J Autom Reasoning (2008) 40:179–194
DOI 10.1007/s10817-007-9089-7

Connection Tableaux with Lazy Paramodulation

Andrei Paskevich

Received: 5 November 2007 / Accepted: 7 November 2007 / Published online: 11 December 2007

Abstract It is well known that the connection refinement of clause tableaux with paramodulation is incomplete (even with weak connections). In this paper, we present a new connection tableau calculus for logic with equality. This calculus is based on a lazy form of paramodulation where parts of the unification step become auxiliary subgoals in a tableau and may be subjected to subsequent paramodulations. Our calculus uses ordering constraints and a certain form of the basicness restriction.

Keywords Connection tableaux · Lazy paramodulation · Basic ordered paramodulation · First-order logic with equality

1 Introduction

The model elimination proof procedure was originally introduced by Loveland as a resolution-based calculus with clauses of a special form [10]. Later it was reconsidered as a clause tableau calculus, where proof search is guided by connections between clauses [8]. In this form, the method is also referred to as *connection tableaux*.

Connection tableaux are a powerful goal-directed refinement of general clause tableaux. Further, strong search pruning methods and efficient implementation techniques were developed for this calculus [9].

It is tempting to adapt connection tableaux for logic with equality by introducing paramodulation. That is, we could make a pair (equality to paramodulate by, literal to paramodulate in) constitute a connection, too, and add rules for paramodulation in a branch. Unfortunately, such a calculus turns out to be incomplete. Consider the

A. Paskevich (✉)
Laboratoire d'Algorithmique, Complexité et Logique,
Université Paris 12, 94010 Créteil Cedex, France
e-mail: andrei@capet.iut-fbleau.fr

clause set $\{a \approx b,\ c \approx d,\ \neg P(f(a),\ f(b)),\ \neg Q(g(c),\ g(d)),\ P(x, x) \vee Q(y, y)\}$. Let us try to build a refutation of $\mathcal{S}$ in that hypothetical calculus.

$$\dfrac{\dfrac{\dfrac{\dfrac{a \approx b}{\neg P(f(a),\ f(b))}}{\neg P(f(b),\ f(b))}}{P(x, x)}}{\dfrac{\bot \cdot (x = f(b))}{}} \qquad \dfrac{Q(y, y)}{?}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg Q(g(c),\ g(d))}{c \approx d}}{\neg Q(g(d),\ g(d))}}{P(x, x)}}{\dfrac{?}{}} \qquad \dfrac{Q(y, y)}{\bot \cdot (y = g(d))}$$

We cannot continue the first inference because the literal $Q(y, y)$ does not match $Q(g(c), g(d))$ and the equality $c \approx d$ cannot be applied to $Q(y, y)$, either. The second inference will fail in a similar way.

The fact that paramodulation works fine in resolution-style calculi [16] and general clause tableaux [4, 7] is due to a flexible order of inferences that is impossible in a goal-directed calculus. The calculus could be made complete if we allow paramodulation into variables and add the axioms of *functional reflexivity* ($f(x) \approx f(x)$, $g(x) \approx g(x)$, etc.) in order to construct new terms [11]. This approach is quite inefficient in practice, however, since functional reflexivity allows us to substitute an arbitrary term for any variable.

To handle problems with equality, competitive connection tableau provers [13] employ various forms of Brand's modification method [2, 3, 15]. This method transforms a clause set with equality into an equiconsistent set where the equality predicate does not occur. In addition, a complete procedure was developed based on a combination of goal-directed proof search in tableaux and a bottom-up equality saturation using basic ordered paramodulation [14].

In this paper we propose an alternative approach for equality handling in connection tableaux that is based on *lazy paramodulation*. This technique was originally introduced by J. Gallier and W. Snyder as a method for general E-unification [6] and used later to overcome incompleteness of the set-of-support strategy (another example of a goal-directed method) in the classical paramodulation calculus [18].

So, what is lazy paramodulation? Above, we noted that the literal $Q(y, y)$ cannot be unified with $Q(g(c), g(d))$. But let us postpone unification until the equality $c \approx d$ is applied to the second literal. Let us make the equality $Q(y, y) = Q(g(c), g(d))$ not a constraint to solve but an additional subgoal to prove. The clause set from the previous counterexample can be easily refuted in such a calculus.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P(x, x)}{\neg P(f(a),\ f(b))}}{f(a) \not\approx x}}{a \approx b}}{f(b) \not\approx x}}{\dfrac{\bot \cdot (f(b) = x)}{}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Q(y, y)}{\neg Q(g(c),\ g(d))}}{f(b) \not\approx x}}{\bot \cdot (f(b) = x)}}{a \not\approx a}}{\bot}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Q(y, y)}{\neg Q(g(c),\ g(d))}}{g(c) \not\approx y}}{c \approx d}}{g(d) \not\approx y}}{\dfrac{\bot \cdot (g(d) = y)}{}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Q(y, y)}{\neg Q(g(c),\ g(d))}}{g(d) \not\approx y}}{\bot \cdot (g(d) = y)}}{c \not\approx c}}{\bot}$$

Although the approach seems to work, an unrestricted procedure will be no better than the use of functional reflexivity. Indeed, if we postpone any unification, we can apply any equality to any nonvariable term. Can we refine the method? Would it be complete? In what follows, we give positive answers to these questions.

This paper is a refined version of the work in Ref. [17]. The introduced calculi have been given a simpler formulation, and some shortcomings in explanation and proofs have been eliminated. The text is organized as follows. The next section contains preliminary material. In Section 3 we explain the method of constrained equality elimination [2] in a form adapted for the completeness proof in the next section. A refined version of connection tableaux with lazy paramodulation is introduced and its completeness proved in Section 4. We conclude with a brief summary and plans for future work.

2 Preliminaries

We work in first-order logic with equality in clausal form. A *clause* is a disjunction of literals; a *literal* is either an atomic formula or the negation of an atomic formula. We consider clauses as unordered multisets.

The equality predicate is denoted by the symbol $\approx$. We abbreviate the negation $\neg(s \approx t)$ as $s \not\approx t$. Negated equalities will be called *disequalities* to be distinguished from inequalities used in constraints (see below). We consider equalities as unordered pairs of terms; that is, $a \approx b$ and $b \approx a$ stand for the same formula.

The symbol $\simeq$ will denote "pseudoequality," a binary predicate without any specific semantics. We use it to replace the symbol $\approx$ when we pass to logic without equality. The order of arguments becomes significant here: $a \simeq b$ and $b \simeq a$ denote different formulas. The expression $s \not\simeq t$ stands for $\neg(s \simeq t)$.

We denote nonvariable terms by the letters p and q and arbitrary terms by l, r, s, and t. Variables are denoted by u, v, w, x, y, and z. Letters with arrows ($\vec{s}$, $\vec{x}$, etc.) stand for sequences of terms or variables. Substitutions are denoted by σ and τ. The result of applying a substitution σ to an expression (term, term sequence, literal, or clause) E is denoted by $E\sigma$. We write $E[s]$ to indicate that s occurs in E (including nonproper occurrence, when E is s), and we write $E[t]$ to denote the expression obtained from E by replacing one occurrence of s by t.

We use *constraints* as defined in [2]. A *constraint* is a, possibly empty, conjunction of *atomic constraints* $s = t$ or $s \succ t$ or $s \succeq t$. The letters γ and δ are used to denote constraints; the symbol $\top$ denotes the empty conjunction. A compound constraint $(a = b \wedge b \succ c)$ can be written in an abbreviated form $(a = b \succ c)$. An equality constraint $(\vec{s} = \vec{t})$ stands for $(s_1 = t_1 \wedge \cdots \wedge s_n = t_n)$.

