Proof and Model Generation with Disconnection Tableaux

Reinhold Letz and Gernot Stenz

Institut für Informatik
Technische Universität München
D-80290 Munich, Germany
{letz,stenzg}@in.tum.de

Abstract. We present the disconnection tableau calculus, which is a free-variable clausal tableau calculus where variables are treated in a non-rigid manner. The calculus essentially consists of a single inference rule, the so-called linking rule, which strongly restricts the possible clauses in a tableau. The method can also be viewed as an integration of the linking rule as used in Plaisted's linking approach into a tableau format. The calculus has the proof-theoretic advantage that, in the case of a satisfiable formula, one can characterise a model of the formula, a property which most of the free-variable tableau calculi lack. In the paper, we present a rigorous completeness proof and give a procedure for extracting a model from a finitely failed branch.

1 Introduction

In the last years considerable progress has been made in the development of tableau-based proof systems for automated deduction. While the tableau framework always was very influential in proof theory and in the development of logics, particularly non-classical ones, it had almost no influence on automated deduction in classical logic. This changed about ten years ago, when it was recognised that it is more natural to view automated deduction calculi like model elimination or the connection method as particular refinements of the tableau calculus. The central new feature of those refinements is the active use of connections as a control mechanism for guiding the proof search. This view had a very fruitful effect on the research in the area. In the meantime, many proof systems developed in automated deduction have been reformulated in tableau style. Furthermore, new calculi have been developed which are based on tableaux and integrate connections in different manners. Currently, some of the most powerful theorem proving systems are based on tableaux with connections.

The main objective in the use of connections is to avoid the blind application of the γ-rule used in Smullyan's tableau system [Smu68]. Instead one uses the substitutions that are induced by the connections in the formula. Since, in general, such substitutions are not ground, this requires the introduction of free variables in tableaux. The question then is how to treat these free variables? The overwhelming majority of the developed free-variable tableau calculi (e.g.,

R. Nieuwenhuis and A. Voronkov (Eds.): LPAR 2001, LNAI 2250, pp. 142–156, 2001.

[Fit96],[LMG94]) treat free variables as *rigid* in the sense that a free variable x is just treated as a placeholder for a single yet unknown (ground) term t. And during the tableau construction this term is successively approximated by applying substitutions to the variable x. The resulting tableau calculi are *destructive* [Let99] in the sense that they not only have expansion inferences as in Smullyan's system but also modification inferences. The negative side-effect is that these calculi lack a fundamental property of Smullyan's tableau method, namely, the possibility of branch saturation. But branch saturation is one of the most appealing features of traditional tableau calculi, since it permits the extraction of models for certain satisfiable formulae. With the new calculi, model generation is impossible. Also, the decision power of free-variable tableau systems is significantly weakened. [1] These negative consequences of free-variable tableau calculi have not sufficiently been recognised so far.

In [Bil96], an alternative clausal free-variable framework is presented, the *disconnection method*. In its tableau format, the method treats a free variable not as rigid but as universally quantified wrt. the respective clause in the tableau. Unfortunately, in [Bil96] many questions concerning the properties of the method remain unanswered. So the completeness proof is only sketched and the problem of model generation is not addressed. In the current paper we give a rigorous presentation of the *disconnection tableau calculus*, as we term it. This includes an elaborated completeness proof. The main emphasis of this paper is on issues of model generation. We develop some new concepts like an *instance-preserving enumeration* of a tableau branch and its associated *Herbrand path*, which permit the convenient extraction of Herbrand models from open saturated tableau branches. As we will demonstrate, these concepts permit a compact representation of Herbrand models.

The paper is organised as follows. Following this introduction, Section 2 motivates the disconnection calculus, defines the required notation and provides an intuitive as well as a formal description of the disconnection calculus. Then, in Section 3 we show how the branch saturation property of our calculus can be used to extract models from saturated branches. Subsequently, the completeness of the disconnection calculus can be proven in a straightforward manner in Section 4. Section 5 shows how the calculus can be strengthened wrt. deductive power and, finally, in Section 6 we give an assessment of our work and address future perspectives.

2 The Disconnection Tableau Calculus

The disconnection tableau calculus was first developed in [Bil96]. Essentially, this proof system can be viewed as an integration of Plaisted's *clause linking* method [PL92] into a tableau control structure. Therefore, in order to comprehend the

[1] For example, Fitting's free-variable tableaux [Fit96] provide no decision procedure for the Bernays-Schoenfinkel class [DG79], unless very rough external criteria like maximal path lengths are used, whereas with a slight modification Smullyan's original tableau method decides the class (see [Let99]).

disconnection tableau calculus, it is instructive to first briefly review the most important features of the clause linking method. This method is in the spirit of the first theorem proving procedures developed in the sixties, which where direct applications of Herbrand's approach to proving the completeness of first-order logic (see [DP60]). Such *Herbrand procedures* consist of two subprocedures, a generator for sets of ground instances and a propositional decision procedure. For some decades this methodology was not pursued in automated deduction, mainly because no guided method of ground instance generation existed. The linking mechanism (and its hyperlinking variant) integrate *unification* into the process of ground instantiation, thus making the ground instantiation more controlled.

