Ordered Tableaux: Extensions and Applications[⋆]

Reiner Hähnle and Christian Pape

Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme
76128 Karlsruhe, Germany, {reiner,pape}@ira.uka.de

Abstract. In this paper several conceptual extensions to the theory of order-restricted free variable clausal tableaux which was initiated in [9, 8] are presented: atom orderings are replaced by the more general concept of a selection function, the substitutivity condition required for lifting is for certain variants of the calculus replaced by a much weaker assumption, and a first version of order-restricted tableaux with theories is introduced. The resulting calculi are shown to be sound and complete. We report on first experiments made with a prototypical implementation and indicate for which classes of problems order-restricted tableaux calculi are likely to be beneficial.

1 Introduction

In this paper we continue to develop the theory of order-restricted free variable clausal tableaux which was initiated in [9, 8].

A-ordered tableaux (full definitions of all notions are given in Section 2 and 3) are regular clause tableaux with two different kinds of extension steps: given an A-ordering [6] $\prec$ on literals, a clause C can be used to extend a tableau branch $\mathbf{B}$ iff (i) C has a maximal connection into $\mathbf{B}$, i.e. the connection literal of C is $\prec$-maximal in C *or* (ii) C has a maximal connection into another clause D, i.e. the connection literals of both clauses are $\prec$-maximal in the clause, where they occur.

On the one hand we present several conceptual extensions of ordered tableaux: in Section 4 atom orderings are replaced by the more general concept of a selection function; in Section 5 we demonstrate how such calculi can be implemented with the help of constraints; for a somewhat less restrictive class of calculi we show that the substitutivity condition required for lifting can be replaced by a much weaker assumption: *stability wrt variable renaming* is already sufficient for completeness. The resulting calculus, called *tableaux with input selection function* is shown to be complete.

Finally, a first version of tableaux with selection function and theories is presented (Section 6.1).

On the other hand, in Section 7 we report our experiences made with a prototypical implementation. We indicate for which classes of problems order-restricted tableaux calculi are likely to be beneficial.

[⋆] This work has been partially supported by the Deutsche Forschungsgemeinschaft, Schwerpunktprogramm "Deduktion".

On a methodological level we compare tableaux with selection function to restart model elimination, recently developed by Baumgartner & Furbach [3] (Section 6.2).

2 Preliminaries

Given a signature Σ, i.e. a set of predicate, function, constant and variable symbols, then **atoms** and **literals** are constructed from Σ and the negation sign $\neg$ as usual. The set of all literals for Σ is denoted by $\mathbf{L}_\Sigma$. We omit the index Σ if no confusion can arise.

We denote substitutions by σ, τ, or write them explicitly as a (finite) set $\sigma = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ with the meaning $\sigma(x_i) = t_i$ and $\sigma(x) = x$ for all $x \neq x_i$. The special case of a variable renaming (all t_i are distinct variables) is denoted by θ. We use such substitutions only for replacing the variables of a clause by new distinct variables.

A **clause** is a sequence $L_1 \vee \ldots \vee L_n, n \geq 1$ of disjunctively connected literals. The variables in such clauses are assumed to be (implicitly) universally quantified. An **instance** of a clause C is a sequence of literals $C\theta$ such that θ replaces the variables of C by new variables. The variables of instances of clauses are only placeholders for terms, they are *not* assumed to be universally quantified. $\mathbf{C}$ is the set of all clauses. We write $L \in C$ for short if a literal L occurs in a clause C. $\overline{L}$ is the **complement** of a literal L, i.e. $\overline{A} = \neg A$ and $\overline{\neg A} = A$ if A is an atom. *Atom* selects the atomic part of literals that is $Atom(A) = Atom(\neg A) = A$ for every atom A.

A **clause tableau** $\mathfrak{T}$ is an ordered tree where the root node is labeled with *true* or a literal and all other nodes are labeled with literals. For a node n of $\mathfrak{T}$ the clause $ClauseOf(n)$ is constructed from the literals of the children of n in the order from left to right. $Predecessor(n)$ denotes the parent node of node n. A path from the root node to a leaf literal of $\mathfrak{T}$ is called a **branch** of $\mathfrak{T}$. A tableau is **closed** if every branch contains (at least) two complementary literals. We sometimes describe a tableau as a finite set of branches and a branch as a finite set of literals. We also often identify branches with the set of literals on them. A branch $\mathbf{B}$ is said to be **regular** if (i) every literal of a node of $\mathbf{B}$ occurs only once on $\mathbf{B}$ and (ii) $ClauseOf(n)$ is not a tautology for every node n of $\mathbf{B}$. A tableau $\mathfrak{T}$ is **regular** if all branches of $\mathfrak{T}$ are regular.

Partial Interpretations are associated with a consistent set of ground literals. An interpretation I **satisfies** a ground clause C iff there exists $L \in C$ with $L \in I$. I is said to be a **model** for a set S of first order clauses iff I satisfies all clauses of every ground instance of S.

3 $\mathcal{A}$-Ordered Tableaux

To ease comparison with previous results in this section we briefly rehash $\mathcal{A}$-ordered clausal ground tableaux as defined in [8].

Definition 1. An A-**ordering** is a binary relation $\prec_A$ on atoms, such that for all atoms A, B, C:

1. $A \not\prec_A A$ (**irreflexivity**),
2. $A \prec_A B$ and $B \prec_A C$ implies $A \prec_A C$ (**transitivity**), and
3. $A \prec_A B$ implies $A\sigma \prec_A B\sigma$ for all substitutions σ (**substitutivity**).