A substitution σ *solves* an atomic constraint $s = t$ if the terms $s\sigma$ and $t\sigma$ are syntactically identical. It is a solution of an atomic constraint $s \succ t$ ($s \succeq t$) if $s\sigma > t\sigma$ ($s\sigma \geqslant t\sigma$, respectively) with respect to some reduction ordering $>$ that is total on ground terms. We say that σ is a solution of a general constraint γ if it solves all atomic constraints in γ; γ is called *satisfiable* whenever it has a solution.

A *constrained clause tableau* is a finite tree $\mathbb{T}$. The root node of $\mathbb{T}$ contains the initial set of clauses to be refuted. The nonroot nodes are pairs $L \cdot \gamma$ where L is a literal and γ is a constraint. When we say that a variable (term, literal) occurs in a tableau, we usually mean occurrence in a nonroot node.

Any branch that contains the literal $\bot$ (denoting the propositional *falsum*) is *closed*. A tableau is *closed*, whenever every branch in it is closed and the overall set of constraints in it is satisfiable. A closed tableau is said to *refute* the clause set in its root node.

An inference starts from the single root node. Each inference step grows some branch in the tableau by adding new leaves under the leaf of the branch in question. Symbolically, we describe an inference rule as follows:

$$\frac{\mathcal{S} \parallel \Gamma}{L_1 \cdot \gamma_1 \quad \cdots \quad L_n \cdot \gamma_n},$$

where $\mathcal{S}$ is the initial set of clauses (the root node), Γ is the branch being augmented (with constraints not mentioned), and $(L_1 \cdot \gamma_1), \ldots, (L_n \cdot \gamma_n)$ are the added nodes (empty constraints will be omitted). Whenever we choose some clause C in $\mathcal{S}$ to participate in the inference, we implicitly rename all the variables in C to some fresh variables.

To illustrate the proposed notation, we present the classical connection tableau calculus (denoted by **CT**) in Fig. 1.

Any tree built by the rules of a tableau calculus can be considered as a tree of inference steps: each nonleaf node in the tableau is mapped to an inference step that grows the branch under that node. We say that *an inference step I' follows an inference step I* in a given tableau whenever I' is the next step to I in some branch of the corresponding inference tree. For example, an inference tree in **CT** always starts with an expansion step; also, that first expansion step can be followed only by another expansion, since a connection step requires at least two literals in a branch.

Let $\mathbb{T}$ be a closed tableau built in **CT** or any other connection tableau calculus introduced in this paper. We say that $\mathbb{T}$ is *strongly connected* whenever every strong connection step in $\mathbb{T}$ follows an expansion step and every expansion step in $\mathbb{T}$ except for the first one is followed by exactly one strong connection step. Since the letter Δ in the rules of weak connection may stand for the empty sequence, the second condition does not forbid two or more literals in the added clause to connect with the parent node. The proviso "exactly" becomes significant as soon as we introduce strong connection rules that are not particular cases of the weak connection rules. The first condition becomes significant as soon as there are connection rules that do not close the branch.

In what follows, we call *refutations* only those closed tableaux that are strongly connected, if not explicitly stated otherwise.

Given an expansion step in a refutation tableau, the added clause will be called the *expansion clause* of that step. In an expansion clause, the literal that is connected at

Fig. 1 Connection tableaux **CT**

Expansion:

$$\frac{\mathcal{S},\ (L_1 \vee \cdots \vee L_k) \parallel \Gamma}{L_1 \quad \cdots \quad L_k}$$

Strong connection:

$$\frac{\mathcal{S} \parallel \Gamma, \neg P(\vec{r}), P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})} \qquad \frac{\mathcal{S} \parallel \Gamma, P(\vec{r}), \neg P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})}$$

Weak connection:

$$\frac{\mathcal{S} \parallel \Gamma, \neg P(\vec{r}), \Delta, P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})} \qquad \frac{\mathcal{S} \parallel \Gamma, P(\vec{r}), \Delta, \neg P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})}$$

the following strong connection step (as required by the definition of connectedness) will be called the *active literal* of that expansion clause or, equivalently, of that strong connection step. In the premise of a strong connection rule, the active literal is at the end of the branch.

The **CT** calculus is sound and complete in first-order logic without equality [9].

Theorem 1 *An equality-free set of clauses $\mathcal{S}$ is unsatisfiable if and only if there exists a refutation of $\mathcal{S}$ in* **CT**. *Moreover, if $\mathcal{S}$ is unsatisfiable but any proper subset of $\mathcal{S}$ is consistent, then for any $C \in \mathcal{S}$, there is a* **CT***-refutation of $\mathcal{S}$ that starts with* (*an expansion by*) *C.*

3 Constrained Equality Elimination

Constrained equality elimination (CEE) was proposed by Bachmair et al. in [2]. This is a variation of Brand's modification method improved by the use of ordering constraints.

Here, we describe CEE-transformation in a slightly modified form as compared with the original explanation in [2]. First, we allow nonequality predicate symbols. Second, we apply the rules of symmetry elimination before elimination of monotonicity and transitivity. Third, we require any two different occurrences of a nonvariable term to be abstracted separately, introducing two fresh variables. Fourth, we work with traditional clauses and incorporate the ordering constraints into inference rules. Fifth, we do not remove variable disequalities $x \not\approx y$ from the clauses but instead "neutralize" them with an appropriate equality constraint.

These modifications are introduced to facilitate our subsequent arguments (essentially, the undoing of the transformation) and do not affect the principal result (Theorem 2).

We split the clause transformation rules of CEE into two groups as shown in Fig. 2. Variables with a caret, called *abstraction variables*, are considered to be fresh in the corresponding clause. Recall that the letters p and q stand for nonvariable terms, whereas l, r, s, and t denote arbitrary terms.

We say that a clause is in *normal form* with respect to a group of CEE-rules if no rule in that group can be applied to the clause. Note that for each group, normal forms are unique up to renaming of abstraction variables.

Given a set of clauses $\mathcal{S}$, we denote by $\mathsf{Sym}(\mathcal{S})$ the set of all normal forms of clauses in $\mathcal{S}$ with respect to symmetry elimination. These rules replace the equality symbol $\approx$ with the nonlogical predicate symbol $\simeq$ and let us forgo using explicit axioms of symmetry for $\simeq$.

We denote by $\mathsf{SMT}(\mathcal{S})$ the set of all normal forms of clauses in $\mathsf{Sym}(\mathcal{S})$ with respect to elimination of monotonicity and transitivity. These rules flatten the terms and split equality literals, thereby making redundant explicit axioms of monotonicity and transitivity for $\simeq$.

The set $\mathsf{CEE}(\mathcal{S})$ is then defined as the union $\mathsf{SMT}(\mathcal{S}) \cup \{x \simeq x\}$. In $\mathsf{CEE}(\mathcal{S})$, resolutions correspond to paramodulations in the initial set. The introduced abstraction variables are, in some sense, "values" of the terms on the left-hand side of new disequalities. By "value" we mean the result of all paramodulations into and under the term.

