Semantic Tableaux with Ordering Restrictions

Stefan Klingenbeck* and Reiner Hähnle

University of Karlsruhe
Institute for Logic, Complexity and Deduction Systems
76128 Karlsruhe, Germany
{klingenb,reiner}@ira.uka.de +49-721-608-{3978,4329}

Abstract. The aim of this paper is to make restriction strategies based on orderings of the Herbrand universe available for semantic tableau-like calculi as well. A marriage of tableaux and ordering restriction strategies seems to be most promising in applications where generation of counter examples is required. In this paper, starting out from semantic trees, we develop a formal tool called *refutation graphs*, which (i) serves as a basis for completeness proofs of both resolution and tableaux, and (ii) is compatible with so-called A-ordering restrictions. The main result is a first-order ground tableau procedure complete for A-ordering restrictions.

Introduction

In recent years one could observe a kind of renaissance of tableau-related methods in automated theorem proving after the field has been dominated by resolution approaches for many years[2]. Tableaux are easy to adjust to nonclassical logics, and they have already a number of advantages for classical first-order logic that have not been paid great attention to for some time, notably: (i) they have a higher ground speed than resolution; (ii) it is easy to incorporate theories [10]; (iii) there are many refinements to remove redundancy from proofs [13, 12]; (iv) lemmaizing and caching can be naturally defined and implemented [1]. All these techniques can be efficiently implemented using Prolog technology [13]. Finally, (v) tableaux are suitable for human interaction. This is a major advantage in program verification, where frequently very large or not valid formulas occur. In the latter case also the use of specialised decision procedures e.g. for certain arithmetical theories is very helpful.

The basic idea of order restricted resolution is as follows:[3] assume we are given a partial ordering $<$ on first-order terms. Now admit only resolution steps wherein the resolved literal is $<$-maximal in the resolvent clause. For certain

* Supported by BMFT within the project KORSO.

[2] We assume the reader is familiar with basic expositions of semantic tableaux like [8] and of resolution like [6].

[3] In order to gain familarity with ordering restricted resolution we recommend [7] as an up-to-date account.

orderings this restriction turn out to be still complete for first-order logic. Moreover, resolution with (variants of) ordering restrictions can be used to decide certain classes of first-order logic [7] and to enhance the performance of basic resolution [18]. Hence, in the light of applications such as program verification it is extremely desirable to make ordering refinements available for tableaux as well, thus bringing together paradigms that have already been proven successful separately.

The part of resolution that corresponds to the selection of a closure substitution in tableau is the selection of a pair of parent clauses and complementary literals therein. A wrong choice does not require backtracking, since resolution is *proof confluent*, but an unnecessary clause can in turn produce many useless clauses, and if resolution proofs are unsuccessful, then usually because the system gets choked with useless clauses.

Every reasonable tableau procedure with free variables and unification must employ backtracking over the possible substitutions that close a branch. For complex problems the resulting number of proofs becomes simply too large even if techniques like caching [1], anti-lemmata or regularity conditions [13] are used to reduce the number of choices. For this reason ordering restrictions are obvious candidates for further enhancement of tableaux, as they are well understood in the case of resolution.

The paper is organized as follows: in Section 1 for convenience of reading we repeat some basic definitions; in Section 2 we introduce refutation graphs which constitute the central link between semantic trees and semantic tableaux, and prove some basic properties. In Section 3 we use refutation graphs to give a completeness proof of (order-restricted) binary resolution, while in Section 4 we do the same for (unrestricted) ground tableaux. Finally, in Section 5 we prove our main result, namely that a straightforward and natural ordering restriction of first-order ground tableau is a complete proof procedure for first-order logic.

Due to space restrictions most proofs had to be omitted. A long version of this paper with full proofs of all theorems can be obtained from the authors on request or via anonymous ftp to `129.13.31.2` (switch to binary mode and get the file `pub/haehnle/Ordered-Tableaux-Long.ps.Z`).

1 Semantic Trees

Throughout the paper we use a standard first-order CNF clause language without equality.

The proof of completeness of resolution via semantic trees is well known. Unfortunately, it is not easy to adapt this proof to (clausal) tableau procedures. The first obstacle is to identify the exact counterpart of clauses and resolvents in a tableau. The propositional resolution rule can be seen as a cut rule restricted to atomic formulas and clausal tableau proofs corresponds to cut free sequent proofs. Cut elimination, however, results in additional copies of certain subformulas which are spread around the proof. Hence, more than one formula in a

tableau corresponds to one generated clause in a resolution proof and one has to keep track of these formulas.

Example 1.

In Figure 1 the closure of the rightmost branch corresponds to an application of the resolution rule with parent clauses $A \vee -B \vee C \vee D$ and $A \vee E \vee -D$. The resolvent $A \vee -B \vee C \vee A \vee E$ occurs as the union of the framed nodes. The parent clauses may be spread over several nodes in the tableau. The only restriction is that the nodes of the literals resolved upon are labelled with a single literal. Note that the parent clauses are not available any longer. Otherwise, a node can belong to different clauses and cause problems, especially in the case of free variable first-order tableaux.

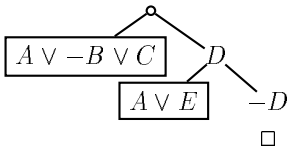

Fig. 1. The tableau is symbolized by a labelled tree. □ denotes the closing of a branch.

The classic completeness proof for resolution [11] is based on two presuppositions: (i) Only subsumed clauses and tautological clauses are discarded; (ii) The resolvent belonging to an inference node retains all interesting ground instances of the parents. As we just have seen we have to discard also clauses in a tableau procedure that are not subsumed. Thus we have to modify the proof idea for completeness based on semantic trees.

We assume the reader is familiar with the basic notions and results of computational logic, in particular with Herbrand bases and Herbrand's theorem (see, for instance [14]). In the following let B_0 be the Herbrand base of our first-order language, and let $B_0 = A_0, A_1, \ldots$ be an arbitrary, but fixed, enumeration; $\sim L$ denotes the complementary of a literal L; sometimes we will treat clauses as sets without mentioning it. $C \backslash L$ means, that each occurrence of the literal L is deleted from the clause C. Most of the following definitions may be found in [14, 7].

