Theorem Proving with Bounded Rigid *E*-Unification*

Peter Backeman and Philipp Rümmer

Uppsala University, Sweden

Abstract. Rigid E-unification is the problem of unifying two expressions modulo a set of equations, with the assumption that every variable denotes exactly one term (*rigid* semantics). This form of unification was originally developed as an approach to integrate equational reasoning in tableau-like proof procedures, and studied extensively in the late 80s and 90s. However, the fact that *simultaneous* rigid E-unification is undecidable has limited practical adoption, and to the best of our knowledge there is no tableau-based theorem prover that uses rigid E-unification. We introduce simultaneous bounded rigid E-unification (BREU), a new version of rigid E-unification that is bounded in the sense that variables only represent terms from finite domains. We show that (simultaneous) BREU is NP-complete, outline how BREU problems can be encoded as propositional SAT-problems, and use BREU to introduce a sound and complete sequent calculus for first-order logic with equality.

1 Introduction

The integration of efficient equality reasoning in tableaux and sequent calculi is a long-standing challenge, and has led to a wealth of theoretically intriguing, yet surprisingly few practically satisfying solutions. Among others, a family of approaches related to the (undecidable) problem of computing *simultaneous rigid* E-*unifiers* have been developed, by utilising incomplete unification procedures in such a way that an overall complete first-order calculus is obtained. To the best of our knowledge, however, none of those procedures has led to competitive theorem provers.

We introduce *simultaneous bounded rigid* E-*unification* (BREU), a new version of rigid E-unification that is bounded in the sense that variables only represent terms from finite domains. BREU is significantly simpler than ordinary rigid E-unification, in terms of computational complexity as well as algorithmic aspects, and therefore a promising candidate for efficient implementation. BREU still enables the design of complete first-order calculi, but also makes combinations with techniques from the SMT field possible, in particular the use of congruence closure to handle ground equations.

* This work was partly supported by the Microsoft PhD Scholarship Programme and the Swedish Research Council.

1.1 Background and Motivating Example

We start by illustrating our approach using the following problem (from [5]):

$$\phi \quad = \quad \exists x, y, u, v.\ \begin{pmatrix} (a \not\approx b \ \vee\ g(x,u,v) \approx g(y, f(c), f(d))) \ \wedge \\ (c \not\approx d \ \vee\ g(u,x,y) \approx g(v, f(a), f(b))) \end{pmatrix}$$

To show validity of ϕ, a Gentzen-style proof (or, equivalently, a tableau) can be constructed, using free variables for x, y, u, v:

$$\dfrac{\dfrac{\dfrac{\mathcal{A}}{a \approx b \ \vdash\ g(X,U,V) \approx g(Y, f(c), f(d))} \qquad \dfrac{\mathcal{B}}{c \approx d \ \vdash\ g(U,X,Y) \approx g(V, f(a), f(b))}}{\vdash\ (a \not\approx b \vee g(X,U,V) \approx g(Y, f(c), f(d))) \wedge (c \not\approx d \vee g(U,X,Y) \approx g(V, f(a), f(b)))}}{\vdash\ \phi}$$

To finish this proof, both $\mathcal{A}$ and $\mathcal{B}$ need to be closed by applying further rules, and substituting concrete terms for the variables. The substitution $\sigma_l = \{X \mapsto Y, U \mapsto f(c), V \mapsto f(d)\}$ makes it possible to close $\mathcal{A}$ through equational reasoning, and $\sigma_r = \{X \mapsto f(a), U \mapsto V, Y \mapsto f(b)\}$ closes $\mathcal{B}$, but neither closes both. Finding a substitution that closes both branches is known as *simultaneous rigid* E*-unification* (SREU), and has first been formulated in [9]:

Definition 1 (Rigid *E*-Unification). *Let E be a set of equations, and s, t be terms. A substitution σ is called a* rigid E-unifier *of s and t if $s\sigma \approx t\sigma$ follows from $E\sigma$ via ground equational reasoning. A* simultaneous rigid E-unifier *σ is a common rigid* E*-unifier for a set $(E_i, s_i, t_i)_{i=1}^n$ of rigid* E*-unification problems.*

In our example, two rigid E-unification problems have to be solved:

$$\begin{aligned} E_1 &= \{a \approx b\}, \qquad s_1 = g(X,U,V), \quad t_1 = g(Y, f(c), f(d)), \\ E_2 &= \{c \approx d\}, \qquad s_2 = g(U,X,Y), \quad t_2 = g(V, f(a), f(b)). \end{aligned}$$

We can observe that $\sigma_s = \{X \mapsto f(a), Y \mapsto f(b), U \mapsto f(c), V \mapsto f(d)\}$ is a simultaneous rigid E-unifier, and suffices to finish the proof of ϕ. In general, of course, the SREU problem famously turned out undecidable [4], which makes the style of reasoning shown here problematic.