Fig. 2 Constrained equality elimination

Elimination of symmetry:

$$\frac{s \approx t \ \vee \ C}{s \simeq t \ \vee \ C \qquad t \simeq s \ \vee \ C} \qquad \frac{x \not\approx y \ \vee \ C}{x \not\simeq y \ \vee \ C}$$

$$\frac{p \not\approx s \ \vee \ C}{p \not\simeq s \ \vee \ C} \qquad \frac{x \not\approx q \ \vee \ C}{q \not\simeq x \ \vee \ C}$$

Elimination of monotonicity and transitivity:

$$\frac{P(\vec{s}[p]) \ \vee \ C}{P(\vec{s}[\hat{u}]) \ \vee \ p \not\simeq \hat{u} \ \vee \ C} \qquad \frac{\neg P(\vec{s}[p]) \ \vee \ C}{\neg P(\vec{s}[\hat{u}]) \ \vee \ p \not\simeq \hat{u} \ \vee \ C}$$

$$\frac{f(\vec{s}[p]) \simeq t \ \vee \ C}{f(\vec{s}[\hat{u}]) \simeq t \ \vee \ p \not\simeq \hat{u} \ \vee \ C} \qquad \frac{f(\vec{s}[p]) \not\simeq t \ \vee \ C}{f(\vec{s}[\hat{u}]) \not\simeq t \ \vee \ p \not\simeq \hat{u} \ \vee \ C}$$

$$\frac{t \simeq f(\vec{s}[p]) \ \vee \ C}{t \simeq f(\vec{s}[\hat{u}]) \ \vee \ p \not\simeq \hat{u} \ \vee \ C} \qquad \frac{t \not\simeq f(\vec{s}[p]) \ \vee \ C}{t \not\simeq f(\vec{s}[\hat{u}]) \ \vee \ p \not\simeq \hat{u} \ \vee \ C}$$

$$\frac{t \simeq q \ \vee \ C}{t \simeq \hat{u} \ \vee \ q \not\simeq \hat{u} \ \vee \ C} \qquad \frac{p \not\simeq q \ \vee \ C}{p \not\simeq \hat{u} \ \vee \ q \not\simeq \hat{u} \ \vee \ C}$$

Next we assign an atomic constraint $p \succeq s$ to each negative literal of the form $p \not\simeq s$ that occurs in $\mathsf{CEE}(\mathcal{S})$. We assign a constraint $x = y$ to each negative literal $x \not\simeq y$ in $\mathsf{CEE}(\mathcal{S})$. We assign a constraint $s \succ t$ to each positive literal $s \simeq t$ in $\mathsf{CEE}(\mathcal{S})$, except for the reflexivity axiom $z \simeq z$, which does not acquire any constraint. A *constrained ground instance* of a clause C from $\mathsf{CEE}(\mathcal{S})$ is any ground clause $C\sigma$ such that the substitution σ is a solution of all atomic ordering constraints assigned for equalities and disequalities in C.

The following proposition is a counterpart of Theorem 4.1 from [2].

Theorem 2 *A clause set $\mathcal{S}$ is satisfiable if and only if the set of all constrained ground instances of clauses from $\mathsf{CEE}(\mathcal{S})$ is satisfiable.*

Consider the calculus $\mathbf{CT}^{\simeq}$ in Fig. 3. In essence, it is just an extension of **CT** with ordering constraints for equality literals.

Theorem 3 *A clause set $\mathcal{S}$ is unsatisfiable if and only if the set $\mathsf{SMT}(\mathcal{S})$ can be refuted in the $\mathbf{CT}^{\simeq}$ calculus.*

Proof First, we show the soundness of $\mathbf{CT}^{\simeq}$ on CEE-transformed clause sets. Consider a $\mathbf{CT}^{\simeq}$-refutation $\mathbb{T}$ of the set $\mathsf{SMT}(\mathcal{S})$ and a substitution σ that solves the overall set of constraints in $\mathbb{T}$.

Let us transform $\mathbb{T}$ into a **CT**-refutation of $\mathsf{CEE}(\mathcal{S})$. To this purpose, we erase the ordering constraints from $\mathbb{T}$, so that equality connection steps of $\mathbf{CT}^{\simeq}$ become connection steps of **CT**. Then we replace the set $\mathsf{SMT}(\mathcal{S})$ in the root node with

Fig. 3 Connection tableaux for CEE-clauses ($\mathbf{CT}^{\simeq}$)

Expansion:

$$\frac{\mathcal{S},\ (L_1 \vee \cdots \vee L_k) \parallel \Gamma}{L_1 \qquad \cdots \qquad L_k}$$

Reduction:

$$\frac{\mathcal{S} \parallel \Gamma, s \not\simeq t}{\bot \cdot (s = t)}$$

Strong connection:

$$\frac{\mathcal{S} \parallel \Gamma, \neg P(\vec{r}),\ P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})} \qquad \frac{\mathcal{S} \parallel \Gamma,\ P(\vec{r}), \neg P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})}$$

$$\frac{\mathcal{S} \parallel \Gamma, p \not\simeq t, l \simeq r}{\bot \cdot (p = l \succ r = t)} \qquad \frac{\mathcal{S} \parallel \Gamma, l \simeq r, p \not\simeq t}{\bot \cdot (p = l \succ r = t)}$$

Weak connection:

$$\frac{\mathcal{S} \parallel \Gamma, \neg P(\vec{r}), \Delta,\ P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})} \qquad \frac{\mathcal{S} \parallel \Gamma,\ P(\vec{r}), \Delta, \neg P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})}$$

$$\frac{\mathcal{S} \parallel \Gamma, p \not\simeq t, \Delta, l \simeq r}{\bot \cdot (p = l \succ r = t)} \qquad \frac{\mathcal{S} \parallel \Gamma, l \simeq r, \Delta,\ p \not\simeq t}{\bot \cdot (p = l \succ r = t)}$$

$\mathsf{CEE}(\mathcal{S})$ (that is, we add the reflexivity axiom) and rewrite every reduction step as an expansion followed by a strong connection step.

$$\frac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, s \not\simeq t}{\bot \cdot (s = t)} \quad \Longrightarrow \quad \frac{\dfrac{\mathsf{CEE}(\mathcal{S}) \parallel \Gamma, s \not\simeq t}{z \simeq z}}{\bot \cdot (z = s \wedge z = t)}$$

On each such transformation, we add the substitution $[s\sigma/z]$ to σ.

In the resulting **CT**-tree $\mathbb{T}'$, each expansion except for the first one is followed by a strong connection step. Furthermore, the resulting substitution σ' solves the overall set of constraints in $\mathbb{T}'$. Therefore, we have a **CT**-refutation of $\mathsf{CEE}(\mathcal{S})$. Let us extend σ' to a ground substitution and apply it to $\mathbb{T}'$. The resulting tree is a **CT**-refutation of a certain set S of ground instances of clauses from $\mathsf{CEE}(\mathcal{S})$. By the soundness of **CT**, the set S is unsatisfiable.

The clauses in S are valid constrained ground instances of clauses from $\mathsf{CEE}(\mathcal{S})$. Indeed, $\mathbb{T}$ is a strongly connected tableau. Therefore, every literal in $\mathbb{T}$ is either reduced or connected with some complement literal in its branch. Each positive equality literal ($l \simeq r$) acquires the strict inequality constraint ($l \succ r$) by an equality connection step. Each disequality ($s \not\simeq t$) in $\mathbb{T}$ is either reduced or connected with a positive equality literal. In both cases, the constraint ($s \succeq t$) will be satisfied by σ. A disequality ($x \not\simeq y$) has to be reduced by reflexivity, so that the constraint ($x = y$) is satisfied by σ, too. Thus, by Theorem 2, the set $\mathcal{S}$ is unsatisfiable.

