\Gamma)$, that:

$$\begin{aligned} T_\Gamma \models t \prec u \quad &\Leftrightarrow \quad T \models t \prec u \\ T_\Gamma \models \neg t \triangleleft^* u \quad &\Rightarrow \quad T \models \neg t \triangleleft^* u. \end{aligned}$$

For all trees T, if $T \models \Gamma$ then $T \models t \prec u \Leftrightarrow [t \prec u] \in \Gamma$, since Γ is complete for $\prec$.

If $T_\Gamma \models \neg t \triangleleft^* u$ then $[\neg t \triangleleft^* u] \in \overline{\Gamma}$. It is not difficult to show, by the construction of $\overline{\Gamma}$ that this implies $[\neg t \triangleleft^* u] \in \Gamma$. Consequently $T \models \neg t \triangleleft^* u$ for all $T \models \Gamma$. □

This justifies naming T_Γ independently of $\overline{\Gamma}$, since the T_Γ constructed for any completion of Γ will be equivalent wrt $\doteq_\Gamma$. Note that the uniqueness of T_Γ (modulo $\doteq$) also implies that Γ is a quasi-tree.

We can now show the correspondence between the T_Γ and the equivalence classes of $\boldsymbol{\mu}\Phi$. We do this by first demonstrating that every equivalence class in $\boldsymbol{\mu}\Phi$ includes some T_Γ and then demonstrating that every T_Γ is a member of some equivalence class in $\boldsymbol{\mu}\Phi$.

Proposition 11 *If Σ is a saturated expansion of Φ then for all $T \in \boldsymbol{\mu}\Phi$ there is some $\Gamma \in \Sigma$ such that $T \doteq_\Phi T_\Gamma$.*

Proof: Suppose $T \in \boldsymbol{\mu}\Phi$. Then $T \preceq_\Phi T'$ or $T' \not\preceq_\Phi T$, for all $T' \in \mathbf{Mod}(\Phi)$.

Since $T \models \Phi$, there is some $\Gamma \in \Sigma$ such that $T \models \Gamma$. Consequently, there is some $\Gamma \in \Sigma$ such that $T_\Gamma \preceq_\Phi T$. (Recalling that $\preceq_\Gamma$ and $\preceq_\Phi$ are the same relation.) Since $T_\Gamma \in \mathbf{Mod}(\Phi)$, $T \preceq_\Phi T_\Gamma$ as well, and therefore $T_\Gamma \doteq_\Phi T$. □

Proposition 12 *If Σ is a saturated expansion of Φ then for all $\Gamma \in \Sigma$ there is some $T \in \boldsymbol{\mu}\Phi$ such that $T_\Gamma \doteq_\Phi T$.*

Proof: By the definition of $\boldsymbol{\mu}\Phi$, for every $\Gamma_i \in \Sigma$ there is some $T \in \boldsymbol{\mu}\Phi$ such that $T \preceq_\Phi T_{\Gamma_i}$ (since $T_{\Gamma_i} \models \Phi$). By the previous proposition for all $T \in \boldsymbol{\mu}\Phi$ there is some $\Gamma_j \in \Sigma$ such that $T_{\Gamma_j} \doteq_\Phi T$.

Suppose then, that $T_0 \in \boldsymbol{\mu}\Phi$ and $T_0 \preceq_\Phi T_{\Gamma_i}$, but $T_{\Gamma_i} \not\doteq_\Phi T_0$. Then there is some $T_{\Gamma_j} \in \Sigma$ such that $T_{\Gamma_j} \doteq_\Phi T_0 \prec_\Phi T_{\Gamma_i}$. Thus it suffices to show for all $\Gamma_i, \Gamma_j \in \Sigma$ that $T_{\Gamma_j} \preceq_\Phi T_{\Gamma_i} \Rightarrow \Gamma_i = \Gamma_j$.

Recall $\mathbf{Mod}(\Gamma_i) = \mathbf{Mod}(\Phi \cup \Lambda_i)$ where Λ_i is the set of formulae introduced into Γ_i by $\boldsymbol{exp}$, and that these are of the form $\neg t \triangleleft^* u$ or $t \not\prec u$.