Definition 1. Let $a_0, a_1, \ldots$ be a fixed enumeration of a set of atoms B. The labelled, binary tree ST, which is defined as the smallest tree obeying the following conditions, is called **semantic tree** for B.

1. The root node of ST is unlabelled; its left child is labelled with a_0, its right child is labelled with $-a_0$.
2. If a node is labelled with a_i or $-a_i$, then its left child is labelled with a_{i+1}, and its right child is labelled with $-a_{i+1}$.

The semantic tree for B_0 is denoted with ST_0. Two nodes in ST with the same parent node are called **siblings**. A **path** in ST is a (finite or infinite) sequence of node labels $\langle l_i, l_{i+1}, \ldots \rangle$, such that the node belonging to l_{j+1} is the child of the node belonging to l_j for all $j \geq i$. An maximal path in ST is a **branch** of ST. Two paths p_1, p_2 are called **sibling-paths** wrt a pair of sibling nodes n_1, n_2 iff p_1 and p_2 are both of the form $\langle l_1, l_2, \ldots, l_{i-1}, l_{n_j}, l_{i+1}, \ldots \rangle$, where l_{n_j} is

the label of n_j for $j = 1, 2$. With each node n of a semantic tree ST we associate its **refutation set** which consists of the labels on the path from the root to n.

Definition 2. A clause C **fails** at a node n of a semantic tree ST iff there is a substitution σ such that for each literal L in $C\sigma$, $\sim L$ is in the refutation set of n. A node n is a **failure node** for a set of clauses M iff some clause from M fails at n, but no clause from M fails at a node above n. A node n is an **inference node** iff both of its children are failure nodes. A semantic tree ST is **closed** by M iff every branch of ST contains a failure node for M.

See Figure 2 for an example illustrating these definitions.

Definition 3. An **A-ordering** on a set of atoms B is a binary relation $<_A$ such that for all $a, b, c \in B$

Irreflexivity $a \not<_A a$.
Transitivity $a <_A b$ and $b <_A c$ imply $a <_A c$.
Substitutivity $a <_A b$ implies $a\sigma <_A b\sigma$ for all substitutions σ.

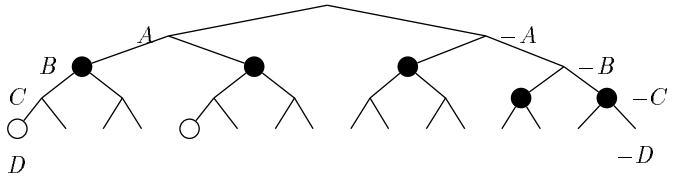

Fig. 2. The failure nodes in a semantic tree for $\{A, B, C, D\}$ for the clause set $\{-A \vee -B,\ -A \vee B,\ A \vee -B,\ A \vee B \vee -C,\ A \vee B \vee C\}$. The paths $\langle A, B, C, D\rangle$ and $\langle A, -B, C, D\rangle$ going from the node labelled with A to the nodes marked with $\circ$ are sibling-paths the sequences of their labels differing only in the second position. The failure nodes are marked with $\bullet$. The node labelled with A is an inference node (among others). Since all branches contain a failure node, the tree is closed.

2 Refutation Graphs[4]

Our goal is to adapt completeness proofs via semantic trees to tableau procedures. In the resolution case, completeness proofs via semantic trees are based on

[4] This should not be confused with a concept bearing the same name introduced by Shostak [16] in the context of clause graph resolution. Since we feel that the phrase 'refutation graph' catches exactly our intention we decided to keep the name nevertheless. Independently, Bibel [4] recently developed the concept of a so-called π-regular graph which is somewhat related to our refutation graphs.

the successive elimination of failure nodes, yielding smaller and smaller closed semantic trees for an inconsistent clause set. At the end the tree consisting solely of the root node is left. Since the only clause that fails at the root node is the empty clause, it must have been inferred by that time.

Unfortunately, this argument does not work in the case of tableaux. Instead of reducing the size of the whole closed semantic tree one has to have a closer look on the nodes of a semantic tree that establish the closure of the tree. We will collect such nodes according to their mutual relationship into more complicated graphs on the node set of the semantic tree. If such a graph meets certain requirements, it can be seen as refutation of a corresponding set of clauses. The number of nodes in this graph will play a similar role in our completeness proof as the size of the semantic tree in the classic proof.

Definition 4. Let M be a set of clauses, and G be a ground instance of a clause $C \in M$. We call a set ch of nodes of ST_0 a **chain** of G (or of C) iff

- all nodes in ch are on the same path in ST_0, and
- L is a label of a node in ch iff $\sim L \in G$.

We denote the set of all chains of M in ST_0 by $CH(M)$, and the set of all nodes occurring in a set CH of chains by $\overline{CH}$. The **empty chain** by definition corresponds to the empty clause.

Informally speaking, a chain is a minimal subset of a path that refutes a clause. We define chains not as sequences but as sets, because we will mainly use them accordingly.

Example 2. Let $M = \{-A \vee -B \vee -C, -A \vee C, -B, A \vee -C, A \vee C, -A \vee B\}$ The chains in Figure 3 marked with $1, \ldots, 7$, respectively, belong to the following clauses: 1 belongs to $-A \vee -B \vee -C$, 2 and 3 belong to $-A \vee C$, 4 belongs to $-B$, 5 belongs to $A \vee -C$, 6 belongs to $A \vee C$, and 7 belongs to $-A \vee B$. Note that different chains can belong to the same clause, because different nodes can have the same labels.

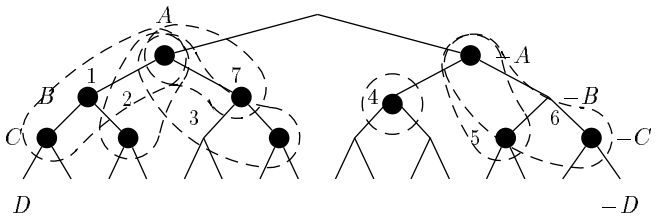

Fig. 3. Chains that belong to various clauses in the semantic tree for the clause set from Example 2.