Now, let us prove the completeness of $\mathbf{CT}^{\simeq}$ on CEE-transformed clause sets. Let $\mathcal{S}$ be an unsatisfiable clause set and S be the set of all constrained ground instances of clauses from $\mathsf{CEE}(\mathcal{S})$. By Theorem 2, S is also unsatisfiable. By Theorem 1, we can build a **CT**-refutation $\mathbb{T}$ of S that does not start with an instance of the reflexivity axiom. Since we do not connect subterms in the **CT** calculus, we can lift $\mathbb{T}$ to a

refutation $\mathbb{T}'$ of $\mathsf{CEE}(\mathcal{S})$ by changing the root node, replacing the constrained ground clause instances with the original clauses from $\mathcal{S}$, and lifting the constraints. Let σ be the substitution that instantiates $\mathbb{T}'$ to $\mathbb{T}$ (thereby satisfying the overall set of constraints in $\mathbb{T}'$).

In the tree $\mathbb{T}'$, any variable disequality $(x \not\simeq y)$ can be expanded (and then strongly connected) only with the reflexivity axiom. Indeed, $x\sigma = y\sigma$ by the definition of a constrained ground instance, and a positive equality literal of the form $(s \simeq s)$ may occur in S only as an instance of the reflexivity axiom (otherwise, it would dissatisfy some strict inequality constraint). Also, there are no disequalities of the form $(x \not\simeq q)$ in $\mathbb{T}'$, by the definition of $\mathsf{Sym}(\mathcal{S})$.

Therefore we transform $\mathbb{T}'$ into a $\mathbf{CT}^{\simeq}$-refutation of $\mathsf{SMT}(\mathcal{S})$. First, we remove the reflexivity axiom $(z \simeq z)$ from the root node of $\mathbb{T}'$ and convert every expansion with this clause to a $\mathbf{CT}^{\simeq}$-reduction (recall that every such expansion in the refutation tableau $\mathbb{T}'$ is necessarily followed by a strong connection step).

$$\dfrac{\dfrac{\mathsf{CEE}(\mathcal{S}) \parallel \Gamma, s \not\simeq t}{z \simeq z}}{\bot \cdot (z = s \wedge z = t)} \quad \Longrightarrow \quad \dfrac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, s \not\simeq t}{\bot \cdot (s = t)}$$

Second, we convert the connection steps.

$$\dfrac{\mathsf{CEE}(\mathcal{S}) \parallel \Gamma, p \not\simeq t, l \simeq r}{\bot \cdot (l = p \wedge r = t)} \quad \Longrightarrow \quad \dfrac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, p \not\simeq t, l \simeq r}{\bot \cdot (p = l \succ r = t)}$$

$$\dfrac{\mathsf{CEE}(\mathcal{S}) \parallel \Gamma, l \simeq r, p \not\simeq t}{\bot \cdot (p = l \wedge t = r)} \quad \Longrightarrow \quad \dfrac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, l \simeq r, p \not\simeq t}{\bot \cdot (p = l \succ r = t)}$$

$$\dfrac{\mathsf{CEE}(\mathcal{S}) \parallel \Gamma, p \not\simeq t, \Delta, l \simeq r}{\bot \cdot (l = p \wedge r = t)} \quad \Longrightarrow \quad \dfrac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, p \not\simeq t, \Delta, l \simeq r}{\bot \cdot (p = l \succ r = t)}$$

$$\dfrac{\mathsf{CEE}(\mathcal{S}) \parallel \Gamma, l \simeq r, \Delta, p \not\simeq t}{\bot \cdot (p = l \wedge r = t)} \quad \Longrightarrow \quad \dfrac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, l \simeq r, \Delta, p \not\simeq t}{\bot \cdot (p = l \succ t = r)}$$

One can easily see that the substitution σ satisfies the newly introduced ordering constraints. Indeed, all the occurrences of the reflexivity axiom have been erased from $\mathbb{T}'$, and every other positive equality literal is provided with the same strict inequality constraint by the definition of a constrained ground instance. Thus, we obtain a well-formed refutation of $\mathsf{SMT}(\mathcal{S})$ in $\mathbf{CT}^{\simeq}$. □

We denote by $\overline{\mathsf{SMT}}(\mathcal{S})$ the closure of $\mathsf{Sym}(\mathcal{S})$ under monotonicity and transitivity elimination. The following lemmata provide several simple properties of this set.

Lemma 1 *For any clause set $\mathcal{S}$, $\mathsf{SMT}(\mathcal{S})$ is a subset of $\overline{\mathsf{SMT}}(\mathcal{S})$.*

Lemma 2 *Any clause in $\overline{\mathsf{SMT}}(\mathcal{S})$ without occurrences of abstraction variables belongs to $\mathsf{Sym}(\mathcal{S})$.*

Lemma 3 *Any clause in* $\overline{\mathsf{SMT}}(\mathcal{S})$ *with occurrences of abstraction variables is of the form* $L[\hat{u}] \vee p \not\simeq \hat{u} \vee C$, *where* (a) *the abstraction variable* $\hat{u}$ *does not occur in* p *and* C, (b) $\hat{u}$ *occurs exactly once in* $L[\hat{u}]$, (c) $L[\hat{u}]$ *is neither* $\hat{u} \simeq s$ *nor* $\hat{u} \not\simeq s$, *and* (d) *the clause* $L[p] \vee C$ *belongs to* $\overline{\mathsf{SMT}}(\mathcal{S})$.

Proof Every step of monotonicity and transitivity elimination adds an abstraction variable to a clause in such a way that the listed properties are satisfied. Note that the variable $\hat{u}$ does not occur in the "unflattened" clause $L[p] \vee C$. □

4 Connection Tableaux with Lazy Paramodulation

In this section we present a refined version of the calculus sketched in the introduction. The inference rules of the **LPCT** calculus are given in Fig. 4. The variables with a bar are considered to be fresh in the whole tableau.

The proposed calculus contains several improvements in comparison with what was sketched at the beginning of the paper. First, we use lazy inference only in the strong connection rules; weak connection does not postpone unification. Second, the "laziness" itself is more restricted now: any two nonvariable terms whose unification is postponed should have the same functional symbol at the top. Third, we use ordering constraints. Fourth, we use basic paramodulation.