Before describing the proof method, we need to introduce some terminology. As usual, a *literal* is an atomic formula or a negated atomic formula. A *clause* is a finite set of literals; occasionally, we will also display clauses as disjunctions of literals. A *literal occurence* is any pair $\langle c, l\rangle$ where c is a clause, l is a literal, and $l \in c$.

Definition 1 (Link, linking instance). *Given two literal occurences $\langle c, l\rangle$ and $\langle c', \neg l'\rangle$, if there is a unifier σ of l and $\neg l'\tau$ where τ is a renaming of c' such that $c'\tau$ is variable-disjoint from c, then the set $\ell = \{\langle c, l\rangle, \langle c', \neg l'\rangle\}$ is called a* connection *or* link (between the clauses c and c'). *The clause $c\sigma$ is a* linking instance *of c* wrt. *the connection ℓ.* [2]

Instead of guessing arbitrary ground instances of clauses as in the theorem proving procedures of the sixties, one can restrict oneself to linking instances (or hyperlinking instances) wrt. the connections in the iteratively increasing clause set. Additionally, from time to time, the current clause set is tested for propositional unsatisfiability. Before this test, the clause set is *grounded*, i.e., all variables are replaced by one and the same constant. If the unsatisfiability test succeeds, then this obviously demonstrates the unsatisfiability of the original clause set.

One of the strengths of the clause linking method is that it avoids the so-called *duplication problem* in resolution, which means that the unresolved parts of a parent clause c are duplicated over and over again in resolvents derived from c. One of the main weaknesses of the clause linking method is that the propositional decision procedure is separated from the generation of the linking instances. And the interfacing problem between the two subroutines may lead to tremendous inefficiencies. The disconnection tableau method provides an intimate integration of the two subroutines. This is achieved by embedding the linking process into a tableau guided control structure. As a result of this embedding, no separate propositional decision procedure is needed and the number of produced linking instances can be significantly reduced. For describing the disconnection tableau method, we need the following notions.

[2] There is also a *hyperlinking* variant, which requires that *each* literal l_i in the clause c has a link with substitution σ_i; the hyperlinking variant of c is $c\sigma$ where σ is the composition of the substitutions σ_i.

Definition 2 (Path). *A* path through *a clause set S is any total mapping $P : S \to \bigcup S$ with $P(c) \in c$, i.e., a set containing exactly one literal occurence $\langle c, l\rangle$ for every $c \in S$. The* set of literals *of P is the set of all l such that $\langle c, l\rangle$ in P. A path P is* complementary *if it contains two literal occurences of the form $\langle c, l\rangle$ and $\langle c', \neg l\rangle$, otherwise P is called* consistent *or* open.

Definition 3 (Tableau). *A* tableau *is a (possibly infinite) downward tree with literal labels at all tree nodes except the root. Given a clause set S, a* tableau for *S is a tableau in which, for every tableau node N, the set of literals $c = l_1, \ldots, l_m$ at the immediate successor nodes $N_1, \ldots, N_m$ of N is an instance of a clause in S; for every N_i $(1 \leq i \leq m)$, c is called* the clause *of N_i. With every tableau node N_i the literal occurence $\langle c, l_i\rangle$ will be associated. Furthermore, a* branch *of a tableau T is any maximal sequence $B = N_1, N_2, N_3, \ldots$ of nodes in T such that N_1 is an immediate successor of the root node and any N_{i+1} is an immediate successor of N_i. With every branch B we will associate a path P_B, viz., the set of literal occurences associated with the nodes in B.*

The specific feature of Billon's disconnection tableau calculus [Bil96] is that it starts the construction of a tableau wrt. to a path through the set S of input clauses, which we call the *initial path* P_S. P_S can arbitrarily chosen but remains fixed throughout the entire tablau construction. The disconnection tableau calculus consists of a single complex inference rule, the so-called *linking rule*.

Definition 4 (Linking rule). *Given an initial path P_S and a tableau branch B with two literal occurrences $\langle c, l\rangle$ and $\langle c', \neg l'\rangle$ in $P_S \cup P_B$, such that $\ell = \{\langle c, l\rangle, \langle c', \neg l'\rangle\}$ is a connection with unifier σ, then*

1. *expand the branch B with a linking instance wrt. ℓ of one of the two clauses, say, with $c\sigma$,*
2. *below the node labeled with $l\sigma$, expand the branch with a linking instance wrt. ℓ of $c'\sigma$.*

In other terms, we perform a clause linking step and attach the coupled linking instances below the leaf node of the current tableau branch. Afterwards, the respective connection must not be used any more below the node on any extension of B, thus "disconnecting" the connected literals. This last feature explains the naming disconnection tableau calculus for the proof method.

As branch closure condition, the standard tableau closure condition is not sufficient, but the same notion as employed in the clause linking method can be used.