For sake of readability we focus on the ground version of A-ordered tableaux. Lifting to first order logic is handled in Section 5 in the context of tableaux with selection function. The results established there hold as well for A-ordered tableaux. As in ordered resolution connections between clauses are restricted to literals that occur $\prec_A$-maximally.

Definition 2. A literal L_j occurs $\prec_A$**-maximally** in a clause $L_1 \vee \ldots \vee L_n$ iff $Atom(L_j) \not\prec_A Atom(L_i)$ for all $i = 1, \ldots, n$.

A clause $C = L_1 \vee \ldots \vee L_n$ possesses a $\prec_A$**-maximal connection** to a clause $C' = L'_1 \vee \ldots \vee L'_{n'}$ iff $L_i = \overline{L'_j}$ for some $1 \leq i \leq n, 1 \leq j \leq n'$, L_i occurs maximally in C, and L'_j occurs maximally in C'. If, moreover, $C, C' \in S$ then C is called a **restart clause** of S.

A clause C has a **maximal connection into a set of literals B** iff C has a maximal connection to a literal of $\mathbf{B}$.

A-ordered tableaux are regular[2] clause tableaux with the additional restriction that a clause $C \in S$ can be used to extend a branch $\mathbf{B}$ only if C has a maximal connection into $\mathbf{B}$ or to another clause of S:

Definition 3. An A**-ordered ground clause tableau** is a regular ground clause tableau $\mathfrak{T}$ such that

1. $C = ClauseOf(n)$ has a maximal connection into the branch ending in n *or*
2. C is a restart clause of S.

At the start of a refutation the initial tableau is empty and only the second extension rule (called **restart rule**) can be used to expand it. But even if a relevant clause for the initial step is used, it is still necessary to allow restarts later on to obtain a complete calculus, as the following example shows:

Example 1.
Take the unsatisfiable clause set $M = \{A \vee \underline{B},\ A \vee \underline{\neg B},\ \neg A \vee \underline{C},\ \neg A \vee \underline{\neg C}\}$ and A-ordering $A < B < C$ (maximal literals are underlined). Each clause of M is a restart clause. Fig. 1 shows a closed A-ordered tableau for M. The solid lines correspond to a (maximal) **extension step**: these branches can be closed immediately with the maximal literal of the maximal connected clause. The dashed arrows indicate a **reduction step**, i.e. the branch is closed by non maximal literals of a clause.

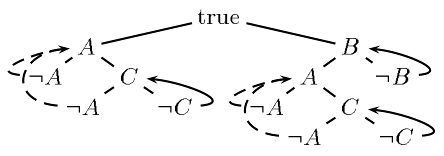

Fig. 1. Restarts are necessary

[2] Regularity and ordering restrictions are orthogonal concepts, but we choose to start out from regular tableaux, because regularity fits naturally into our completeness proof below. All results do still hold, of course, if regularity is dropped.

Note that without a second application of the restart rule no closed A-ordered tableau for M can be constructed, independently of the choice of clause for the initial step.

Theorem 4 [9]. *For any unsatisfiable set S of ground clauses and A-ordering $\prec_A$ exists a closed $\prec_A$-ordered tableau for S.*

In the ground case the procedure stays complete even if restart steps are delayed until no extension steps are possible.

The first order case is handled as usual with Herbrand's Theorem and a lifting lemma which, by substitutivity of $\prec_A$, is straightforward. In Section 5 we will see that lifting is even possible under a weaker assumption.

L**-orderings** (and L-ordered tableaux) are defined exactly as A-orderings, but on literals instead of atoms. Hence, each A-ordering is also an L-ordering, but not vice versa. It is easy to show that the previous theorem holds for L-orderings as well.

4 Tableaux with Selection Function

In this section we still work with ground clauses. The first order case is considered in the following section.

As shown in the previous section, in ordered tableaux only the literals which are maximal wrt the clause in which they occur need to be considered for an extension step. This kind of restriction can be generalized with the help of certain functions.

Each A-ordering $\prec_A$ induces a function $f_{\prec_A} : \mathbf{C} \rightarrow (2^{\mathbf{L}} - \{\emptyset\})$ by stipulating $f_{\prec_A}(C) = \{L | \, L \text{ is } \prec_A\text{-maximal in } C\}$. On the other hand, not every function f from $\mathbf{C}$ to $(2^{\mathbf{L}} - \emptyset)$ can be realized with an A-ordering.[3]

Definition 5. A **selection function** is a total function $f : \mathbf{C} \rightarrow (2^{\mathbf{L}} - \{\emptyset\})$ such that all literals in $f(C)$ occur also in C for all $C \in \mathbf{C}$.

A selection function f is **deterministic** iff $|f(C)| = 1$ for all $C \in \mathbf{C}$.

Example 2. Consider a *total L-ordering* $\prec_L$ on ground literals. Then exactly one literal in each ground clause is maximal. Thus $\prec_L$ defines a deterministic selection function.

A total A-ordering on ground literals does not define a deterministic selection function in general as, for example, in $L \vee \overline{L}$ both literals are maximal.

We define particular deterministic selection functions f_{last} and f_{first} which select the exactly last, respectively, the first literal of a clause.

In tableaux with selection function the selected literals play the rôle of maximal literals in ordered tableaux. Accordingly, connections between clauses are restricted to selected literals.