Since all nodes of a chain are contained in the same path of a semantic tree they are ordered in a natural way. The successor relation together with the relation of being sibling nodes is used to define the graphs mentioned above.

Definition 5. We write $n_1 > n_2$, if n_2 is closer to the root. Moreover, a node n_1 of a chain ch is called a **successor node** of $n_2 \in ch$, if $n_1 > n_2$ and there is no node $n_3 \in ch$ such that $n_1 > n_3 > n_2$. A node without a successor node is called an **end node** of that chain.

Definition 6. For an arbitrary set of chains CH in ST_0, we define a binary relation $\prec_{CH}$ on the nodes of ST_0: $n_1 \prec_{CH} n_2$ iff n_1 and n_2 are sibling nodes or belong to the same chain in CH and n_2 is a successor of n_1.

A set CH of chains **closes** a subtree ST of ST_0 iff it contains the empty chain or each path of ST_0 through the root of ST contains a chain of CH whose end node is in ST.

Note that for each clause set M the semantic tree ST_0 is closed by $CH(M)$ iff ST_0 is closed by M in the sense of Definition 2.

Definition 7. Given a non-empty set of chains CH and a subtree ST of ST_0, the **graph** defined by ST and CH is $G(CH, ST) = \langle \overline{CH} \cap ST, \prec_{CH} \rangle$. A graph $G(CH, ST)$ is called a **refutation graph** iff it is finite and for all nodes $n_1 \in G(CH, ST)$, $n_2 \in ST_0$ we have that $n_1 \prec_{CH} n_2$ implies $n_2 \in G(CH, ST)$. A graph $G(CH, ST)$ is a **semi refutation graph** iff it is finite and for each node, but the root node of ST, also its sibling is contained in $G(CH, ST)$. A (semi) refutation graph $G(CH, ST)$ is **minimal** iff for no $CH_1 \subsetneq CH$ $G(CH_1, ST)$ is a (semi) refutation graph. A refutation graph $G(CH, ST)$ is **connected** if there are no $\emptyset \neq CH_1, CH_2 \subsetneq CH$, $CH_1 \neq CH_2$ such that $G(CH_1, ST)$ and $G(CH_2, ST)$ are both refutation graphs.

Note that the empty graph that corresponds to the set of chains consisting only of the empty chain is a refutation graph, and is called the **empty refutation graph**.

Example 3. Let M be as in Example 2. The chains of Example 2 do not establish a refutation graph, since sibling nodes for nodes in the chains 3 and 4 are missing. However, consider the subgraph consisting of the chains $CH_1 = \{1, 2, 5, 6, 7\}$. The corresponding refutation graph $G(CH_1, ST_0)$ is shown on top in Figure 4. Note that the clauses corresponding to chains $3, 4$ are not necessary for a refutation by resolution or tableau. We get a much simpler refutation graph $G(CH_2, ST_0)$ (depicted on bottom in Figure 4), if we take the chains $5, 6, 7$ and 8 as a chain set CH_2 with the new chain 8 stemming from the clause $-B$. One can consider chain 8 as a copy of chain 4 placed in the left subtree of ST_0 instead of the right. The next definition will deal with this copy operation more rigorously.

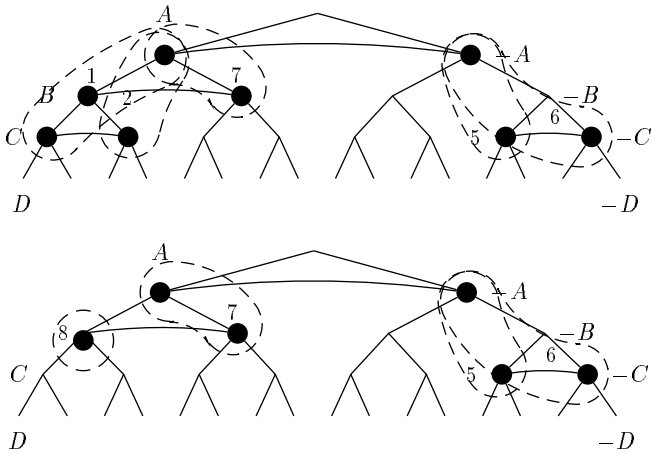

Fig. 4. The refutation graphs $G(CH_1, ST_0)$ and $G(CH_2, ST_0)$ from Example 3.

Definition 8. Let ch be a chain on a path p in the tree ST_0 with end node n. Let $c \notin ch$ be a node with $c < n$ and let q be the sibling-path of p with respect to c and its sibling. We denote with $copy(ch, c)$ the unique chain situated on q, whose nodes have the same labels as ch. The extension of the definition $copy(CH, c)$ to sets of chains CH is obvious.

Example 4. Using the numbering of the previous examples, $copy(4, n_A)$=8, where n_A is the node labelled with A.

Theorem 9. *Let M be a clause set.*

1. *M is unsatisfiable iff ST_0 is closed by M.*
2. *If ST_0 is closed by M, then there is a set of chains $CH \subseteq CH(M)$ such that $G(CH, ST_0)$ is a refutation graph.*
3. *If CH is a set of chains belonging to clauses in M, and $G(CH, ST_0)$ is a refutation graph, then ST_0 is closed by M.*

Proof. 1. Proved in [7].
2. The semi refutation graph $G(CH, ST_0)$ constructed in Lemma 10 is sufficient, because ST_0 contains all nodes.
3. Proved in Lemma 11 for arbitrary subtrees ST of ST_0.

Lemma 10. *Let M be a set of clauses and ST a subtree of ST_0 closed by $CH(M)$. There is a set of chains $CH \subseteq CH(M)$ such that*

- *CH closes ST*
- *CH contains only chains that have end nodes in ST*
- *no node in CH is end node of different chains*

– *$G(CH, ST)$ is a semi refutation graph*

Proof. The proof is by induction over the maximal distance d between any inference node and the root node n_0 of ST.