Definition 5 (Closure). *A tableau branch B is* closed *if its associated path is complementary. B is* closed wrt. a term t, t-closed *for short, if B becomes closed when all the variables in the literals on B are substituted by the term t. B is* universally closed, $\forall$-closed *for short, if B is t-closed for any term t; if B is not $\forall$-closed it is called* open.

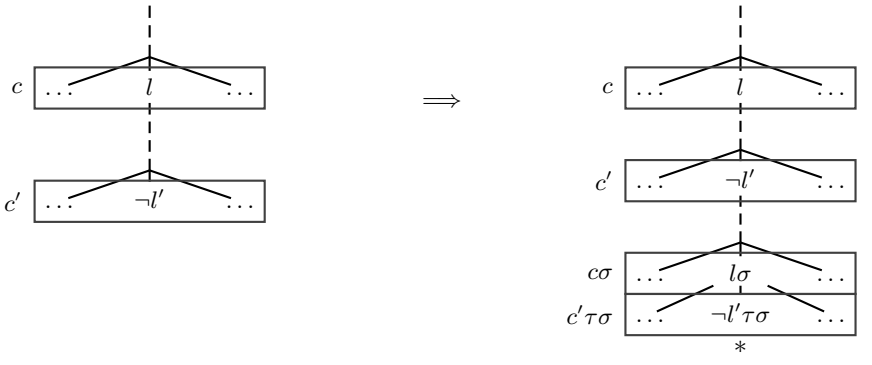

Fig. 1. Illustration of a linking step.

That is, a branch of a tableau is ∀-closed if it contains two literals l and k such that $l\theta$ is the complement of $k\theta$ where θ is a substitution identifying all variables in the tableau. Applied to the tableau in Figure 1, this means that after the linking step at least the middle branch is ∀-closed, as indicated with an asterisk.

The *disconnection tableau calculus* then simply consists of the linking rule plus a rule for the selection of an initial path applicable merely once at the beginning of the proof construction. The calculus is sound and complete for any initial path selection, i.e., a clause set S is unsatisfiable if and only if, for any initial path P_S, there exists a finite disconnection tableau T for S and P_S such that all branches of T are ∀-closed (or t-closed for an arbitrary term t). Both notions of closure can be used, but using t-closure may lead to shorter proofs if the respective term is selected in a fortunate manner. We will present all our results for the weaker notion of ∀-closure, since then they automatically hold for the t-closure case.

Definition 6 (Disconnection tableau sequence). *Given a set of clauses S and an initial path P_S through S, a* disconnection tableau sequence *for S and P_S is any (possibly infinite) sequence $\mathcal{T} = T_0, T_1, T_2, \ldots$ satisfying the following properties:*

- *T_0 is the trivial tableau consisting just of the root node,*
- *any T_{i+1} in $\mathcal{T}$ can be obtained from T_i by application of a linking step.*

Any tableau in $\mathcal{T}$ is called a disconnection tableau *for S and P_S.*

Figure 2 displays a ∀-closed disconnection tableau for the clause set consisting of the transitivity clause $\{P(x,z), \neg P(x,y), \neg P(x,z)\}$, and the three unit clauses $\{P(c,b)\}$, $\{P(a,b)\}$, and $\{\neg P(a,c)\}$. As usual, upper-case letters denote predicate symbols, lower-case letters from the end of the alphabet like x, y, z denote variables, the other function symbols. The selected initial path passes through every clause at the first literal position. In the figure, we have shown the set of input clauses, with the initial path marked, in a box at the beginning

of the actual tableau. We have also depicted the connections as arcs, and immediately before every linking instance c in the tableau we have annotated from which connection c derives.

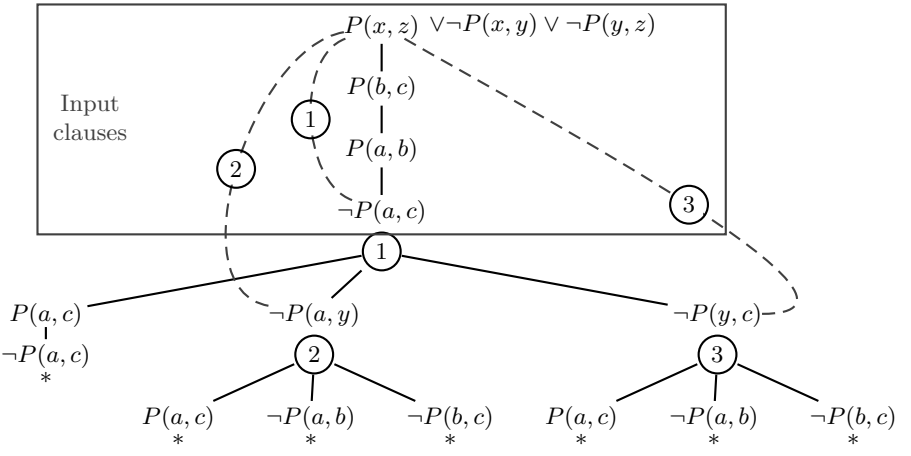

Fig. 2. A ∀-closed disconnection tableau.

As usual, a pure calculus is nothing without refinements. The most important of these for the disconnection tableau calculus is the following.