[3] Consider the clauses $C_1 = L_1 \vee L_2$, $C_2 = L_2 \vee L_1$, and assume $f(C_i) = \{L_i\}$ for $i = 1, 2$. Any ordering $\prec_A$ such that $f_{\prec_A} = f$ must be cyclic which contradicts irreflexivity.

Definition 6. Given a selection function f, two ground clauses C and D have a **connection via** f iff there are $L \in f(C)$, $M \in f(D)$ such that L is complementary to M. If, moreover, $C, D \in S$ then we say that C is a **restart clause of** S **(wrt** f**)**.

A clause C has a **weak connection into a set of literals B via** f iff some $L \in f(C)$ is complementary to a literal of $\mathbf{B}$.

In tableaux with selection function the only admissible extension steps are with clauses that have a weak connection into the branch they extend or if they are restart clauses wrt f (see Definition 7). In the latter case $C = D$ is not excluded, however, such clauses are tautologies and, by regularity, are not allowed to be considered for extension steps.

Note that the set of restart clauses can easily be computed in a pre-processing step and thus causes no additional cost during proof search.

Definition 7. Given a selection function f and a set of ground clauses S, a **tableau with selection function** f **for** S is a regular ground clause tableau $\mathfrak{T}$ such that

1. $C = \mathit{ClauseOf}(n)$ has a weak connection via f into the branch ending in n *or*
2. C is a restart clause of S wrt f.

Example 3. Reconsider the set M of Example 1. The selection function f_{last} selects the underlined literals and thus the ordered tableau displayed in Fig. 1 is also a tableaux with selection function f_{last} for M.

Because each restart clause can extend any tableau branch at any time (provided that it was not used before in this branch which would violate regularity), one should be very careful in the choice of a selection function. To gain a maximum of search space restriction for a given set of clauses, one should choose a selection function for which the number of restart clauses in S is minimal. This prohibits extensive use of restarts and leads to stronger connected proofs.

By virtue of regularity of tableaux with selection function it is not possible to build infinite branches in an attempt to refute a finite set S of ground clauses. In our completeness proof this property ensures that a model of S can be constructed from an open *finite* tableau branch which cannot be extended further. As a consequence, ground tableaux with selection function are proof confluent.

Definition 8. Let S be a set of ground clauses, f a selection function, and $\mathfrak{T}$ a tableau with selection function f for S. $\mathfrak{T}$ is **saturated** iff there is no tableau $\mathfrak{T}'$ with selection function f for S such that $\mathfrak{T}$ is a proper subtree of $\mathfrak{T}'$.

Theorem 9. *For any finite unsatisfiable set S of ground clauses and selection function f there exists a closed tableau with selection function f for S.*

Proof. Assume there were no closed tableaux with selection function f for S. We construct a model of S.

Regularity does not permit to extend tableau branches with tautologies, hence assume wlog that S does not contain any tautologies.

Let $\mathfrak{T}$ be any saturated tableau with selection function f for S which is finite by regularity and finiteness of S. $\mathfrak{T}$ is not closed, so it has a finite open branch $\mathbf{B}$. The literals on $\mathbf{B}$ form a partial interpretation $I_{\mathbf{B}}$ which satisfies at least the clauses of S that were used to extend $\mathbf{B}$.

Let $S' \subseteq S$ be the set of clauses not satisfied by $I_{\mathbf{B}}$. The clauses in S' have the following properties:

1. There are no clauses $C, D \in S'$ such that C is connected to D via f. Otherwise, C and D were restart clauses and thus were used to extend $\mathbf{B}$, because $\mathfrak{T}$ is saturated. But then C and D are satisfied by definition of $I_{\mathbf{B}}$.
2. There is no clause $C \in S'$ such that C has a weak connection via f into $\mathbf{B}$. Otherwise, C was used to extend $\mathbf{B}$ and, as before, is satisfied by definition of $I_{\mathbf{B}}$.

By 1. $J = \bigcup_{C \in S'} f(C)$ is a well-defined partial interpretation which is trivially also a model of S'.

By 2. $I_{\mathbf{B}} \cup J$ is a well-defined interpretation and thus a model of S, because $I_{\mathbf{B}}$ is a model of $S - S'$ and J is a model of S'. □

This proof can be adapted to formulas in negation normal form (as done for A-ordered tableaux in [8]) or even to arbitrary first order formulas, see [15] for details. Likewise, the proof works as well for infinite sets of ground clauses provided that the clauses are selected in a fair manner for extension.

It is also easy to see that the proof goes through unaltered for a generalized procedure in which before each restart or extension step the selection function f can be changed arbitrarily.

In the case of A-orderings the proof can be made even shorter by making use of a result by Bachmair & Ganzinger [1] that any set of clauses which is saturated wrt ordered resolution and does not contain the empty clause is satisfiable. If this is assumed then the model building part in our proof can be omitted and it simply remains to show that if $\mathbf{B}$ is a saturated ordered open tableau branch for S, then $\mathbf{B} \cup S$ is saturated wrt ordered resolution, an observation due to Bachmair (personal communication).

We refrain from using the latter insight, because the results in [1] (i) are only for the clausal case, whereas our approach can be generalized to non-normal-form [15], and (ii) are only for the theory of equality, whereas we employ other theories as well.

5 Lifting

As usual in semantic tableaux, there are (at least) two different ways for lifting ground tableaux with selection function to first order logic.