Different solutions have been proposed to address this situation, including potentially non-terminating, but complete E-unification procedures [8], and terminating but incomplete algorithms that are nevertheless sufficient to create complete proof procedures [5, 11]. The practical impact of such approaches has been limited; to the best of our knowledge, there is no (at least no actively maintained) theorem prover based on such explicit forms of SREU.

This paper introduces a new approach, *bounded rigid* E*-unification* (BREU), which belongs to the class of "terminating, but incomplete" algorithms for SREU. In contrast to ordinary SREU, our method only considers E-unifiers where substituted terms are taken from some predefined finite set. This directly

implies decidability of the unification problem; as we will see later, the problem is in fact NP-complete, even for the simultaneous case, and can be handled efficiently using SAT technology. In our experiments, cases with hundreds of simultaneous unification problems and thousands of terms were well in reach, and future advances in terms of algorithm design and efficient implementation are expected to further improve scalability.

For sake of presentation, BREU operates on formulae that are normalised by means of flattening (observe that ϕ and ϕ' are equivalent):

$$\begin{aligned}\phi' \;=\; & \forall z_1, z_2, z_3, z_4.\, \big(f(a) \not\approx z_1 \vee f(b) \not\approx z_2 \vee f(c) \not\approx z_3 \vee f(d) \not\approx z_4 \vee \\ & \exists x, y, u, v.\, \forall z_5, z_6, z_7, z_8.\, \left(\begin{array}{l} g(x,u,v) \not\approx z_5 \vee g(y, z_3, z_4) \not\approx z_6 \vee \\ g(u,x,y) \not\approx z_7 \vee g(v, z_1, z_2) \not\approx z_8 \vee \\ ((a \not\approx b \vee z_5 \approx z_6) \wedge (c \not\approx d \vee z_7 \approx z_8)) \end{array}\right)\big)\end{aligned}$$

A proof constructed for ϕ' has the same structure as the one for ϕ, with the difference that all function terms are now isolated in the antecedent:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}'}{\ldots, g(X,U,V) \approx o_5, a \approx b \;\vdash\; o_5 \approx o_6} \quad \dfrac{\mathcal{B}'}{\ldots, g(U,X,Y) \approx o_7, c \approx d \;\vdash\; o_7 \approx o_8}}{\vdots}}{f(a) \approx o_1 \vee f(b) \approx o_2 \vee f(c) \approx o_3 \vee f(d) \approx o_4 \;\vdash\; \exists x, y, u, v.\, \forall z_5, z_6, z_7, z_8.\; \ldots}\,(*)}{\vdots}}{\vdash\; \forall z_1, z_2, z_3, z_4.\; \ldots}$$

To obtain a *bounded* rigid E-unification problem, we now restrict the terms considered for instantiation of X, Y, U, V to the symbols that were in scope when the variables were introduced (at $(*)$ in the proof): X ranges over constants $\{o_1, o_2, o_3, o_4\}$, Y over $\{o_1, o_2, o_3, o_4, X\}$, and so on. Since the problem is flat, those sets contain representatives of all existing ground terms at point $(*)$ in the proof. It is therefore possible to find a simultaneous E-unifier, namely the substitution $\sigma_b = \{X \mapsto o_1, Y \mapsto o_2, U \mapsto o_3, V \mapsto o_4\}$.

It has long been observed that this restricted instantiation strategy gives rise to a complete calculus for first-order logic with equality. The strategy was first introduced as *dummy instantiation* in the seminal work of Kanger [13] (in 1963, i.e., even before the introduction of unification), and later studied under the names *subterm instantiation* and *minus-normalisation* [6, 7]; the relationship to SREU was observed in [5]. The impact on practical theorem proving was again limited, however, among others because no efficient search procedures for dummy instantiation were available [7]. The present paper addresses this topic and makes the following main contributions:

- we define bounded rigid E-unification, as a restricted version of SREU, and investigate its complexity (Sect. 3);
- we present a sound, complete, and backtracking-free BREU-based sequent calculus for first-order with equality (Sect. 4–6);
- we give a preliminary experimental evaluation, comparing with other tableau-based theorem provers (Sect. 7).