Definition 7 (Variant freeness). *A disconnection tableau T is* variant-free *if, for no node N with clause c in T, there exists an ancestor node N' in T with clause c' such that c and c' are* variants *of each other, i.e., c can be obtained from c' by renaming its variables. (It should be noted that this restriction does not extend to the initial path.)*

With this refinement the disconnection framework can be used for proving the satisfiability of certain clause sets, and from the respective tableau a model for S can be extracted, as we shall demonstrate in the next section. In the following, we assume all disconnection tableaux to be variant free. In order to illustrate the decision power of the disconnection calculus, we give a very simple satisfiable clause sets, which is very hard for other calculi. For example, none of the generally successful theorem provers based on resolution or model elimination terminate. The clause set is displayed on the left-hand side of Figure 3. It is obtained from the clause set displayed in Figure 2 by two simple modifications. First, we have exchanged the positions of the constants b and c in the unit clause $P(b, c)$, which makes the set satisfiable. Furthermore, we have switched the signs of the literals, which does evidently not change the logic. [3]

[3] But it significantly changes the behaviour of resolution systems, for example. Without the second modification, since the set is Horn, we can use unit resolution, which terminates on this set.

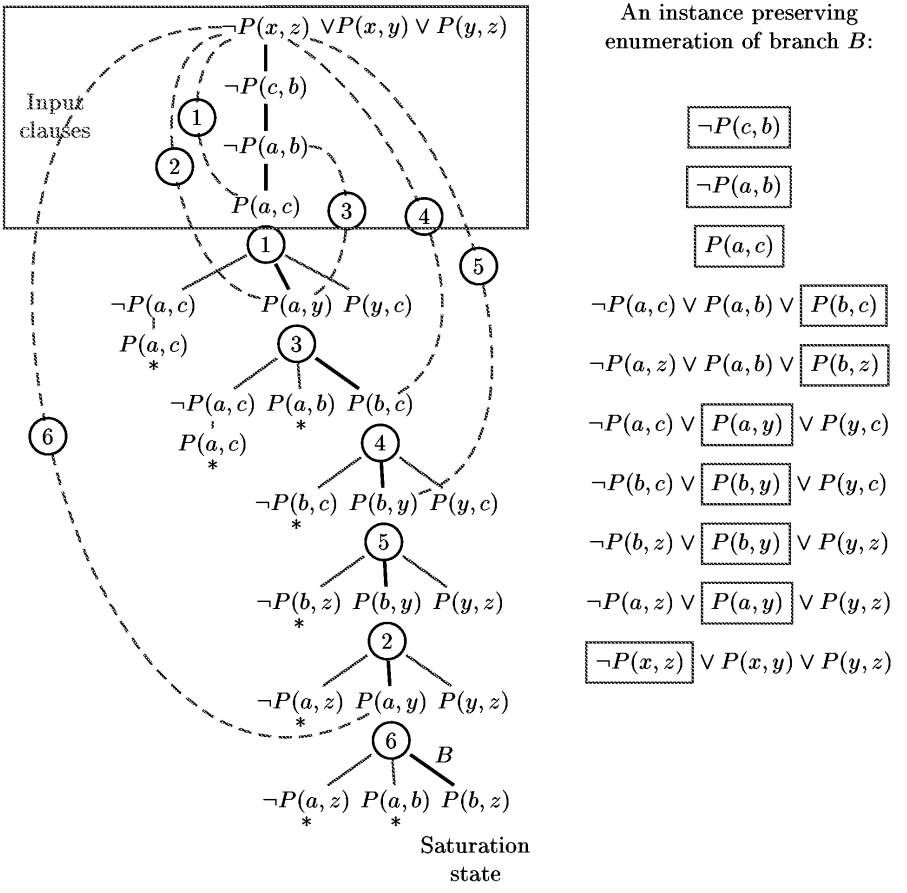

Fig. 3. A saturated open branch and an instance-preserving enumeration.

When considering the branch B in the tableau, we observe that any further linking step on B produces a variant of a clause already on the branch. This proves the satisfiability of the set of input clauses. In fact, the disconnection tableau method is a decision procedure for the class of finite clause sets without function symbols of arity > 0, which corresponds to the Bernays-Schoenfinkel class [DG79].

3 Model Extraction from Disconnection Tableaux

In [Bil96] only a sketch of a completeness proof is given and the problem of extracting a model from the failed construction of a $\forall$-closed tableau is left open. In this paper, we address both topics. We present a full and self-contained completeness proof which is different from the approach in [Bil96]. For example,

we do not need auxiliary concepts like the *connective depth* as used in [Bil96]. Furthermore, we show how a Herbrand model can be extracted from such a failed tableau construction. Since the completeness proof uses the model property, we start with the part concerned with model generation.