One can enforce a fair selection of all the ground instances $\hat{S}$ of a first order clause set S to build a (possibly infinite) ground tableau with selection function for $\hat{S}$ which in turn gives a first-order tableau with selection function for S. Completeness of this calculus follows directly from Herbrand's Theorem and Theorem 9. This gives a calculus in the spirit of Smullyan [17].

For efficiency reasons we favor the so-called *free variable* approach (cf., for example, [7]): rather than guessing the "right" instantiation of a universally quantified formula, one uses free variables and unification to search for a closing substitution. For this purpose we generalize the notions of connection and of selection function.

Definition 10. A clause C has a **connection** with a clause C' iff there exist literals $L \in C$, $L' \in C'$, such that $L\sigma = \overline{L'}\sigma$ with mgu σ. L and L' are called **connection literals**.

Given a selection function f, C and C' have a **connection via** f iff they have a connection with literals L and L' and mgu σ such that $L\sigma \in f(C\sigma)$ and $L'\sigma \in f(C'\sigma)$. If, moreover, $C, C' \in S$ we say that C is a **restart clause** in S (with connection literal L and mgu σ).

By Herbrand's Theorem we know that for any unsatisfiable set S of first order clauses there is a finite unsatisfiable set $\hat{S}$ of ground instances of S. For $\hat{S}$ there is a closed ground tableau with selection function by Theorem 9. It is straightforward to lift this tableau to a closed first order tableau provided that the selection function used for the latter has the following property:

Definition 11. A selection function f is **stable wrt substitutions** iff $L\sigma \in f(C\sigma)$ implies $L \in f(C)$ for all substitutions σ and clauses C.

Example 4. The functions f_{last} and f_{first} defined in Example 2 are stable wrt substitutions.

Theorem 12. *For any unsatisfiable set S of first order clauses and selection function f which is stable wrt substitutions exists a closed tableau with selection function f for S.*

In general it is not possible to extend a deterministic selection function on ground clauses to a deterministic selection function on first order clauses which is stable wrt substitutions as the following simple example shows (one can obtain, however, a slightly different notion of selection function by using literal *occurrences* in the definition of stability which admits deterministic extension):

Example 5. Consider any deterministic selection function f and the first order clause $C = p(x) \vee p(f(y))$. For the substitution $\sigma = \{x \leftarrow f(a), y \leftarrow a\}$ obviously $f(C\sigma) = \{p(x)\sigma, p(f(y))\sigma\} = \{p(f(a))\}$ holds. Then every extension $\hat{f}$ of f to first order clauses must, by the substitutivity condition (Def. 11), select both literals $p(x)$ and $p(f(y))$. Therefore $\hat{f}$ is not deterministic.

Table 1 shows a first order proof calculus based on Definition 7. The additional set $\mathfrak{C}$ is a constraint which guarantees that every ground instance of the resulting tableau is also a ground tableau with selection function. More precisely, $\mathfrak{C}$ is a set of literal/clause pairs $\langle L, C\rangle$ which record the so far selected literals so one can check whether the substitution associated with a connection respects the selection function. Accordingly, one defines $\langle L, C\rangle\sigma = \langle L\sigma, C\sigma\rangle$ and $\mathfrak{C}$ to be **satisfiable** iff $L \in f(C)$ for all $\langle L, C\rangle \in \mathfrak{C}$. The idea of using constraints to express global restrictions on tableau search is due to [10], where constraints were used to enforce regularity. To increase readability these regularity constraints are omitted in Table 1.

$\dfrac{\mathfrak{T} \cup \{\mathbf{B}\} \mid\mid \mathfrak{C}}{(\mathfrak{T} \cup \bigcup_{\substack{L \neq L' \\ L \in C}} \{\mathbf{B} \cup \{L\theta\}\})\sigma \mid\mid (\mathfrak{C} \cup \{\langle L', C\rangle\theta\})\sigma}$	where C has a weak connection via f into $\mathbf{B}$ with connection literal L' and mgu σ, $\mathfrak{C}\sigma$ is satisfiable, and θ is a variable renaming of C.
$\dfrac{\mathfrak{T} \cup \{\mathbf{B}\} \mid\mid \mathfrak{C}}{(\mathfrak{T} \cup \bigcup_{L \in C} \{\mathbf{B} \cup \{L\theta\}\})\sigma \mid\mid (\mathfrak{C} \cup \{\langle L', C\rangle\theta\})\sigma}$	where C is a restart clause in S with connection literal L' and mgu σ, $\mathfrak{C}\sigma$ is satisfiable, and θ is a variable renaming of C.
$\dfrac{\mathfrak{T} \cup \{\mathbf{B}\} \mid\mid \mathfrak{C}}{(\mathfrak{T} - \{\mathbf{B}\})\sigma \mid\mid \mathfrak{C}\sigma}$	if σ is an mgu of $\{L, \overline{L'}\} \subseteq \mathbf{B}$ and $\mathfrak{C}\sigma$ is satisfiable.

Table 1. First order tableaux with selection function

Checking validity of a constraint $\mathfrak{C}$ can be expensive. For example, the total A-ordering based on the lexicographical path ordering (LPO) leads to a satisfiability test for $\mathfrak{C}$ which is NP-complete [13].

A compromise is to simply omit the constraints and apply the selection function on uninstantiated clauses. The resulting calculus is still complete, but somewhat less restrictive. It corresponds to the rules in Table 1 if constraints and the conditions imposed on them are removed. In other words, one constructs a tableau in which each extension step *at the time when it is performed* is restricted by a selection function f, but this is not necessarily the case for the final closed tableau.

