Connection Tableaux with Lazy Paramodulation

Andrei Paskevich

Université Paris 12, Laboratoire d'Algorithmique, Complexité et Logique,
94010 Créteil Cedex, France
andrei@capet.iut-fbleau.fr

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.

1 Introduction

The Model Elimination proof procedure was originally introduced by Loveland as a resolution-based calculus with clauses of a special form [1]. Later it was reconsidered as a clause tableau calculus, where proof search is guided by connections between clauses [2]. In this form, the method is also referred to as *connection tableaux*.

Connection tableaux are a powerful goal-directed refinement of general clause tableaux. Moreover, strong search pruning methods and efficient implementation techniques were developed for this calculus [3].

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

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

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 [4] and general clause tableaux [5, 6] is due to a flexible order of inferences which 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 [7]. However, this approach is quite inefficient in practice, since functional reflexivity allows us to substitute an arbitrary term for any variable.

In order to solve problems with equality, existing competitive connection tableau provers [8] employ various forms of Brand's modification method [9–11]. 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 upon a combination of goal-directed proof search in tableaux and a bottom-up equality saturation using basic ordered paramodulation [12].

In this paper we propose an alternative approach for equality handling in connection tableaux which is based on *lazy paramodulation*. This technique was originally introduced by J. Gallier and W. Snyder as a method for general E-unification [13] and used later to overcome incompleteness of the set-of-support strategy (another example of a goal-directed method) in the classical paramodulation calculus [14].

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 counter-example can be easily refuted in such a calculus:

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

Though 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 non-variable term. Can we refine the method? Would it be complete? In what follows, we give positive answers to these questions.

The paper is organized as follows. The next section contains preliminary material. In Section 3 we explain the method of constrained equality elimination [11] 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 is 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, i.e. $a \approx b$ 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 $\neg(s \simeq t)$.

In what follows, we denote non-variable terms by the letters p and q, and arbitrary terms with the letters l, r, s, and t. Substitutions are denoted by the letters σ and τ. The result of applying a substitution σ to an expression (term, literal, or clause) E is denoted by $E\sigma$. We write $E[s]$ to indicate that s is a subterm of E and write $E[t]$ to denote the expression obtained from E by replacing one occurrence of s by t. Letters in bold ($\boldsymbol{s}$, $\boldsymbol{x}$, etc) stand for sequences of terms and variables.

We use *constraints* as defined in [11]. 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 $(\boldsymbol{s} = \boldsymbol{t})$ 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 non-root nodes are pairs $L \cdot \gamma$ where L is a literal and γ is a constraint.

Any branch that contains the literal $\bot$ (denoting the propositional *falsum*) is considered as *closed*. A tableau is *closed*, whenever each branch in it is closed and the overall set of constraints in it is satisfiable.

An inference starts from the single root node (the initial clause set). Each inference step expands 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 expanded (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 variables in C to some fresh variables.

A closed tableau built from the initial set $\mathcal{S}$ will be called a *refutation* of $\mathcal{S}$.

In order to illustrate the proposed notation, we present the classical connection tableau calculus (denoted by **CT**) in Figure 1.

Start rule:

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

Expansion rules:

$$\frac{\mathcal{S},\ (\neg P(\boldsymbol{s}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, P(\boldsymbol{r})}{\bot \cdot (\boldsymbol{s} = \boldsymbol{r}) \qquad L_1 \qquad \cdots \qquad L_k} \qquad \frac{\mathcal{S},\ (P(\boldsymbol{s}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, \neg P(\boldsymbol{r})}{\bot \cdot (\boldsymbol{s} = \boldsymbol{r}) \qquad L_1 \qquad \cdots \qquad L_k}$$

Termination rules:

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

Fig. 1. Connection tableaux **CT**

This calculus is sound and complete in first-order logic without equality [3]:

Proposition 1. *An equality-free set of clauses $\mathcal{S}$ is unsatisfiable if and only if $\mathcal{S}$ can be expanded to a closed* **CT***-tableau. 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 C.*

3 Constrained equality elimination

Constrained equality elimination (CEE) was proposed by L. Bachmair et al. in [11]. 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 [11]. First, we allow non-equality predicate symbols. Second, we require any two different occurrences of a non-variable subterm to be abstracted separately, introducing two different fresh variables during monotonicity elimination (*flattening*). Third, we apply the monotonicity elimination rules after symmetry elimination. Fourth, we work with traditional clauses and incorporate the ordering constraints into the inference rules. Fifth, we handle occurrences of negated variable equality $x \not\approx y$ in a different way. These modifications are minor and do not affect the main result (Proposition 2).