We employ the following notions. The *Herbrand set* S^* of a set of clauses S is defined as usual, i.e., a clause c is in S^* if and only if c is a ground instance of a clause in S wrt. the Herbrand universe of S. As usual, a *Herbrand interpretation* for S can be viewed as a set I of literals containing, for each atom A in the Herbrand base of S, exactly one of A or $\neg A$. The set I is a *Herbrand model* for S if there is a path P through the Herbrand set S^* of S such that the set of literals in P is a subset of I. For completeness and model generation, it is even more elegant to work with *partial* Herbrand interpretations and models. Any subset I of an Herbrand interpretation for S is called a *partial Herbrand interpretation. I* is a *partial Herbrand model* for S if all Herbrand interpretations for S which are supersets of I are Herbrand models for S. Obviously, the following proposition holds.

Proposition 1. *The set of literals I_P of any consistent path P through the Herbrand set S^* of a set of clauses S is a partial Herbrand model for S, and any partial Herbrand interpretation $I \supseteq I_P$ is a partial Herbrand model for S.*

An appealing feature of the disconnection tableau calculus is that it provides us with a natural representation of partial Herbrand models for any saturated tableau branch B which is not $\forall$-closed or t-closed. Under certain circumstances this representation may be more compact than just explicitly specifying a partial Herbrand model as a set of ground literals. The idea which permits such a representation is to enumerate the literal occurences of a branch in an order which respects the instance relation of the contained clauses. This is formalised with the following notions.

Definition 8. *Given any set of literal occurences P, an* instance-preserving enumeration of P *is any sequence $E = L_1, L_2, L_3, \ldots$ in which exactly the elements of P occur and in such an order that, for any $L_i = \langle c_i, l_i \rangle$ and $L_j = \langle c_j, l_j \rangle$, when c_i is a proper instance of c_j, then $i < j$. Let, furthermore, S^* be the Herbrand set of the clauses occuring in P. Then, with any instance-preserving enumeration E of P, we associate a path P^* through S^* as follows: a literal occurence $\langle c, l \rangle$ is in P^* if and only if there is a literal occurence $L_k = \langle c_k, l_k \rangle$ in E and there is no $L_j = \langle c_j, l_j \rangle$ with $j < k$ in E such that c is an instance of c_j; we term L_k* the minimal matcher *in E of the clause c. The path P^* is called* the Herbrand path *of E and an* Herbrand path *of P.*

In case the Herbrand path P^* of such an enumeration E is consistent, then the set of literals of P^* is a partial Herbrand model. As an example, consider the sequence E of literal occurences displayed on the right-hand side of Figure 3. E is an instance-preserving enumeration of the open tableau branch on the left-hand side. The set of literals of its Herbrand path is

$$\{\neg P(c,b), \neg P(a,b), P(a,c), P(b,c), P(b,a), P(b,b), P(a,a), \neg P(c,a), \neg P(c,c)\}$$

which is an Herbrand interpretation for S. This technique has similarities with Baumgartner's exception rule [Bau00] and with Peltier's method [Pel99] of annotating clauses with equational constraints. However, due to lack of space we cannot present a detailed comparison in this paper.

For the generation of models, we have to formalise the notion of a tableau which cannot be $\forall$-closed. In contrast to tableau calculi like connection tableaux [LMG94] or Fitting's free-variable tableaux [Fit96], the disconnection tableau calculus is *nondestructive*, i.e., any inference step performs just an expansion of the previous tableau and any tableau of a disconnection tableau sequence $\mathcal{T}$ contains all tableaux with smaller indices in the sequence as initial segments. This permits that we can form the union $\bigcup\mathcal{T}$ of all tableaux in $\mathcal{T}$. In the case of a finite sequence, $\bigcup\mathcal{T}$ is just the last element in the sequence. In general, $\bigcup\mathcal{T}$ may be an infinite tableau. We call $\bigcup\mathcal{T}$ *the* tableau of the sequence $\mathcal{T}$.

Definition 9 (Saturation). *A branch B in a (possibly infinite) tableau T is called* saturated *if B is open and cannot be extended in a variant-free manner by the linking rule. The tableau $\bigcup\mathcal{T}$ of a disconnection tableau sequence $\mathcal{T}$ is called* saturated *if either all its branches are closed or it has a saturated branch.*

As a final preparation of the main result of this section, we mention the following straightforward property of disconnection tableaux.

Lemma 1. *Let B be a saturated branch in a disconnection tableau. If B contains a node N_1 with literal l and clause c and a node N_2 with literal $\neg l'$ and clause c' such that $\ell = \{\langle c, l\rangle, \langle c', \neg l'\rangle\}$ is a connection with unifier σ which is no renaming substitution, then B must contain a linking instance of c or c' wrt. ℓ.*

Proposition 2 (Model characterisation). *Given a branch B of the tableau of a disconnection tableau sequence for a clause set S and an initial path P_S, let P be the path associated with B. If B is saturated, then the set of literals I of any Herbrand path P^* of $P \cup P_S$ is a partial Herbrand model for S.*