We note that there are two forms of the basicness restriction. The first one forbids paramodulation into terms introduced by instantiation. The corresponding

Fig. 4 Connection tableaux with lazy paramodulation **LPCT**

Expansion:

$$\frac{\mathcal{S},\ (L_1 \vee \cdots \vee L_k) \parallel \Gamma}{L_1 \qquad \cdots \qquad L_k}$$

Reduction:

$$\frac{\mathcal{S} \parallel \Gamma, s \not\approx t}{\bot \cdot (s = t)}$$

Strong connection:

$$\frac{\mathcal{S} \parallel \Gamma, \neg P(\vec{r}), P(\vec{s})}{\bot \cdot (\vec{v} = \vec{r}) \qquad s_1 \not\approx \bar{v}_1 \qquad \cdots \qquad s_n \not\approx \bar{v}_n}$$

$$\frac{\mathcal{S} \parallel \Gamma, P(\vec{r}), \neg P(\vec{s})}{\bot \cdot (\vec{v} = \vec{r}) \qquad s_1 \not\approx \bar{v}_1 \qquad \cdots \qquad s_n \not\approx \bar{v}_n}$$

$$\frac{\mathcal{S} \parallel \Gamma, L[p], z \approx r}{L[\bar{w}] \cdot (p = z \succ \bar{w}) \qquad r \not\approx \bar{w}}$$

$$\frac{\mathcal{S} \parallel \Gamma, L[p], f(\vec{s}) \approx r}{L[\bar{w}] \cdot (p = f(\vec{\bar{v}}) \succ \bar{w}) \qquad r \not\approx \bar{w} \qquad s_1 \not\approx \bar{v}_1 \qquad \cdots \qquad s_n \not\approx \bar{v}_n}$$

$$\frac{\mathcal{S} \parallel \Gamma, l \approx r, L[f(\vec{s})]}{L[\bar{w}] \cdot (f(\vec{\bar{v}}) = l \succ r = \bar{w}) \qquad s_1 \not\approx \bar{v}_1 \qquad \cdots \qquad s_n \not\approx \bar{v}_n}$$

Weak connection:

$$\frac{\mathcal{S} \parallel \Gamma, \neg P(\vec{r}), \Delta, P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})} \qquad \frac{\mathcal{S} \parallel \Gamma, P(\vec{r}), \Delta, \neg P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})}$$

$$\frac{\mathcal{S} \parallel \Gamma, L[p], \Delta, l \approx r}{L[\bar{w}] \cdot (p = l \succ r = \bar{w})} \qquad \frac{\mathcal{S} \parallel \Gamma, l \approx r, \Delta, L[p]}{L[\bar{w}] \cdot (p = l \succ r = \bar{w})}$$

refinement of lazy paramodulation was described by Moser [12]. This restriction is fully adopted in **LPCT**, since we work with constrained literals and do not apply substitutions in the course of inference.

The second and stronger form additionally prevents paramodulation into terms introduced by the earlier paramodulation steps [1]. In this form, basicness is used in **LPCT**, too (note the variables with a bar), although not everywhere: when a paramodulating equality is the active literal in a strong connection, the inserted term is left "on the surface," allowed for subsequent paramodulations.

Let us illustrate the rules of **LPCT** in the example discussed in the introduction. To fit the tableau onto a page, we will "halve" the problem (the other part is treated in the same way) and consider the clause set $\mathcal{S} = \{a \approx b,\ \neg P(f(a),\ f(b)),\ P(x, x)\}$ and the ordering $f \succ a \succ b$. The following tableau is a well-formed refutation of $\mathcal{S}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{S}}{P(x,x)}}{\neg P(f(a),\ f(b))}}{\bot \cdot (\bar{v}_1 = x \wedge \bar{v}_2 = x)} \qquad \dfrac{\dfrac{\dfrac{f(a) \not\approx \bar{v}_1}{a \approx b}}{\dfrac{f(\bar{w}) \not\approx \bar{v}_1 \cdot (a = a \succ \bar{w})}{\bot \cdot (f(\bar{w}) = \bar{v}_1)}} \qquad \dfrac{b \not\approx \bar{w}}{\bot \cdot (b = \bar{w})}}{} \qquad \dfrac{f(b) \not\approx \bar{v}_2}{\bot \cdot (f(b) = \bar{v}_2)}$$

The soundness of **LPCT** can be shown directly, by checking that inference rules generate only what follows from the initial clause set and the current branch.

Theorem 4 *For any unsatisfiable clause set $\mathcal{S}$ there exists a refutation of $\mathcal{S}$ in* **LPCT**.

Proof We prove the completeness of the **LPCT** calculus by transforming a $\mathbf{CT}^{\simeq}$-refutation of the set of CEE-rewritten clauses into an **LPCT**-refutation of the initial clause set. To this purpose, we introduce an intermediate calculus $\mathbf{LPCT}^{\simeq}$, whose inference rules are those of **LPCT** with the equality symbol $\approx$ replaced with $\simeq$.

At the first stage we build a $\mathbf{CT}^{\simeq}$-refutation of the set $\mathsf{SMT}(\mathcal{S})$. Such a refutation exists by Theorem 3. Then we extend the root node of the tableau to $\overline{\mathsf{SMT}}(\mathcal{S})$ (by Lemma 1) and transform the tree into an $\mathbf{LPCT}^{\simeq}$-refutation $\mathbb{T}$ of $\mathsf{SMT}(\mathcal{S})$. In Fig. 5, we show how the connection rules of $\mathbf{CT}^{\simeq}$ can be simulated in $\mathbf{LPCT}^{\simeq}$ so that generated constraints stay essentially the same.

At the second stage we unflatten the clauses. Let us denote the tableau $\mathbb{T}$ as $\mathbb{T}^{(0)}$. We are going to construct a sequence of well-formed $\mathbf{LPCT}^{\simeq}$-refutations of the set $\overline{\mathsf{SMT}}(\mathcal{S})$ such that for every $i > 0$, there are fewer different abstraction variables in $\mathbb{T}^{(i)}$ than in $\mathbb{T}^{(i-1)}$. Once we obtain a refutation tableau where abstraction variables do not occur, we can replace $\overline{\mathsf{SMT}}(\mathcal{S})$ in the root node with $\mathsf{Sym}(\mathcal{S})$ (by Lemma 2) and proceed to the third stage.

Consider some lowermost expansion clause from $\overline{\mathsf{SMT}}(\mathcal{S})\backslash\mathsf{Sym}(\mathcal{S})$ in $\mathbb{T}^{(i-1)}$. By "lowermost" we mean that there are no expansion clauses with abstraction variables under this clause. According to Lemma 3, the clause is of the form $L[\hat{u}] \vee p \not\simeq \hat{u} \vee C$

Strong connection:

$$\frac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, \neg P(\vec{r}),\ P(\vec{s})}{\bot \cdot (\vec{r} = \vec{s})} \quad \Longrightarrow \quad \frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma, \neg P(\vec{r}),\ P(\vec{s})}{\bot \cdot (\vec{v} = \vec{r}) \qquad \frac{s_1 \not\simeq \bar{v}_1}{\bot \cdot (s_1 = \bar{v}_1)} \qquad \cdots \qquad \frac{s_n \not\simeq \bar{v}_n}{\bot \cdot (s_n = \bar{v}_n)}}$$

Strong equality connection:

$$\frac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma,\ p \not\simeq t, z \simeq r}{\bot \cdot (p = z \succ r = t)} \quad \Longrightarrow \quad \frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ p \not\simeq t, z \simeq r}{\frac{\bar{w} \not\simeq t \cdot (p = z \succ \bar{w})}{\bot \cdot (\bar{w} = t)} \qquad \frac{r \not\simeq \bar{w}}{\bot \cdot (r = \bar{w})}}$$

$$\frac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma,\ p \not\simeq t,\ f(\vec{s}) \simeq r}{\bot \cdot (p = f(\vec{s}) \succ r = t)} \quad \Longrightarrow \quad \frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ p \not\simeq t,\ f(\vec{s}) \simeq r}{\frac{\bar{w} \not\simeq t \cdot (p = f(\vec{v}) \succ \bar{w})}{\bot \cdot (\bar{w} = t)} \qquad \frac{r \not\simeq \bar{w}}{\bot \cdot (r = \bar{w})} \qquad \frac{s_1 \not\simeq \bar{v}_1}{\bot \cdot (s_1 = \bar{v}_1)} \qquad \cdots \qquad \frac{s_n \not\simeq \bar{v}_n}{\bot \cdot (s_n = \bar{v}_n)}}$$

$$\frac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma, l \simeq r,\ f(\vec{s}) \not\simeq t}{\bot \cdot (f(\vec{s}) = l \succ r = t)} \quad \Longrightarrow \quad \frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma, l \simeq r,\ f(\vec{s}) \not\simeq t}{\frac{\bar{w} \not\simeq t \cdot (f(\vec{v}) = l \succ r = \bar{w})}{\bot \cdot (\bar{w} = t)} \qquad \frac{s_1 \not\simeq \bar{v}_1}{\bot \cdot (s_1 = \bar{v}_1)} \qquad \cdots \qquad \frac{s_n \not\simeq \bar{v}_n}{\bot \cdot (s_n = \bar{v}_n)}}$$