1.2 Further Related Work

For a general overview of research on equality handling in sequent calculi and related systems, as well as on SREU, we refer the reader to the detailed handbook chapter [6]. The following paragraphs survey some of the more recent work.

Our work is partly motivated by a recent line of research on backtracking-free tableau calculi with free variables [10], capturing unification conditions as constraints that are attached to literals or tableau branches. This calculus was extended to handle equality using superposition-style inferences in [11], building on results from [5]. Our work resembles both [5, 11] in that we define an incomplete version of SREU, but show it to be sufficient for complete first-order reasoning. Our variant BREU is incomparable in completeness to the SREU solving in [5, 11]: BREU is able to derive a solution for the example shown in Sect. 1.1, which [5, 11] cannot; on the other hand, the procedures in [5, 11] are able to synthesise new terms of unbounded size as unifiers, whereas our procedure only considers terms from predefined bounded domains. The calculus in [11] was further extended to handle linear integer arithmetic in [14], however, excluding functions (but including uninterpreted predicates, to which functions can be reduced via axioms), leading to a further unification problem that is incomparable in expressiveness.

Equality handling was integrated into hyper tableaux in [2], again using superposition-style inferences, and also including redundancy criteria. This work deliberately avoids the use of rigid free variables shared between multiple tableau branches, so that branches can be closed one at a time, and there is no need for simultaneous E-unification. The calculus was implemented in the Hyper prover, against which we compare our implementation in Sect. 7.

2 Preliminaries

We assume familiarity with classical first-order logic and Gentzen-style calculi (see e.g., [8]). Given countably infinite sets C of constants (denoted by $c, d, \dots$), V_b of bound variables (written $x, y, \dots$), and V of free variables (denoted by $X, Y, \dots$), as well as a finite set F of fixed-arity function symbols (written $f, g, \dots$), the syntactic categories of *formulae* ϕ and *terms* t are defined by

$$\phi \ ::= \ \phi \wedge \phi \ \big| \ \phi \vee \phi \ \big| \ \neg\phi \ \big| \ \forall x.\phi \ \big| \ \exists x.\phi \ \big| \ t \approx t \ , \qquad t \ ::= \ c \ \big| \ x \ \big| \ X \ \big| \ f(t, \dots, t) \ .$$

Note that we distinguish between constants and zero-ary functions for reasons that will become apparent later. We generally assume that bound variables x only occur underneath quantifiers $\forall x$ or $\exists x$. Semantics of terms and formulae without free variables is defined as is common using first-order structures (U, I) consisting of a non-empty universe U, and an interpretation function I.

We call constants and (free or bound) variables *atomic terms*, and all other terms *compound terms*. A *flat equation* is an equation between atomic terms, or an equation of the form $f(t_1, \dots, t_n) \approx t_0$, where $t_0, \dots, t_n$ are atomic terms. A *flat formula* is a formula ϕ in which functions only occur in flat equations. A

formula ϕ is *positively flat* (*negatively flat*) if it is flat, and every occurrence of a function symbol is underneath an even (odd) number of negations. Note that every formula can be transformed to an equivalent positively flat (negatively flat) formula; we will usually assume that such preprocessing has been applied to formulae handled by our procedures. This kind of preprocessing is also standard for congruence closure procedures [1], and similarly used in SMT solvers.

If Γ is a finite set of positively flat formulae (the *antecedent*), and Δ a finite set of negatively flat formulae (the *succedent*), then $\Gamma \vdash \Delta$ is called a *sequent.* A sequent $\Gamma \vdash \Delta$ without free variables is called *valid* if the formula $\bigwedge \Gamma \to \bigvee \Delta$ is valid. A calculus rule is a binary relation between finite sets of sequents (the premises) and sequents (the conclusion).

A substitution is a mapping of variables to terms, such that all but finitely many variables are mapped to themselves. Symbols $\sigma, \theta, \ldots$ denote substitutions, and we use post-fix notation $\phi\sigma$ or $t\sigma$ to denote application of substitutions. An *atomic substitution* is a substitution that maps variables only to atomic terms. We write $u[r]$ do denote that r is a sub-expression of a term or formula u.

