Hyper Tableaux with Equality

Peter Baumgartner[1], Ulrich Furbach[2], and Björn Pelzer[2]

[1] NICTA, Canberra, Australia, Peter.Baumgartner@nicta.com.au
[2] Universität Koblenz-Landau, Koblenz, Germany, {uli,bpelzer}@uni-koblenz.de

Abstract. In most theorem proving applications, a proper treatment of equational theories or equality is mandatory. In this paper we show how to integrate a modern treatment of equality in the hyper tableau calculus. It is based on splitting of positive clauses and an adapted version of the superposition inference rule, where equations used for paramodulation are drawn (only) from a set of positive unit clauses, the candidate model. The calculus also features a generic, semantically justified simplification rule which covers many redundancy elimination techniques known from superposition-style theorem proving. Our main theoretical result is the soundness and completeness of the calculus. The calculus is implemented, and we also report on practical experiments.

1 Introduction

Tableau calculi play an important role in theorem proving, knowledge representation and in logic programming. Yet, for automated first-order theorem proving the influence of tableau calculi decreased in the last decade. The CASC competition [SS06] is dominated by saturation-based provers, and a tableau system like SETHEO, which was several times among CASC winners, is not even entering the competition any more. One of the reasons can be seen in the problems tableau calculi have with efficient handling of equality. Of course there are numerous papers on equality handling in tableau calculi. Various approaches are discussed, for instance, in [Bec97]. It is not clear, however, whether they are a basis for high performance theorem proving. This has to do with the usage of free variables in most semantic tableau calculi. The nature of these free variables, their rigidness, seems to be a major source for difficulties to define efficient proof procedures, even without equality. For instance, proof procedures often suffer from excessive backtracking and enumerate whole tableaux in an iterative-deepening fashion, typically based on the number of γ-rule applications in a tableau.

To avoid the problems of rigid variables for equality reasoning, in [DV96] the authors combine a superposition based equality reasoning system with a top down semantic tableau reasoner. Yet, certain substitutions still have to be applied globally to all variables in the tableau, which thus are still treated rigidly. As with most free-variable tableau calculi, the important property of *proof confluence* does not hold or is not known to hold.

Other free-variable tableau methods are based on solving (simultaneous) rigid E-unifiability problems [DV98] but still face the same problem of not exploiting proof confluence.

A more recent stream of equality handling in free-variable tableaux has been initiated by Martin Giese. It is (also) motivated by addressing the excessive backtracking of the methods mentioned above. In [Gie02] the author gives a calculus for free variable tableaux with superposition-type inference and proves completeness by adapting the model generation technique for superposition [BG98,NR01]. One improvement, compared with [DV96] and other free-variable methods is that unification constraints leading to a closed tableau are now held locally together with tableau literals. This allows avoiding backtracking over the tableaux generated in a derivation, but instead amounts to combine local substitutions in a compatible way for the purpose to witness a closed tableau (see [Gie01] for details). A drawback of this approach is its potentially high memory consumption, as, in essence, it does not admit a one-branch-at-a-time proof procedure.

In [Gie03], simplification rules and reasoning with universal variables[3] are added to the framework of [Gie02], *but without equality.* Putting equality aside, the most relevant contribution in [Gie03] from the viewpoint of this paper is the instantiation of the calculus there to a variant of the hyper tableau calculus [BFN96].[4] An important difference to [BFN96] is that [Gie03] uses rigid variables for variables that are shared between positive literals in clauses. For instance, a clause like $\forall x,y\ (p(x,y) \vee q(x))$ then is treated by β-expansion with the formulas $\forall y\ p(X,y)$ and $q(X)$, where X is a rigid variable shared between branches. In contrast, the hyper tableau of [BFN96] would branch out on the formulas $\forall y\ p(t,y)$ and $q(t)$, where t is some "guessed" ground term of the input signature.[5]

In this paper we stick with the hyper tableau calculus and its "obviously inefficient" approach of guessing ground terms for shared variables, as opposed to using free variables. More precisely, we show how to incorporate efficient ordering-based equality inference rules and redundancy elimination techniques from the superposition calculus [BG98,NR01] in a tableau calculus. We believe the hyper tableau calculus [BFN96] is a good basis for doing that, for the following reasons.

- All variables in a hyper tableau are universally quantified in the branch literal they occur. This facilitates the adaption of the superposition framework and enables powerful redundancy criteria.
- As far as we know, none of the free-variable calculi mentioned above can be used as a non-trivial decision procedure for clause logic without non-nullary function symbols (i.e. disjunctive Datalog). The same holds true for any known resolution refinement.

  On the other hand, our calculus is a non-trivial decision procedure for this fragment (with equality), which captures the complexity class NEXPTIME. Many practically

[3] Variables that are local to a clause or literal and that are universally quantified.

[4] Hyper tableaux is a tableau model generation method, which is applied to clauses and needs only one inference rule, which can be seen as a tableaux β-rule. It is applied in a "hyper-way", such that all negative literals are "resolved away" by positive literals in the branch. The remaining literals are positive and are splitted then. This basis idea stems from SATCHMO [MB88], which is extended in hyper tableau by making better use of universally quantified variables.

[5] Notice that Resolution- or Superposition calculi, also those with Splitting [Wei01], reason with the clause $\forall x,y\ (p(x,y) \vee q(x))$.

relevant problems are NEXPTIME-complete, e.g. satisfiability of SHOIQ knowledge bases and first-order model expansion (relevant for constraint solving).

- Advanced techniques are available to restrict the domain of the ground terms (like t above) to be guessed. For instance, the preprocessing technique in [BS06] can readily be used in conjunction with our calculus without any change.
- Specific to the theory of equality and in presence of simplification inference rules, that domain can even be further reduced. This occasionally shows unexpected (positive) effects, leading to termination of our system, where e.g. superposition based systems don't terminate. See Section 5 for details.
- The hyper tableau calculus is the basis of the KRHyper prover, which is used in various applications [FO06,BF03,BFGHS04, e.g.] from which we learned that an efficient handling of equality would increase its usability even more.

The closest approximation of the superposition calculus to E-hyper tableaux is obtained by using a selection function that selects all negative literals in a clause and using a prover that supports splitting (of variable-disjoint subclauses) like SPASS [Wei01]. Even then, there remain differences. We discuss these issues in Section 5.

The article [LS02] discusses various ways of integrating equality reasoning in disconnection tableaux. It includes a variant based on ordered paramodulation, where paramodulation inferences are determined by inspecting connections between literals of two clauses. Only comparably weak redundancy criteria are available. Related to that calculus, in [BT05], the model evolution calculus is extended by equality. Model evolution can be seen as a lifting of DPLL to the first order case together with a sophisticated model construction method, which also admits semantically justified redundancy elimination criteria. Both caluli belong to the family of instance-based methods, which are conceptually rather different to resolution- or tableau calculi as considered here.

This paper is organised as follows: we start with preliminaries in the following section. In Section 3 we present superposition inference rules for clauses together with a static completeness result. In Section 4 we introduce E-hyper tableaux and correctness and completeness properties. In Section 5 we consider improvements for splitting and discuss the relation with splitting in the SPASS prover. Section 6 describes the implementation of the E-KRHyper system. Detailed proofs of all results are given in an appendix.

2 Preliminaries

Most of the notions and notation we use in this paper are the standard ones in the field. We report here only notable differences and additions.

We will use an infinite sets of variables X, and x and y denote elements of X. We fix a signature Σ throughout the paper. Unless otherwise specified, when we say term we will mean Σ-term. If t is a term we denote by $\mathcal{V}\mathrm{ar}(t)$ the set of t's variables. A term t is *ground* iff $\mathcal{V}\mathrm{ar}(t) = \emptyset$.

A substitution ρ is a *renaming (on X)* iff it is a bijection of X onto itself. We say that s is *a variant of* t, and write $s \sim t$, iff there is a renaming ρ such that $s\rho = t$. If s

and t are two terms, we write $s \gtrsim t$, iff there is a substitution σ such that $s\sigma = t$.[6] The notation $s[t]_p$ means that the term t occurs in the term s at position p, as usual.

All of the above is extended from terms to literals in the obvious way.

In this paper we restrict to equational clause logic. Therefore, and essentially without loss of generality, we assume that the only predicate symbol in Σ is $\simeq$. Any atom A that is originally not an equation can be represented as the equation $A \simeq \mathbf{t}$, where $\mathbf{t}$ is some distinguished constant not appearing elsewhere. (But we continue to write, say, $P(a)$ instead of the official $P(a) \simeq \mathbf{t}$.) This move is harmless, in particular from an operational point of view.[7] An atom then is always an equation, and a literal then is always an equation or the negation of an equation. Literals of the latter kind, i.e., literals of the form $(s \simeq t)$ are also called *negative equations* and generally written $s \not\simeq t$ instead. We call a literal *trivial* if it is of the form $t \simeq t$ or $t \not\simeq t$.

We denote atoms by the letters A and B, literals by the letters K and L and by $\overline{L}$ the complement of a literal L.

A clause is a finite multiset of literals, written as a disjunction $A_1 \vee \cdots \vee A_m \vee B_1 \vee \cdots \vee B_n$ or an implication $A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$, where $m, n \geq 0$. Each atom A_i, for $i = 1, \ldots, m$, is called a *head atom*, and each atom B_j, for $j = 1, \ldots, n$, is called a *body atom*. We write $A, \mathcal{A} \leftarrow B, \mathcal{B}$ to denote any clause whose head atoms are $\{A\} \cup \mathcal{A}$ and whose body atoms are $\{B\} \cup \mathcal{B}$, for some atoms A and B, and multisets of atoms $\mathcal{A}$ and $\mathcal{B}$. As usual, clauses are implicitly universally quantified.

We suppose as given a reduction ordering $\succ$ that is total on ground Σ-terms. [8] The non-strict ordering induced by $\succ$ is denoted by $\succeq$, and $\prec$ and $\preceq$ denote the converse of $\succ$ and $\succeq$. The reduction ordering $\succ$ has to be extended to rewrite rules, equations and clauses. Following usual techniques [BG98,NR01, e.g.], to a given ground clause $\mathcal{A} \leftarrow \mathcal{B}$ we associate to each head atom $s \simeq t$ in $\mathcal{A}$ the multiset $\{s,t\}$ and to each body atom $u \simeq v$ in $\mathcal{B}$ the multiset $\{u,u,v,v\}$. Two atoms then (head or body) are compared by using the multiset extension of $\succ$, which is also denoted by $\succ$. This will have the effect of a lexicographic ordering, where, first, the bigger terms of two equations are compared, then the sign (body atoms are bigger) and at last the smaller sides of the equations. To compare clauses the two-fold multiset extension of $\succ$ is used, likewise denoted by $\succ$. For the purpose of comparing ground rewrite rules they are treated as positive unit clauses.

A central notion for hyper tableaux is that of a *pure* clause [BFN96]: a clause $A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ is called *pure* iff $\mathcal{V}\mathrm{ar}(A_i) \cap \mathcal{V}\mathrm{ar}(A_j) = \emptyset$, for all $1 \leq i, j \leq m$ with $i \neq j$. That is, in a pure clause variables are not shared among head literals. (Below we will need this concept for positive clauses only.) Any substitution that turns a clause C into a pure instance $C\pi$ is called a *purifying substitution (for C)*.

[6] Note that many authors would write $s \lesssim t$ in this case.

[7] Strictly speaking, one has to move to a two-sorted signature with different signatures for function symbols and predicate symbols, and all variables are of the sort of terms. We ignore this aspect throughout the paper because it does not cause any complications.

[8] A *reduction ordering* is a strict partial ordering that is well-founded and is closed unter context i.e., $s \succ s'$ implies $t[s] \succ t[s']$ for all terms t, and liftable, i.e., $s \succ t$ implies $s\delta \succ t\delta$ for every term s and t and substitution δ.

A *(Herbrand) interpretation* I is a set of ground Σ-equations—those that are true in the interpretation. Satisfiability/validity of ground Σ-literals, Σ-clauses, and clause sets in a Herbrand interpretation is defined as usual. We write $I \models F$ to denote the fact that I satisfies F, where F is a ground Σ-literal or a Σ-clause (set).

Since every interpretation defines in effect a binary relation on ground Σ-terms, and every binary relation on such terms defines an interpretation, we will identity the two notions in the following.

An *E-interpretation* is an interpretation that is also a congruence relation on the Σ-terms. If I is an interpretation, we denote by I^{E} the smallest congruence relation on the Σ-terms that includes I, which is an E-interpretation. We say that I *E-satisfies* F iff $I^{\mathrm{E}} \models F$. Instead of $I^{\mathrm{E}} \models F$ we generally write $I \models_{\mathrm{E}} F$. We say that F *E-entails* F', written $F \models_{\mathrm{E}} F'$, iff every E-interpretation that satisfies F also satisfies F'. We say that F and F' are *E-equivalent* iff $F \models_{\mathrm{E}} F'$ and $F' \models_{\mathrm{E}} F$.