Weak equality connection:

$$\frac{\mathsf{SMT}(\mathcal{S}) \parallel \Gamma,\ p \not\simeq t, \Delta, l \simeq r}{\bot \cdot (p = l \succ r = t)} \quad \Longrightarrow \quad \frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ p \not\simeq t, \Delta, l \simeq r}{\frac{\bar{w} \not\simeq t \cdot (p = l \succ r = \bar{w})}{\bot \cdot (\bar{w} = t)}}$$

Fig. 5 Transforming $\mathbf{CT}^{\simeq}$ to $\mathbf{LPCT}^{\simeq}$

such that $\hat{u}$ does not occur in p and C, $\hat{u}$ occurs exactly once in $L[\hat{u}]$, $L[\hat{u}]$ is not of the form $\hat{u} \simeq s$, and the clause $L[p] \vee C$ belongs to $\overline{\mathsf{SMT}}(\mathcal{S})$.

We are going to replace the literals $L[\hat{u}]$ and $p \not\simeq \hat{u}$ with $L[p]$ and combine two corresponding subtrees of $\mathbb{T}^{(i-1)}$ so that all the paramodulations in and under the term p are made first and the connections to L follow them.

Let $\mathbb{T}^{\bullet}$ be the subtree of $\mathbb{T}^{(i-1)}$ that grows from the literal $p \not\simeq \hat{u}$. We can affirm that $\hat{u}$ occurs in $\mathbb{T}^{\bullet}$ only in disequalities of the form $t \not\simeq \hat{u}$ and in constraints $(t = \hat{u})$

introduced by a reduction step; moreover, $\hat{u}$ does not occur in these terms t. Indeed, all we can do in **LPCT**$^{\simeq}$ with a literal $t \not\simeq \hat{u}$ is to paramodulate in t or reduce the branch.

The choice of the second subtree, denoted $\mathbb{T}^{\circ}$, depends on whether $L[\hat{u}]$ is the active literal in the expansion clause in question. If $L[\hat{u}]$ is not the active literal, then $\mathbb{T}^{\circ}$ will be the subtree that grows from $L[\hat{u}]$. Otherwise, consider seven possible strong connection steps in Fig. 6. In each case, a strong connection rule generates exactly one node where $\hat{u}$ occurs. Further, in that node, $\hat{u}$ occurs exactly once in the literal and does not occur in the constraint (if any). Then we denote by $\mathbb{T}^{\circ}$ the subtree growing from that node.

Let $M[\hat{u}] \cdot \gamma$ be the root node of $\mathbb{T}^{\circ}$. For an arbitrary term t, we denote by $\mathbb{T}^{\circ}[t]$ the tree $\mathbb{T}^{\circ}$ where every occurrence of $\hat{u}$ (both in literals and constraints) is replaced with t. For an arbitrary constraint δ, we denote by $\mathbb{T}^{\circ}[t] \cdot \delta$ the tree $\mathbb{T}^{\circ}[t]$ where the root node constraint is replaced with δ.

We construct the combined tree $[\mathbb{T}^{\bullet}]^{\mathbb{T}^{\circ}}$ from $\mathbb{T}^{\bullet}$ as follows:

- Each nonreduced literal of the form $t \not\simeq \hat{u}$ is replaced with $M[t]$;
- Each branch end of the form $\dfrac{t \not\simeq \hat{u} \cdot \delta}{\bot \cdot (t = \hat{u})}$ is replaced with $\mathbb{T}^{\circ}[t] \cdot \delta$.

Then we transform $\mathbb{T}^{(i-1)}$ as follows:

(a) The expansion clause $L[\hat{u}] \vee p \not\simeq \hat{u} \vee C$ is replaced with $L[p] \vee C$;
(b) The subtree $\mathbb{T}^{\bullet}$ is removed from the tableau;
(c) The subtree $\mathbb{T}^{\circ}$ is replaced with $[\mathbb{T}^{\bullet}]^{\mathbb{T}^{\circ}} \cdot \gamma$.

An example of this transformation is shown in Fig. 7. Note how occurrences of the abstraction variable $\hat{u}$ in $\mathbb{T}^{\bullet}$ change in $[\mathbb{T}^{\bullet}]^{\mathbb{T}^{\circ}}$.

Fig. 6 Abstraction variable occurs in an active literal

$$\frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ \neg P(\vec{r}),\ P(\vec{s}[\hat{u}])}{\bot \cdot (\vec{v} = \vec{r}) \qquad s_1 \not\simeq \bar{v}_1 \quad \cdots \quad s_i[\hat{u}] \not\simeq \bar{v}_i \quad \cdots \quad s_n \not\simeq \bar{v}_n}$$

$$\frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ P(\vec{r}),\ \neg P(\vec{s}[\hat{u}])}{\bot \cdot (\vec{v} = \vec{r}) \qquad s_1 \not\simeq \bar{v}_1 \quad \cdots \quad s_i[\hat{u}] \not\simeq \bar{v}_i \quad \cdots \quad s_n \not\simeq \bar{v}_n}$$

$$\frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ M[q],\ z \simeq r[\hat{u}]}{M[\bar{w}] \cdot (q = z \succ \bar{w}) \qquad r[\hat{u}] \not\simeq \bar{w}}$$

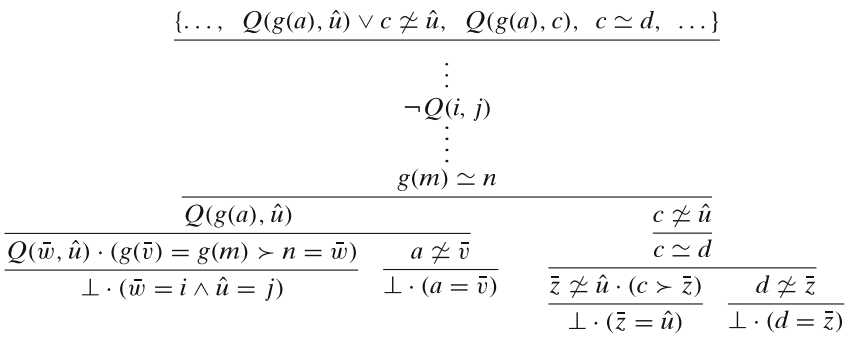
$$\frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ M[q],\ f(\vec{s}[\hat{u}]) \simeq r}{M[\bar{w}] \cdot (q = f(\vec{\bar{v}}) \succ \bar{w}) \quad r \not\simeq \bar{w} \quad s_1 \not\simeq \bar{v}_1 \quad \cdots \quad s_i[\hat{u}] \not\simeq \bar{v}_i \quad \cdots \quad s_n \not\simeq \bar{v}_n}$$

$$\frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ M[q],\ f(\vec{s}) \simeq r[\hat{u}]}{M[\bar{w}] \cdot (q = f(\vec{\bar{v}}) \succ \bar{w}) \qquad r[\hat{u}] \not\simeq \bar{w} \qquad s_1 \not\simeq \bar{v}_1 \quad \cdots \quad s_n \not\simeq \bar{v}_n}$$