Definition 2 (Replacement relation [16]). *The* replacement relation $\to_E$ *induced by a set of equations E is defined by: $u[l] \to u[r]$ if $l \approx r \in E$. The relation $\leftrightarrow_E^*$ represents the reflexive, symmetric and transitive closure of $\to_E$.*

3 Bounded Rigid *E*-Unification

We present *bounded* rigid E-Unification, a restriction of rigid E-unification in the sense that we now require solutions to be atomic substitutions such that variables are only mapped to smaller atomic terms according to a given partial order $\preceq$. This order takes over the role of an occurs-check of regular unification.

Definition 3 (BREU). *A* bounded rigid E-unification (BREU) problem *is a triple $U = (\preceq, E, e)$, with $\preceq$ being a partial order over atomic terms such that for all variables X the set $\{s \mid s \preceq X\}$ is finite; E is a finite set of flat equations; and $e = s \approx t$ is an equation between atomic terms (the target equation). An atomic substitution σ is called a* bounded rigid E-unifier *of s and t if $s\sigma \leftrightarrow_{E\sigma}^* t\sigma$ and $X\sigma \preceq X$ for all variables X.*

Note that the partial order $\preceq$ is in principle an infinite object. However, only a finite part of it is relevant for defining and solving a BREU problem, which ensures that BREU problems can effectively be represented.

Definition 4 (Simultaneous BREU). *A* simultaneous bounded rigid E-unification problem *is a pair $(\preceq, (E_i, e_i)_{i=1}^n)$ such that each triple $(\preceq, E_i, e_i)$ is a bounded rigid* E*-unification problem. An atomic substitution σ is a* simultaneous bounded rigid E-unifier *for $(\preceq, (E_i, e_i)_{i=1}^n)$ if σ is a bounded rigid* E*-unifier for each problem $(\preceq, E_i, e_i)$.*

A solution to a simultaneous BREU problem can be used to close all branches in a proof tree. In Sect. 4 we present the connection in detail.

Example 5. We revisit the example introduced in Sect. 1.1, which leads to the following simultaneous BREU problem $(\preceq, \{(E_1, e_1), (E_2, e_2)\})$:

$$E_1 = E \cup \{a \approx b\}, \quad e_1 = o_5 \approx o_6, \qquad E_2 = E \cup \{c \approx d\}, \quad e_2 = o_7 \approx o_8,$$
$$E = \left\{ \begin{array}{l} f(a) \approx o_1, f(b) \approx o_2, f(c) \approx o_3, f(d) \approx o_4, \\ g(X, U, V) \approx o_5, g(Y, o_3, o_4) \approx o_6, g(U, X, Y) \approx o_7, g(V, o_1, o_2) \approx o_8 \end{array} \right\}$$

with $\{a, b, c, d\} \prec o_1 \prec o_2 \prec o_3 \prec o_4 \prec X \prec Y \prec U \prec V \prec o_5 \prec o_6 \prec o_7 \prec o_8$.

A unifier to this problem is sufficient to close all goals of the tree up to equational reasoning; one solution is $\sigma = \{X \mapsto o_1, Y \mapsto o_2, U \mapsto o_3, V \mapsto o_4\}$.

While SREU is undecidable in the general case, BREU is decidable; the existence of bounded rigid E-unifiers can be decided in non-deterministic polynomial time, since it can be verified in polynomial time that a substition σ is a solution of a (possibly simultaneous) BREU problem (and since an E-unifier only has to consider variables that occur in the problem, it can be represented in space linear in the size of the BREU problem). Hardness follows from the fact that propositional satisfiability can be reduced to BREU, by virtue of the following construction.

3.1 Reduction of SAT to BREU

Consider propositional formulae ϕ_b, which are assumed to be constructed using the following operators:

$$\phi_b \ ::= \ p \ \big| \ \neg\phi_b \ \big| \ \phi_b \vee \phi_b$$

where p is a propositional symbol.

A formula ϕ_b of this kind is converted to a BREU problem by introducing two constants $\mathbf{0}$ and $\mathbf{1}$; two function symbols f_{or} and f_{not}; for each propositional symbol p in ϕ_b, a variable X_p such that $\mathbf{0} \prec X_p$ and $\mathbf{1} \prec X_p$; and for each sub-formula ψ of ϕ_b, a constant c_ψ and an equation:

$$\begin{array}{ll} X_p \approx c_\psi & \text{if } \psi = p, \\ f_{not}(c_{\psi_1}) \approx c_\psi & \text{if } \psi = \neg\psi_1, \\ f_{or}(c_{\psi_1}, c_{\psi_2}) \approx c_\psi & \text{if } \psi = \psi_1 \vee \psi_2. \end{array}$$