If $d = 0$ (that is, the root node of ST is an inference node), then both its successors are failure nodes with corresponding chains ch_1, ch_2. Let $CH = \{ch_1, ch_2\}$; all conditions are fulfilled trivially.

If $d \neq 0$ several cases must be distinguished. Consider the successors n_1, n_2 of n_0. We proof only the semi refutation graph property, all others are trivial. If n_i is the end node of a chain ch_i, then set $K_i = \{ch_i\}$. Otherwise, if n_i is the root of a closed subtree, then let K_i be the set of chains obtained by applying the induction hypothesis.

Case 1: $n_i \in \overline{K_i}$ for $i = 1, 2$ or $n_i \notin \overline{K_i}$ for $i = 1, 2$. Set $CH = K_1 \cup K_2$.

Case 2: $n_1 \in \overline{K_1}$, but $n_2 \notin \overline{K_2}$. This time we cannot take the union of K_1 and K_2 as CH, because, by $n_1 \in \overline{K_1}$, its sibling n_2 would have to be in $\overline{CH}$ which is not the case. We remedy this situation by discarding all chains ending in the subtree rooted at n_1, and use instead the semi refutation graph we have already below n_2. Formally, let $\tilde{K_1} := copy(K_2, n_2)$. Set $CH = \tilde{K_1} \cup K_2$. All chains in $\tilde{K_1}$ have their end nodes in the subtree rooted in n_1 and neither n_1 nor n_2 is in $G(CH, ST)$. Hence, by the induction hypothesis, for all nodes of $G(CH, ST)$, with the exception of n_0, also its sibling is in $G(CH, ST)$.

Case 3: $n_2 \in \overline{K_2}$, but $n_1 \notin \overline{K_1}$. Analogous to case 2.

Lemma 11. *If ST is a subtree of ST_0, $CH \subseteq CH(M)$, and $G(CH, ST)$ is a refutation graph, then ST_0 is closed by M.*

Proof. By definition, CH is non-empty. If CH contains the empty chain, the result is trivial. Assume p is a branch of ST_0 that has no failure node. Then there is also no chain $ch \in CH(M)$ with $ch \subseteq p$. We will construct such a chain $ch' \subseteq p$ and thus find a contradiction. We find ch' by successively decreasing the number of chains in CH until only ch' is left.

With each node n_i of p we associate its sibling node $\sim n_i$ and the subtrees $ST_i, \sim ST_i$ rooted in n_i, respectively, in $\sim n_i$. For each n_i we recursively define a set of chains CH_i and a semi refutation graph $G(CH_i, ST_i)$:

$$CH_0 := CH$$

$$CH_i := \begin{cases} \{ch \in CH_{i-1} |\, ST_i \cap ch \neq \emptyset\} & \overline{CH_{i-1}} \cap ST_i \neq \emptyset \\ \{copy(ch, \sim n_i) |\, ch \in CH_{i-1}, \sim ST_i \cap ch \neq \emptyset\} & \overline{CH_{i-1}} \cap ST_i = \emptyset \text{ and} \\ & \overline{CH_{i-1}} \cap \sim ST_i \neq \emptyset \\ CH_{i-1} & \text{otherwise} \end{cases}$$

By this definition all those chains are removed from CH_{i-1} that are surely no candidates for ch'. The copy operation in the above definition is well defined, provided $G(CH_{i-1}, ST_{i-1})$ is a semi refutation graph. For in this case neither the

node n_i nor the node $\sim n_i$ are contained in any chain of CH_{i-1} if $\overline{CH_{i-1}} \cap ST_i = \emptyset$.

Each $G(CH_i, ST_i)$ is either empty or it is a semi refutation graph, because for all semi refutation graphs $G(CH, ST)$ and every subtree ST' of ST the following holds: $G(CH, ST')$ either is a semi refutation graph or it is empty. If it is not empty and $\overline{CH}$ does not contain the root node n_0 of ST', then $G(copy(CH, n_0), ST')$ is a semi refutation graph. Recall that copied chains belong to the same clauses as their originals.

The case when $G(CH_i, ST_i)$ is empty gives rise to $CH_j := CH_{j-1}$ for all $j > i$. Now it is easy to see, that for all i the following properties hold:

1. $\overline{CH_i} \neq \emptyset$ (although $CH_i \cap ST_i$ may be empty).
2. $\overline{CH_i} \subseteq p \cup ST_i$ (all chains in CH_i are on the path p at least up to the node n_i).

Since CH_0 is finite (by definition of a semi refutation graph), it has a node with maximal distance to the root node. Let n_{max} be this node on p. Because of $ST_{max} \cap \overline{CH_i} = \emptyset$ for all i, we know that $ch' \subseteq p$ for any $ch' \in CH_{max}$.

3 Refutation Graphs and Resolution

We intend to use refutation graphs for completeness proofs. In the present section we outline how this is done in the case of resolution. If we can prove that a refutation procedure finds and recognizes a refutation graph of an unsatisfiable clause set, as a consequence of Theorem 9, it is refutation complete. First we will redefine some notions related to semantic trees in the context of refutation graphs.

Definition 12. Let M be a clause set, CH a subset of its chains, and $G(CH, ST)$ a refutation graph. A node $n \in \overline{CH}$ is a **failure node** of $G(CH, ST)$, iff it is an end node of a chain $ch \in CH$ and no other chain of CH than ch contains a node $n_1 \geq n$.

We call the parent node of two sibling nodes that are failure nodes[5] an **inference node** of $G(CH, ST)$ and any two chains having siblings as failure nodes **inference chains**. If ch_1, ch_2 are a pair of inference chains with failure nodes n_1 and n_2, then the chain with nodes $(ch_1 \cup ch_2) \backslash \{n_1, n_2\}$ is called **resolvent chain** of ch_1 and ch_2.

The next lemma shows that each minimal, non-empty refutation graph contains at least one inference node.