Such an intermediate calculus is possible, because in tableau calculi each clause used for an extension step, by definition, is an *input clause*. Let us, therefore, call the resulting tableau calculi—in analogy to input resolution—**tableaux with input selection function**.

It turns out that for tableaux with input selection function substitutive selection functions are not required: below we show completeness when the selection function merely is stable wrt variable renaming:

Definition 13. A selection function f is **stable wrt variable renaming** iff $L \in f(C)$ implies $L\theta \in f(C\theta)$ for all variable renamings θ and clauses C.

Obviously, each selection function stable wrt substitutions is also a selection function stable wrt variable renaming. Moreover, the inclusion is proper:

Example 6. Let $f_{\#}$ be the function which selects the first literal (from the left) among the literals containing a maximal number of (constant and function) symbols.

It is easy to show that $f_{\#}$ is stable wrt variable renaming. $f_{\#}$ is not stable wrt substitutions: let $C = P(f(x)) \vee Q(y)$, then $f_{\#}(C) = P(f(x))$, but $f_{\#}(C\{y \leftarrow f(f(u))\}) = Q(f(f(u)))$.

Obviously, one has much greater flexibility in the choice of a suitable selection function for a given problem in this larger class.

For the following completeness proof we have to modify some of our notions. A ground clause is considered to be a sequence of disjunctive connected *indexed literals* $L : I$, where the additional index I can be chosen from any set. It is straightforward to adapt the definition of ground tableau with selection function and the proof of Theorem 9 to indexed clauses.

Theorem 14. *Given an unsatisfiable set S of first order clauses and a selection function f which is stable wrt variable renaming. Then there exists a closed tableau with input selection function f for S.*

For the proof of this theorem we need the following technical lemma. Its easy proof is based on the fact that for unifiable terms idempotent mgus always exist.

Lemma 15. *Given a tableau $\mathfrak{T}$ and a ground substitution ν for $\mathfrak{T}$ such that $\mathfrak{T}\nu$ is closed. Let $\mathbf{B}$ be a branch of $\mathfrak{T}$ closed by two complementary literals $L\nu$ and $L'\nu$ and let σ be the mgu of $L, \overline{L'} \in \mathbf{B}$.*

Then $\mathfrak{T}\sigma\nu = \mathfrak{T}\nu$, and hence: $\mathfrak{T}$ can be closed by applying the mgus of all complementary literals in arbitrary order.

Proof of Theorem 14. By Herbrand's Theorem there exists a finite unsatisfiable set $\widehat{S}$ of ground instances of S.

From S and $\widehat{S}$ we construct the set $\widehat{S}{:}S$ of *indexed* ground clauses as follows: for each $(L_1 \vee \cdots \vee L_n)\sigma \in \widehat{S}$ with $L_1 \vee \cdots \vee L_n \in S$ let $L_1\sigma{:}L_1\theta \vee \cdots \vee L_n\sigma{:}L_n\theta$ be in $\widehat{S} : S$, where θ is a renaming of the variables in $L_1 \vee \cdots \vee L_n$.

We extend f from S to $\widehat{S}{:}S$ by stipulating $\widehat{L}{:}L \in \widehat{f}(\widehat{C}{:}C)$ iff $L \in f(C)$.

$\widehat{f}$ is well-defined, because f is stable wrt variable renaming. $\widehat{S}{:}S$ is a set of indexed ground clauses. From Theorem 9 we know that a closed tableau $\widehat{\mathfrak{T}}$ with selection function $\widehat{f}$ for $\widehat{S}{:}S$ exists.

From $\widehat{\mathfrak{T}}$ we build a tableau $\mathfrak{T}$ with selection function f for S:

1. Obtain $\mathfrak{T}$ from $\widehat{\mathfrak{T}}$ by replacing each literal $\widehat{L} : L$ in $\widehat{\mathfrak{T}}$ by L.
2. If two complementary literals $\widehat{L} : L$ and $\widehat{L'}{:}L'$ close a branch in $\widehat{\mathfrak{T}}$ with mgu σ, then apply σ to $\mathfrak{T}$. Lemma 15 guarantees that the corresponding branch in $\mathfrak{T}\sigma$ is closed, too.

The result is a closed tableau that can be constructed with the rules of Table 1, neglecting constraints. Thus it is a tableau with input selection function f.

□

6 Extensions and Related Calculi

6.1 Theory Connections

With theory reasoning we mean a general method to integrate theories such as the equality theory (efficiently) into deductive systems. Various sound and complete methods have been developed for several calculi including resolution, the connection method, and tableaux. All of the more efficient methods for tableau-like calculi share two main ideas: (i) an extension of the notion of connection and connection unifiers to theory connections and theory unifiers and (ii) the partition of the automated deduction system into a general purpose foreground reasoner and a theory-specific background reasoner.

In the case of **total theory reasoning** the background reasoner calculates theory unifiers for a given input set of formulas given by the foreground reasoner, whereas in **partial theory reasoning** the background reasoner in addition derives new formulas, so-called *residues*, that have to be used by the foreground reasoner. In our context we are only interested in total theory reasoning and we give only a very brief overview of the necessary constituents of theory reasoning. [4] is a detailed survey of theory reasoning in tableau calculi.

In general, every satisfiable set of first order clauses defines a theory $\mathcal{T}$. The notions and semantics of an interpretation, the satisfiability of formulas, etc. are restricted relative to a given theory.

Definition 16. A **theory** $\mathcal{T}$ is a satisfiable set of first order clauses.