The four groups of rewriting rules in Figure 2 are applied in the order of their appearance to clauses from the initial set $\mathcal{S}$. Let us denote the intermediate

1. Elimination of symmetry:

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

2. Elimination of monotonicity:

$$\frac{P(\boldsymbol{s}_1, p, \boldsymbol{s}_2) \ \vee \ C}{P(\boldsymbol{s}_1, \hat{u}, \boldsymbol{s}_2) \ \vee \ p \not\simeq \hat{u} \ \vee \ C} \qquad \frac{\neg P(\boldsymbol{s}_1, p, \boldsymbol{s}_2) \ \vee \ C}{\neg P(\boldsymbol{s}_1, \hat{u}, \boldsymbol{s}_2) \ \vee \ p \not\simeq \hat{u} \ \vee \ C}$$

$$\frac{f(\boldsymbol{s}_1, p, \boldsymbol{s}_2) \simeq t \ \vee \ C}{f(\boldsymbol{s}_1, \hat{u}, \boldsymbol{s}_2) \simeq t \ \vee \ p \not\simeq \hat{u} \ \vee \ C} \qquad \frac{f(\boldsymbol{s}_1, p, \boldsymbol{s}_2) \not\simeq t \ \vee \ C}{f(\boldsymbol{s}_1, \hat{u}, \boldsymbol{s}_2) \not\simeq t \ \vee \ p \not\simeq \hat{u} \ \vee \ C}$$

$$\frac{t \simeq f(\boldsymbol{s}_1, p, \boldsymbol{s}_2) \ \vee \ C}{t \simeq f(\boldsymbol{s}_1, \hat{u}, \boldsymbol{s}_2) \ \vee \ p \not\simeq \hat{u} \ \vee \ C} \qquad \frac{t \not\simeq f(\boldsymbol{s}_1, p, \boldsymbol{s}_2) \ \vee \ C}{t \not\simeq f(\boldsymbol{s}_1, \hat{u}, \boldsymbol{s}_2) \ \vee \ p \not\simeq \hat{u} \ \vee \ C}$$

3. Elimination of transitivity:

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

4. Introduction of reflexivity:

$$\frac{}{z \simeq z}$$

Fig. 2. Constrained equality elimination

result of transformation after the i-th group by $\mathrm{CEE}^i(\mathcal{S})$. Variables with caret are considered to be new in the corresponding clause. Recall that p and q stand for non-variable terms, whereas l, r, s, and t denote arbitrary terms.

The first group replaces the equality symbol $\approx$ with the non-logical predicate symbol $\simeq$ and eliminates the need for explicit symmetry axioms for $\simeq$. The second group flattens the terms, thus eliminating the need for explicit monotonicity axioms for $\simeq$. The third group splits equality literals, thus eliminating the need for explicit transitivity axioms for $\simeq$. The last rule explicitly adds the reflexivity axiom to the clause set.

In the resulting set of clauses $\mathrm{CEE}(\mathcal{S})$ $(= \mathrm{CEE}^4(\mathcal{S}))$, resolutions correspond to paramodulations in the initial set. The introduced 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.

Now, we assign an atomic constraint $p \succeq s$ to each negative literal $p \not\simeq s$ that occurs in $\mathrm{CEE}(\mathcal{S})$. We assign a constraint $x = y$ to each negative literal $x \not\simeq y$ in $\mathrm{CEE}(\mathcal{S})$. We assign a constraint $s \succ t$ to each positive literal $s \simeq t$ in $\mathrm{CEE}(\mathcal{S})$, except for the reflexivity axiom $z \simeq z$ which does not acquire any constraint. A *constrained ground instance* of a clause C from $\mathrm{CEE}(\mathcal{S})$ 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 [11].