Lemma 13. *Let $G(CH, ST)$ be a minimal semi refutation graph, ST a subtree of ST_0, and n the end node of a chain $ch \in CH$. There is no chain $ch \neq ch_1 \in CH$ containing a node $n_1 \geq n$.*

[5] Recall that by definition for each node in a refutation graph also its sibling node must be present.

Proof. The proof is by induction over the maximal distance d between any end node and the root node n_0 of ST.

If $d = 0$, then the root node of ST must be an end node (hence, $ST_0 \neq ST$). The semi refutation graph $G(CH, ST)$ is minimal, so it consists exactly of the chain ending in the root node of ST.

If $d > 0$ the root node is no end node, otherwise $G(CH, ST)$ would not be minimal. Let $CH_1, CH_2 \subseteq CH$ be the sets of chains having their end nodes in the left subtree ST_1 of ST, respectively, in the right subtree ST_2 of ST. We are done if we can ensure that $G(CH_i, ST_i)$ is a minimal semi refutation graph for $i = 1, 2$.

This claim is proved by contradiction: let, for example, $G(CH_1, ST_1)$ be not minimal and let $G(CH_1', ST_1)$ be a semi refutation graph with $CH_1' \subsetneq CH_1, CH_1' \neq CH_1$. Since $G(CH, ST)$ is minimal, $G((CH \backslash CH_1) \cup CH_1', ST)$ is no semi refutation graph. (CH_1', ST_1) being a semi refutation graph implies that the root node of ST_1 is not contained in $(CH \backslash CH_1) \cup CH_1'$, but its sibling is. The contradiction comes with the fact that $G(CH_1', ST)$ already establishes a semi refutation graph and, therefore, $G(CH, ST)$ would not be minimal.

In a refutation graph the sibling of a failure node is also contained in the graph. Since each refutation graph is finite, a minimal, non-empty refutation graph contains at least one inference node.

Theorem 14. *Let $G(CH, ST)$ be a minimal refutation graph, let ST be a subtree of ST_0, and e the resolvent chain of a pair c, d of inference chains. Then the graph $G((CH \backslash \{c, d\}) \cup \{e\}, ST)$ is a refutation graph and it has lesser nodes than $G(CH, ST)$.*

Proof. We abbreviate $(CH \backslash \{c, d\}) \cup \{e\}$ with CH'. We have to prove that for all nodes $n_1 \in G(CH', ST), n_2 \in ST_0$ $n1 \prec_{CH'} n2$ implies $n_2 \in G(CH', ST)$.

Since we did not add nodes to CH, all nodes of CH' are still nodes of ST and each chain is completely contained in ST. We deleted a pair of failure nodes. Since $G(CH, ST)$ is minimal, by the previous lemma we know that no other nodes loose their siblings.

To each chain there belongs a unique ground clause (the empty clause to the empty chain). Thus the well-known lifting lemma (see [14] for the versions for basic resolution and for ordered resolution) tells us that resolving of chains can be lifted to basic resolution (with factorisation). Moreover, since in our setting only maximal nodes (literals) are involved in a resolution step, the restrictions imposed by any A-ordering are obeyed, provided the ordering is compatible with the enumeration of the Herbrand base chosen for ST_0.

Example 5. Consider the set of chains CH_2 used in the lower part of Figure 4. Chains 5 and 6 constitute a pair of inference chains with failure nodes labelled with C and $-C$. $G(CH_2, ST_0)$ is a minimal refutation graph. If we resolve on chains 5 and 6 we obtain the minimal refutation graph $G(CH_3, ST_0)$ depicted in Figure 5, where $CH_3 = (CH_2 \backslash \{\{-A, C\}, \{-A, -C\}\}) \cup \{\{-A\}\}$. Note that

CH_2 corresponds to the set of clauses $\{-B, -A \vee B, A \vee -C, A \vee C\}$, and the application of Theorem 14 corresponds to resolving the last two clauses on C.

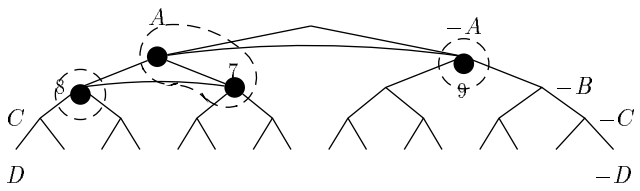

Fig. 5. The minimal refutation graph $G(CH_3, ST_0)$ from Example 5.

Robinson resolution or A-resolution can recognize a refutation graph that consists of a pair of sibling nodes (that are resolved to the empty clause). The search strategies of these procedures enrich the set of clauses. Let us think about adding all resolvents of factors of parents in a clause set S_i yielding the new clause set S_{i+1} as a single step. We know that S_{i+1} contains a refutation graph with lesser nodes. For an unsatisfiable clause set S_0 the well-founded order on sets of clauses $S < T$ iff $\alpha(S) < \alpha(T)$, where $\alpha(S) = \min\{|G(CH', ST)| \ : \ CH' \subseteq CH(S), ST \subseteq ST_0, G(CH', ST)$ a is a refutation graph$\}$ garuantees termination since under this ordering $S_i > S_{i+1}$ for all $i \geq 0$.

Resolution takes care of all graphs at each step and uses the fact that certain resolution steps decrease the number of nodes of a refutation graph. A-resolution employs that these resolution steps are among the A-resolvents. Therefore, instead of resolving each possible pair of parent clauses, A-resolution looks for certain patterns, namely for inference nodes. The corresponding pair of clauses is A-resolvable.

4 Refutation Graphs for Tableaux

We will deal with tableaux for ground clauses in a enumeration strategy for first-order logic. Thus it is convenient to modify our notion of tableaux in order to reflect the clausal form (cf. also [12] for *clausal tableaux*). The reader, who is familiar with Fitting's [8] conception of tableaux, can consider our tableaux as generated by a single extension rule that, at the same time, generalizes the β extension rule to any finite number of disjuncts, and combines it with the substitution rule.