Redundant Clauses. Intuitively, a clause is redundant iff it follows from a set of smaller clauses. We will formalize this now, following [BG98]. There is a related notion of "redundant inference" which will be introduced in Section 3.1 below.

If D is a ground clause and $\mathcal{C}$ is a set of ground clauses then let $\mathcal{C}_D = \{C \in \mathcal{C} \mid D \succ C\}$. When $\mathcal{C}$ is a set of non-ground clauses and when writing $\mathcal{C}_D$ we identify $\mathcal{C}$ with the set of all ground instances of all its clauses.

Now, a ground clause D is *redundant wrt. a set of clauses* $\mathcal{C}$ iff $\mathcal{C}_D \models_{\mathrm{E}} D$. That is, D is redundant wrt. $\mathcal{C}$ iff D follows from smaller clauses taken from $\mathcal{C}$.[9] When D is a non-ground clause we say that D is redundant wrt. $\mathcal{C}$ iff every ground instance of D is redundant wrt. $\mathcal{C}$. For instance, using any simplification ordering, the clause $P(f(a)) \leftarrow$ is redundant wrt. $\{P(a) \leftarrow, f(x) \simeq x \leftarrow\}$.

3 Inference Rules on Clauses

The following three inference rules are taken from the superposition calculus [BG98] and adapted to our needs. We need in addition a splitting rule that will be defined afterwards. All rules will below be embedded into the hyper tableau derivation rules.

When writing an equation $l \simeq r$ we always mean it as a nondeterministic notation that also stands for its symmetric version $r \simeq l$.

The sup-left rule (*superposition left*) applies a superposition step to a body literal:

$$\textsf{sup-left}(\sigma)\ \frac{\mathcal{A} \leftarrow s[l'] \simeq t, \mathcal{B} \qquad l \simeq r \leftarrow}{(\mathcal{A} \leftarrow s[r] \simeq t, \mathcal{B})\sigma} \quad \text{if} \begin{cases} l' \text{ is not a variable,} \\ \sigma \text{ is a mgu of } l \text{ and } l', \\ l\sigma \not\preceq r\sigma, \text{ and} \\ s\sigma \not\preceq t\sigma \end{cases}$$

The last condition can be dropped, and the resulting inference rule is then called *ordered paramodulation left*.

[9] By compactness, even from a *finite* set of clauses.

The unit-sup-right rule (*unit superposition right*) applies a superposition step to a positive *unit* clause:

$$\text{unit-sup-right}(\sigma)\ \frac{s[l'] \simeq t \leftarrow \qquad l \simeq r \leftarrow}{(s[r] \simeq t \leftarrow)\sigma} \quad \text{if} \begin{cases} l' \text{ is not a variable,} \\ \sigma \text{ is a mgu of } l \text{ and } l', \\ (s \simeq t)\sigma \not\preceq (l \simeq r)\sigma, \\ l\sigma \not\preceq r\sigma, \text{ and} \\ s\sigma \not\preceq t\sigma \end{cases}$$

The last condition can be dropped, and the resulting inference rule is then called *ordered unit paramodulation right*.

The general superposition right inference rule of [BG98] between non-unit clauses is not needed, essentially due to the presence of the splitting rule below.

The ref rule (*reflexivity*) eliminates a body literal on the grounds of being trivially true (after applying a substitution).

$$\text{ref}(\sigma)\ \frac{\mathcal{A} \leftarrow s \simeq t, \mathcal{B}}{(\mathcal{A} \leftarrow \mathcal{B})\sigma} \quad \text{if } \sigma \text{ is a mgu of } s \text{ and } t$$

Finally, the announced splitting rule. It takes a disjunctive fact, applies a purifying substitution π to it and returns the instantiated head atoms, one conclusion per head atom.

$$\text{split}(\pi)\ \frac{A_1, \ldots, A_m \leftarrow}{A_1\pi \leftarrow \quad \cdots \quad A_m\pi \leftarrow} \quad \text{if} \begin{cases} m \geq 2, \text{ and} \\ \pi \text{ is a purifying substitution } \pi \text{ for } A_1, \ldots, A_m \leftarrow \end{cases}$$

3.1 Redundant Inferences and Saturation

We write $C, D \Rightarrow_{\text{sup-left}(\sigma)} E$ to denote a sup-left inference, i.e., an instance of the sup-left inference rule with left premise C, right premise D, conclusion E and substitution σ that satisfies the rule's side condition. We use analogous notation for an application of the sup-right inference rule, and for an application of ref we write, similarly, $C \Rightarrow_{\text{ref}(\sigma)} E$. Likewise, $C \Rightarrow_{\text{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ denotes a split inference with premise C, purifying substitution π and conclusions $A_1 \leftarrow, \ldots, A_m \leftarrow$.

An R-inference, with $R \in \{\text{sup-left}, \text{unit-sup-right}, \text{ref}\}$ is *ground* iff its constituent clauses C, D and E are ground. The substitution σ in a ground inference is irrelevant and may be assumed, without loss of generality, to be the empty substitution ε.

If $C, D \Rightarrow_{R(\sigma)} E$ is an R-inference (with D absent in the case of ref) and γ is a substitution such that $C\sigma\gamma, D\sigma\gamma \Rightarrow_{R(\varepsilon)} E\gamma$ is a ground inference, then the latter inference is called a *ground instance* of the inference $C, D \Rightarrow_{R(\sigma)} E$.

For instance, by taking $\gamma = \{x \mapsto a\}$ one sees that the ground inference

$$(P(f(a)) \leftarrow), (f(a) \simeq a \leftarrow) \Rightarrow_{\text{sup-right}(\varepsilon)} P(a) \leftarrow$$

is a ground instance of the inference

$$(P(f(x)) \leftarrow), (f(y) \simeq y \leftarrow) \Rightarrow_{\text{sup-right}(\{y \mapsto x\})} P(x) \leftarrow \quad .$$

In contrast,

$$(P(f(f(a))) \leftarrow\ ), (f(a) \simeq a \leftarrow\ ) \Rightarrow_{\textsf{sup-right}(\varepsilon)} P(f(a)) \leftarrow$$

is not a ground instance of the inference above, for no substitution γ. Intuitively, only such ground inferences can be ground instances of inferences where paramodulation takes place at positions that exist also at the non-ground level. This excludes ground inferences that are not liftable because they would require paramodulation into or below variables. We can define these notions for the split rule analogously: a split inference is *ground* if the premise is ground (and hence all its conclusions are ground). Similarly as above for the other rules, the purifying substitution π can always be assumed to be the empty substitution then.

If $C \Rightarrow_{\textsf{split}(\pi)} A_1 \leftarrow\ , \ldots, A_m \leftarrow$ is a split inference and γ is a substitution such that $C\pi\gamma \Rightarrow_{\textsf{split}(\varepsilon)} A_1\gamma \leftarrow\ , \ldots, A_m\gamma \leftarrow$ is a ground split inference, then the latter inference is called a *ground instance* of the former inference.

Let $\mathcal{D}$ be a set of clauses, not necessarily ground. A ground inference $C, D \Rightarrow_{\textsf{sup-left}(\varepsilon)} E$ or $C, D \Rightarrow_{\textsf{sup-right}(\varepsilon)} E$ is *redundant wrt.* $\mathcal{D}$ iff E is redundant wrt. $\mathcal{D}_C \cup \{D\}$. A ground inference $C \Rightarrow_{\textsf{ref}(\varepsilon)} E$ is *redundant wrt.* $\mathcal{D}$ iff E is redundant wrt. $\mathcal{D}_C$. And a ground inference $C \Rightarrow_{\textsf{split}(\varepsilon)} A_1 \leftarrow\ , \ldots, A_m \leftarrow$ is *redundant wrt.* $\mathcal{D}$ iff there is an i with $1 \le i \le m$ such that $A_i \leftarrow$ is redundant wrt. $\mathcal{D}_C$.

For all inference rules sup-left, unit-sup-right, ref and split, a (possibly non-ground) inference is *redundant wrt.* $\mathcal{D}$ iff each of its ground instances is redundant wrt. $\mathcal{D}$.

Intuitively a ground inference is redundant wrt. $\mathcal{D}$ iff its conclusion follows from a set of smaller clauses than the left premise, while fixing the right premise. Because all (ground) inferences work in a strictly order-decreasing way, adding the conclusion of an inference to the clause set the premises are taken from renders the inference redundant wrt. that set.[10] For instance, adding $P(a) \leftarrow$ to the set $\{(P(f(a)) \leftarrow\ ), (f(a) \simeq a \leftarrow\ )\}$ renders the obvious sup-right inference redundant wrt. the resulting set.

It is not only redundant inferences that can be neglected. Also inferences where one or both parent clauses are redundant can be neglected. This is captured by the following definition.

Definition 3.1 (Saturation up to redundancy). *A clause set* $\mathcal{C}$ *is* saturated up to redundancy *iff for all clauses* $C \in \mathcal{C}$ *such that* C *is not redundant wrt.* $\mathcal{C}$ *all of the following hold:*

1. *Every inference* $C \Rightarrow_{\textsf{split}(\pi)} A_1 \leftarrow\ , \ldots, A_m \leftarrow$ *such that* $C\pi$ *is not redundant wrt.* $\mathcal{C}$*, is redundant wrt.* $\mathcal{C}$*.*
2. *Every inference* $C, D \Rightarrow_{R(\sigma)} E$*, where* $R \in \{\textsf{sup-left}, \textsf{unit-sup-right}\}$ *and* D *is a fresh variant of a positive unit clause from* $\mathcal{C}$*, such that neither* $C\sigma$ *nor* $D\sigma$ *is redundant wrt.* $\mathcal{C}$*, is redundant wrt.* $\mathcal{C}$*.*
3. *Every inference* $C \Rightarrow_{\textsf{ref}(\sigma)} E$ *such that* $C\sigma$ *is not redundant wrt.* $\mathcal{C}$*, is redundant wrt.* $\mathcal{C}$*.*

For instance, the (satisfiable) propositional clause set $\mathcal{C} = \{(A, B \leftarrow\ ), (\ \leftarrow A)\}$ is *not* saturated up to redundancy. By an application of the split rule to $A, B \leftarrow$ one can infer

[10] This property makes it obvious that fair derivations, as defined below, exists.

$A \leftarrow$ and $B \leftarrow$, and adding, say, $A \leftarrow$ to $\mathcal{C}$ renders the clause $A, B \leftarrow$ redundant. Notice that with two more (obvious) inferences the empty clause can then be derived, which results in a saturated state. This example also shows that neglecting cases, as for $B \leftarrow$, easily leads to unsoundness.

As an example for a non-ground split inference consider a clause $P(x), Q(x) \leftarrow$ from some clause set. One may want to avoid applying all purifying substitutions to it. Fortunately, Definition 3.1-1 does not prescribe that at all. For instance, when the clause set includes an equation $a \simeq b \leftarrow$ (where $a \succ b$) then purifying $P(x), Q(x) \leftarrow$ by $\pi = \{x/b\}$, yielding $P(b), Q(b) \leftarrow$, and adding $P(b) \leftarrow$ to the clause set is sufficient to render the split inference with purifying substitution $\{x/a\}$ redundant, as the clause $P(a) \leftarrow$ follows from $P(b) \leftarrow$ and $a \simeq b \leftarrow$, both of which are smaller than $P(a), Q(a) \leftarrow$.

Theorem 3.2 (Static Completeness). *Let $\mathcal{C}$ be a clause set saturated up to redundancy. If $\square \notin \mathcal{C}$ then $\mathcal{C}$ is* E*-satisfiable.*

The proof employs the model-construction technique originally developed for the superposition calculus, but adapted to our needs.[11]

Notice that Theorem 3.2 applies to a *statically* given clause set $\mathcal{C}$. The connection to the *dynamic* derivation process of the E-hyper tableau calculus will be given later, and Theorem 3.2 will be essential then in proving the completeness of the E-hyper tableau calculus.

4 E-Hyper Tableaux

In [BFN96], based on [LMG94], hyper tableau have been introduced as labeled trees over *literals* (which are universally quantified, and hence can be seen as unit clauses). For our purposes, however, a generalization towards trees over *clauses* is better suited. This is, because *new* clauses can now be derived as the derivation proceeds, and these clauses are context dependant (branch local), and tableau are an obvious data structure to deal with this context dependency.

A *labeled tree over a set M* is a pair $(\mathbf{t}, \lambda)$ consisting of a finite, ordered tree $\mathbf{t}$ and a labeling function λ that maps each node of $\mathbf{t}$ to some element from M. A *(clausal) tableau over a signature* Σ is a labeled tree over the set of Σ-clauses.

We use the letter $\mathbf{T}$ to denote tableaux.