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

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

In the Start and Expansion rules, the chosen clause is not $(z \simeq z)$

Start rule:

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

Reduction rule:

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

Expansion rules:

$$\frac{\mathcal{S},\ (\neg P(\boldsymbol{s}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, P(\boldsymbol{r})}{\bot \cdot (\boldsymbol{s} = \boldsymbol{r}) \qquad L_1 \qquad \cdots \qquad L_k}$$

$$\frac{\mathcal{S},\ (P(\boldsymbol{s}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, \neg P(\boldsymbol{r})}{\bot \cdot (\boldsymbol{s} = \boldsymbol{r}) \qquad L_1 \qquad \cdots \qquad L_k}$$

$$\frac{\mathcal{S},\ (p \not\simeq t \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, l \simeq r}{\bot \cdot (p = l \succ r = t) \qquad L_1 \qquad \cdots \qquad L_k}$$

$$\frac{\mathcal{S},\ (l \simeq r \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, p \not\simeq t}{\bot \cdot (p = l \succ r = t) \qquad L_1 \qquad \cdots \qquad L_k}$$

Termination rules:

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

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

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

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

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

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

Proof. We give just an outline of the proof, since the details are quite obvious. First, let us show the soundness of $\mathbf{CT}^{\simeq}$ with respect to CEE-transformed clause sets. Consider a closed $\mathbf{CT}^{\simeq}$-tableau $\mathbb{T}$ refuting the set $\mathrm{CEE}(\mathcal{S})$.

The substitution that solves the overall set of constraints from $\mathbb{T}$, can be completed to a ground substitution, giving us a set of ground instances of clauses from $\mathrm{CEE}(\mathcal{S})$. This set is unsatisfiable since we can transform $\mathbb{T}$ to a well-formed **CT**-refutation by erasing ordering constraints.

It is easy to see that these ground instances are valid constrained ground instances mentioned in Proposition 2. Indeed, each positive equality literal $(l \simeq r)$ (except the reflexivity axiom) that takes part in the inference acquires the corresponding strict inequality constraint $(l \succ r)$ during an expansion or a termination step. Each disequality $(s \not\simeq t)$ is either reduced with the help of the

reflexivity axiom or resolved with some positive equality literal. In both cases, the constraint $(s \succeq t)$ will be satisfied. A disequality $(x \not\approx y)$ can only be reduced by reflexivity, so that the constraint $(x = y)$ will be satisfied, too.

Let us prove the completeness of $\mathbf{CT}^{\simeq}$ with respect to CEE-transformed clause sets. Consider the set S of all constrained ground instances of clauses from CEE($\mathcal{S}$). By Proposition 2, S is unsatisfiable, therefore we can build a $\mathbf{CT}$-refutation of S that does not start with the reflexivity axiom $(z \simeq z)$. Then we simply lift the inference to the first order and transform that $\mathbf{CT}$-tableau into a $\mathbf{CT}^{\simeq}$-refutation of CEE($\mathcal{S}$). □

4 Connection tableaux with lazy paramodulation

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

The proposed calculus contains several improvements in comparison with what was sketched at the beginning of the paper. First of all, we use lazy paramodulation only in expansion steps; paramodulation and termination steps do not postpone unification. Second, the "laziness" itself is more restricted now: any two non-variable terms whose unification is postponed should have the same functional symbol at the top. Third, we use ordering constraints. Fourth, we use basic paramodulation.

It should be noted that there are two different forms of the basicness restriction. The first one forbids paramodulation into terms introduced by instantiation. The corresponding refinement of lazy paramodulation was described by M. Moser [15]. 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, stronger form additionally prevents paramodulation into terms introduced by the earlier paramodulation steps [16]. In this form, basicness is used in **LPCT**, too (note the variables with bar), though not everywhere: two of the three equality expansion rules leave the inserted term "on the surface", allowed for subsequent paramodulations.