Definition 15. Let M be a clause set. We call a finitely branching tree T a **tableau** for a clause set M, if

- the root node is labelled with T, all other nodes are labelled with a literal, and

– if $D = L_1 \vee \cdots \vee L_n$, where $\{L_1, \ldots, L_n\}$ are the literals labelling the set of direct successor nodes of some node, then there is a substitution σ and a clause $C \in M$, such that $D = C\sigma$.

T is a **ground tableau** iff all its labels are ground. A ground tableau is **closed** iff every branch contains two complementary literals. A **subtableau** is a proper subtree of a tableau (hence, the root of a subtableau is labelled with a literal).

We use the following notation for tableaux: $T = (L, T_1, \ldots, T_n)$, where L is the label of the root of T, and $T_1, \ldots, T_n$ are immediate subtableaux.

Tableau procedures for clause sets regard the chains of each clause as a part of a possible refutation graph. To find a refutation graph they look for a chain that contains a sibling node of a node of a chain already present in the tableau. If a suitable chain is found, a corresponding clause can be used to extend the current tableau and close a branch of the extended tableau. If some siblings of the used chain (clause) are not yet on the extended branch one has to find counterparts for these nodes also. One can consider constructing a tableau as walking through graphs. The next definition makes this idea precise.

Definition 16. Let *seq* be a finite sequence $\langle n_1, n_2, \ldots, n_m \rangle$ of nodes of a graph $G(CH, ST)$, $m > 1$. We call *seq* a **connection** in $G(CH, ST)$ iff $n_i \neq n_j$ for $i \neq j$ (i.e. *seq* is repetition free), all n_i, n_{i+1} are siblings for odd i, and all n_i, n_{i+1} are members of the same chain in CH for even i. We say that n_m is **connected** to n_1. If n_m is connected to n_1, then obviously there is a chain that contains n_m and all of whose nodes are also connected to n_1. We call the set $cc(CH, n_1)$ of chains in $G(CH, ST)$ all of whose nodes are connected to n_1 the **connected component** of n_1.

Example 6. Consider the graph $G(CH, ST)$ in Figure 6 with chains as indicated, without the shaded chain. The nodes $\langle n_1, n_2, n_3, n_4, n_7 \rangle$ form a connection from n_1 to n_7. All chains in CH but $\{n_1\}$ are connected to n_1, hence $cc(CH, n_1) = CH \backslash \{\{n_1\}\}$. On the other hand, none of the nodes in $\{n_1, \ldots, n_7\}$ is connected to n_7, thus $cc(CH, n_7) = \emptyset$. Hence, we see immediately that the connectedness relation is irreflexive and not symmetric.

Lemma 17. *Let $n_1, n_2, \ldots, n_m$ be a connection in $G(CH, ST)$ and let n'_m be the sibling of n_m. If there is a path of ST through both n_1 and n'_m, then $n'_m = n_{m-1}$.*

Proof. By induction over the length of the connection. The base case $m = 2$ is clear, because n_1 and n_2 are siblings by definition.

For $m > 2$ let ST_2 be the tree rooted in n_2. No node of ST_2 but n_2 is a sibling to any node that shares a path with n_1, therefore, n_m is not in ST_2. If ch is a chain including n_2, then all nodes of the chain are in ST_2 or predecessors of n_1. Let n_j be the first node in $n_2, \ldots, n_m$ not in ST_2. There is such a node since n_m is not in ST_2. Then j is odd, because the sibling of every node, but the root

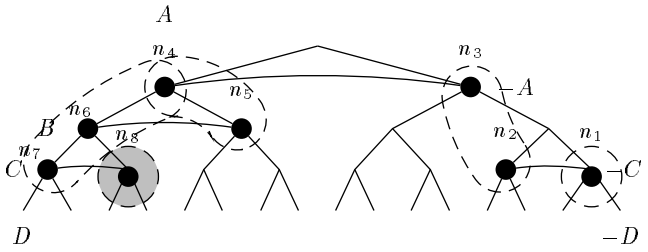

Fig. 6. Connections in a graph.

node, in ST_2 is again in ST_2, hence the node in the connection before n_j cannot be a sibling of n_j. In addition, n_j and n_1 must have a path of ST in common and n_j is a predecessor of n_1. Thus there is a path through both n_j and n'_m and the lemma follows after the induction hypothesis is applied to $n_j, \ldots, n_m$.

Let us adopt the convention $\sim \Box = \mathsf{T}$.

Definition 18. Let CH be an arbitrary set of chains, T a ground (sub)tableau and L a label. We say that T **represents** the pair (CH, L) iff the following holds: If CH is empty, then $T = (\sim L)$, else $T = (\sim L, T_1, \ldots, T_n)$ with subtableaux T_i representing (CH_i, L_i), where the L_i are the labels of the nodes n_i of a chain $ch \in CH$ and the $CH_i = cc(CH \backslash \{ch\}, n_i)$ are connected components of the roots of the subtrees (each of the CH_i has less chains than CH provided CH is finite).

If we fix a rule telling us how to choose the next chain $ch \in CH$, when there are several, we obtain a unique tableau representant of a given pair (CH, L). This is the case, for example, when there is an enumeration of the ground instances of a given clause set M. If the chains of CH belong to ground instances of M, one can take the chain with the lowest index.

Example 7. Assume that the graph of Figure 6 is extended to a refutation graph $G(CH, ST)$ by the shaded chain. In order to construct a tableau that represents $(CH, \Box)$, we begin by taking an arbitrary chain from CH, say $\{n_4, n_5\}$, and extend the root tableau node with the complements of its labels, see Figure 7(a). We number the branches, so that we can identify them more easily.

The next chain that is used to extend branch (1) must be taken from the set $cc(CH \backslash \{\{n_4, n_5\}\}, n_4) = \{\{n_1\}, \{n_2, n_3\}\}$. We take $\{n_1\}$ and extend (1) with C. After that only one chain, namely $\{n_2, n_3\}$, is left, and we obtain the tableau in Figure 7(b).