Let $\mathbf{B}$ be a branch of a tableau $\mathbf{T}$ of length n, i.e., a sequence of nodes $(\mathbf{N}_1, \ldots, \mathbf{N}_n)$, for some $n \geq 0$, where $\mathbf{N}_1$ is the root and $\mathbf{N}_n$ is the leaf of $\mathbf{B}$. Each of the clauses $\lambda(\mathbf{N}_i)$, for $i = 1, \ldots, n$, is called a *(tableau) clause of* $\mathbf{B}$.

Occasionally it is convenient to read a branch $\mathbf{B}$ as the multiset of its tableau clauses $\lambda(\mathbf{B}) := \{D \mid D \text{ is a tableau clause of } \mathbf{B}\}$. This allows to write, for instance, $C \in \mathbf{B}$ instead of $C \in \lambda(\mathbf{B})$. Further, if $\mathbf{B}$ is branch of a tableau $\mathbf{T}$ we write $\mathbf{B} \cdot C$ and mean the tableau obtained from $\mathbf{T}$ by adding an edge from the leaf of $\mathbf{B}$ to a fresh node labeled with C. Further, we write $\mathbf{B} \cdot \mathbf{B}'$ to denote the branch obtained by concatenating the branch $\mathbf{B}$ and the node sequence $\mathbf{B}'$.

[11] Proofs are in the appendix.

4.1 Extension Rules

We define two derivation rules for extending branches in a given tableau.

The Split rule branches out on an instance of a positive clause; its conclusions are labeled as “decision clauses”, as indicated by the annotation $^{\mathrm{d}}$. The rï¿½e of this labeling will become clear below in Section 4.2.

$$\textsf{Split}\quad \frac{\mathbf{B}}{\mathbf{B}\cdot A_1 \leftarrow^{\mathrm{d}} \quad \cdots \quad \mathbf{B}\cdot A_m \leftarrow^{\mathrm{d}}} \quad \text{if} \begin{cases} \text{there is a clause } C \in \mathbf{B} \text{ and} \\ \text{a substitution } \pi \text{ such that} \\ \quad C \Rightarrow_{\textsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow \text{ and} \\ \quad \mathbf{B} \text{ contains no variant of } A_i \leftarrow, \\ \quad \text{for any } i = 1, \ldots, m \end{cases}$$

The clause C is called the *selected clause (of a* Split *inference)*.

The Equality rule applies an inference rule for equality reasoning from Section 3 to a body literal.

$$\textsf{Equality}\quad \frac{\mathbf{B}}{\mathbf{B}\cdot E} \quad \text{if} \begin{cases} \text{there is a clause } C \in \mathbf{B}, \\ \text{a fresh variant } D \text{ of a positive unit clause in } \mathbf{B}, \text{ and} \\ \text{a substitution } \sigma \text{ such that} \\ \quad C, D \Rightarrow_{R(\sigma)} E \text{ with } R \in \{\textsf{sup-left}, \textsf{unit-sup-right}\} \text{ or} \\ \quad C \Rightarrow_{\textsf{ref}(\sigma)} E, \text{ and} \\ \mathbf{B} \text{ contains no variant of } E \end{cases}$$

In both rules, the test for the conclusion(s) being not contained in **B** is needed in interplay with deletion of clauses based on non-proper subsumption (see the Del below).

For later use, we say that an application of a Split, Sup-left, Unit-sup-right or Ref derivation rule to a branch **B** is *redundant* iff its conclusion (at least one of its conclusions, in the case of Split) is redundant wrt. **B**.

4.2 Deletion and Simplification Rules

From a practical point of view, deletion of redundant clauses and simplification operations on clauses are crucial. We will introduce these now. Adding such rules is a major addition to the hyper tableau calculus and involves a more sophisticted technical treatment than that in [BFN96]. This is, because hyper tableau as defined in [BFN96] are *non-destructive*, in the sense that extending a branch goes along with increasing the set of its corresponding labels (unit clauses). This is no longer the case in presence of, for instance, the Del rule (*deletion*) below, which removes a clause that is redundant in a branch or subsumed by another clause in the branch.

Also, to preserve the calculus’ correctness, arbitrary deletion of redundant clauses is not possible. Only clauses can be deleted the deletion of which is not justified by clauses introduced at a later decision level. This is formalized next.

$$\textsf{Del}\quad \frac{\mathbf{B}\cdot C^{(\mathrm{d})}\cdot \mathbf{B}_1 \cdot \mathbf{B}_2}{\mathbf{B}\cdot \mathbf{t} \simeq \mathbf{t} \leftarrow^{(\mathrm{d})} \cdot \mathbf{B}_1 \cdot \mathbf{B}_2} \quad \text{if} \begin{cases} (1)\ C \text{ is redundant wrt. } \mathbf{B}\cdot\mathbf{B}_1, \text{ or some} \\ \text{clause in } \mathbf{B}\cdot\mathbf{B}_1 \text{ non-properly subsumes } C, \text{ and} \\ (2)\ \mathbf{B}_1 \text{ does not contain a decision clause} \end{cases}$$

The notation $^{(\mathrm{d})}$ is meant to say that if there is a label $^{\mathrm{d}}$, it is preserved when replacing C by $\mathbf{t} \simeq \mathbf{t} \leftarrow$.

Observe that our redundancy notion does not cover non-proper subsumption.[12] For instance, the clause $P(a) \leftarrow$ is *not* redundant wrt. $\{P(x) \leftarrow\}$ (and neither is the clause $P(y) \leftarrow$). Therefore, deletion of non-properly subsumed clauses has been taken care of explicitly.

The condition (2) is needed to guarantee the completeness of the calculus. Without it, a Del step would be possible that is justified at or below some later decision level (Split application). This would render the calculus incomplete.

The next rule, Simp (simplification), replaces a clause by another one that is smaller in the ordering:

$$\mathsf{Simp}\ \frac{\mathbf{B} \cdot C^{(\mathrm{d})} \cdot \mathbf{B}_1 \cdot \mathbf{B}_2}{\mathbf{B} \cdot D^{(\mathrm{d})} \cdot \mathbf{B}_1 \cdot \mathbf{B}_2} \quad \text{if } \begin{cases} \mathbf{B} \cdot C \cdot \mathbf{B}_1 \models_{\mathrm{E}} D, \\ C \text{ is redundant wrt. } \mathbf{B} \cdot D \cdot \mathbf{B}_1, \text{ and} \\ \mathbf{B}_1 \text{ does not contain a decision clause} \end{cases}$$

The Simp rule covers, for instance, standard rewriting by unit clauses.

Notice that a similar condition as for the Del rule above, that $\mathbf{B}_1$ does not contain a decision clause, is also needed in Simp rule. This time, the condition is even necessary to obtain a *sound* calculus.

Lemma 4.1. *For each of the derivation rules* Split, Equality, Del *and* Simp*, if the premise of the rule is* E*-satisfiable, then one of its conclusions is* E*-satisfiable as well.*

A slightly different approach to deletion and simplification is implemented in the SPASS prover [Wei01]. The delete rule in SPASS is even more general than ours as it allows to ignore the decision levels. Of course, a deleted clause must be restored on backtracking to an earlier decision level if the deletion depends on the current decision level. This is never necessary in our case. Similar argumentation holds for the Simp rule. At present, it is not clear to us what approach is preferrable in practice. We intend to clarify this issue by experiments.

4.3 Derivations

We say that a branch of a tableau is *closed* iff it contains the empty clause $\square$.[13] A branch that is not closed is also called *open*. A tableau is *closed* iff each of its branches is closed, and it is *open* iff it is not closed (i.e., if it has an open branch).

An (E*-hyper tableau) derivation* of a set $\{C_1, \ldots, C_n\}$ of Σ-clauses is a possibly infinite sequence of tableaux $\mathbf{D} = (\mathbf{T}_i)_{0 \leq i < \kappa}$ such that

1. $\mathbf{T}_0$ is the clausal tableau over Σ that consists of a single branch of length n with tableau clauses $C_1, \ldots, C_n$.[14], and

[12] A clause C non-properly subsumes a clause D iff there is a substitution σ such that $C\sigma = D$.

[13] We write $\square$ instead of " $\leftarrow$ ".

[14] The order does not matter, as the collection of tableaux clauses of a branch will be seen as sets. For technical reasons we assume that no clause C_i is a variant of a clause C_j, for all $1 \leq i < j \leq n$, but this is obviously not an essential restriction.

2. for all $i > 0$, $\mathbf{T}_i$ is obtained from $\mathbf{T}_{i-1}$ by a single application of one of the derivation rules in Sections 4.1 and 4.2 to some open branch of $\mathbf{T}_{i-1}$, called the *selected branch*.

Recall that a tableau $\mathbf{T}$ is of the form $(\mathbf{t}, \lambda)$, where $\mathbf{t}$ is a tree, i.e., a pair $(\mathbf{N}, \mathbf{E})$ where $\mathbf{N}$ is the set of the nodes of $\mathbf{t}$ and $\mathbf{E}$ is the set of the edges of $\mathbf{t}$.

Each derivation $\mathbf{D} = ((\mathbf{N}_i, \mathbf{E}_i), \lambda_i)_{i<\kappa}$ determines a *limit tree* $((\bigcup_{i<\kappa} \mathbf{N}_i, \bigcup_{i<\kappa} \mathbf{E}_i)$. It is easy to show that a limit tree of a derivation $\mathbf{D}$ is indeed a (possibly infinite) tree.

Now let $\mathbf{t}$ be the limit tree of some derivation, let $\mathbf{B} = (\mathbf{N}_i)_{i<\kappa}$ be a (possibly infinite) branch in $\mathbf{t}$ with κ nodes, and let $\mathbf{B}_i = (\mathbf{N}_1, \ldots, \mathbf{N}_i)$ be the initial segment of $\mathbf{B}$ with i nodes, for all $i < \kappa$. Define $\mathbf{B}_\infty = \bigcup_{i<\kappa} \bigcap_{i \le j < \kappa} \lambda_j(\mathbf{B}_j)$, the set of *persistent clauses (of* $\mathbf{B}$*)*.

Definition 4.2 (Exhausted Branch). *Let* $\mathbf{t}$ *be a limit tree, and let* $\mathbf{B} = (\mathbf{N}_i)_{i<\kappa}$ *be a branch in* $\mathbf{t}$ *with* κ *nodes. The branch* $\mathbf{B}$ *is* exhausted *iff it does not contain the empty clause, and for every clause* $C \in \mathbf{B}_\infty$ *and every fresh variant* D *of every positive unit clause in* $\mathbf{B}_\infty$ *such that neither* C *nor* D *is redundant wrt.* $\mathbf{B}_\infty$ *all of the following hold, for all* $i < \kappa$ *such that* $C \in \mathbf{B}_i$ *and* D *is a variant of a clause in* $\mathbf{B}_i$*:*

1. *if* Split *is applicable to* $\mathbf{B}_i$ *with underlying inference* $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ *and* $C\pi$ *is not redundant wrt.* $\mathbf{B}_i$*, then there is a* $j < \kappa$ *such that the inference* $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ *is redundant wrt.* $\mathbf{B}_j$*.*
2. *if* Equality *is applicable to* $\mathbf{B}_i$ *with underlying inference* $C, D \Rightarrow_{R(\sigma)} E$*, for some* $R \in \{\mathsf{sup\text{-}left}, \mathsf{unit\text{-}sup\text{-}right}\}$*, and neither* $C\sigma$ *nor* $D\sigma$ *is redundant wrt.* $\mathbf{B}_i$*, then there is a* $j < \kappa$ *such that the inference* $C, D \Rightarrow_{R(\sigma)} E$ *is redundant wrt.* $\mathbf{B}_j$*.*
3. *if* Equality *is applicable to* $\mathbf{B}_i$ *with underlying inference* $C \Rightarrow_{\mathsf{ref}(\sigma)} E$ *and* $C\sigma$ *is not redundant wrt.* $\mathbf{B}_i$*, then there is a* $j < \kappa$ *such that the inference* $C \Rightarrow_{\mathsf{ref}(\sigma)} E$ *is redundant wrt.* $\mathbf{B}_j$*.*

A *refutation of a clause set* $\mathcal{C}$ is a finite derivation of $\mathcal{C}$ that ends in a closed tableau.

A derivation is *fair* iff it is a refutation or its limit tree has an exhausted branch.

In the preceeding definition, actually carrying out a Split inference with a clause C and (irreducible) purifying substitution π, when applicable, will achieve the conclusion, i.e. make $C\pi$ redundant wrt. $\mathbf{B}_j$. The analogous holds for the Equality inferences in items 2 and 3. This observation indicates that proof procedures implementing fair derivations indeed can be given.