An interpretation I is a **$\mathcal{T}$-interpretation** iff I satisfies $\mathcal{T}$.

A formula ϕ is **$\mathcal{T}$-satisfiable** iff ϕ is satisfied by a $\mathcal{T}$-interpretation; ϕ is **$\mathcal{T}$-unsatisfiable** otherwise.

Example 7. The clause set $\mathcal{O} = \{\neg x < x,\ \neg x < y \vee \neg y < z \vee x < z\}$ defines the theory of strict orderings.

For a given signature Σ the equality theory $\mathcal{E}_\Sigma$ can be defined by the axioms of reflexivity, symmetry, transitivity, and monotonicity for function and predicate symbols of the equality predicate $\approx$. $\mathcal{E}_\Sigma$ is finite iff Σ is.

The notions of complementary literal and connection unifier are relativized to theories as well.

Definition 17. Let $\mathbf{B}$ be a set of literals. $\mathbf{B}$ is **$\mathcal{T}$-complementary** iff the existential closure of the conjunction of the members of $\mathbf{B}$ is $\mathcal{T}$-unsatisfiable.

A substitution σ is a **$\mathcal{T}$-unifier** for $\mathbf{B}$ iff $\mathbf{B}\sigma$ is $\mathcal{T}$-complementary. If, moreover, $\mathbf{B}\sigma$ is a minimal set of $\mathcal{T}$-complementary literals, i.e. no proper subset of it is $\mathcal{T}$-complementary, then $\mathbf{B}$ is a **$\mathcal{T}$-connection** and σ is a **$\mathcal{T}$-connection unifier for $\mathbf{B}$**. As usual, a **$\mathcal{T}$-mgu** is a most general $\mathcal{T}$-unifier.

The minimality condition for theory connections is due to [2], where a connection calculus with theories is presented for the first time. In [16] $\mathcal{T}$-connections are used without minimality condition and the rôle of $\mathcal{T}$-connections is taken by a so-called set of complete $\mathcal{T}$-connections. The latter gives more flexible control over the $\mathcal{T}$-unifiers that have to be calculated to obtain a complete calculus. This can be of importance in theories where decidability is not guaranteed for $\mathcal{T}$-connections but for complete sets of $\mathcal{T}$-connections. Our notion of $\mathcal{T}$-connections has the advantage that it leads straightforwardly to a theory version of tableaux with selection function. It can be adapted easily to the terminology of [16].

Example 8. In the equality theory $\mathcal{E}$ the literal set $\mathbf{B} = \{a \approx b,\ p(a),\ \neg p(b),\ p(x)\}$ is $\mathcal{E}$-complementary, but it is no $\mathcal{E}$-connection. The substitution $\sigma = \{x \leftarrow a\}$ is an $\mathcal{E}$-unifier for $\mathbf{B}' = \{a \approx b,\ \neg p(b),\ p(x)\}$ which is an $\mathcal{E}$-connection.

If $\mathcal{T} = \emptyset$, then the standard notions of complementary literals, connection, and connection unifier are instances of the above definitions.

To build theories into tableaux with selection function we must modify our definition of connection via a selection function:

Definition 18. A set S of instances of first order clauses has a **$\mathcal{T}$-connection via** f iff there is a $\mathcal{T}$-connection $\mathbf{B}$ with unifier σ such that $L \in f(C)$ with $C \in S$ for all $L \in \mathbf{B}$.

Depending on the given theory, more than two clauses may be involved in a connection via f. Instead of defining $\mathcal{T}$-restart clauses directly, we generalize the notion of a connection into a tableau branch, i.e., into a set of literals.

Definition 19. Given a set of literals $\mathbf{B}$, a selection function f, a theory $\mathcal{T}$, and a set S of first order clauses, then a clause $C \in S$ has a **$\mathcal{T}$-connection into $\mathbf{B}$** iff there is a set $S' \subseteq \{C | C$ is an instance of a clause in S or $C \in \mathbf{B}\}$ with $C \in S'$ such that S' has a $\mathcal{T}$-connection via f.

$f(C)$ is a **connection literal** of S'.

If S' contains only clauses of S then every $C \in S'$ is (an instance of) a **restart clause** and if $S' \cap \mathbf{B} \neq \emptyset$ then every $C \in (S' - \mathbf{B})$ has a **weak connection** into $\mathbf{B}$.

Table 2 shows $\mathcal{T}$-tableaux with selection function. The main difference to the previous version is the uniform handling of extension steps and, as a consequence, there is no closure involved in any extension step. In theory reasoning it does not make sense to separate out the case, where an immediate closure is possible, because in general a tableau branch may have to be extended more than once before any new open branch can be closed with a theory unifier.

Theorem 20. *Given a theory $\mathcal{T}$, a $\mathcal{T}$-unsatisfiable set S of first order clauses and a selection function f which is stable wrt substitutions. Then there exists a closed $\mathcal{T}$-tableau with selection function f for S.*

It is possible to modify the previous completeness proofs. Lifting to first order logic is possible with the help of a theory version of Herbrand's Theorem [15, 16].