Suppose $T_{\Gamma_i} \preceq_\Phi T_{\Gamma_j}$.

$T_{\Gamma_i} \models \Gamma_i \Rightarrow T_{\Gamma_i} \models \Lambda_i$.

$T_{\Gamma_i} \preceq_\Phi T_{\Gamma_j} \Rightarrow T_{\Gamma_j} \models \Lambda_i$.

$T_{\Gamma_j} \models \Lambda_i \Rightarrow (\forall T \models \Gamma_j)\,[T \models \Lambda_i]$, since $T_{\Gamma_j} \preceq_\Phi T$.

$(\forall T \models \Gamma_j)\,[T \models \Lambda_i] \Rightarrow \mathbf{Mod}\,\Gamma_j \subseteq \mathbf{Mod}\,\Gamma_i$.

Then, by Proposition 10, $\Gamma_j = \Gamma_i$. □

Corollary 4 *If Σ is a saturated expansion of Φ then $\doteq_\Phi$ establishes a 1-1 correspondence between $\{T_\Gamma \mid \Gamma \in \Sigma\}$ and $\{[T]_\Phi \mid T \in \boldsymbol{\mu}\Phi\}$. Thus, since $\{T_\Gamma \mid \Gamma \in \Sigma\} \subseteq \boldsymbol{\mu}\Phi$, the set $\{T_\Gamma \mid \Gamma \in \Sigma\}$ is, modulo $\doteq_\Phi$, the set of minimal referents of Φ.*

8 Conclusion

In this paper we have explored the problems of determining when a partial description of trees is satisfiable and, if it is, of constructing an appropriate representative tree that satisfies it. These problems arise in recent developments (Vijay-Shanker, 1992; Vijay-Shanker & Schabes, 1992) in the TAG framework which involve manipulation of these partial descriptions. Our work provides formal foundations for these developments and potentially for other work employing descriptions of this sort.

Vijay-Shanker (1992) introduced an interpretation of TAGs in which adjunction is understood as a logical operation on quasi-trees—partial descriptions of trees for which every tree satisfying the description can be derived by adjunction from some minimal tree which also satisfies the description. This interpretation provides insight into the nature of the adjunction operation and supports a variety of generalizations of it. In this paper we have formalized the notion of quasi-trees by fixing a quantifier-free language in which descriptions of trees can be expressed and then defining a relation on the theories of trees in that language which corresponds to the relation between a tree and those trees which can be derived from it by adjunction. A quasi-tree is then any set of formulae which has a unique minimal model wrt that relation or, equivalently, the set of trees which satisfies such a set of formulae. This gives us our notion of what counts as an appropriate representative of a set of trees—that minimal tree. A description unambiguously determines a minimal tree in this sense only if it is a quasi-tree.

The partial descriptions employed in (Vijay-Shanker & Schabes, 1992) are not assumed to be quasi-trees and they are combined by operations which generalize adjunction. Nonetheless, these operations are used to recover the elementary trees of a TAG grammar, and it is assumed that a particular tree can ultimately be extracted from their result. Thus that work presupposes the existence of mechanisms to verify that such a tree can always be recovered and to construct that tree. We have provided these mechanisms. The first is an extension of a resolution theorem prover which, given an arbitrary description produces a logically equivalent set of quasi-trees. If this set is not empty the given description is satisfiable, if it is singleton the description is unambiguous. Thus this mechanism provides a means of checking the correctness of the encoding of a grammar. The second mechanism constructs the unique minimal model of these quasi-trees. Thus it serves to construct an elementary tree from its description.

References

Fitting, M. (1990). *First-order Logic and Automated Theorem Proving.* New York: Springer-Verlag. p. 242.

Rogers, J. & Vijay-Shanker, K. (1992). Reasoning with descriptions of trees. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, (pp. 72–80).

Rogers, J. & Vijay-Shanker, K. (1993). A mechanism for obtaining trees from their descriptions. Technical Report 94-18, University of Delaware, Wilmington, DE. p. 36.

Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes, Number 4. Stanford, CA: CSLI. p. 105.

Shieber, S. M. (1989). Parsing and type inference for natural and computer languages. Technical Report 460, SRI International. p. 147.

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

Vijay-Shanker, K. & Schabes, Y. (1992). Structure sharing in lexicalized tree-adjoining grammars. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'92)*, (pp. 205–212).

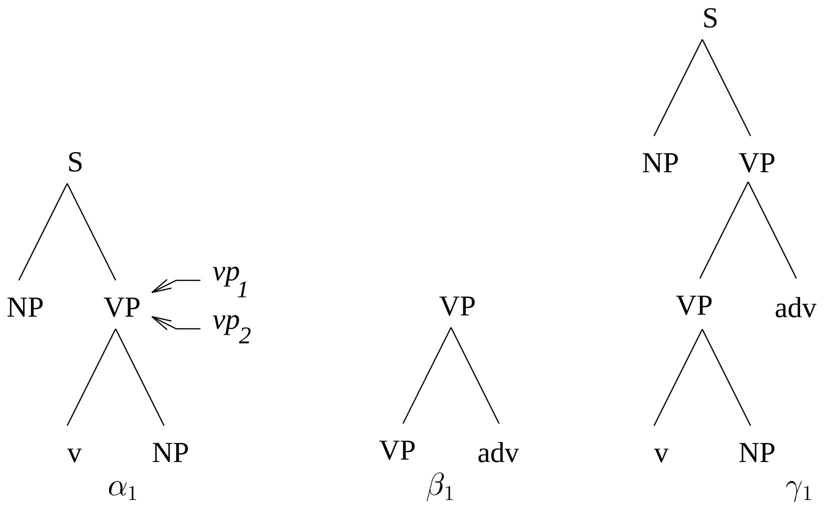

Figure 1: Adjunction

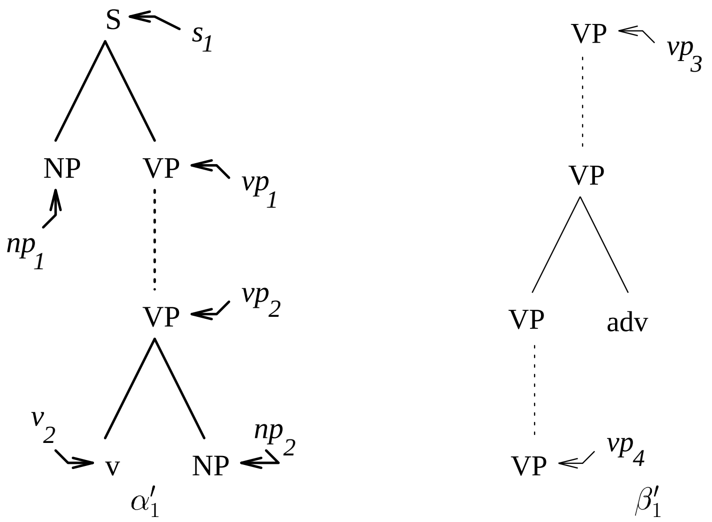

Figure 2: Quasi-trees

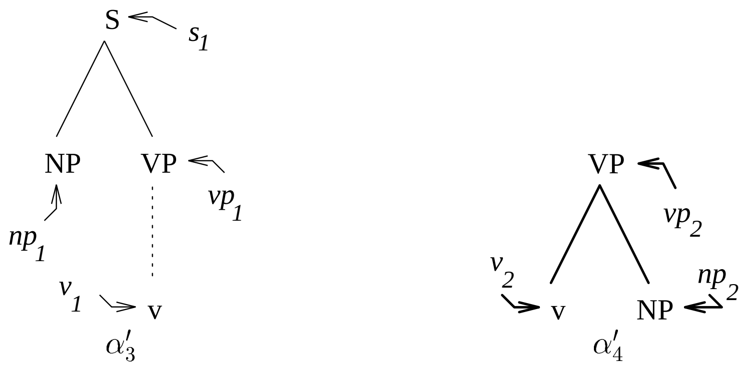

Figure 3: Structure sharing