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.

We prove completeness of **LPCT** by transforming a $\mathbf{CT}^{\simeq}$-refutation of the set of CEE-rewritten clauses into an **LPCT**-refutation of the initial clause set.

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

Proof. We begin by introducing 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 closed $\mathbf{CT}^{\simeq}$-tableau refuting the set CEE($\mathcal{S}$) (by Proposition 3) and transform it into an $\mathbf{LPCT}^{\simeq}$-refutation $\mathbb{T}$ of CEE^3($\mathcal{S}$). In Figure 5, we show how the termination and expansion rules of $\mathbf{CT}^{\simeq}$ can be

Start rule:

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

Reduction rule:

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

Expansion rules:

$$\frac{\mathcal{S},\ (\neg P(\boldsymbol{s}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, P(\boldsymbol{r})}{\bot \cdot (\bar{\boldsymbol{v}} = \boldsymbol{r}) \quad s_1 \not\approx \bar{v}_1 \quad \cdots \quad s_n \not\approx \bar{v}_n \quad L_1 \quad \cdots \quad L_k}$$

$$\frac{\mathcal{S},\ (P(\boldsymbol{s}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, \neg P(\boldsymbol{r})}{\bot \cdot (\bar{\boldsymbol{v}} = \boldsymbol{r}) \quad s_1 \not\approx \bar{v}_1 \quad \cdots \quad s_n \not\approx \bar{v}_n \quad L_1 \quad \cdots \quad L_k}$$

Equality expansion rules:

$$\frac{\mathcal{S},\ (z \approx r \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, L[p]}{L[\bar{w}] \cdot (p = z \succ \bar{w}) \quad r \not\approx \bar{w} \quad L_1 \quad \cdots \quad L_k}$$

$$\frac{\mathcal{S},\ (f(\boldsymbol{s}) \approx r \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, L[p]}{L[\bar{w}] \cdot (p = f(\bar{\boldsymbol{v}}) \succ \bar{w}) \quad r \not\approx \bar{w} \quad s_1 \not\approx \bar{v}_1 \quad \cdots \quad s_n \not\approx \bar{v}_n \quad L_1 \quad \cdots \quad L_k}$$

$$\frac{\mathcal{S},\ (L[f(\boldsymbol{s})] \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, l \approx r}{L[\bar{w}] \cdot (f(\bar{\boldsymbol{v}}) = l \succ r = \bar{w}) \quad s_1 \not\approx \bar{v}_1 \quad \cdots \quad s_n \not\approx \bar{v}_n \quad L_1 \quad \cdots \quad L_k}$$

Paramodulation rules:

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

Termination rules:

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

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

simulated in $\mathbf{LPCT}^{\simeq}$, so that leaves in open branches and generated constraints stay the same. Recall that every equality or disequality in $\mathrm{CEE}(\mathcal{S})$ has a variable on the right hand side (by definition of CEE).

At the second stage we unflatten the clauses. We will call *suspicious* those variables with caret which were introduced by CEE-transformation. We will call a clause *suspicious* if a suspicious variable occurs in it. We will call an $\mathbf{LPCT}^{\simeq}$-inference step *suspicious* if it is a start step or expansion step or equality expansion step that involves a suspicious initial clause.

Let $\mathcal{S}^{(0)}$ be $\mathrm{CEE}^3(\mathcal{S})$ and $\mathbb{T}^{(0)}$ be $\mathbb{T}$. We are going to construct a sequence of clause sets and $\mathbf{LPCT}^{\simeq}$-refutations such that the following statements will hold for every $i > 0$:

- $\mathbb{T}^{(i)}$ is a well-formed refutation of $\mathcal{S}^{(i)}$ in $\mathbf{LPCT}^{\simeq}$;
- there are fewer different suspicious variables in $\mathbb{T}^{(i)}$ than in $\mathbb{T}^{(i-1)}$;
- any suspicious variable $\hat{u}$ occurs exactly twice in a clause from $\mathcal{S}(i)$: once in a disequality of the form $(p \not\simeq \hat{u})$ and once in some other literal (but never as the left argument of a (dis)-equality);
- any non-suspicious clause in $\mathcal{S}^{(i)}$ belongs to $\mathrm{CEE}^1(\mathcal{S})$.