Theorem 4.3 (Correctness of E-Hyper Tableaux). *Let* $\mathcal{C}$ *be a clause set that has a refutation. Then* $\mathcal{C}$ *is* E*-unsatisfiable.*

For the completeness direction we need the following result:

Proposition 4.4 (Exhausted branches are saturated up to redundancy). *If* $\mathbf{B}$ *is an exhausted branch of a limit tree of some fair derivation then* $\mathbf{B}_\infty$ *is saturated up to redundancy.*

Proposition 4.4 and Theorem 3.2 entails our main result:

Theorem 4.5 (Completeness of E**-Hyper Tableaux).** *Let $\mathcal{C}$ be a clause set and* **T** *be the limit tree of a fair derivation* **D** *of $\mathcal{C}$. If* **D** *is not a refutation then $\mathcal{C}$ is satisfiable.*

Because the proof of this theorem refers to the proof of Theorem 3.2, the model constructed in the proof of Theorem 3.2 provides a strengthening of Theorem 4.5 by being more specific.

Corollary 4.6 (Bernays-Schönfinkel Class with Equality). *The* E*-hyper tableau calculus can be used as a decision procedure for the Bernays-Schönfinkel class with equality, i.e., for sentences with the quantifier prefix $\exists^*\forall^*$.*

5 Restricting Split and the Relation to Splitting in SPASS

For performance reasons it is mandatory to restrict the search space induced by having to apply purifying substitutions in Split rule applications. The fairness criteria in Definition 4.2 already support that. For instance, one can take advantage of avoiding purifying substitutions that are *reducible*, as they lead to redundant inferences.

Definition 5.1 (Reducible substitution). *Let $\mathcal{C}$ be a clause set and σ a substitution. We say that σ is* reducible wrt. $\mathcal{C}$ *iff there is a term $t \in \mathcal{R}\mathrm{an}(\sigma)$*[15]*, a unit clause $l \simeq r \leftarrow\ \in \mathcal{C}$ and a (matching) substitution μ such that $l\mu$ occurs in t and $l\mu \succ r\mu$.*

We say that σ is *irreducible wrt. $\mathcal{C}$* if σ is not reducible wrt. $\mathcal{C}$.

Obviously, for any (positive) clause $C = A_1, \ldots, A_m \leftarrow$ in a branch **B** and any purifying substitution π_0 for C there is a maximal chain $C\pi_0 \succ C\pi_1 \succ \cdots \succ C\pi_n$, for some $n \geq 0$, where π_i is obtained from π_{i-1} by one-step rewriting a term of its range with a positive unit clause from **B** and such that π_n is irreducible wrt. **B**. It is not difficult to see that, by equality, applying Split with $C\pi_n$ renders the Split inferences with $C\pi_0, \ldots, C\pi_{n-1}$ redundant then (wrt. all branches obtained by splitting $C\pi_n$). No reducible purifying substitution needs therefore ever be considered in Split inferences to obtain an exhausted branch.

An example of such a situation is $C = P(x), Q(x) \leftarrow$, $a \simeq b \leftarrow\ \in \mathbf{B}$, $a \succ b$, $\pi_0 = \{x/a\}$ and $\pi_1 = \{x/b\}$. Split with $P(b), Q(b) \leftarrow$ alone to extend **B** is sufficient.

A significantly different split rule is implemented in the SPASS prover [Wei01]. It does not apply a purifying substitution to force partitioning a clause into variable disjoint parts. Instead, it can split on clauses only that are already partitioned that way.

We do not claim that our approach is always preferrable in practice. Yet, there are situations where indeed it is. By way of example, consider the following clauses

$$f(a) \simeq a \leftarrow \quad (1) \qquad\qquad f(g(x)) \simeq g(f(x)) \leftarrow \quad (3)$$
$$g(a) \simeq a \leftarrow \quad (2) \qquad\qquad p(f(x)), p(g(x)) \leftarrow \quad (4)$$

Suppose a precedence $f \succ g \succ a$ (or $g \succ f \succ a$, as the problem is symmetric in f and g), lifted to any simplification ordering. All superposition inferences among the clauses

[15] As usual, the *range* of a substitution σ is $\mathcal{R}\mathrm{an}(\sigma) = \{x\sigma \mid x\sigma \neq x\}$.

1-3 are redundant, and a prover like SPASS will detect that. Among others, there is a superposition inference between clause 4 and 3, which yields the clause

$$p(g(f(x))), p(g(g(x))) \leftarrow \quad . \tag{5}$$

In fact this inference is redundant, too. To see this, consider any ground substitution γ. It must map x to some term comprised of a combination of fs, gs and (one) a, e.g. $\gamma = \{x/f(f(g(f(a))))\}$. Now, any ground instance obtained from clause 5 in this way can be reduced by the unit clauses 1-3 in one or more steps to the clause $p(f(a)), p(g(a)) \leftarrow$ (they can be reduced even further), which is a ground instance of clause 4 and which is smaller in the ordering than the ground instance of clause 5 we started with. By this argument the superposition inference leading to clause 5 is redundant (and need not be carried out).

Notice that this argumentation takes the clause sets signature into account. However, the commonly implemented redundancy criteria don't do that. In particular, for instance, SPASS does not find a finite saturation of the clause set above. In contrast, E-hyper tableau are aware of the input signature and the redundancy criteria based on irreducible purifying substitution, as mentioned above, are strong enough to achieve termination.[16] To see this, it is enough to observe that any purifying substitution, like $\pi = \{x/f(f(g(f(a))))\}$, is reducible (to $\pi = \{x/a\}$) wrt. any branch containing clauses 1 and 2. Thus, the *only* instance of clause 4 to be considered for splitting (in presence of 1-3) is $p(f(a)), p(g(a)) \leftarrow$ (which can be simplified further). Moreover, this can easily be achieved by adding the following "logic program"

$$ran(a) \leftarrow \quad (6) \qquad ran(f(x)) \leftarrow ran(x) \quad (7) \qquad ran(g(x)) \leftarrow ran(x) \quad (8)$$

which, in combination with rewriting by unit clauses will enumerate in its *ran* predicate the ground terms of the input signature that are irreducible wrt. the orientable current positive unit clauses, i.e. the terms that are considered as the range of purifying substitutions. In presence of clauses 1 and 2 this is the singleton $\{a\}$. The general form of the "logic program" has, of course, already been used within SATCHMO [MB88] and some descendants. To our knowledge, though, it was never observed before that equational reasoning can help to confine the *ran*-predicate.

6 Implementation

We have implemented the E-hyper tableau calculus by extending our existing KRHyper system. KRHyper is a hyper tableaux theorem prover, and as such it lacked equality handling in the original version. The modified system, called E-KRHyper, adapts the methods of its precursor to accommodate the new inferences, while at the same time retaining the original functionality regarding input clause sets without equality.

The derivation proceeds in a bottom up manner. Internally, clauses are divided into three sets, one containing the positive non-equational units (*facts*), the other consisting of the positive non-unit clauses (*disjunctions*), and the third including both the unit

[16] More precisely, there is a finite derivation in the E-hyper tableau calculus, and any reasonable implementation, like our E-KRHyper system, will find it.

equations and the clauses with negative literals (*rules*). The hyper extension inference of KRHyper is equivalent to a series of Sup-left, Ref and Split applications, and therefore it is kept in place in E-KRHyper as a shortcut inference for the resolution of non-equational atoms. The E-hyper tableau is generated depth first, with the current state of the three clause sets always representing a single branch. The Split on a disjunction is only executed when the other inference possibilities have been exhausted. An iterative deepening strategy with a limit on the maximum term weight of generated clauses ensures the refutational completeness.

Clauses are derived by a loop iterating over the rules, with each rule in turn accessing indexes in the search for inference partners. The inferred clauses are added to their respective sets after having passed the weight and subsumption tests. The dynamic nature of the rule set represents a major change compared to the previous system version. As the hyper tableaux calculus has no inferences that generate new rule clauses, this set remained fixed throughout the derivation of KRHyper, and many optimizations on the input could be delegated to preprocessing. Operations like the clause subsumption test are necessary for the new calculus, and they are now employed to optimize the input clauses as well.

The superposition inferences utilize a discrimination-tree based index over the subterms of clauses, and terms are ordered according to the recursive path ordering (RPO). As an option, the backtracking mechanism allows the removal of redundant clauses from the entire current branch, beyond the limits set in Section 4.2.

Category	**NNE**	**HEQ**	**NEQ**	**UEQ**
Attempted	20	20	70	100
Solved	6	9	13	2
Av. time	0.01s	0.88s	2.56s	1.93s

We have tested E-KRHyper with several problem sets used in the last CASC system competition, in 2006, at a timeout of 400 seconds; some results are given in the table on the right. The column NNE (non-Horn without equality) can be used to compare E-KRHyper with KRHyper (not shown in the table): which also solved 6 out of the 20 problems, but with a better performance (E-KRHyper is in average 28 percent slower on these examples). In the category HEQ (Horn with equality) E-KRHyper compares with the competitions participants in the intermediate ranks, which we would consider satisfactory for a newcomer. For those problems there is no need for enumerating ground terms for splitted variables; this is the case for the NEQ (non-Horn with equality) examples, where the performance compares to the slower participants of the competition. Also for UEQ (units with equality) E-KRHyper is not yet really competitive. We consider this version of E-KRHyper as a first step towards an efficiently applicable tableau prover with equality. More details about the system can be found in [PW07]; it is available under the GNU Public License from the E-KRHyper website at `http://www.uni-koblenz.de/~bpelzer/ekrhyper`.

7 Conclusion

We have presented a tableau calculus with equality, by integrating superposition based inference rules into the hyper tableau calculus rules. Our main result is its correctness and completeness, the latter in combination with redundancy criteria. The calculus is implemented in the E-KRHyper system, an extension of our existing KRHyper prover.

References

[Bec97] B. Beckert. Semantic Tableaux With Equality. *Journal of Logic and Computation*, 7(1):39–58, 1997.

[BF03] P. Baumgartner and U. Furbach. Automated Deduction Techniques for the Management of Personalized Documents. *Annals of Mathematics and Artificial Intelligence – Special Issue on Mathematical Knowledge Management*, 38(1), 2003.

[BFGHS04] P. Baumgartner, U. Furbach, M. Gross-Hardt, and A. Sinner. Living Book – Deduction, Slicing, and Interaction. *Journal of Automated Reasoning*, 32(3), 2004.

[BFN96] P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. JELIA 96*, LNAI 1126, Springer, 1996.

[BG98] L. Bachmair and H. Ganzinger. Chapter 11: Equational Reasoning in Saturation-Based Theorem Proving. In W. Bibel and P. H. Schmitt, eds., *Automated Deduction. A Basis for Applications*, Volume I, Kluwer, 1998.

[BS06] P. Baumgartner and R. Schmidt. Blocking and Other Enhancements for Bottom-up Model Generation Methods. In U. Furbach and N. Shankar, eds., *Proc. IJCAR*, LNAI 4130, Springer, 2006.

[BT05] P. Baumgartner and C. Tinelli. The Model Evolution Calculus with Equality. In R. Nieuwenhuis, ed., *Proc. CADE-20*, LNAI 3632, Springer, 2005.

[DV96] A. Degtyarev and A. Voronkov. Equality Elimination for the Tableau Method. In *Proc. DISCO-96*, LNAI 1128, Springer, 1996.

[DV98] A. Degtyarev and A. Voronkov. What you Always Wanted to Know About Rigid E-Unification. *Journal of Automated Reasoning*, 20(1):47–80, 1998.

[FO06] U. Furbach and C. Obermaier. Applications of Automated Reasoning. In *Proc. of the 29th German Conference on AI.* LNAI 4314, Springer 2007.

[Gie01] M. Giese. Incremental Closure of Free Variable Tableaux. In R. Goré, A. Leitsch, and T. Nipkow, eds., *Proc. IJCAR 2001*, LNAI 2083, Springer, 2001.

[Gie02] M. Giese. A Model Generation Style Completeness Proof For Constraint Tableaux With Superposition. In U. Egly and C. G. Fermüller, eds., *Proc. TABLEAUX 2002*, LNCS 2381, Springer, 2002.

[Gie03] M. Giese. Simplification Rules for Constrained Formula Tableaux. In M. C. Mayer and F. Pirri, eds., *Proc. TABLEAUX 2003*, LNCS, Springer, 2003.

[LMG94] R. Letz, K. Mayr, and C. Goller. Controlled Integrations of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, 13, 1994.

[LS02] R. Letz and G. Stenz. Integration of Equality Reasoning into the Disconnection Calculus. In U. Egly and C. G. Fermüller, eds., *Proc. TABLEAUX*, LNAI 2381, Springer, 2002.

[MB88] R. Manthey and F. Bry. SATCHMO: a Theorem Prover Implemented in Prolog. In E. Lusk and R. Overbeek, eds., *Proc. CADE-9*, LNCS 310, Springer, 1988.

[NR01] R. Nieuwenhuis and A. Rubio. Paramodulation-based Theorem Proving. In J. A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.