$\dfrac{\mathfrak{T} \cup \{\mathbf{B}\} \mid\mid \mathfrak{C}}{(\mathfrak{T} \cup \bigcup_{L \in C} \{\mathbf{B} \cup \{L\theta\}\})\sigma \mid\mid (\mathfrak{C} \cup \{\langle L', C\rangle\theta\})\sigma}$	if C has a $\mathcal{T}$-connection via f into $\mathbf{B}$ with connection literal L' and $\mathcal{T}$-mgu σ, $\mathfrak{C}\sigma$ is satisfiable, and θ is a variable renaming of C.
$\dfrac{\mathfrak{T} \cup \{\mathbf{B}\} \mid\mid \mathfrak{C}}{(\mathfrak{T} - \{\mathbf{B}\})\sigma \mid\mid \mathfrak{C}\sigma}$	if there is a $\mathcal{T}$-connection $\mathbf{B}' \subseteq \mathbf{B}$ with mgu σ and $\mathfrak{C}\sigma$ is satisfiable.

Table 2. $\mathcal{T}$-tableaux with selection function

The background reasoner is used for two tasks in this calculus: First, it is used to calculate theory connections and, second, it is used to check for closure. Both can be done by calculating theory unifiers.

The calculation of theory unifiers can be very expensive and even undecidable. Without any restriction one has to consider each set $\mathbf{B}$ with $C \in S$ for all $L \in \mathbf{B}$. Therefore, a restriction on certain literals as, for example, in tableaux with selection function, is extremely useful.

6.2 Restart Model Elimination

In [3] a modification of Model Elimination [11] is introduced which bears some similarities to tableaux with selection function. This calculus, called Restart Model Elimination, also uses a selection function f to restrict connections to those clauses which contain literals selected by f.

In contrast to tableaux with selection function, (i) only *positive* literals of a clause are selected by f and are considered as connection literals and (ii) a clause can extend a branch $\mathbf{B}$ only if it has a **strong connection via f into $\mathbf{B}$**, i.e. if it has a connection to the *leaf*-literal of $\mathbf{B}$. Furthermore strong connections are restricted to *negative* leaf literals.

Like in tableaux with selection function [8] imposing the strong connection condition leads to an incomplete calculus. To avoid this negative clauses are admitted for restarts.

The strong connection condition also implies that regularity in restart model elimination has to be relaxed in order to obtain a complete calculus. A branch $\mathbf{B}$ needs only be regular wrt all *positive* literals and within each *block* of $\mathbf{B}$, where a block is defined as a sequence of literals in $\mathbf{B}$ which lies totally between two subsequent applications of the restart rule.

The completeness proof for restart model elimination is based on the literal-excess method. Unfortunately, it gives no clue on how the calculus can be modified such that the selection function does yield negative literals as well.

7 Implementation and Application

$\mathcal{T}$-Tableaux with (input) selection function were implemented in $_3T^AP$ [5], an automated theorem prover for sorted multiple-valued full first order logic with equality.

(Equality) theory reasoning is used to close a branch if the problem contains equalities, but is not used to calculate theory connections. Instead, the test is performed with a weaker condition (giving more candidates for theory connections than necessary) that can be computed in polynomial time [18]. This latter method is, of course, restricted to selected literals.

Completely unrestricted extensions of tableau branches with restart clauses is problematic, because such clauses might be totally independent from the branch on which they are used. It turns out that a good heuristic for avoiding extensions with potentially useless restart clauses is to prefer such restart clauses that have a connection, not necessarily via the selection function, into the current branch.

$_3T^AP$ uses a further technique due to [14] to avoid redundant extension steps in hindsight: if a subtableau below a node n can be closed without using the literal of this node, then all open branches produced by *ClauseOf*(*Predecessor*(n)) are *pruned*, i.e. closed immediately.

$$\begin{tikzcd}
 & \Delta \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\
\cup & \cap & - \\
 & \in \arrow[ul, no head] \arrow[u, no head] \arrow[ur, no head] &
\end{tikzcd}$$

Fig. 2. Hierarchy of predicate symbols

We have found that tableaux with selection function work quite well when the formulas of a problem to be solved are in some sense hierarchical. Mathematical theories, for example, are based on definitions which themselves are based on definitions of more primitive notions. Most of these definitions introduce new predicate or function symbols. Thus, these symbols form a hierarchy.

Let us consider a first order axiomatization of a fragment of set theory. Symmetrical difference Δ between two sets can be defined by the union $\cup$, intersection $\cap$, and set difference $-$, all of which are based on the membership predicate $\in$.

Fig. 2 shows the hierarchy induced by these definitions. Formally, such a hierarchy is an irreflexive, partial ordering $<_H$ on predicate, function and constant symbols. In our example we have $\in \; <_H \; \cup, \, \cap, \, - \; <_H \; \Delta$.

Definition 21. Let $<$ be any irreflexive, partial ordering on function and constant symbols. Then for any terms $s = f(s_1, \dots, s_n)$, $t = g(t_1, \dots, t_m)$ we define $s \prec_L t$ to hold iff both of the following conditions hold:

1. $f < g$ or $f = g$, $n = m$, and $t_i \prec_L s_i$ for $i = 1, \dots, n$;
2. the variables of s are a subset of the variables of t.

$\prec_L$ can be extended to an A-(L-)ordering in an obvious way provided that $<$ is an ordering on predicate symbols as well.

In the set theory example the ordering $<_H$ gives a good correspondence between the literals ordered by $\prec_L$ and the hierarchy of set theory. $\prec_L$ often prevents literals from unrelated hierarchies to be unifiable and, as a consequence, a clause C has no $\prec_L$-connection into a branch which only contains literals from a hierarchy that is not reachable from a hierarchy of the maximal literals of C.