Consider a lowermost suspicious inference I in $\mathbb{T}^{(i-1)}$ (i.e. there are no suspicious steps under that one). Let $\hat{u}$ be a suspicious variable that comes into the tableau with this step. The corresponding suspicious clause is of the form $(L[\hat{u}] \vee p \not\simeq \hat{u} \vee C)$. Let $\mathcal{S}^{(i)}$ be $\mathcal{S}^{(i-1)} \cup \{\, L[p] \vee C \,\}$.

Note that one of these two literals (containing an occurrence of $\hat{u}$) may be an "active literal" in I. This literal will not appear in $\mathbb{T}^{(i-1)}$ in its original form. Nevertheless, we can affirm that $\mathbb{T}^{(i-1)}$ contains two disjoint subtrees, $\mathbb{T}^{\circ}$ and $\mathbb{T}^{\bullet}$, such that the following holds:

- $\mathbb{T}^{\circ}$ and $\mathbb{T}^{\bullet}$ are introduced at the step I;
- $\hat{u}$ does not occur outside of $\mathbb{T}^{\circ}$ and $\mathbb{T}^{\bullet}$;
- the root literal of $\mathbb{T}^{\bullet}$ is of the form $s \not\simeq \hat{u}$ and $\hat{u}$ does not occur in s;
- moreover, $\hat{u}$ occurs in $\mathbb{T}^{\bullet}$ only in disequalities $(t \not\simeq \hat{u})$ and constraints $(t = \hat{u})$ introduced by a reduction step; $\hat{u}$ does not occur in these t (indeed, all we can do with $(t \not\simeq \hat{u})$ is to reduce it or to paramodulate in t);
- $\hat{u}$ occurs exactly once in the root literal of $\mathbb{T}^{\circ}$;
- $\hat{u}$ does not occur in the root node constraint of $\mathbb{T}^{\circ}$.

Let $\mathbb{T}^{\circ}$ have the form:

$$\frac{M[\hat{u}] \cdot \delta}{\mathbb{T}_1 \quad \cdots \quad \mathbb{T}_n}$$

Below, $\mathbb{T}_k[\hat{u} \leftarrow t]$ denotes the tree $\mathbb{T}_k$ where all occurrences of $\hat{u}$ (both in literals and constraints) are replaced with t. It is easy to see that this substitution does not make $\mathbb{T}_k$ ill-formed, provided that $\hat{u}$ and t are equal with respect to the

$\mathbf{CT}^{\simeq}$-Termination $\implies$ $\mathbf{LPCT}$-Paramodulation:

$$\frac{\mathcal{S} \parallel \Gamma, f(\boldsymbol{x}) \not\simeq y, \Delta, l \simeq u}{\bot \cdot (f(\boldsymbol{x}) = l \succ u = y)} \quad \Longrightarrow \quad \frac{\mathcal{S} \parallel \Gamma, f(\boldsymbol{x}) \not\simeq y, \Delta, l \simeq u}{\dfrac{\bar{w} \not\simeq y \cdot (f(\boldsymbol{x}) = l \succ u = \bar{w})}{\bot \cdot (\bar{w} = y)}}$$

$\mathbf{CT}^{\simeq}$-Expansion $\implies$ $\mathbf{LPCT}$-Expansion:

$$\frac{\mathcal{S},\ (\neg P(\boldsymbol{x}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, P(\boldsymbol{y})}{\bot \cdot (\boldsymbol{x} = \boldsymbol{y}) \qquad L_1 \qquad \cdots \qquad L_k} \quad \Longrightarrow$$

$$\frac{\mathcal{S},\ (\neg P(\boldsymbol{x}) \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, P(\boldsymbol{y})}{\bot \cdot (\bar{\boldsymbol{v}} = \boldsymbol{y}) \qquad \dfrac{x_1 \not\simeq \bar{v}_1}{\bot \cdot (x_1 = \bar{v}_1)} \qquad \cdots \qquad \dfrac{x_n \not\simeq \bar{v}_n}{\bot \cdot (x_n = \bar{v}_n)} \qquad L_1 \qquad \cdots \qquad L_k}$$