Proof. First, we show that P^* is a consistent Herbrand path through S^*. Assume, indirectly, that P^* is not. This means that P^* contains two literal occurences $\langle c, l\rangle$ and $\langle c', \neg l\rangle$. Since P^* is the Herbrand path of an instance-preserving enumeration E of $P \cup P_S$, $P \cup P_S$ must contain two literal occurences which are more general than $\langle c, l\rangle$ and $\langle c', \neg l\rangle$, respectively. We select their minimal matchers $o_1 = \langle c_1, k\rangle$ and $o_2 = \langle c_2, \neg k'\rangle$ in E. Obviously, k and k' must be unifiable. Let σ be a most general unifier of k and $k'\tau$ where $c_2\tau$ is a variant of c_2 which is variable-disjoint from c_1. Either σ is a renaming substitution. In this case B would be $\forall$-closed, which contradicts our assumption. Or, σ is no renaming substitution. Then we can apply Lemma 1 which assures that a proper instance c_3 of c_1 or c_2 must be on the branch B and c_3 is a linking instance of that clause wrt. the connection $\{o_1, o_2\}$. Then c_3 is a matcher of c or c', but this contradicts our selection of o_1 and o_2 as minimal matchers of c and c', respectively. Since we have considered all cases, P^* must be a consistent Herbrand path through S^*. Therefore, by Proposition 1, the set of literals I of P^* is a partial Herbrand model for S. □

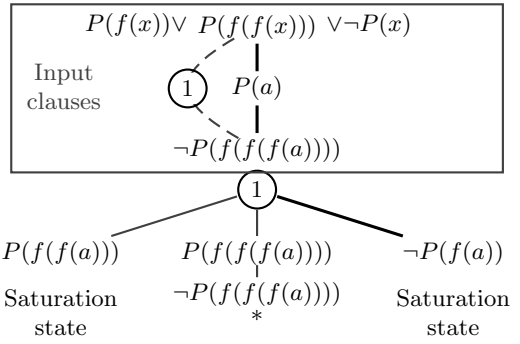

Fig. 4. A saturated open disconnection tableau for a clause set with an infinite Herbrand universe.

In principle, this result suffices for the explicit extraction of Herbrand models as sets of ground literals from saturated branches. But, in certain cases, a finite branch can even represent an infinite Herband model. As an example, consider the saturated tableau in Figure 4 and its right-most branch. The enumeration

$$\neg P(f(f(f(a)))),\ \neg P(f(a)),\ P(a),\ P(f(f(x)))$$

of this branch represents an infinite Herbrand model for the respective clause set S, viz.,

$$\{\neg P(f(f(f(a)))), \neg P(f(a)), P(a)\} \cup \{P(f(f(\mathfrak{s}))) : \mathfrak{s} \neq f(a)\}$$

where $\mathfrak{s}$ ranges over the Herbrand universe of S. From every finite enumeration, one can compute such a schematic expression with instantiation exceptions. Many important questions concerning Herbrand models can be solved efficiently in this framework.

Given an finite enumeration E of a saturated branch B in a disconnection tableau for a clause set S and any literal l, then the following problems (among others) can be decided in time polynomial in the size of E and l, even if the Herband universe of S is infinite:

1. is an instance of l in the partial Herbrand model I represented by E,
2. are all instances of l in I,
3. compute a schematic expression of all instances of l in I.

This shows the potential of the framework for the extraction and representation of models. We should emphasise, however, that this approach does not subsume the term schematisation concepts developed, e.g., in [Sal92,HG97].

4 Completeness of the Disconnection Tableau Calculus

With the model characterisation proposition at hand, the completeness of the disconnection tableau calculus can be recognised with a technique which is more

traditional for tableaux. The key approach is to use a *systematic* or *fair* inference strategy [Smu68]. In general, the disconnection tableau calculus is nondeterministic. An *inference strategy* makes the tableau construction deterministic for any input set. Formally, an inference strategy $\mathfrak{f}$ is a mapping which associates with every clause set S a disconnection tableau sequence $\mathcal{T}$ for S. We call $\bigcup\mathcal{T}$ *the* tableau for S and $\mathfrak{f}$.

In detail, an inference strategy has to deal with the following three kinds of indeterminisms:

1. the selection of the initial path,
2. the selection of the next branch B to be expanded,
3. the selection of the next linking step to be performed on B.

Definition 10 (Fairness). *An inference strategy $\mathfrak{f}$ is called* systematic *or* fair *if, for any clause set, the tableau for S and $\mathfrak{f}$ is saturated.*

Obviously, there are fair inference strategies for the disconnection tableau calculus. It can easily be checked that the following describes such a fair strategy.

Example 1 (A fair inference strategy).

1. For the initial path: select the first literal in each clause,
2. for branch selection: choose the left-most open branch B,
3. for link selection: from all links on B where the sum of the depths[4] of the connected nodes is minimal, select the one where the depth of the upper node is minimal.