Now consider branch (3) in Figure 7(b). Observe that node n_5 is connected to every chain in $CH \backslash \{\{n_4, n_5\}\}$. Hence, $CH_{(1)} := cc(CH \backslash \{\{n_4, n_5\}\}, n_5) = CH \backslash \{\{n_4, n_5\}\}$ is the next connected component. We choose $\{n_2, n_3\}$ and extend the tableau with the corresponding labels in Figure 7(c).

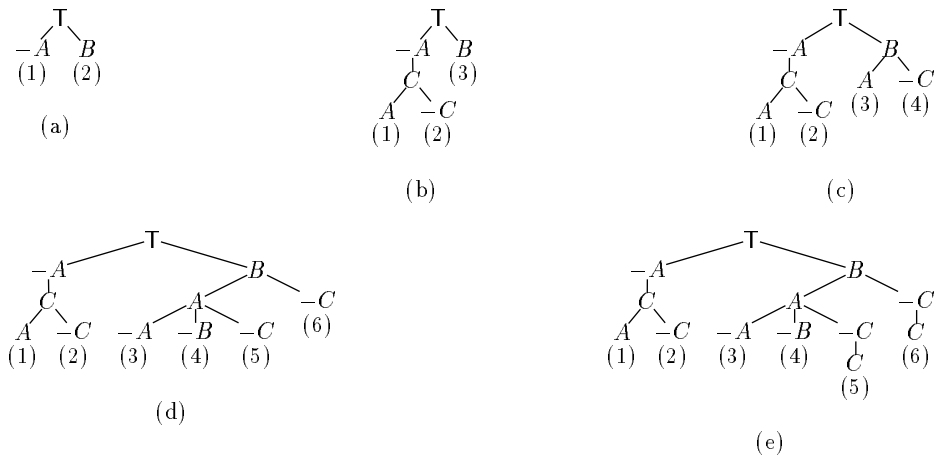

Fig. 7. Constructing a closed tableau that represents a refutation graph.

We focus on branch (3) in Figure 7(c). The leaf A is the label of n_3. We must look for the chains in $CH_{(2)} := cc(CH_{(1)} \backslash \{\{n_2, n_3\}\}$ that are connected with n_3. These are $\{\{n_4, n_6, n_7\}, \{n_8\}\}$. We expand the tableau with the labels corresponding to $\{n_4, n_6, n_7\}$, and obtain Figure 7(d). The node corresponding to the leaf of (3) is n_4. If we look for the chains in $CH_{(3)} := cc(CH_{(2)} \backslash \{\{n_4, n_6, n_7\}\}, n_4)$ we see that this is the empty set, therefore, we cannot expand the branch anymore. Indeed, branch (3) in Figure 7(d) is closed with $(-A, A)$. The same is true for branches (1), (2), and (4). In each case the connected component is empty.

The node corresponding to the leaf of (5) is n_7. The connected component for n_7 in $CH_{(4)} := cc(CH_{(2)} \backslash \{\{n_4, n_6, n_7\}\}, n_7)$ is $\{n_8\}$, and we extend (5) with C. The situation in branch (6) is similar, and we finally obtain the closed tableau shown in Figure 7(e).

The following lemma states that it was not by coincidence that the tableau in the example was closed.

Lemma 19. *If T represents $(CH, \Box)$ and $G(CH, ST)$ is a refutation graph, then T is closed. Moreover, if a subtableau T' of T represents a pair (CH', L), then for each node n in CH' the following holds:*

(*) Either the sibling of n is in $\overline{CH'}$ or the literal that labels n is already on the branch.

Proof. By definition, no leaf ($\sim L$) of T can be expanded, because the corresponding connected component $CH_L = cc(CH'' \backslash \{ch\}, n_L)$ is empty, where CH'' is the set of chains that (L) must represent. If (*) holds for CH'' and CH_L is empty, then the sibling of n_L is not in CH'', and L is already on the branch,

which is the complementary literal to the leaf ($\sim L$). So the closedness of T indeed follows from (*).

Now to the proof of (*). It holds for CH, because $G(CH, ST)$ is a refutation graph and thus for each node in $\overline{CH}$ also its sibling is present.

Otherwise consider the subtableau T_i representing (CH_i, L_i), where L_i is the label of the node $n_i \in ch$ and $CH_i = cc(CH'\backslash\{ch\}, n_i)$ Provided (*) holds for each node of CH', then it also holds for those nodes n_m of $CH'\backslash\{ch\}$ whose siblings n'_m are not in ch. The remaining case is when n_m is connected to n_i and the sibling n'_m in ch, but then we can apply Lemma 17. With $n_m \neq n_i$ also its sibling node is in $CH_i = cc(CH\backslash\{ch\}, n_i)$. With n_i also the literal labelling its sibling is on the branch.

In the light of the previous lemma and Theorem 9 it is sufficient for a tableau procedure in order to be complete to ensure that, if there is a refutation graph for a clause set, a tableau representing it is found. One can guruantee this, for example, by backtracking over all possible tableaux, or by successively bringing all ground instances of all clauses on each branch. In the latter case there is a tableau for each refutation graph compatible with the chosen enumeration strategy. If unnecessary literals are put onto a branch the closing steps have to be repeated in each superfluous resulting branch. Of course, there is no need to bring the same ground instance of a literal more than once on a branch. One can easily construct a closed tableau without multiple occurrences of ground instances on a single branch if a closed tableau without this restriction is given. For the sake of simplicity we did not address this problem in Lemma 19.

A tableau with free variables as in [8] deals simultaneously with all ground instance tableaux that are obtained by the application of a ground substitution to the variables of the tableau. One has to ensure then that it can be closed, if one of its instances can be closed. This problem will be addressed in a forthcoming paper.

5 Tableaux with A-Ordering

In this section we explain how to incorporate A-orderings into a tableau refutation procedure for clause sets.

Definition 20. Let T be a ground (sub)tableau representing (CH, L) and $d(n)$ the distance of a node n from the root node. We say T is **ordered** iff either

- T is a leaf or
- all immediate subtableaux $T_1, \ldots, T_n$ of T are ordered, and there is no node $d(n) > d(n'_{\max})$ in $\overline{CH}$, where $n'_{\max}$ is the sibling of the maximal node of the chain $ch \in CH$ that determines $T_1, \ldots, T_n$.