[PW07] B. Pelzer and C. Wernhard. System Description: E-KRHyper Fachberichte Informatik 13-2007, Universität Koblenz-Landau, 2007.

[SS06] Geoff Sutcliffe and Christian Suttner. The State of CASC. *AI Communications*, 19(1):35–48, 2006.

[Wei01] C. Weidenbach. Combining Superposition, Sorts and Splitting. In A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*. North Holland, 2001.

A Proofs

The general technique to prove the E-hyper tableau calculus complete is taken from the completeness proof of the superposition calculus [BG98,NR01] but adapted to our needs. One of the key concepts concerns the construction of a model of a clause set under certain conditions. That model constructed is presented as a (convergent) rewrite system. We will describe these concepts next.

A.1 Orderings and Rewrite Rules

Our approach makes heavy use of term rewrite systems and term orderings. We only mention here some details specific to our framework and refer to the literature [BG98,NR01, e.g.] for standard definitions otherwise.

We suppose as given a reduction ordering $\succ$ that is total on ground Σ-terms.[17] The non-strict ordering induced by $\succ$ is denoted by $\succeq$, and $\prec$ and $\preceq$ denote the converse of $\succ$ and $\succeq$, respectively.

A *(rewrite) rule* is an expression of the form $l \to r$ where l and r are Σ-terms. A *rewrite system* is a (possibly infinite) set of rewrite rules. A ground rewrite system R is *ordered by* $\succ$ iff $l \succ r$, for every rule $l \to r \in R$, and R is *lhs-irreducible* if it contains no two different rules of the forms $l \to r$ and $s[l] \to t$. In other words, no left hand side of a rule can be rewritten by another rule.

Notice that any ground rewrite system ordered by $\succ$ and without lhs-overlaps is a convergent ground rewrite system.[18] It is well known that for any convergent rewrite system R, and any two terms s and t, $R \models_{\mathrm{E}} s \simeq t$ if and only if there is a (exactly one) term u such that $s \to_R^\star u$ and $t \to_R^\star u$. This result thus applies in particular to ground lhs-irreducible convergent rewrite systems.

In the sequel, the letter R will always denote a ground lhs-irreducible rewrite system.

By a slight abuse of notation we will write $R \models F$ for a ground rewrite system R and clause (set) F iff the interpretation $\{l \simeq r \mid l \to r \in R\}$ satisfies F. (Similarly for $R \models_{\mathrm{E}} F$.)

A.2 Model Construction

This section is presenting the proof of Theorem 3.2 (Static Completeness). Let $\mathcal{C}$ be a (possibly infinite) set of clauses. (In the completeness proof $\mathcal{C}$ will be obtained as a certain limit branch of a tableau.) We show how $\mathcal{C}$ induces a ground lhs-irreducible rewrite system $R_{\mathcal{C}}$.

First, for a positive ground Σ-clause C we define by induction on the term ordering $\succ$ sets of rewrite rules ε_C and R_C as follows (we leave the parameter $\mathcal{C}$ implicit). Assume

[17] A *reduction ordering* is a strict partial ordering that is well-founded and is closed unter context i.e., $s \succ s'$ implies $t[s] \succ t[s']$ for all terms t, and liftable, i.e., $s \succ t$ implies $s\delta \succ t\delta$ for every term s and t and substitution δ.

[18] A *convergent* rewrite system is one that is confluent and terminating.

that ε_D has already been defined for all ground Σ-clauses D with $C \succ D$. Where $R_C = \bigcup_{C \succ D} \varepsilon_D$, define

$$\varepsilon_C = \begin{cases} \{l \to r\} & \text{if } C = l \simeq r \leftarrow \text{ is a ground instance of some positive} \\ & \text{unit clause in } \mathcal{C},\ l \succ r, \text{ and } l \text{ is irreducible wrt. } R_C \\ \emptyset & \text{otherwise} \end{cases}$$

Then, $R_{\mathcal{C}} = \bigcup_C \varepsilon_C$ where C ranges over all ground Σ-clauses.

By construction, $R_{\mathcal{C}}$ has no critical pairs thus is an lhs-irreducible rewrite system. Since $\succ$ is a well-founded ordering, $R_{\mathcal{C}}$ is a convergent rewrite system by construction. The given clause set $\mathcal{C}$ comes into play as stated only in the first condition of the definition of ε_C. An important detail is that according to our convention the equations $s \simeq t$ and $t \simeq s$ are treated as the same. Thus, if $s \prec t$ then $s \simeq t \leftarrow$ may still be turned into the rewrite rule $t \to s$ in $R_{\mathcal{C}}$ by means of its symmetric version $t \simeq s \leftarrow$.

Observe that even if $\mathcal{C}$ is a set of positive unit clauses, then $R_{\mathcal{C}}$, even if convergent, maybe incomplete wrt. the equational theory presented by it. For instance, with $\mathcal{C} = \{(a \simeq b \leftarrow), (a \simeq c \leftarrow)\}$ and the ordering $a \succ b \succ c$ the induced rewrite system $R_{\mathcal{C}} = \{a \to c\}$ is clearly incomplete wrt. the equational theory $\{a \simeq b, a \simeq c\}$. In general then it might be necessary to add enough positive unit clauses to $\mathcal{C}$ to make $R_{\mathcal{C}}$ complete. The E-hyper tableau calculus does that, but not for the positive non-unit clauses, which are handled differently, by splitting.

The following lemma states that satisfaction of a clause C in R_C is preserved as R_C is being extended.

Lemma A.1. *Let $\mathcal{C}$ be a clause set, C a ground clause, and R and R' rewrite systems such that $R_C \subseteq R \subseteq R' \subseteq R_{\mathcal{C}}$. If $R \models_{\mathrm{E}} C$ then $R' \models_{\mathrm{E}} C$.*

Proof. Writing C as the clause $\mathcal{A} \leftarrow \mathcal{B}$, we suppose $R \models_{\mathrm{E}} \mathcal{A} \leftarrow \mathcal{B}$ and show $R' \models_{\mathrm{E}} \mathcal{A} \leftarrow \mathcal{B}$.

If $R \models_{\mathrm{E}} \mathcal{B}$ (reading $\mathcal{B}$ as a conjunction of atoms) then with $R \models_{\mathrm{E}} \mathcal{A} \leftarrow \mathcal{B}$ it follows $R \models_{\mathrm{E}} A$, for some head atom A of $\mathcal{A} \leftarrow \mathcal{B}$. From monotonicity of first-order logic with equality, and with $R' \supseteq R$ it follows $R' \models_{\mathrm{E}} A$ and, trivially, $R' \models_{\mathrm{E}} \mathcal{A} \leftarrow \mathcal{B}$. Hence assume $R \not\models_{\mathrm{E}} \mathcal{B}$ from now on.

By way of contradiction assume $R' \models_{\mathrm{E}} \mathcal{B}$ but $R' \not\models_{\mathrm{E}} A$, for any head atom A (of $\mathcal{A} \leftarrow \mathcal{B}$). That $R' \models_{\mathrm{E}} \mathcal{B}$ holds but $R \models_{\mathrm{E}} \mathcal{B}$ does not hold means there is at least on body equation $s \simeq t$ in $\mathcal{B}$ such that $R' \models_{\mathrm{E}} s \simeq t$ but $R \not\models_{\mathrm{E}} s \simeq t$. Because $R_{\mathcal{C}}$ is convergent (this follows easily from its construction) and hence also its subsets R and R' are convergent, conclude that $s \simeq t$ is joinable by R' but not by R.

Every rule $l \to r \in R_{\mathcal{C}}$ is obtained from a ground instance $l \simeq r \leftarrow$ of a positive unit clause from the clause set $\mathcal{C}$. From $l \to r \in (R' \setminus R)$ and $R \supseteq R_C$ it follows $l \to r \notin R_C$. By definitition of R_C then $(l \simeq r \leftarrow) \succeq C$. (In fact even $(l \simeq r \leftarrow) \succ C$ because these two clauses are different.) This entails that the head atom $l \simeq r$ (of the unit clause $l \simeq r \leftarrow$) is greater or equal than the body atom $s \simeq t$, i.e. $\{l, r\} \succeq \{s, s, t, t\}$. It follows that l is greater than even the maximum of s and t. But then it is impossible (essentially, by the subterm property of reduction orderings) that the rule $l \to r$ can be used to rewrite the term s or the term t. Because this holds for every rule in $(R' \setminus R)$, the R'- and R-normalforms of s and t are the same. This leads to a contradiction to the conclusion that

$R' \models_{\mathrm{E}} s \simeq t$ holds but $R \models_{\mathrm{E}} s \simeq t$ does not hold. Hence, the assumption that $R' \models_{\mathrm{E}} \mathcal{B}$ holds but $R_C \models_{\mathrm{E}} A$ does not hold must be given up. This entails $R' \not\models_{\mathrm{E}} \mathcal{B}$ or $R' \models_{\mathrm{E}} A$, for some head atom A. Equivalently, $R' \models_{\mathrm{E}} \mathcal{A} \leftarrow \mathcal{B}$. □

Occasionally the following lemma comes handy.

Lemma A.2. *Let $\mathcal{C}$ be a clause set and C and D ground clauses. If $C \succ D$ then $R_D \cup \varepsilon_D \subseteq R_C$*

Proof. By definition $R_C = \bigcup_{C \succ E} \varepsilon_E$ and $R_D = \bigcup_{D \succ E} \varepsilon_E$. With $C \succ D$ it follows $\varepsilon_D \subseteq R_C$ and $R_D \subseteq R_C$. Together, thus, $R_D \cup \varepsilon_D \subseteq R_C$. □

Proposition A.3 (Model construction). *Let $\mathcal{C}$ be a clause set that is saturated up to redundancy and such that $\square \notin \mathcal{C}$. Then, for every ground instance C of every clause from $\mathcal{C}$ the following holds:*

1. *If $\mathcal{C}_C \models_{\mathrm{E}} C$ then $\varepsilon_C = \emptyset$ and $R_C \models_{\mathrm{E}} C$.*
2. *If $\mathcal{C}_C \not\models_{\mathrm{E}} C$ then $R_C \cup \varepsilon_C \models_{\mathrm{E}} C$.*

That is, either C is redundant wrt. $\mathcal{C}$ and R_C already satisfies C, or else, when C is not redundant wrt. $\mathcal{C}$, extension of R_C by ε_C will satisfy C. However, the case $\varepsilon_C = \emptyset$ is possible. For example, when $C = \ \leftarrow a \simeq b$ and $\mathcal{C}_C = \emptyset$.

But, in any case the proposition gives $R_C \cup \varepsilon_C \models_{\mathrm{E}} C$.

Proof. The claim is proved by well-founded induction on the ground instances of the clauses from $\mathcal{C}$. Hence chose any ground instance C of a clause from $\mathcal{C}$ arbitrarily and assume the proposition holds for all ground instances D of all clause from $\mathcal{C}$ such that $C \succ D$.

1. $\mathcal{C}_C \models_{\mathrm{E}} C$.
Regarding item 1, assume $\mathcal{C}_C \models_{\mathrm{E}} C$, i.e. C is redundant wrt. $\mathcal{C}$. By induction, combining cases 1 and 2, we get $R_D \cup \varepsilon_D \models_{\mathrm{E}} D$, for every clause $D \in \mathcal{C}_C$. With Lemma A.2 conclude $R_D \cup \varepsilon_D \subseteq R_C$, and with Lemma A.1 it follows $R_C \models_{\mathrm{E}} D$, for every clause $D \in \mathcal{C}_C$. Equivalently, $R_C \models_{\mathrm{E}} \mathcal{C}_C$. With $\mathcal{C}_C \models_{\mathrm{E}} C$ conclude $R_C \models_{\mathrm{E}} C$, as desired. This completes the proof of the first part of item 1.

To show $\varepsilon_C = \emptyset$ assume, by contradiction, $\varepsilon_C = \{l \to r\}$, where $C = l \simeq r \leftarrow$. Recall we have just shown $R_C \models_{\mathrm{E}} \mathcal{C}_C$. As for any convergent rewrite system, two (ground) terms are equal in the E-interpretation induced by R_C iff their normal forms wrt. R_C are the same. Applied to the situation here, this means that l and r have the same R_C-normal form. In particular, thus, some rule from R_C must be applicable to the larger term (wrt. $\succ$) of l and r, which is l. But then, by definition we have $\varepsilon_C = \emptyset$ then. A plain contradiction. This completes the proof of the first item.

2. $\mathcal{C}_C \not\models_{\mathrm{E}} C$.
Turning to item 2, suppose from now on $\mathcal{C}_C \not\models_{\mathrm{E}} C$, i.e., C is not redundant wrt. $\mathcal{C}$. It follows that no clause $D \in \mathcal{C}$ that C is a ground instance of can be redundant wrt. $\mathcal{C}$ either. We use this fact below to enable using items 1-3 of Definition 3.1.