$\mathbf{CT}^{\simeq}$-Expansion $\implies$ $\mathbf{LPCT}$-EqualityExpansion:

$$\frac{\mathcal{S},\ (f(\boldsymbol{x}) \not\simeq y \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, l \simeq u}{\bot \cdot (f(\boldsymbol{x}) = l \succ u = y) \qquad L_1 \qquad \cdots \qquad L_k} \quad \Longrightarrow$$

$$\frac{\mathcal{S},\ (f(\boldsymbol{x}) \not\simeq y \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, l \simeq u}{\dfrac{\bar{w} \not\simeq y \cdot (f(\bar{\boldsymbol{v}}) = l \succ u = \bar{w})}{\bot \cdot (\bar{w} = y)} \qquad \dfrac{x_1 \not\simeq \bar{v}_1}{\bot \cdot (x_1 = \bar{v}_1)} \qquad \cdots \qquad \dfrac{x_n \not\simeq \bar{v}_n}{\bot \cdot (x_n = \bar{v}_n)} \qquad L_1 \qquad \cdots \qquad L_k}$$

$$\frac{\mathcal{S},\ (z \simeq u \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, f(\boldsymbol{x}) \not\simeq y}{\bot \cdot (f(\boldsymbol{x}) = z \succ u = y) \qquad L_1 \qquad \cdots \qquad L_k} \quad \Longrightarrow$$

$$\frac{\mathcal{S},\ (z \simeq u \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, f(\boldsymbol{x}) \not\simeq y}{\dfrac{\bar{w} \not\simeq y \cdot (f(\boldsymbol{x}) = z \succ \bar{w})}{\bot \cdot (\bar{w} = y)} \qquad \dfrac{u \not\simeq \bar{w}}{\bot \cdot (u = \bar{w})} \qquad L_1 \qquad \cdots \qquad L_k}$$

$$\frac{\mathcal{S},\ (f(\boldsymbol{z}) \simeq u \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, f(\boldsymbol{x}) \not\simeq y}{\bot \cdot (f(\boldsymbol{x}) = f(\boldsymbol{z}) \succ u = y) \qquad L_1 \qquad \cdots \qquad L_k} \quad \Longrightarrow$$

$$\frac{\mathcal{S},\ (f(\boldsymbol{z}) \simeq u \vee L_1 \vee \cdots \vee L_k) \parallel \Gamma, f(\boldsymbol{x}) \not\simeq y}{\dfrac{\bar{w} \not\simeq y \cdot (f(\boldsymbol{x}) = f(\bar{\boldsymbol{v}}) \succ \bar{w})}{\bot \cdot (\bar{w} = y)} \qquad \dfrac{u \not\simeq \bar{w}}{\dfrac{\bot \cdot}{(u = \bar{w})}} \qquad \dfrac{z_1 \not\simeq \bar{v}_1}{\dfrac{\bot \cdot}{(z_1 = \bar{v}_1)}} \qquad \cdots \qquad \dfrac{z_n \not\simeq \bar{v}_n}{\dfrac{\bot \cdot}{(z_n = \bar{v}_n)}} \qquad L_1 \quad \cdots \quad L_k}$$

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

constraints in $\mathbb{T}^{(i-1)}$. A tree transformation $[T]^{\mathbb{T}^\circ}$ is defined as follows:

$$\left[\frac{t \not\approx \hat{u} \cdot \gamma}{\bot \cdot (t = \hat{u})}\right]^{\mathbb{T}^\circ} \quad \Longrightarrow \quad \frac{M[t] \cdot \gamma}{\mathbb{T}_1[\hat{u} \leftarrow t] \quad \cdots \quad \mathbb{T}_n[\hat{u} \leftarrow t]}$$

$$\left[\frac{t \not\approx \hat{u} \cdot \gamma}{T_1 \quad \cdots \quad T_n}\right]^{\mathbb{T}^\circ} \quad \Longrightarrow \quad \frac{M[t] \cdot \gamma}{[T_1]^{\mathbb{T}^\circ} \quad \cdots \quad [T_n]^{\mathbb{T}^\circ}}$$

$$\left[\frac{L \cdot \gamma}{T_1 \quad \cdots \quad T_n}\right]^{\mathbb{T}^\circ} \quad \Longrightarrow \quad \frac{L \cdot \gamma}{[T_1]^{\mathbb{T}^\circ} \quad \cdots \quad [T_n]^{\mathbb{T}^\circ}}$$