The above, together with the set of equations $\{f_{or}(\mathbf{0}, \mathbf{0}) \approx \mathbf{0}, f_{or}(\mathbf{0}, \mathbf{1}) \approx \mathbf{1}, f_{or}(\mathbf{1}, \mathbf{0}) \approx \mathbf{1}, f_{or}(\mathbf{1}, \mathbf{1}) \approx \mathbf{1}, f_{not}(\mathbf{0}) \approx \mathbf{1}, f_{not}(\mathbf{1}) \approx 0\}$ defining the semantics of the Boolean operators, and a target equation $c_{\phi_b} \approx \mathbf{1}$ yields a BREU problem that is naturally equivalent to the problem of checking satisfiability of ϕ_b. Indeed, every E-unifier can be translated to an assignment A of the propositional symbols such that $A \models \phi_b$.

Theorem 6. *Satisfiability of BREU problems is NP-complete.*

3.2 Generalisations

A number of generalisations in the definition of BREU are possible, but can uniformly be reduced to BREU as formulated in Def. 3, without causing a blow-up in the size of the BREU problem.

General target constraints. Most importantly, there is no need to restrict BREU to single target equations e, instead arbitrary positive Boolean combinations of equations can be solved; this observation is useful for integration of BREU into calculi. Any such combination of equations can be transformed to a single target equation using a construction resembling that in Sec. 3.1, at the cost of introducing a linear number of new symbols and defining equations.

For the remainder of the paper, we assume that e in Def. 3 can indeed be any positive Boolean combination of atomic equations.

Arbitrary equations. BREU problems containing arbitrary (i.e., possibly non-flat) equations in E or as target equation can be handled by reduction to equisatisfiable BREU problems with only flat equations, in a manner similar to [1]. Any non-flat equation of the form $t[f(\bar{c})] \approx s$ can be replaced by two new equations $t[d] \approx s$ and $f(\bar{c}) \approx d$, where d is a fresh constant; the symmetric case, and non-flat target equations are handled similarly. Iterating this reduction eventually results in a problem with only flat equations.

Non-atomic E*-unifiers.* It is further possible to consider partial orders $\preceq$ over arbitrary terms, as long as the set $\{s \mid s \preceq X\}$ is still finite for all variables X. Reduction to problems as in Def. 3 is done by introducing a fresh constant c_t and a (possibly non-flat) equation $t \approx c_t$ for each compound term t occurring in a set $\{s \mid s \preceq X\}$ for some variable X in the BREU problem. A new order $\preceq'$ is defined by replacing compound terms t with constants c_t, in such a way that

$$\{s \mid s \preceq' X\} \;=\; \{s \mid s \preceq X, s \text{ is atomic}\} \cup \{c_t \mid t \preceq X, t \text{ is compound}\}\ .$$

With this in mind, it is possible to relax Def. 3 by including non-atomic unifiers σ (which might map variables to compound terms) as solutions to a BREU problem, as long as the condition $X\sigma \preceq X$ holds for all variables X.

Example 7. Consider the generalised BREU problem $B = (\preceq, E, e)$ defined by

$$E = \{f(f(a,b),c) \approx g(b), f(X,Y) \approx c, g(b) \approx a\}, \qquad e = a \approx c,$$
$$a \prec b \prec c \prec f(a,a) \prec f(a,b) \prec f(b,a) \prec f(b,b) \prec X \prec Y.$$

Intuitively, the order $\preceq$ encodes the fact that an E-unifier has to be constructed that maps every variable to a term with at most one occurrence of f, and no occurrence of g. A solution is the substitution $\sigma = \{X \mapsto f(a,b), Y \mapsto c\}$.

An equisatisfiable BREU problem according to Def. 3 is $B' = (\preceq', E', e'\}$:

$$E' = \left\{ \begin{array}{l} f(d_1,c) \approx d_2, f(a,b) \approx d_1, g(b) \approx d_2, f(X,Y) \approx c, g(b) \approx a, \\ f(a,a) \approx d_3, f(a,b) \approx d_4, f(b,a) \approx d_5, f(b,b) \approx d_6 \end{array} \right\},$$
$$e' = e = a \approx c, \qquad a \prec' b \prec' c \prec' d_3 \prec' d_4 \prec' d_5 \prec' d_6 \prec' X \prec' Y,$$

with the E-unifier $\sigma' = \{X \mapsto d_4, Y \mapsto c\}$.