Proposition 3 (Completeness). *If S is an unsatisfiable clause set and $\mathfrak{f}$ is a fair inference strategy, then the tableau T for S and $\mathfrak{f}$ is a finite $\forall$-closed disconnection tableau for S.*

Proof. First, we show that every branch of T is $\forall$-closed. Assume not, then, because of the fairness of the strategy $\mathfrak{f}$, T would be saturated and contain a saturated branch. Then, by Proposition 2, S would be satisfiable, contradicting our assumption. The $\forall$-closedness entails the finiteness of every branch. Since T is finitely branching, by König's Lemma, it must be finite. □

5 Refinements

Our research regarding the disconnection tableau calculus was not for theoretical purposes only. There also exists an implementation of the disconnection tableau calculus, which was presented in [LS01]. When designing a new calculus for implementation in a powerful theorem prover, it is impossible to ignore the successful paradigms developed in the field of automated theorem proving

[4] The *depth* of a tableau node is the number of its ancestor nodes.

over the recent decades. Therefore, when new issues such as proof length and redundancy elimination came into focus also from a practical point of view, we incorporated a number of completeness preserving refinements into the calculus that are described in the aforementioned paper.

These refinements include different variations of subsumption, a strong unit theory element and several deletion strategies for redundancy elimination. For lack of space, however, we cannot describe these refinements here in depth or give proof of their preserving the completeness of the disconnection calculus.

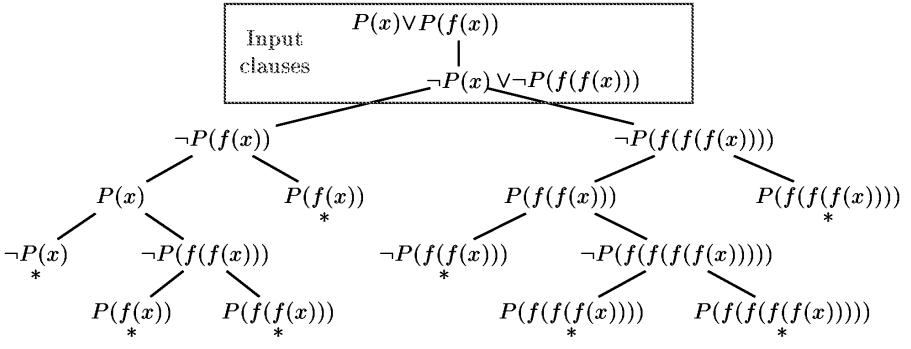

Fig. 5. ∀-closed plain disconnection tableau.

Here, we will concentrate on one particular facet of these refinements. One inherent characteristic of plain tableau calculi is that they are inherently cut-free. This property may lead to the generation of large proofs containing many redundant inferences and these redundancies also occur when searching for models. The standard approach for solving this problem is the use of *lemmas* or *controlled cuts* [LMG94].

The following example demonstrates the possibility of introducing unit lemmas, which simulate certain atomic cut steps. Figure 5 shows a ∀-closed plain disconnection tableau for the unsatisfiable clause set

$$S = \{\{P(x), P(f(x))\}, \{\neg P(x), \neg P(f(f(x)))\}\}.$$

The structure of the subtableau below the node $\neg P(f(x))$ strongly resembles the structure of the subtableau below the node $\neg P(f(f(f(x))))$, which indicates that a part of the proof might be redundant. If we look at the tableau in Figure 6, we see that this redundancy indeed exists and how, by the use of *folding up* and *unit simplification*, we can avoid this redundancy. Folding up works as follows. Assume a subtableau T with root literal l is ∀-closed without refering to literals on the branch above l. This amounts to the generation of a new unit lemma, namely, the complement of $l\sigma$ where σ identifies all variables in l—just taking the complement of l would be unsound because of the branch closure condition,

which identifies all variables. This feature is a special case of the so-called *folding up* procedure [LMG94]. In the example, the new unit lemma $P(f(x))$ is inserted at the root of the tableau after closure step 4.

Unit simplification allows the immediate closure of a branch if it contains a literal k which can be unified with the complement of the literal in an input unit clause or a unit lemma. Unit simplification is used to close the right-most branch of the tableau in Figure 6, thus eliminating the entire right subtableau.

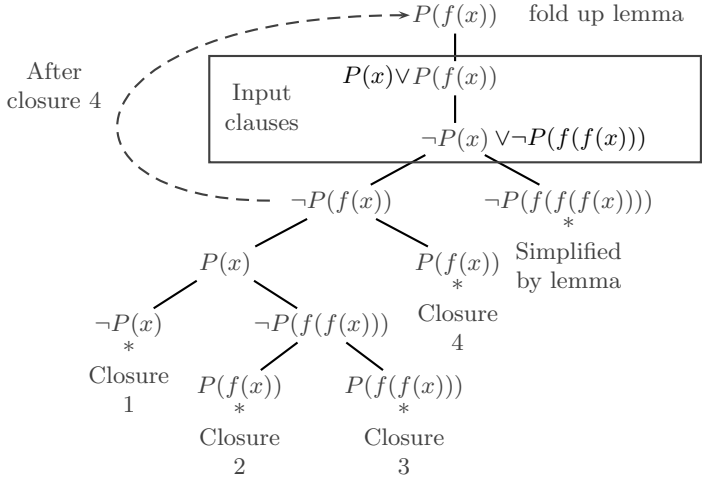

Fig. 6. ∀-closed disconnection tableau with unit lemmas.