We distinguish various cases on the form of C, most of them leading to a contradiction, though, thus ruling out that these forms are possible (in fact, when C is of any of these forms it will be redundant wrt. $\mathcal{C}$). For the (two) non-contradictory subcases we will show $R_C \cup \varepsilon_C \models_{\mathrm{E}} C$.

2-1. $C = (D[x])\gamma$ and $x\gamma$ is reducible wrt. R_C.
Suppose $C = D\gamma$, for some clause $D \in \mathcal{C}$ and some (grounding) substitution γ, that D contains a variable x, i.e., $D = D[x]$, and $x\gamma$ is reducible wrt. R_C. That is, $x\gamma = x\gamma[l]$ for some rule $l \to r \in R_C$.

Let γ' be the substitution that is the same as γ, except for x, where we set $x\gamma' = x\gamma[r]$. That is, γ' is like γ but with the rewrite rule $l \to r$ applied to $x\gamma$. From $l \succ r$ it follows $D\gamma' \prec D\gamma$. By the induction hypothesis $R_{D\gamma'} \cup \varepsilon_{D\gamma'} \models_{\mathrm{E}} D\gamma'$. From $D\gamma' \prec D\gamma$ conclude $R_{D\gamma'} \cup \varepsilon_{D\gamma'} \subseteq R_{D\gamma}$. Together with Lemma A.1 it follows $R_{D\gamma} \models_{\mathrm{E}} D\gamma'$. Because of $l \to r \in R_C$, $D\gamma = C$ and by definition of γ' conclude with congruence $R_C \models_{\mathrm{E}} C$, a plain contradiction to $\mathcal{C}_C \not\models_{\mathrm{E}} C$ as assumed above.

2-2. $C = (\mathcal{A} \leftarrow s \simeq t, \mathcal{B})\gamma$ and $s\gamma = t\gamma$.
If $C = (\mathcal{A} \leftarrow s \simeq t, \mathcal{B})\gamma$, for some clause $(\mathcal{A} \leftarrow s \simeq t, \mathcal{B}) \in \mathcal{C}$ and grounding substitution γ, and $s\gamma = t\gamma$ then there is an inference $(\mathcal{A} \leftarrow s \simeq t, \mathcal{B}) \Rightarrow_{\mathsf{ref}(\sigma)} (\mathcal{A} \leftarrow \mathcal{B})\sigma$, where σ is a mgu of s and t (and there is a substitution δ such that $\gamma = \sigma\delta$).

Neither the clause $\mathcal{A} \leftarrow s \simeq t, \mathcal{B}$ nor the clause $(\mathcal{A} \leftarrow s \simeq t, \mathcal{B})\sigma$ is redundant wrt. $\mathcal{C}$. This follows trivially from the assumption of case 2, that their instance C is not redundant wrt. $\mathcal{C}$. By saturation (Definition 3.1-3) the inference $(\mathcal{A} \leftarrow s \simeq t, \mathcal{B}) \Rightarrow_{\mathsf{ref}(\sigma)} (\mathcal{A} \leftarrow \mathcal{B})\sigma$ is redundant wrt. $\mathcal{C}$. In particular, thus, its ground instance $C \Rightarrow_{\mathsf{ref}(\varepsilon)} (\mathcal{A} \leftarrow \mathcal{B})\gamma$ is redundant wrt. $\mathcal{C}$. By definition of redundancy, $\mathcal{C}_C \models_{\mathrm{E}} (\mathcal{A} \leftarrow \mathcal{B})\gamma$. It follows trivially $\mathcal{C}_C \models_{\mathrm{E}} C$, a plain contradiction to $\mathcal{C}_C \not\models_{\mathrm{E}} C$ as assumed above.

2-3. $C = (\mathcal{A} \leftarrow s \simeq t, \mathcal{B})\gamma$, $s\gamma \succ t\gamma$ and $s\gamma$ is irreducible wrt. R_C.
Assume $C = (\mathcal{A} \leftarrow s \simeq t, \mathcal{B})\gamma$ for some clause $(\mathcal{A} \leftarrow s \simeq t, \mathcal{B}) \in \mathcal{C}$ and grounding substitution γ. We may assume $s\gamma \neq t\gamma$ because otherwise case 2-2 applies. Without loss of generality let $s\gamma$ be the larger side of the equation $(s \simeq t)\gamma$, i.e. $s\gamma \succ t\gamma$. Assume further $s\gamma$ is irreducible wrt. R_C. This entails that $s\gamma$ and $t\gamma$ are not joinable wrt. R_C. Thus, $R_C \not\models_{\mathrm{E}} s\gamma \simeq t\gamma$, which trivially entails $R_C \models_{\mathrm{E}} C$. Finally, as C is not a positive unit clause we have trivially $\varepsilon_C = \emptyset$, which concludes this case.

2-4. $C = (s \simeq t \leftarrow\ )\gamma$, $s\gamma \succ t\gamma$ and $s\gamma$ is irreducible wrt. R_C.
Assume $C = (s \simeq t \leftarrow\ )\gamma$ for some positive unit clause $(s \simeq t \leftarrow\ ) \in \mathcal{C}$ and grounding substitution γ. We may assume $s\gamma \neq t\gamma$ because otherwise the claim follows trivially. Without loss of generality let $s\gamma$ be the larger side of the equation $(s \simeq t)\gamma$, i.e. $s\gamma \succ t\gamma$. Assume further $s\gamma$ is irreducible wrt. R_C. Thus, $\varepsilon_C = \{s\gamma \to t\gamma\}$, which trivially entails $R_C \cup \varepsilon_C \models_{\mathrm{E}} C$.

2-5. $C = (A_1, \ldots, A_m \leftarrow\ )\gamma$, for some $m \geq 2$.
Assume $C = D\gamma$ for some positive non-unit clause $D = (A_1, \ldots, A_m \leftarrow\ ) \in \mathcal{C}$, where $m \geq 2$, and grounding substitution γ. It is not difficult to see that γ can be obtained by composition of some purifying substitution π for D and some other substitution δ, i.e. $\gamma = \pi\delta$. Such a substitution π always exists, it could be γ itself.

Neither D nor $D\pi$ is redundant wrt. $\mathcal{C}$. This follows trivially from the assumption of case 2, that their instance $C = D\gamma = D\pi\delta$ is not redundant wrt. $\mathcal{C}$. By saturation (Definition 3.1-1) the inference $D \Rightarrow_{\mathsf{split}(\pi)} A_1\pi \leftarrow\ , \ldots, A_m\pi \leftarrow$ is redundant wrt. $\mathcal{C}$.

In particular, thus, its ground instance $C \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\gamma \leftarrow, \ldots, A_m\gamma \leftarrow$ is redundant wrt. $\mathcal{C}$. By definition of redundancy, $\mathcal{C}_C \models_{\mathrm{E}} A_i\gamma \leftarrow$, for some i with $1 \leq i \leq m$. It follows trivially $\mathcal{C}_C \models_{\mathrm{E}} C$, a plain contradiction to $\mathcal{C}_C \not\models_{\mathrm{E}} C$ as assumed above.

2-6. $C = (D[s])\gamma$ and $s\gamma$ is reducible at a non-variable position.
To make the case analysis exhaustive assume that C does not fall into one of the cases 2-1 – 2.5. We analyze further the form C can take. Assume $C = D\gamma$ for some clause $D \in \mathcal{C}$ and grounding substitution γ

In the first case D has a non-empty body. Because the cases 2-2 and 2-3 are excluded, D can be written as $\mathcal{A} \leftarrow s \simeq t, \mathcal{B}$, where $s\gamma \succ t\gamma$ and $s\gamma$ is reducible wrt. R_C.

In the second case D has an empty body. Because the cases 2-4 and 2-5 are excluded, and we are given that $\mathcal{C}$ does not contain the empty clause, D must be a positive unit clause and can be written as $s \simeq t \leftarrow$, where $s\gamma \succ t\gamma$ and $s\gamma$ is reducible wrt. R_C.

Doing both cases together, consider any rule $l \to r \in R_C$ that rewrites $s\gamma$. Because case 2-1 is excluded, $l \to r$ does not rewrite $s\gamma$ at or below a variable position of s. That is, any position p such that $s\gamma[l]_p$ holds is a non-variable position of s.

We continue the proof doing both cases together.

By construction the rewrite rule $l \to r$ is obtained from a ground instance of some positive unit equation from $\mathcal{C}$. Let $E = l' \simeq r' \leftarrow$ be a fresh variant of that positive unit equation. Because it is fresh, we may assume γ has been extended so as to give $l'\gamma = l$ and $r'\gamma = r$.

We must have $C \succ E\gamma(= (l' \simeq r' \leftarrow)\gamma)$ because otherwise $l'\gamma \to r'\gamma \in R_C$ (i.e., $l \to r \in R_C$) would be impossible. Therefore we can apply induction to $E\gamma$. If case 1 applies, i.e. $\mathcal{C}_{E\gamma} \models_{\mathrm{E}} E\gamma$ then $\varepsilon_{E\gamma} = \emptyset$ and so $l'\gamma \to r'\gamma(= l \to r)$ could not be a rewrite rule in R_C and thus neither in R_C. Case 1 is thus impossible. Therefore we must have $\mathcal{C}_{E\gamma} \not\models_{\mathrm{E}} E\gamma$. In other words, $E\gamma$ is not redundant wrt. $\mathcal{C}$.

As said above, D can take two different forms. If D is of the form $\mathcal{A} \leftarrow s \simeq t, \mathcal{B}$ consider the ground sup-left inference

$$(\mathcal{A}\gamma \leftarrow s\gamma[l'\gamma]_p \simeq t\gamma, \mathcal{B}\gamma), E\gamma \Rightarrow_{\mathsf{sup\text{-}left}(\varepsilon)} (\mathcal{A}\gamma \leftarrow s\gamma[r'\gamma]_p \simeq t\gamma, \mathcal{B}\gamma) \quad . \tag{1}$$

Because p is a position of a non-variable term in s, say, l'', the sup-left inference

$$(\mathcal{A} \leftarrow s[l'']_p \simeq t, \mathcal{B}), E \Rightarrow_{\mathsf{sup\text{-}left}(\sigma)} (\mathcal{A} \leftarrow s[r']_p \simeq t, \mathcal{B})\sigma \tag{2}$$

exists, where σ is a mgu of l' and l'', and $\gamma = \sigma\delta$ for some substitution δ. The ground sup-left inference (1) then is a ground instance of the sup-left inference (2).
(*) Above we concluded that $E\gamma$ is not redundant wrt. $\mathcal{C}$. Therefore the more general clause $E\sigma$ cannot be redundant wrt. $\mathcal{C}$ either. A global assumption in case 2 is that D is not redundant wrt. $\mathcal{C}$. By saturation (Definition 3.1-2) the inference (2) is redundant wrt. $\mathcal{C}$. In particular, thus, its ground instance (1) is redundant wrt. $\mathcal{C}$. For economy of notation let $F = \mathcal{A}\gamma \leftarrow s\gamma[r'\gamma]_p \simeq t\gamma, \mathcal{B}\gamma$ be the conclusion of the inference (1).

By definition of redundancy $\mathcal{C}_C \cup \{E\gamma\} \models_{\mathrm{E}} F$. By induction, combining cases 1 and 2, we get $R_G \cup \varepsilon_G \models_{\mathrm{E}} G$, for every clause $G \in \mathcal{C}_C$. With Lemma A.2 conclude $R_G \cup \varepsilon_G \subseteq R_C$, and with Lemma A.1 it follows $R_C \models_{\mathrm{E}} G$, for every clause $G \in \mathcal{C}_C$. Equivalently, $R_C \models_{\mathrm{E}} \mathcal{C}_C$.

Because $E\gamma = (l' \simeq r')\gamma$ is present as a rewrite rule $(l'\gamma \to r'\gamma) = (l \to r) \in R_C$ it follows trivially $R_C \models_{\mathrm{E}} E\gamma$. Together with $R_C \models_{\mathrm{E}} \mathcal{C}_C$ and $\mathcal{C}_C \cup \{E\gamma\} \models_{\mathrm{E}} F$ (by redundancy of the inference, as mentioned above) conclude $R_C \models_{\mathrm{E}} F$. From $l \to r \in R_C$ conclude by congruence $R_C \models_{\mathrm{E}} C$, which is a plain contradiction to $\mathcal{C}_C \not\models_{\mathrm{E}} C$ as assumed to hold for case 2. This case is thus impossible.