3.3 Encoding of E-Unification into SAT

Since satisfiability of BREU problems is NP-complete, a natural approach to compute solutions is an encoding as a propositional SAT problem, so that the performance of modern SAT solvers can be put to use. A procedure for solving a BREU problem will consist of three steps: (i) generating a candidate E-unifier σ; (ii) using congruence closure [1] to calculate the equivalence relation induced by the candidate σ and the equations of the BREU problem; and (iii) checking if the BREU target equation is satisfied by this relation.

Each of these steps can be encoded into SAT. Candidate E-unifiers σ are represented by a set of bit-vector variables storing the index of the term $X\sigma$ that each variable X is mapped to. To guess candidate E-unifiers, it is then just necessary to encode the conditions $X\sigma \preceq X$ as a propositional formula.

A congruence closure procedure can be modelled by representing intermediate results (i.e., equivalence relations) as a sequence of union-find data structures. To represent such a data structure in SAT, it suffices to introduce one bit-vector variable per atomic term t occurring in the BREU problem, storing the index of the parent of t in the union-find forest. Propositional constraints are added to characterise well-formed union-find forests, and to define the derivation of each forest from the previous one.

Lastly, to check the correctness of the candidate σ, it is asserted that the target equation is satisfied in the last union-find structure.

4 A First-order Logic Calculus with E-Unification

We will now introduce our sequent calculus for first-order logic with equality. The calculus operates only on flat formulae, and is kept quite minimalist to illustrate the use of free variables and BREU for delayed instantiation; for practical purposes, many refinements are possible, some of which are outlined in Sect. 6. The BREU procedure is utilised to define a global closure rule that discharges all goals of a proof tree simultaneously. Proof construction is intended to be done in upward direction and backtracking-free manner, following the proof procedures presented in [10, 14]; this is possible because all calculus rules are non-destructive and the overall calculus proof-confluent. We will show that fair application of the proof rules is complete.

The propositional, first-order, and equational rules of the calculus are shown in Table 1. Propositional and first-order rules mostly correspond to the classical system LK [8], however, keeping all structural rules implicit (Γ and Δ are sets of formulae). The first-order rules use Skolem symbols $c \in C$ for existential quantifiers in the antecedent, and fresh free variables $X \in V$ for universal quantifiers; and similarly for formulae in the succedent.

The equational rules simplify terms by means of ordered ground rewriting. Given a proof tree, we introduce a strict partial order $\prec \subseteq (C \cup V)^2$ over constants

Table 1. Our sequent calculus for first-order logic with equality. In rules $\forall\text{L}$ and $\exists\text{R}$, X is a fresh variable, whereas the rules $\exists\text{L}$ and $\forall\text{R}$ introduce a fresh constant c. In $\approx\text{L}$ and $\approx\text{R}$, the equation $(t' \approx s')[t/s]$ is the result of replacing all occurrences of t with s.

$$\frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta}\ \wedge\text{L} \qquad \frac{\Gamma \vdash \phi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \wedge \psi, \Delta}\ \wedge\text{R} \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma, \neg\phi \vdash \Delta}\ \neg\text{L}$$

$$\frac{\Gamma, \phi \vdash \Delta \quad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \vee \psi \vdash \Delta}\ \vee\text{L} \qquad \frac{\Gamma \vdash \phi, \psi, \Delta}{\Gamma \vdash \phi \vee \psi, \Delta}\ \vee\text{R} \qquad \frac{\Gamma, \phi \vdash \Delta}{\Gamma \vdash \neg\phi, \Delta}\ \neg\text{R}$$

$$\frac{\Gamma, \forall x.\phi, \phi[x/X] \vdash \Delta}{\Gamma, \forall x.\phi \vdash \Delta}\ \forall\text{L} \qquad \frac{\Gamma \vdash \phi[x/c], \Delta}{\Gamma \vdash \forall x.\phi, \Delta}\ \forall\text{R} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, s \approx s \vdash \Delta}\ \approx\text{ELIM}$$

$$\frac{\Gamma, \phi[x/c] \vdash \Delta}{\Gamma, \exists x.\phi \vdash \Delta}\ \exists\text{L} \qquad \frac{\Gamma \vdash \exists x.\phi, \phi[x/X], \Delta}{\Gamma \vdash \exists x.\phi, \Delta}\ \exists\text{R} \qquad \frac{*}{\Gamma \vdash s \approx s, \Delta}\ \approx\text{CLOSE}$$