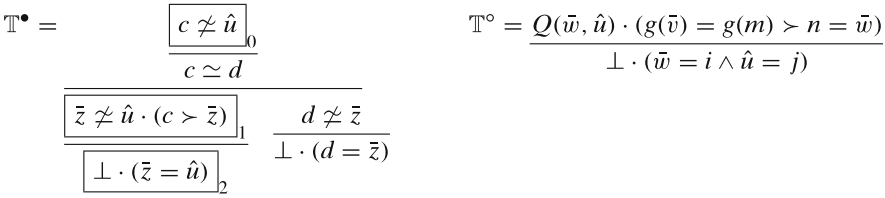
$$\frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ l \simeq r,\ L[f(\vec{s}[\hat{u}])]}{L[\bar{w}] \cdot (f(\vec{\bar{v}}) = l \succ r = \bar{w}) \qquad s_1 \not\simeq \bar{v}_1 \quad \cdots \quad s_i[\hat{u}] \not\simeq \bar{v}_i \quad \cdots \quad s_n \not\simeq \bar{v}_n}$$

$$\frac{\overline{\mathsf{SMT}}(\mathcal{S}) \parallel \Gamma,\ l \simeq r,\ L[f(\vec{s}),\ \hat{u}]}{L[\bar{w}, \hat{u}] \cdot (f(\vec{\bar{v}}) = l \succ r = \bar{w}) \qquad s_1 \not\simeq \bar{v}_1 \quad \cdots \quad s_n \not\simeq \bar{v}_n}$$

Before elimination:



Subtrees:



After elimination:

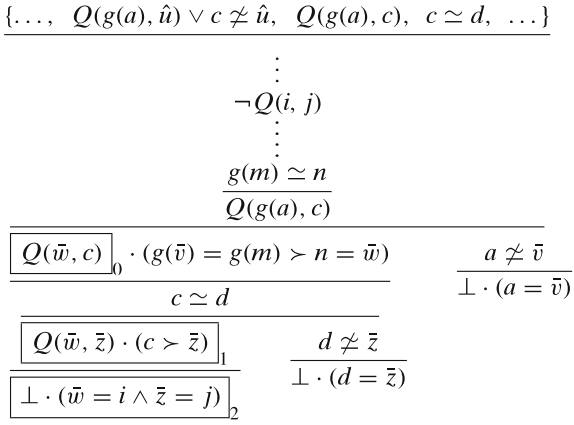


Fig. 7 Elimination of an abstraction variable

Let $\mathbb{T}^{(i)}$ be the resulting tableau. Let us demonstrate that this tree satisfies all the necessary conditions:

1. $\mathbb{T}^{(i)}$ *is a well-formed* **LPCT**$^{\simeq}$*-tableau.* First of all, note that the root literal of $[\mathbb{T}^{\bullet}]^{\mathbb{T}^{\circ}}$ is $M[p]$. If $L[\hat{u}]$ was the active literal in the original expansion clause, then $M[p] \cdot \gamma$ is exactly the node that is generated by a strong connection rule when $L[p]$ replaces $L[\hat{u}]$. (If $L[\hat{u}]$ was not active, then M and L are the same literal). Furthermore, the only possible connection with a literal of the form $t \not\simeq \hat{u}$ in $\mathbb{T}^{\bullet}$ is paramodulation in the term t. Then the same connection steps can be made with

the corresponding literals $M[t]$ in $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ (also in case where $p \not\simeq \hat{u}$ is the active literal in the original expansion clause).
Finally, each literal $t \not\simeq \hat{u}$ reduced in $\mathbb{T}^\bullet$ appears as $M[t]$ in $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ and grows further as the subtree $\mathbb{T}^\circ[t]$. One can easily see that this substitution does not make $\mathbb{T}^\circ[t]$ ill-formed. Indeed, the only inference rule in **LPCT**$^\simeq$ that requires a variable in a premise is the third rule of strong connection, where the active literal is of the form $z \simeq r$. However, an abstraction variable never occurs at the left-hand side of an equality in a clause from $\overline{\mathsf{SMT}}(\mathcal{S})$.

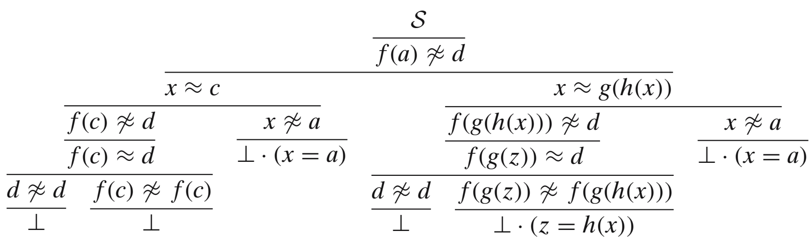
2. $\mathbb{T}^{(i)}$ *is a closed tableau.* Obviously, every branch in $\mathbb{T}^{(i)}$ is closed. Furthermore, for each literal $t \not\simeq \hat{u}$ reduced in $\mathbb{T}^\bullet$, $\hat{u}$ and t are equal with respect to the constraints in $\mathbb{T}^{(i-1)}$. Hence the constraints in the new subtree $\mathbb{T}^\circ[t]$ can be solved with the same substitution.
3. $\mathbb{T}^{(i)}$ *is strongly connected.* If $L[\hat{u}]$ is the active literal in the original expansion clause, then $L[p]$ becomes the active literal in the new tree. That is the reason why we choose $\mathbb{T}^\circ$ growing from a child node and not from $L[\hat{u}]$ itself. Here is also where laziness of strong connection steps is crucial for our proof. If the abstraction variable could hide into a constraint, then the only way to eliminate it would be replacing the subtree growing from $L[\hat{u}]$, thereby giving up the connectedness of the tree.
If $p \not\simeq \hat{u}$ is the active literal in the original expansion clause, the subtree growing from $L[p]$ in $\mathbb{T}^{(i)}$ (namely, $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$) follows the structure of $\mathbb{T}^\bullet$. Thus, $L[p]$ becomes (again) the active literal in the new tree.
If the strong connection has been made elsewhere in the original clause, it is unaffected by our transformation.
4. $\mathbb{T}^{(i)}$ *contains strictly less distinct abstraction variables than* $\mathbb{T}^{(i-1)}$. The variable $\hat{u}$ is eliminated from the expansion clause and cannot occur in any tree $\mathbb{T}^\circ[t]$. Every occurrence of $\hat{u}$ in $\mathbb{T}^\bullet$ is eliminated from $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ by construction. Therefore, $\hat{u}$ does not occur in $\mathbb{T}^{(i)}$.
Since no abstraction variable was introduced under the considered expansion step in $\mathbb{T}^{(i-1)}$, new abstraction variables could not appear in $\mathbb{T}^{(i)}$ because of multiple copies of $\mathbb{T}^\circ$ in $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$.