Note that even if the ordering does not perfectly reflect the hierarchy within a problem (or if there is no real hierarchy), completeness of the calculus still guarantees a proof can be found, though proof search might not be influenced as favorably.

name	ordering	R	CB	${}_3T^AP$[s]	Otter[s]
set001	$\prec_L$	30	11	1.18	0.06
	none	10	4	0.12	
set002	$\prec_L$	546	206	27.82	2.32
	none			> 500	
set003	$\prec_L$	12	5	0.26	0.08
	none	48	8	0.66	
set004	$\prec_L$	12	5	6.12	0.05
	none	48	8	0.72	
set005	any			> 500	> 300
set006	$\prec_L$	650	40	52.15	0.05
	none			> 500	
set007	any			> 500	> 300
set008	$\prec_L$	1406	536	102.29	0.96
	none			> 500	
set009	$\prec_L$	22	10	0.27	0.47
	none	22	10	0.17	

Fig. 3. Results for TPTP-problem `SET`

Fig. 3 shows some results for a few problems from the TPTP problem class `SET` [19]. The calculus used are tableaux with input selection function. None of the problems is formulated with equality.

As the time needed for refutation is only a rough measure of the complexity of a proof, we give the number of rule applications (R)[4] and of branches (CB) in the closed tableau. Lines corresponding to the smallest proof are darkly shaded. The final column shows the times Otter [12] needs for a proof.[5] Otter is usually faster, but the proofs it generates are typically longer, sometimes drastically. Also the basic speed of the experimental system ${}_3T^AP$ is slower than that of Otter by a factor of several hundred.

8 Conclusion

We introduced tableaux with selection function which are a generalization of A-ordered tableaux. In addition we showed that lifting is possible with respect to a large class of selection functions which makes it easy to find a suitable selection function for certain problems.

It turns out that theories can be build into tableaux with selection function using the concepts of theory connection and theory reasoning.

Our first experiments with an implementation of this calculus show promising results with problems that have a hierarchical structure.

In [15] tableaux with selection function are extended to the non-clausal case and also an implementation with constraints is discussed.

Acknowledgements

We owe Peter Baumgartner and an anonymous referee many useful comments on a previous version of this paper.

[4] In ${}_3T^AP$ clauses are quantified explicitly. To extend a tableau branch with a clause several γ and β-rule applications have to be made. Because of this, R is usually several times larger than the actual number of extension steps.

[5] The figures were obtained with Otter version 3.0.4 in June 1996. They are available via `ftp://info.mcs.anl.gov/pub/Otter/www-misc/otter304-tptp121.txt.gz`. Both systems were run on roughly comparable machines.

References

1. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. of Logic and Computation*, 4(3):217–247, 1994.
2. P. Baumgartner. A Model Elimination Calculus with Built-in Theories. In H.-J. Ohlbach, editor, *Proceedings of the 16-th German AI-Conference (GWAI-92)*, pages 30–42. Springer-Verlag, 1992. LNAI 671.
3. P. Baumgartner and U. Furbach. Model Elimination without Contrapositives and its Application to PTTP. *J. of Automated Reasoning*, 13:339–359, 1994.
4. B. Beckert. Equality and other theory inferences. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods.* Kluwer, Dordrecht, to appear, 1997.
5. B. Beckert, R. Hähnle, P. Oel, and M. Sulzmann. The tableau-based theorem prover $_{3}T^{A}P$, version 4.0. In *Proceedings, 13th International Conf. on Automated Deduction (CADE), New Brunswick, NJ, USA*, LNCS 1104, pages 303–307. Springer, 1996.
6. C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Methods for the Decision Problem.* LNAI 679. Springer-Verlag, 1993.
7. M. C. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, New York, second edition, 1996.
8. R. Hähnle and S. Klingenbeck. A-ordered tableaux. *J. of Logic and Computation*, 6(6):819–834, 1996.
9. S. Klingenbeck and R. Hähnle. Semantic tableaux with ordering restrictions. In A. Bundy, editor, *Proc. 12th Conf. on Automated Deduction CADE, Nancy*, volume 814 of *LNCS*, pages 708–722. Springer-Verlag, 1994.
10. R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A high-perfomance theorem prover. *J. of Automated Reasoning*, 8(2):183–212, 1992.
11. D. Loveland. Mechanical theorem proving by model elimination. *Journal of the ACM*, 15(2), 1968.
12. W. W. McCune. *OTTER 3.0 Reference Manual and Guide.* Argonne National Laboratory/IL, USA, 1994.
13. R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, 1993.
14. F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *J. of Automated Reasoning*, 4:69–100, 1988.
15. C. Pape. Vergleich und Analyse von Ordnungseinschränkungen für freie Variablen Tableau. Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, May 1996. In German. Available via anonymous ftp to `ftp.ira.uka.de` under `pub/uni-karlsruhe/papers/techreports/1996/1996-30.ps.gz`.
16. U. Petermann. How to build in an open theory into connection calculi. *Journal on Computers and Artificial Intelligence*, 2:105–142, 1992.
17. R. M. Smullyan. *First-Order Logic.* Dover Publications, New York, second corrected edition, 1995. First published 1968 by Springer-Verlag.
18. M. Sulzmann. Ausnutzung von KIV-Spezifikationen in $_{3}T^{A}P$. Studienarbeit, Fakultät für Informatik, Universität Karlsruhe, July 1995.
19. G. Sutcliffe, C. Suttner, and T. Yemenis. The TPTP problem library. In A. Bundy, editor, *Proc. 12th Conf. on Automated Deduction CADE, Nancy*, LNAI, pages 252–266. Springer, 1994.