Similarly, if D is of the form $s \simeq t \leftarrow$ consider the ground unit-sup-right inference

$$(s\gamma[l'\gamma]_p \simeq t\gamma \leftarrow), E\gamma \Rightarrow_{\text{unit-sup-right}(\varepsilon)} (s\gamma[r'\gamma]_p \simeq t\gamma \leftarrow) \ . \tag{3}$$

Because p is a position of a non-variable term in s, say, l'', the unit-sup-right inference

$$(s[l'']_p \simeq t \leftarrow), E \Rightarrow_{\text{unit-sup-right}(\sigma)} (s[r']_p \simeq t \leftarrow)\sigma \tag{4}$$

exists, where σ is a mgu of l' and l'', and $\gamma = \sigma\delta$ for some substitution δ. The ground unit-sup-right inference (3) then is a ground instance of the unit-sup-right inference (4).

The rest of the proof of this case is the same as from (*) above and is omitted (obviously, $F = (s\gamma[r'\gamma]_p \simeq t\gamma \leftarrow)$ this time).

In conclusion, the case 2-6 is impossible, too. $\square$

Theorem 3.2 (Static Completeness). *Let $\mathcal{C}$ be a clause set saturated up to redundancy. If $\square \notin \mathcal{C}$ then $\mathcal{C}$ is* E-*satisfiable.*

Proof. Suppose $\square \notin \mathcal{C}$. To show that $\mathcal{C}$ is E-satisfiable it suffices we show that $R_{\mathcal{C}}$ is an E-model of $\mathcal{C}$. For this, it suffices to show $R_{\mathcal{C}} \models_{\mathrm{E}} C\gamma$ for an arbitrarily chosen clause $C \in \mathcal{C}$ and anarbitrarily chosen grounding substitution γ for C. To prove $R_{\mathcal{C}} \models_{\mathrm{E}} C\gamma$, we first use Proposition A.3 and conclude $R_{C\gamma} \cup \varepsilon_{C\gamma} \models_{\mathrm{E}} C\gamma$. From that, $R_{\mathcal{C}} \models_{\mathrm{E}} C\gamma$ follows immediately by Lemma A.1. $\square$

A.3 Correctness

Lemma A.5. *For each of the derivation rules* Split, Equality, Del *and* Simp, *if the premise of the rule is* E-*satisfiable, then one of its conclusions is* E-*satisfiable as well.*

Proof. Let us first focus on the inference rules sup-left and unit-sup-right. Assume the premises of such a rule is E-satisfiable and let I be a E-model; from the axioms of congruence we can immediately conclude for both rules, that I is an E-model for the conclusion as well. For ref the claim follows directly from reflexivity.

For Equality the claim is an immediate consequence from the above. For Split assume that there is an E-model I for the premise $\mathbf{B}$. Let $A_1, \cdots, A_m \leftarrow$ be the selected clause from $\mathbf{B}$. Then I is an E-model for $(A_1, \cdots, A_m \leftarrow)\pi$ where π is the purifying subsitution for $A_1, \ldots, A_m$. $A_1\pi, \ldots, A_m\pi$ have no variables in common and all variables are implicitly universally quantified; hence $\forall(A_1\pi \vee \ldots \vee A_m\pi)$ is equivalent to $\forall A_1\pi \vee \ldots \vee \forall A_m\pi$ and we conclude that I is an E-model for $\forall A_1\pi \vee \ldots \vee \forall A_m\pi$.

Hence there is an E-model for one of $\mathbf{B} \cdot A_1\pi \leftarrow^{\mathrm{d}}, \ldots, \mathbf{B} \cdot A_m\pi \leftarrow^{\mathrm{d}}$.

For Del the claim holds obviously and for Simp assume an E-model I for the premise. Let $C, D, \mathbf{B}$ and $\mathbf{B}_1$ as in the definition of Simp; from $(\mathbf{B} \cdot C \cdot \mathbf{B}_1) \models_{\mathrm{E}} D$, we conclude that D also holds in I. $\square$

Theorem A.6 (Correctness of E-Hyper Tableaux). *Let $\mathcal{C}$ be a clause set that has a refutation. Then $\mathcal{C}$ is* E-*unsatisfiable.*

Proof. Let $\mathbf{T}$ be the resulting closed tree of the refutation. From the contrapositive of Lemma A.5 we conclude that if a tree $\mathbf{T}_i$ of a derivation contains only E-unsatisfiable branches, this holds for its predecessor $\mathbf{T}_{i-1}$ as well. The final tableau T of the refutation clearly consists only of E-unsatisfiable branches and hence by induction of the length of the refutation (which is by definition a finite derivation), we can conclude that the initial tableau $\mathbf{T}_0$, which consists of one branch with the tableau clauses from C, is E-unsatisfiable. $\square$

A.4 Completeness

Lemma A.7. *Let C_1 and C_2 be ground clauses and $\mathcal{C}$ a set of ground clauses. If $(\mathbf{B}_j)_{C_1} \cup \mathcal{C} \models_{\mathrm{E}} C_2$ for some $j < \kappa$ then $(\mathbf{B}_\infty)_{C_1} \cup \mathcal{C} \models_{\mathrm{E}} C_2$.*

Proof. The proof is by well-founded induction. Suppose the result to hold for all ground constrained clauses C_1' and C_2' such that $C_1' \prec C_1$.[19]

Suppose $(\mathbf{B}_j)_{C_1} \cup \mathcal{C} \models_{\mathrm{E}} C_2$ holds for some $j < \kappa$. If $(\mathbf{B}_j)_{C_1} \subseteq (\mathbf{B}_\infty)_{C_1}$ then the result follows from the monotonicity of first-order logic with equality. Otherwise let $(\mathbf{B}_j)_{C_1}$ itself denote a finite subset of $(\mathbf{B}_j)_{C_1}$ such that the entailment in the premise of the lemma statement holds. Such a finite set exists by compactness of first-order logic with equality.

Let $\mathbf{B}' := (\mathbf{B}_j)_{C_1} \setminus (\mathbf{B}_\infty)_{C_1}$ be those clauses from $(\mathbf{B}_j)_{C_1}$ that are not an instance of any persisting clause in $\mathbf{B}_\infty$. Chose any clause $C' \in \mathbf{B}'$ arbitrary. By construction, it is a ground instance of some clause $C \in \mathbf{B}_j$ such that $C \notin \mathbf{B}_\infty$. This means that C has been removed from the clause set $\mathbf{B}_k$ labeling the node $\mathbf{N}_k$ of the branch $\mathbf{B}$, for some $k < \kappa$. In other words, the Del or Simp derivation rule has been applied to $\mathbf{B}_k$ with selected clause C. We treat the application of both derivation rules in one, but distinguish two subcases.

In the first subcase C has been removed by non-proper subsumption from $\mathbf{B}_k$. Let $D \in \mathbf{B}_k$ be the clause non-properly subsuming C. As an easy inductive consequence of the definition of the Split and Equality derivation rules, no constrained clause set derived can contain a constrained clause and a variant of it. Hence, D cannot be a variant of C and C must be a proper instance of D. Because the ordering based on the converse relation, proper generalization, is well-founded, by induction there is a clause D' in $\mathbf{B}_\infty$ that non-properly subsumes C (it could be D). Now, with C' being an instance of C, C' is an instance of D' as well. With $D' \in \mathbf{B}_\infty$, C' thus is an instance of a persisting clause in $\mathbf{B}_\infty$. With this contradiction to the construction of $\mathbf{B}'$ conclude this subcase is impossible.

Hence, as the second subcase, by definition of the Del and Simp derivation rules, the clause C, and hence its instance C' is redundant wrt. a specific subset $\mathbf{B}'' \subseteq \mathbf{B}_{k+1}$ That subset $\mathbf{B}''$ is specified in the definition of the Del and Simp derivation rules. For our purpose the only important fact is that with $\mathbf{B}'' \subseteq \mathbf{B}_{k+1}$ it follows (trivially) that C' is redundant wrt. $\mathbf{B}_{k+1}$ as well.

That C' is redundant wrt. $\mathbf{B}_{k+1}$ means by definition of redundancy $(\mathbf{B}_{k+1})_{C'} \models_{\mathrm{E}} C'$. This implies by monotonicity of first-order logic with equality $(\mathbf{B}_{k+1})_{C'} \cup \mathcal{C} \models_{\mathrm{E}} C'$

[19] Thus, formally, this is induction on the lexicographic extension of the ordering $\prec$ on pairs, which compares $(C_1', C_2') \prec (C_1, C_2)$.

With $C' \in \mathbf{B}' \subseteq (\mathbf{B}_j)_{C_1}$ it follows $C' \prec C_1$. By the induction hypothesis then

$$(\mathbf{B}_\infty)_{C'} \cup \mathcal{C} \models_{\mathrm{E}} C' \ . \tag{5}$$

From $C' \prec C_1$ it follows easily $(\mathbf{B}_\infty)_{C'} \subseteq (\mathbf{B}_\infty)_{C_1}$. Together with (5) and by monotonicity of first-order logic with equality it follows

$$(\mathbf{B}_\infty)_{C_1} \cup \mathcal{C} \models_{\mathrm{E}} C' \ . \tag{6}$$

Because of this entailment, the constrained clause C' can be removed from the premise $(\mathbf{B}_j)_{C_1}$ in the given entailment at the cost of adding the stronger set $(\mathbf{B}_\infty)_{C_1} \cup \mathcal{C}$. More formally, from $(\mathbf{B}_j)_{C_1} \cup \mathcal{C} \models_{\mathrm{E}} C_2$ and (6) it follows

$$((\mathbf{B}_\infty)_{C_1} \cup \mathcal{C}) \cup ((\mathbf{B}_j)_{C_1} \setminus \{C'\}) \cup \mathcal{C} \models_{\mathrm{E}} C_2 \ . \tag{7}$$

Repeating this procedure for each of the (finitely many) members of $\mathbf{B}'$ allows to conclude

$$((\mathbf{B}_\infty)_{C_1} \cup \mathcal{C}) \cup ((\mathbf{B}_j)_{C_1} \setminus \mathbf{B}') \cup \mathcal{C} \models_{\mathrm{E}} C_2 \ . \tag{8}$$

Recall that $\mathbf{B}' = (\mathbf{B}_j)_{C_1} \setminus (\mathbf{B}_\infty)_{C_1}$, which implies by elementary set theory $(\mathbf{B}_j)_{C_1} \setminus \mathbf{B}' \subseteq (\mathbf{B}_\infty)_{C_1}$. But then, $(\mathbf{B}_\infty)_{C_1} \cup \mathcal{C} \models_{\mathrm{E}} C_2$ follows from (8) immediately. □

Lemma A.8. *If C is redundant wrt. $\mathbf{B}_j$, for some $j < \kappa$ then C is redundant wrt. $\mathbf{B}_\infty$.*

Proof. Suppose C is redundant wrt. $\mathbf{B}_j$, for some $j < \kappa$. Let D be an arbitrarily chosen ground instance of C. By definition, D is redundant wrt. $\mathbf{B}_j$, which means $(\mathbf{B}_j)_D \models_{\mathrm{E}} D$. With Lemma A.7 it follows $(\mathbf{B}_\infty)_D \models_{\mathrm{E}} D$. In other words D is redundant wrt. $\mathbf{B}_\infty$. Because D was chosen as an arbitrary ground instance of C, C is redundant wrt. $\mathbf{B}_\infty$. □

Lemma A.9. *If $R \models_{\mathrm{E}} \mathcal{C}$ and C is redundant wrt. $\mathcal{C}$ then $R \models_{\mathrm{E}} C$.*

Proof. Suppose $R \models_{\mathrm{E}} \mathcal{C}$ and C is redundant wrt. $\mathcal{C}$. Let D be an arbitrarily chosen ground instance of C. It suffices to show $R \models_{\mathrm{E}} D$. Since C is redundant wrt. $\mathcal{C}$, by definition, its ground instance D is redundant wrt. $\mathcal{C}$. Equivalently, $\mathcal{C}_D \models_R D$, which entails $R \models_{\mathrm{E}} D$ provided $R \models_{\mathrm{E}} \mathcal{C}_D$ holds. The latter however follows immediately from $R \models_{\mathrm{E}} \mathcal{C}$ and the trivial fact that $\mathcal{C}_D$ is a subset of the set of all ground instances of all clauses from $\mathcal{C}$. □

Lemma A.10. *Let C be a clause and D a positive unit clause. Then, any inference $C, D \Rightarrow_{R(\sigma)} E$, where $R \in \{\textsf{sup-left}, \textsf{unit-sup-right}\}$, or $C \Rightarrow_{\textsf{ref}(\sigma)} E$ that is redundant wrt. $\mathbf{B}_j$, for some $j < \kappa$, is redundant wrt. $\mathbf{B}_\infty$.*

Proof. Suppose an inference $C, D \Rightarrow_{R(\sigma)} E$, where $R \in \{\textsf{sup-left}, \textsf{unit-sup-right}\}$, redundant wrt. $\mathbf{B}_j$, for some $j < \kappa$. Let γ be an arbitrary ground substitution for C and D such that $\gamma = \sigma\delta$ for some substitution δ and such that $C\gamma \Rightarrow_{R(\varepsilon)} E\delta$ is a ground instance of $C, D \Rightarrow_{R(\sigma)} E$. Because chosen arbitrarily, it suffices to show that this ground instance $C\gamma \Rightarrow_{R(\varepsilon)} E\delta$ is redundant wrt. $\mathbf{B}_\infty$.