Example 8. The tableau in Figure 7(e) is not ordered, because $d(n_5)$ is not maximal in the set $\overline{CH}$ of Example 7. On the other hand, both subtrees below the root node are ordered.

The following theorem is a consequence of Lemma 19.

Theorem 21. *If T is ordered, represents $(CH, \Box)$, and $G(CH, ST)$ is a refutation graph, then T is closed.*

Proof. Each subtableau T' of T represents a pair (CH', L). The theorem is a consequence of Lemma 19, provided that CH' contains a chain that obeys the restriction of Definition 20. In each chain set there are chains whose end nodes have maximal distance from the root. Every such chain is suitable.

One can combine this strategy with enumeration, if one selects the chain with the lowest index if several candidates are present.

Definition 22. Let $<_A$ be an A-ordering on a clause set M. An **A-ordered ground instance enumeration tableau** is a tableau constructed by the following restricted expansion rule:

Tableau branches are expanded solely by ground instances of clauses in M not already on the branch

1. that have a maximal literal complementary to a literal already on the branch or
2. to a maximal literal of another ground instance of a clause in M

Theorem 23. *If M is an unsatisfiable clause set, then there is a closed A-ordered ground instance enumeration tableau.*

Proof. If M is an unsatisfiable clause set, then let $CH \subseteq CH(M)$, and T' an ordered tableau representing $(CH, \Box)$ such that $G(CH, ST_0)$ is a refutation graph. This is possible by Theorem 9. By Theorem 21, T' is closed.

The property (*) of Lemma 19 holds for T'. By definition of ordered tableaux, the sibling of the maximal literal of an expansion is either complementary to the maximal literal of a ground instance or not in the corresponding connected component. In the latter case, by (*), the complement of the maximal literal is already on the branch. Thus all the instances in T' are permitted by the restriction imposed in the A-ordered ground instance enumeration tableau procedure.

6 Conclusions and Related Work

In this paper we can only lay the ground for further investigations. We provided a technique for proving completeness of first-order enumeration ground tableaux and Robinson resolution in a uniform way. The completeness proofs can be adapted to ordering refinements for resolution and tableau as well. The outlined ground tableau procedure can be immediately implemented in tableau provers like HARP [15] or Tatzelwurm [10].

In future work we intend to investigate the combination of ordering restrictions with free variable tableaux, tableaux with equality, and tableaux as decision procedures.

There have been several approaches to involve ordering restrictions into first-order theorem proving. Soon after the introduction of basic resolution by J. A. Robinson a wide variety of refinements for resolution followed. Ordering restrictions and semantic clash were first investigated by Slagle [17]. Kowalski and Hayes [11] combined ordering restrictions with a completeness proof using semantic trees. Joyner used this refinement as a decision procedure for a certain class of first-order formulas [9]. His results, however, were not widely discussed until recently Fermüller et al. [7] revived the ideas of Joyner. Another approach to decidability stems from the inverse method due to Maslov, and was further developed by Zamov [19] and Tammet [18] in a very similar way as resolution in the work of Joyner. Both approaches contribute to [7]. Further, we mention the work of Bachmaier and Ganzinger [2]. They generalize completion procedures similar to the well known Knuth-Bendix completion to refutational first-order theorem proving with equality. A different approach to using tableaux as a decision procedure for certain first-order problems is due to Caferra [5]. It is not based on ordering restrictions, but on keeping track of models with a decidable equality language.

References

1. O. Astrachan and M. Stickel. Caching and lemmaizing in model elimination theorem provers. In D. Kapur, editor, *Proc. 11th Conference on Automated Deduction, Albany/NY, USA*, pages 224–238. Springer LNAI 607, 1992.
2. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. Technical Report MPI–I-9-1–208, Max-Planck-Institut für Informatik, MPI at Saarbrücken, Germany, 1991.
3. W. Bibel. *Automated Theorem Proving.* Vieweg, Braunschweig, second revised edition, 1987.
4. W. Bibel and E. Eder. Decomposition of tautologies into regular formulas and strong completeness of connection graph resolution. Technical Report AIDA-94-0?, FG Intellektik, FB Informatik, TH Darmstadt, Mar. 1994.
5. R. Caferra and N. Zabel. A tableaux method for systematic simultaneous search for refutations and models using equational problems. *Journal of Logic and Computation*, 3(1):3–26, 1993.
6. C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, London, 1973.
7. C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Methods for the Decision Problem.* Springer LNAI 679, 1993.
8. M. C. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer, New York, 1990.
9. W. H. Joyner. Resolution strategies as decision procedures. *Journal of the Association for Computing Machinery*, 23(3):398–417, 1976.
10. T. Käufl and N. Zabel. Cooperation of decision procedures in a tableau-based theorem prover. *Revue d'Intelligence Artificielle*, 4(3):99–126, 1990.
11. R. Kowalski and P. Hayes. Semantic trees in automatic theorem-proving. *Machine Intelligence*, 4:87–101, 1969.
12. R. Letz. *First-Order Calculi and Proof Procedures for Automated Deduction.* PhD thesis, TH Darmstadt, June 1993.

13. R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A high-perfomance theorem prover. Technical report, Forschungsgruppe Künstliche Intelligenz, TU München, 1991.
14. D. W. Loveland. *Automated Theorem Proving. A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science*. North-Holland, 1978.
15. F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *Journal of Automated Reasoning*, 4:69–100, 1988.
16. R. E. Shostak. Refutation graphs. *Artificial Intelligence*, 7:51–64, 1976.
17. J. R. Slagle. Automatic theorem proving with renamable and semantic resolution. *Journal of the Association for Computing Machinery*, 14(4):687–697, 1967.
18. T. Tammet. The resolution program, able to decide some solvable classes. In *Proceedings COLOG-88, Talinn*, pages 300–312. Springer, LNCS 417, 1990.
19. N. Zamov. Maslov's inverse method and decidable classes. *Annals of Pure and Applied Logic*, 42:165–194, 1989.