$$\frac{\Gamma, t \approx s \vdash \Delta}{\Gamma, s \approx t \vdash \Delta}\ \approx\text{ORIENT} \qquad \text{where } t \succ s$$

$$\frac{\Gamma, t \approx s, (t' \approx s')[t/s] \vdash \Delta}{\Gamma, t \approx s, t' \approx s' \vdash \Delta}\ \approx\text{L}$$

where $t \succ s$ and $t' \succ s'$, the term t occurs in $t' \approx s'$, and if $t = t'$ then $s' \succ s$

$$\frac{\Gamma, t \approx s \vdash (t' \approx s')[t/s], \Delta}{\Gamma, t \approx s \vdash t' \approx s', \Delta}\ \approx\text{R}$$

where $t \succ s$ and the term t occurs in $t' \approx s'$

$$\frac{\dfrac{*}{\Gamma_1 \vdash \Delta_1}\ \cdots\ \dfrac{*}{\Gamma_n \vdash \Delta_n}\ \text{BREU}}{\Gamma \vdash \Delta}$$

where $\Gamma_1 \vdash \Delta_1, \ldots, \Gamma_n \vdash \Delta_n$ are all open goals of the proof, $E_i = \{t \approx s \in \Gamma_i\}$ are flat antecedent equations, $e_i = \bigvee\{t \approx s \in \Delta_i\}$ are succedent equations, and the simultaneous BREU problem $(\preceq, (E_i, e_i)_{i=1}^n)$ is solvable

and free variables reflecting the order in which symbols are introduced by the rules $\forall\text{L}$, $\forall\text{R}$, $\exists\text{L}$, $\exists\text{R}$: we define $s \prec t$ if the constant or variable t was introduced *above* the symbol s, or if s is a symbol already occurring in the root sequent and t is introduced by some rule in the proof. For instance, for the proof shown in Sect. 1.1, the partial order shown in Example 5 is derived.

By slight abuse of notation, we also write $s \prec f(t_1, \ldots, t_n)$ if s does not start with a function symbol. The rule $\approx$ORIENT moves the bigger term to the left-hand side of an equation. $\approx\text{L}$ and $\approx\text{R}$ can be used to replace occurrences of the (bigger) left-hand side term of an equation with the smaller right-hand side term; this rewriting is purely ground and does not unify expressions containing free variables (unification is entirely left to the BREU closure rule discussed in the next paragraph). As a consequence, and since $\prec$ is well-founded, rewriting is terminating and confluent, and in fact implements a congruence closure procedure [1] that eventually replaces every term with a unique representative term of its equivalence class modulo equations in the antecedent.

The BREU rule operates globally and closes all remaining goals of a proof if a global E-unifier σ exists that solves some succedent equation in each goal. The rule makes use of the non-strict partial order $\preceq$ corresponding to $\prec$, with the implication that every variable X can be mapped to symbols that were

introduced prior to X in the proof. To encode non-emptiness of the universe, we assume that there is some constant $c_\bot \in C$ below all variables $X \in V$ in a proof ($c_\bot \prec X$ for all $X \in V$); if the proof itself does not contain such a constant, it is assumed that $c_\bot$ is some fresh constant with $c_\bot \prec X$ for all variables X.

5 Properties of the Calculus

5.1 Soundness

The soundness of the calculus from Table 1 can be shown by substituting constants for all free variables, and observing the local soundness of each rule.

Lemma 8. *Suppose $\Gamma \vdash \Delta$ is a sequent without free variables. If a closed proof can be constructed for $\Gamma \vdash \Delta$ using the calculus in Table 1, then $\Gamma \vdash \Delta$ is valid.*

Proof. We assume that a proof for $\Gamma \vdash \Delta$ was closed using rule BREU, with a unifier σ that maps every variable X occurring in the proof to a constant $X\sigma \in C$ with $X\sigma \prec X$. In case all goals were closed using $\approx$CLOSE, $X\sigma$ can be some arbitrary constant with $X\sigma \prec X$.