Consider the tableau $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$. We can affirm the following:

- The suspicious variable $\hat{u}$ does not occur in $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$.
- Given that $(s \not\approx \hat{u})$ is the root literal of $\mathbb{T}^\bullet$, the literal $M[s]$ is the root literal of $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$.
- Every paramodulation made in a literal of the form $(t \not\approx \hat{u})$ in $\mathbb{T}^\bullet$ was made in the term t. Therefore it can also be made in the corresponding literal $M[t]$ in $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$.
- Every literal $(t \not\approx \hat{u})$ reduced in $\mathbb{T}^\bullet$ becomes the literal $M[t]$ in $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ and is extended further with the subtrees $\mathbb{T}_i[\hat{u} \leftarrow t]$. Since $\hat{u}$ and t are equal with respect to the constraints of $\mathbb{T}^{(i-1)}$, the tree $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ is closed.

Then we add the constraint δ to the root node constraint of $[\mathbb{T}^\bullet]^{\mathbb{T}^\circ}$ and replace $\mathbb{T}^\circ$ and $\mathbb{T}^\bullet$ in $\mathbb{T}^{(i-1)}$ with that tree. Also, we replace $\mathcal{S}^{(i-1)}$ with $\mathcal{S}^{(i)}$ in the root. The resulting well-formed closed $\mathbf{LPCT}^{\simeq}$-tableau will be $\mathbb{T}^{(i)}$.

In $\mathbb{T}^{(i)}$, the step I is made with the clause $L[p] \vee C$. Then we make all paramodulations in p that were made in $\mathbb{T}^\bullet$ and proceed where needed with the inferences that were made in $\mathbb{T}^\circ$. The variable $\hat{u}$ disappeared from $\mathbb{T}^{(i)}$ and no other suspicious variables were introduced. It is not difficult to verify that other required conditions are satisfied, too.

By repeating this procedure, we will eventually get a closed tableau $\mathbb{T}^{(N)}$ where suspicious variables do not occur at all. This tableau is, essentially, an $\mathbf{LPCT}^{\simeq}$-refutation of the set $\mathrm{CEE}^1(\mathcal{S})$. It remains to undo the symmetry elimination step. We replace the symbol $\simeq$ with $\approx$ and reorient equalities to their initial form in $\mathcal{S}$.

Altogether, 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 variables with caret introduced in CEE-clauses 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, each term has exactly one "value". It is not the case for **LPCT**.

Let $\mathcal{S}$ be the set $\{\, x \approx c \vee x \approx g(h(x)),\ f(c) \approx d,\ f(g(z)) \approx d,\ f(a) \not\approx d \,\}$. The following tableau built in a simplified version of **LPCT** cannot be obtained

from any $\mathbf{CT}^{\simeq}$-refutation of $\mathrm{CEE}(\mathcal{S})$.

$$
\begin{array}{c}
\mathcal{S} \\ \hline
f(a) \not\approx d \\ \hline
\begin{array}{cc}
\begin{array}{c}
x \approx c \\ \hline
\begin{array}{cc}
\begin{array}{c}
f(c) \not\approx d \\ \hline
f(c) \approx d \\ \hline
\begin{array}{cc}
\dfrac{d \not\approx d}{\bot} & \dfrac{f(c) \not\approx f(c)}{\bot}
\end{array}
\end{array}
&
\dfrac{x \not\approx a}{\bot \cdot (x = a)}
\end{array}
\end{array}
&
\begin{array}{c}
x \approx g(h(x)) \\ \hline
\begin{array}{cc}
\begin{array}{c}
f(g(h(x))) \not\approx d \\ \hline
f(g(z)) \approx d \\ \hline
\begin{array}{cc}
\dfrac{d \not\approx d}{\bot} & \dfrac{f(g(z)) \not\approx f(g(h(x)))}{\bot \cdot (z = h(x))}
\end{array}
\end{array}
&
\dfrac{x \not\approx a}{\bot \cdot (x = a)}
\end{array}
\end{array}
\end{array}
\end{array}
$$