Because the inference $C, D \Rightarrow_{R(\sigma)} E$ is redundant wrt. $\mathbf{B}_j$ its instance $C\gamma \Rightarrow_{R(\varepsilon)} E\delta$ is redundant wrt. $\mathbf{B}_j$. By definition of redundancy this means

$$(\mathbf{B}_j)_{C\gamma} \cup \{D\gamma\} \models_{\mathrm{E}} E\delta \ . \tag{9}$$

By Lemma A.7 then

$$(\mathbf{B}_\infty)_{C\gamma} \cup \{D\gamma\} \models_{\mathrm{E}} E\delta \ , \tag{10}$$

which, by definition, means that the inference $C\gamma \Rightarrow_{R(\varepsilon)} E\delta$ is redundant wrt. $\mathbf{B}_\infty$, which was to be shown.

The proof of the case of an inference $C \Rightarrow_{\mathsf{ref}(\sigma)} E$ is similar and is omitted. □

Lemma A.11. *Let C be a positive clause and π a purifying substitution for C. If the inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ is redundant wrt. $\mathbf{B}_j$, for some $j < \kappa$, then it is redundant wrt. $\mathbf{B}_\infty$.*

Proof. Suppose an inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ that is redundant wrt. $\mathbf{B}_j$, for some $j < \kappa$. Let γ be an arbitrary ground substitution for C such that $\gamma = \pi\delta$ for some substitution δ and such that $C\gamma \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\delta \leftarrow, \ldots, A_m\delta \leftarrow$ is a ground instance of $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$. Because chosen arbitrarily, it suffices to show that the ground $C\gamma \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\delta \leftarrow, \ldots, A_m\delta \leftarrow$ inference is redundant wrt. $\mathbf{B}_\infty$.

Because the inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ is redundant wrt. $\mathbf{B}_j$ its instance $C\gamma \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\delta \leftarrow, \ldots, A_m\delta \leftarrow$ is redundant wrt. $\mathbf{B}_j$. By definition of redundancy this means that $A_i\delta \leftarrow$ is redundant wrt. $\mathbf{B}_j$, for some i with $1 \leq i \leq m$. By Lemma A.8 then $A_i\delta \leftarrow$ is redundant wrt. $\mathbf{B}_\infty$. It follows immediately that the ground inference $C\gamma \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\delta \leftarrow, \ldots, A_m\delta \leftarrow$ is redundant wrt. $\mathbf{B}_\infty$, which remained to be shown. □

Proposition 4.4 (Exhausted branches are saturated up to redundancy). *If $\mathbf{B}$ is an exhausted branch of a limit tree of some fair derivation then $\mathbf{B}_\infty$ is saturated up to redundancy.*

Proof. Suppose $\mathbf{B}$ is an exhausted branch of a limit tree of some fair derivation. According to Definition 3.1 it suffices to chose arbitrarily a clause $C \in \mathbf{B}_\infty$ that is not redundant wrt. $\mathbf{B}_\infty$ and prove the properties 1-3 claimed there for C.

Before doing that, notice that if there is a $j < \kappa$ such that C is redundant wrt. $\mathbf{B}_j$ then by Lemma A.8 the clause C is redundant wrt. $\mathbf{B}_\infty$ and nothing remains to be shown for C. Hence suppose from now on that C is not redundant wrt. $\mathbf{B}_j$, for all $j < \kappa$.

1. $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$
Suppose there is an inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$. It suffices to show that this inference is redundant wrt. $\mathbf{B}_\infty$, or that $C\pi$ is redundant wrt. $\mathbf{B}_\infty$.

If there is a $j < \kappa$ such that $C\pi$ is redundant wrt. $\mathbf{B}_j$ then by Lemma A.8 $C\pi$ is redundant wrt. $\mathbf{B}_\infty$, and nothing remains to be shown. Hence suppose that $C\pi$ is not redundant wrt. $\mathbf{B}_j$, for all $j < \kappa$.

It suffices to show that an arbitrarily chosen ground instance of the inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$. is redundant wrt. $\mathbf{B}_\infty$. Hence let γ be an arbitrary ground substitution for C such that $\gamma = \pi\delta$ for some substitution δ, and such that $C\gamma \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\delta \leftarrow$

$,\ldots,A_m\delta \leftarrow$ is a ground instance of of the inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$. We will show that this ground inference is redundant wrt. $\mathbf{B}_\infty$.

From $C \in \mathbf{B}_\infty$ it follows there is an $i < \kappa$ such that for all $j \geq i$ with $j < \kappa$ it holds $C \in \mathbf{B}_j$. Because of $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ Split is applicable (in particular) to $\mathbf{B}_i$ with underlying inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ unless $A_j \leftarrow$, for some j with $i \leq j \leq m$ is contained as a variant in $\mathbf{B}_i$. In this case, by virtue of the ground instance $A_j\delta \leftarrow$ of $A_j \leftarrow$ it follows that the ground instance $C\gamma \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\delta \leftarrow, \ldots, A_m\delta \leftarrow$ is redundant wrt. $\mathbf{B}_i$ and nothing remains to be shown.

Recall we are considering the case that $C\pi$ is not redundant wrt. $\mathbf{B}_j$, for every $j < \kappa$.

But then, by Definition 4.2-1 there is a $k < \kappa$ such that the inference $C \Rightarrow_{\mathsf{split}(\pi)} A_1 \leftarrow, \ldots, A_m \leftarrow$ is redundant wrt. $\mathbf{B}_k$. By Lemma A.11 then, this inference is redundant wrt. $\mathbf{B}_\infty$. Therefore, in particular its (ground) instance $C\gamma \Rightarrow_{\mathsf{split}(\varepsilon)} A_1\delta \leftarrow, \ldots, A_m\delta \leftarrow$ is redundant wrt. $\mathbf{B}_\infty$, which remained to be shown.

2. $C, D \Rightarrow_{R(\sigma)} E$, where $R \in \{\mathsf{sup\text{-}left}, \mathsf{unit\text{-}sup\text{-}right}\}$
This case is concerned with Equality inferences. More precisely, suppose there is an inference $C, D \Rightarrow_{R(\sigma)} E$, where $R \in \{\mathsf{sup\text{-}left}, \mathsf{unit\text{-}sup\text{-}right}\}$ and D is a fresh variant of a positive unit clause from $\mathbf{B}_\infty$ and σ is some substitution.

It suffices to show that this inference is redundant wrt. $\mathbf{B}_\infty$, or that $C\sigma$ or $D\sigma$ is redundant wrt. $\mathbf{B}_\infty$.

If there is a $j < \kappa$ such that $C\sigma$ is redundant wrt. $\mathbf{B}_j$ then by Lemma A.8 $C\sigma$ is redundant wrt. $\mathbf{B}_\infty$, and nothing remains to be shown. Hence suppose that $C\sigma$ is not redundant wrt. $\mathbf{B}_j$, for all $j < \kappa$. By exactly the same argumentation, this time applied to $D\sigma$, we may assume that $D\sigma$ is not redundant wrt. $\mathbf{B}_j$, for all $j < \kappa$.

It suffices to show that an arbitrarily chosen ground instance of the inference $C, D \Rightarrow_{R(\sigma)} E$ is redundant wrt. $\mathbf{B}_\infty$. Hence let γ be an arbitrary ground substitution for C and D such that $\gamma = \sigma\delta$ for some substitution δ, and such that $C\gamma, D\gamma \Rightarrow_{R(\varepsilon)} E\delta$ is a ground instance of of the inference $C, D \Rightarrow_{R(\sigma)} E$. Hence we will show that this ground inference $C\gamma, D\gamma \Rightarrow_{R(\varepsilon)} E\delta$ is redundant wrt. $\mathbf{B}_\infty$.

From $C \in \mathbf{B}_\infty$ it follows there is an $i < \kappa$ such that for all $j \geq i$ with $j < \kappa$ it holds $C \in \mathbf{B}_j$. Likewise, from D being a variant of a clause in $\mathbf{B}_\infty$ it follows there is an i' such that for all $j' \geq i'$ it holds D is a variant of a clause in $\mathbf{B}_{j'}$. Without loss of generality assume $i \geq i'$. It follows D is a variant of a clause in $\mathbf{B}_j$, for all $j \geq i$.

From the just said, and because of $C, D \Rightarrow_{R(\sigma)} E$, Equality is applicable (in particular) to $\mathbf{B}_i$ with underlying inference $C, D \Rightarrow_{R(\sigma)} E$ unless E is contained as a variant in $\mathbf{B}_i$. In this case, the inference $C, D \Rightarrow_{R(\sigma)} E$ is redundant wrt. $\mathbf{B}_i$ and nothing remains to be shown.

Recall we are currently considering the case that neither $C\sigma$ nor $D\sigma$ is redundant wrt. $\mathbf{B}_j$, for every $j < \kappa$.

But then, by Definition 4.2-2 there is a $k < \kappa$ such that the inference $C, D \Rightarrow_{R(\sigma)} E$ is redundant wrt. $\mathbf{B}_k$. By Lemma A.10 then, this inference is also redundant wrt. $\mathbf{B}_\infty$. Therefore, in particular its (ground) instance $C\gamma, D\gamma \Rightarrow_{R(\varepsilon)} E\delta$ is redundant wrt. $\mathbf{B}_\infty$, which remained to be shown.

3. $C \Rightarrow_{\mathsf{ref}(\sigma)} E$

This case is concerned with an Equality inference, more precisely with an application of the ref rule. The proof is done analogously to case 2 and is omitted.

□

Theorem 4.5 (Completeness of E**-Hyper Tableaux).** *Let $\mathcal{C}$ be a clause set and* $\mathbf{D}$ *a fair derivation of $\mathcal{C}$. If* $\mathbf{D}$ *is not a refutation then $\mathcal{C}$ is satisfiable.*

Proof. Suppose that $\mathbf{D}$ is not a refutation. Therefore its limit tree $\mathbf{T}$ has an exhausted branch. Let $\mathbf{B}$ be any such exhausted branch.

By Proposition 4.4 the clause set $\mathbf{B}_\infty$ is saturated up to redundancy. Moreover, $\mathbf{B}_\infty$ cannot contain the empty clause, because, if it did, $\mathbf{B}$ would contain it too, but no exhausted branch can contain the empty clause.

With Theorem 3.2 it follows $\mathbf{B}_\infty$ is satisfiable. Moreover, the proof of Theorem 3.2 gives us a convergent rewrite system $R_{\mathbf{B}_\infty}$ such that $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} \mathbf{B}_\infty$.

To prove the theorem it suffices to show $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} \mathcal{C}$. To show that, let C be any clause from $\mathcal{C}$, and it suffices to show $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} C$. By definition of derivation, $C \in \mathbf{B}_0$, where $\mathbf{B}_0$ is the (single) branch of the initial tableau $\mathbf{T}_0$ of the derivation $\mathbf{D}$.

If $C \in \mathbf{B}_\infty$ then with $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} \mathbf{B}_\infty$ immediately conclude $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} C$. Hence suppose $C \notin \mathbf{B}_\infty$ from now on.

From $C \in \mathbf{B}_0$ and $C \notin \mathbf{B}_\infty$ it follows that C has been removed at some time $k < \kappa$ from the clause set $\mathbf{B}_k$ by an application of the Del or the Simp derivation rule. We distinguish both cases at once.

In the first subcase C has been removed by non-proper subsumption from $\mathbf{B}_k$. Let $D \in \mathbf{B}_k$ be the clause non-properly subsuming C. As an easy inductive consequence of the definition of the Split and Equality derivation rules, no clause set $\mathbf{B}_i$ derived can contain both a clause and a variant of it. Hence, D cannot be a variant of C and C must be a proper instance of D. Because the ordering based on the converse relation, proper generalization, is well-founded, by induction on this ordering there is a clause D' in $\mathbf{B}_\infty$ that non-properly subsumes C (it could be D). Now, to D' the case $D' \in \mathbf{B}_\infty$ above applies and, clearly, $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} D'$ entails $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} C$.

In the second subcase C is redundant wrt. a specific subset $\mathbf{B}'$ of the derived branch $\mathbf{B}_{k+1}$, where $\mathbf{B}'$ is specified in the definition of the Del and Simp derivation rules. Because $\mathbf{B}' \subseteq \mathbf{B}_{k+1}$ it follows trivially that C is redundant wrt. $\mathbf{B}_{k+1}$.

By Lemma A.8, C is redundant wrt. $\mathbf{B}_\infty$. With $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} \mathbf{B}_\infty$ and Lemma A.9 it follows $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} C$, which trivially implies $R_{\mathbf{B}_\infty} \models_{\mathrm{E}} C$. □