It is the use of refinements like the ones described above that turns the implementation of a calculus into a potentially successful proof procedure. It must be noted, however, that the disconnection calculus is not compatible with certain refinements or is only compatible with weakened forms of those refinements. Examples for such refinements are hyperlinking or goal orientation. Goal orientation may be introduced into the proof procedure to a certain degree by adjusting the weighting and selection functions, but the fairness condition required to sustain the branch saturation property prevents us from exploiting goal orientation to the full.

6 Conclusions and Further Work

In this paper, we have presented the disconnection tableau calculus, a promising framework for proof and model finding in automated reasoning. We have given rigorous self-contained proofs of completeness and model generation. In order to make the calculus practically relevant, we have shown how the special handling of unit clauses can be integrated into the framework. Proof-theoretically, the disconnection framework has a number of advantages. Like Plaisted's linking

method, the disconnection approach avoids the duplication problem inherent in calculi like resolution. Furthermore, it is branch saturating and combines the appealing properties of traditional tableau methods wrt. model generation with a controlled method of generating clause instances purely guided by connections.

As future promising research directions we would like to mention the following three issues. First, it is absolutely necessary to integrate an efficient equality handling into the framework. In [Bil96], a method was briefly sketched which is a form of paramodulation adapted to the tableau framework. Since we have no structural tableau restrictions (like, e.g., the connectedness condition in connection tableau) which are incompatible with orderings, even the integration of an ordered equality handling is possible [BG98].

A further interesting research line would be to consider the integration of term schematisation techniques [Sal92,HG97] into the disconnection tableau framework, which would require an extension of the unification procedure. Such techniques would permit a further condensation of the representation of models.

Finally, it could be interesting to envisage the generation of finite general (i.e., non-Herbrand) models. The current methods for the generation of general finite models like *Finder* [Sla94] or *Mace* [McC94] work by a more or less blind identification of terms. Since the literals and terms on a (non-saturated) branch of a disconnection tableau often induce certain promising identifications of terms, it seems promising to integrate such a subcomponent into a disconnection tableau prover. With such a subcomponent, the decision power of the system could significantly be improved.

References

[Bau00] Peter Baumgartner. FDPLL – A First-Order Davis-Putnam-Logeman-Loveland Procedure. In David McAllester, editor, *CADE-17 – The 17th International Conference on Automated Deduction*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 200–219. Springer, 2000.

[BG98] Leo Bachmair and Harald Ganzinger. Equational reasoning in saturation-based theorem proving. In Wolfgang Bibel and Peter H. Schmidt, editors, *Automated Deduction: A Basis for Applications. Volume I, Foundations: Calculi and Methods*, pages 353–398. Kluwer Academic Publishers, Dordrecht, 1998.

[Bil96] Jean-Paul Billon. The disconnection method: a confluent integration of unification in the analytic framework. In P. Migliolo, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Proceedings of the 5th International Workshop on Theorem Proving with analytic Tableaux and Related Methods (TABLEAUX)*, volume 1071 of *LNAI*, pages 110–126, Berlin, May15–17 1996. Springer.

[DG79] Burton Dreben and Warren D. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas.* Addison-Wesley Publishing Company, Reading, MA, 1979.

[DP60] Martin Davis and Hilary Putman. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, July 1960.

[Fit96] Melvin C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, second edition, 1996.

[HG97] Miki Hermann and Roman Galbavý. Unification of infinite sets of terms schematized by primal grammars. *Theoretical Computer Science*, 176(1–2):111–158, April 1997.

[Let99] Reinhold Letz. First-Order Tableaux Methods. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 125–196. Kluwer, Dordrecht, 1999.

[LMG94] R. Letz, K. Mayr, and C. Goller. Controlled integration of the cut rule into connection tableau calculi. *Journal of Automated Reasoning*, 13(3):297–337, December 1994.

[LS01] Reinhold Letz and Gernot Stenz. DCTP: A Disconnection Calculus Theorem Prover. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR-2001), Siena, Italy*, volume 2083 of *LNAI*, pages 381–385. Springer, Berlin, June 2001.

[McC94] William McCune. A Davis-Putnam program and its application to finite first-order model search: quasigroup existence problems. Technical Memorandum ANL/MCS-TM-194, Argonne National Laboratories, IL/USA, September 1994.

[Pel99] Nicolas Peltier. Pruning the search space and extracting more models in tableaux. *Logic Journal of the IGPL*, 7(2):217–251, 1999.

[PL92] David A. Plaisted and Shie-Jue Lee. Eliminating duplication with the hyper-linking strategy. *Journal of Automated Reasoning*, 9(1):25–42, 1992.

[Sal92] Gernot Salzer. The unification of infinite sets of terms and its applications. In A. Voronkov, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *LNAI*, pages 409–420, St. Petersburg, Russia, July 1992. Springer Verlag.

[Sla94] John Slaney. FINDER: Finite domain enumerator. In Alan Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, volume 814 of *LNAI*, pages 798–801, Berlin, June/July 1994. Springer.

[Smu68] Raymond Smullyan. *First-Order Logic*. Springer, 1968.