Here, we replace the constant a in the starting clause with two different 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 [13]. This form of unification is much weaker than *top unification* (introduced in [17] and used in [14]) which descends down to variables. Top unification allows us to restrict drastically 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 ordering $a > b > c$ and the set $\mathcal{S} = \{ P(c) \vee Q(c), \neg P(a), \neg Q(b), b \approx c, a \approx c \}$. Ordering constraints prohibit paramodulations into c. The only way to refute $\mathcal{S}$ in **LPCT** is to resolve $P(c)$ 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. The author is not aware about 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.

For the future, 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. It is also interesting to study more restricted forms of laziness, probably, giving up orderings and basicness.

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

Acknowledgment. The author is grateful to Alexander Lyaletski and Konstantin Verchinine for their guidance and expertise through the course of this work.

References

1. Loveland, D.W.: Mechanical theorem proving by model elimination. Journal of the ACM **16**(3) (1968) 349–363
2. Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO: a high-performance theorem prover. Journal of Automated Reasoning **8**(2) (1992) 183–212
3. Letz, R., Stenz, G.: Model elimination and connection tableau procedures. In Robinson, A., Voronkov, A., eds.: Handbook for Automated Reasoning. Volume II. Elsevier Science (2001) 2017–2116
4. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In Robinson, A., Voronkov, A., eds.: Handbook for Automated Reasoning. Volume I. Elsevier Science (2001) 371–443
5. Degtyarev, A., Voronkov, A.: What you always wanted to know about rigid E-unification. Journal of Automated Reasoning **20**(1) (1998) 47–80
6. 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. Volume 2381 of Lecture Notes in Computer Science., Springer-Verlag (2002) 130–144
7. Loveland, D.W.: Automated Theorem Proving: A Logical Basis. Volume 6 of Fundamental studies in Computer Science. North-Holland (1978)
8. Moser, M., Ibens, O., Letz, R., Steinbach, J., Goller, C., Schumann, J., Mayr, K.: SETHEO and E-SETHEO — the CADE-13 systems. Journal of Automated Reasoning **18**(2) (1997) 237–246
9. Brand, D.: Proving theorems with the modification method. SIAM Journal of Computing **4** (1975) 412–430
10. Moser, M., Steinbach, J.: STE-modification revisited. Technical Report AR-97-03, Fakultät für Informatik, Technische Universität München, München (1997)
11. 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. Volume 1421 of Lecture Notes in Computer Science., Springer-Verlag (1998) 175–190
12. 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)
13. Gallier, J., Snyder, W.: Complete sets of transformations for general E-unification. Theoretical Computer Science **67** (1989) 203–260

14. Snyder, W., Lynch, C.: Goal directed strategies for paramodulation. In Book, R., ed.: Rewriting Techniques and Applications: 4th International Conference, RTA 1991. Volume 488 of Lecture Notes in Computer Science., Springer-Verlag (1991) 150–161
15. Moser, M.: Improving transformation systems for general E-unification. In Kirchner, C., ed.: Rewriting Techniques and Applications: 5th International Conference, RTA 1993. Volume 690 of Lecture Notes in Computer Science., Springer-Verlag (1993) 92–105
16. Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W.: Basic paramodulation. Information and computation **121**(2) (1995) 172–192
17. Dougherty, D.J., Johann, P.: An improved general E-unification method. In Stickel, M.E., ed.: Automated Deduction: 10th International Conference, CADE-10. Volume 449 of Lecture Notes in Computer Science., Springer-Verlag (1990) 261–275