By repeating this procedure, we will eventually get a refutation tableau $\mathbb{T}^{(N)}$ where abstraction variables do not occur at all. Therefore, every expansion clause in $\mathbb{T}^{(N)}$ belongs to the set $\mathsf{Sym}(\mathcal{S})$ by Lemma 2. So we replace $\overline{\mathsf{SMT}}(\mathcal{S})$ with $\mathsf{Sym}(\mathcal{S})$ in the root node of $\mathbb{T}^{(N)}$ and obtain an **LPCT**$^\simeq$-refutation of $\mathsf{Sym}(\mathcal{S})$.

At the third stage we undo the symmetry elimination step. We replace the symbol $\simeq$ with $\approx$ and reorient equalities to their initial form in $\mathcal{S}$. Thus we obtain an **LPCT**-refutation of $\mathcal{S}$. □

Despite the way in which we prove completeness of the calculus, **LPCT** is not just a reformulation of the CEE method. In fact, there is an essential difference between flattening and lazy paramodulation. We said above that abstraction variables introduced by CEE can be considered as "values" of the terms they replace. That is, the term that is finally substituted for a variable $\hat{u}$ in fact is the result of all paramodulations made under and in the term t which was replaced with $\hat{u}$ by CEE. Therefore, in a given CEE-clause, every term has exactly one "value." It is not the case for **LPCT**.

Let $\mathcal{S}$ be $\{x \approx c \vee x \approx g(h(x)),\ f(c) \approx d,\ f(g(z)) \approx d,\ f(a) \not\approx d\}$. The following tableau built in a simplified version of **LPCT** cannot be obtained from any $\mathbf{CT}^{\simeq}$-refutation of $\mathsf{CEE}(\mathcal{S})$.



Here, we replace the constant a in the starting clause with two terms, c and $g(h(x))$. If we make inferences with CEE-clauses, we should take two different instances of the starting clause. Based on this example, one can show that **LPCT** can give an exponential shortening of the minimal inference size as compared with $\mathbf{CT}^{\simeq}$ (but at the same time the number of possible inferences increases).

Another noteworthy point is the weakness of unification. The lazy unification procedure used in **LPCT**, which matches top functional symbols immediately and postpones the rest is the one proposed for lazy paramodulation in [6]. This form of unification is much weaker than *top unification* (introduced in [5] and used in [18]) which descends down to variables. Top unification allows us to restrict dramatically the weight of postponed "unification obligations." In particular, top unifiability of two ground terms is decided immediately.

Unfortunately, top unification and ordering constraints cannot be used together in the framework of connection tableaux. Consider the set $\mathcal{S} = \{P(c) \vee Q(c),\ \neg P(a),\ \neg Q(b),\ b \approx c,\ a \approx c\}$ and the ordering $a > b > c$. Ordering constraints prohibit paramodulations into c. The only way to refute $\mathcal{S}$ in **LPCT** is to connect $P(c)$ with $\neg P(a)$, or $Q(c)$ with $\neg Q(b)$. However, these pairs are not top unifiable.

It is unclear whether ordered inferences for a stronger kind of lazy unification is a good trade-off. We are not aware of any adaptation of connection tableaux for lazy paramodulation with top unification. One of the directions for further research is to develop and study one.

5 Conclusion

We have presented a new connection tableau calculus for first-order clausal logic with equality. This calculus employs lazy paramodulation with ordering constraints and a restricted form of basicness. The refutational completeness of the calculus is demonstrated by transforming proofs given by the (almost) traditional connection tableau calculus applied to a set of flattened clauses (in the spirit of Brand's modification method). Thus a connection is established between lazy paramodulation and equality elimination via problem transformation.

We plan to investigate the compatibility of the proposed calculus with various refinements of connection tableaux; first of all, with the regularity restriction.

Unfortunately, the existing completeness proof is not well suited for this task; some semantic argument would be useful here. Another interesting topic to study is more restricted forms of laziness, probably giving up orderings and basicness.

We also hope to implement the proposed calculus and compare it in practice with other methods of equality handling in tableau calculi.

Acknowledgements We are grateful to Alexander Lyaletski and Konstantin Verchinine for their guidance and expertise.

References

1. Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W.: Basic paramodulation. Inf. Comput. **121**(2), 172–192 (1995)
2. Bachmair, L., Ganzinger, H., Voronkov, A.: Elimination of equality via transformation with ordering constraints. In: Kirchner, C., Kirchner, H. (eds.) Automated Deduction: 15th International Conference, CADE-15. Lecture Notes in Computer Science, vol. 1421, pp. 175–190 (1998)
3. Brand, D.: Proving theorems with the modification method. SIAM J. Comput. **4**, 412–430 (1975)
4. Degtyarev, A., Voronkov, A.: What you always wanted to know about rigid *E*-unification. J. Autom. Reason. **20**(1), 47–80 (1998)
5. Dougherty, D.J., Johann, P.: An improved general *E*-unification method. In: Stickel, M.E. (ed.) Automated Deduction: 10th International Conference, CADE-10. Lecture Notes in Computer Science, vol. 449, pp. 261–275 (1990)
6. Gallier, J., Snyder, W.: Complete sets of transformations for general *E*-unification. Theor. Comp. Sci. **67**, 203–260 (1989)
7. Giese, M.: A model generation style completeness proof for constraint tableaux with superposition. In: Egly, U., Fermüller, C.G. (eds.) Automated Reasoning with Analytic Tableaux and Related Methods: International Conference, TABLEAUX 2002. Lecture Notes in Computer Science, vol. 2381, pp. 130–144 (2002)
8. Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO: a high-performance theorem prover. J. Autom. Reason. **8**(2), 183–212 (1992)
9. Letz, R., Stenz, G.: Model elimination and connection tableau procedures. In: Robinson, A., Voronkov, A. (eds.) Handbook for Automated Reasoning, vol. II, Chapt. 28, pp. 2017–2116. Elsevier Science (2001)
10. Loveland, D.W.: Mechanical theorem proving by model elimination. J. ACM **16**(3), 349–363 (1968)
11. Loveland, D.W.: Automated Theorem Proving: A Logical Basis. Fundamental Studies in Computer Science, vol. 6. North-Holland (1978)
12. Moser, M.: Improving transformation systems for general *E*-unification. In: Kirchner, C. (ed.) Rewriting Techniques and Applications: 5th International Conference, RTA 1993. Lecture Notes in Computer Science, vol. 690, pp. 92–105 (1993)
13. Moser, M., Ibens, O., Letz, R., Steinbach, J., Goller, C., Schumann, J., Mayr, K.: SETHEO and E-SETHEO – the CADE-13 systems. J. Autom. Reason. **18**(2), 237–246 (1997)
14. Moser, M., Lynch, C., Steinbach, J.: Model elimination with basic ordered paramodulation. Technical Report AR-95-11, Fakultät für Informatik, Technische Universität München, München (1995)
15. Moser, M., Steinbach, J.: STE-modification revisited. Technical Report AR-97-03, Fakultät für Informatik, Technische Universität München, München (1997)
16. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook for Automated Reasoning, vol. I, Chapt. 7, pp. 371–443. Elsevier Science (2001)
17. Paskevich, A.: Connection tableaux with lazy paramodulation. In: Furbach, U., Shankar, N. (eds.) Automated Reasoning, 3rd International Joint Conference IJCAR 2006. Lecture Notes in Computer Science, vol. 4130, pp. 112–124. Seattle WA, USA (2006)
18. Snyder, W., Lynch, C.: Goal directed strategies for paramodulation. In: Book, R. (ed.) Rewriting Techniques and Applications: 4th International Conference, RTA 1991. Lecture Notes in Computer Science, vol. 488, pp. 150–161 (1991)