By induction, it can be shown that the instance $(\Gamma' \vdash \Delta')\sigma = \Gamma'\sigma \vdash \Delta'\sigma$ of every sequent $\Gamma' \vdash \Delta'$ occurring in the proof is valid. This is the case for every goal discharged using rule BREU by definition. For all other rules, it is the case that if the σ-instance of the premises is valid, then also the σ-instance of the conclusion is valid. We show two cases, the other rules are verified similarly:

- $\exists$L: assume that the instantiated premise $(\Gamma, \phi[x/c] \vdash \Delta)\sigma$ is valid. Since c is fresh, we know that $X \prec c$ for all free variables X in $\Gamma, \exists x.\phi \vdash \Delta$. Therefore $X\sigma \prec c$, and it follows that $(\Gamma, \exists x.\phi \vdash \Delta)\sigma$ does not contain c. Validity of $(\Gamma, \phi[x/c] \vdash \Delta)\sigma$ then implies validity of $\forall x.(\bigwedge \Gamma \wedge \phi \to \bigvee \Delta)\sigma$, and equivalently of $(\Gamma, \exists x.\phi \vdash \Delta)\sigma$.
- $\approx$L: assume that $(\Gamma, t \approx s, (t' \approx s')[t/s] \vdash \Delta)\sigma$ is valid. Then the conclusion $(\Gamma, t \approx s, t' \approx s' \vdash \Delta)\sigma$ is valid, too, since the conjunctions $(t \approx s \wedge t' \approx s')\sigma$ and $(t \approx s \wedge (t' \approx s')[t/s])\sigma$ are equivalent.

Since the root sequent $\Gamma \vdash \Delta$ does not contain any free variables, it is implied that $(\Gamma \vdash \Delta)\sigma = \Gamma \vdash \Delta$ is valid. □

5.2 Completeness

The completeness of the calculus can be shown using a model construction argument (e.g., [8]), which also implies that every attempt to construct a proof of a valid sequent in a "fair" manner will ultimately be successful; this ensures that proofs can always be found without the need for backtracking (although backtracking might sometimes lead to success more quickly, of course).

We call a proof search strategy for the calculus in Table 1 *fair* if the propositional and first-order rules $\wedge$L, $\wedge$R, $\vee$L, $\vee$R, $\neg$L, $\neg$R, $\forall$L, $\forall$R, $\exists$L, $\exists$R are always eventually applied when they are applicable to some formula, and if every proof

goal in which one of those rules is applicable is eventually expanded. This implies, in particular, that $\forall$L and $\exists$R are applied unboundedly often to every quantified formula. Fairness does not mandate the application of the equational rules, which are subsumed by BREU; eager application of equational rules is in practice cheap and advisable for performance, however.

Lemma 9 (Completeness of fair proof search). *Suppose $\Gamma \vdash \Delta$ is a sequent without free variables, and suppose that a proof is constructed in a fair manner. If $\Gamma \vdash \Delta$ is valid, then eventually a proof tree will be obtained that can be closed using the rule* BREU.

In order to prove this lemma, we first consider a "ground" version GC of our calculus, obtained by removing the rule BREU, and by replacing $\forall$L and $\exists$R with the following ground rules:

$$\frac{\Gamma, \forall x.\phi, \phi[x/c] \;\vdash\; \Delta}{\Gamma, \forall x.\phi \;\vdash\; \Delta}\;\forall \mathrm{L}_g\,, \qquad \frac{\Gamma \;\vdash\; \exists x.\phi, \phi[x/c], \Delta}{\Gamma \;\vdash\; \exists x.\phi, \Delta}\;\exists \mathrm{R}_g$$

where c is an arbitrary constant. GC has the property that systematic application of the rules will either eventually produce a closed proof, or lead to a *saturated* (possibly infinite) branch from which a model can be derived:

Definition 10. *An open proof branch in GC labelled with sequents $\Gamma_0 \vdash \Delta_0$, $\Gamma_1 \vdash \Delta_1, \ldots$ (where $\Gamma_0 \vdash \Delta_0$ is the root of the proof) is called* saturated *if*

(i) the branch is finite and no rule is applicable in the goal sequent $\Gamma_n \vdash \Delta_n$; or
(ii) the branch is infinite, and for the limit sets $\Gamma^\infty, \Delta^\infty$ of formulae occurring on the branch, as well as the sets Γ^p, Δ^p of persistent *formulae*

$$\Gamma^\infty = \bigcup_{i\geq 0} \Gamma_i\,, \quad \Delta^\infty = \bigcup_{i\geq 0} \Delta_i\,, \quad \Gamma_p = \bigcup_{i\geq 0} \bigcap_{j\geq i} \Gamma_j\,, \quad \Delta_p = \bigcup_{i\geq 0} \bigcap_{j\geq i} \Delta_j$$

it is the case that (a) Γ_p only contains equations and $\forall$-quantified formulae; (b) Δ_p only contains equations and $\exists$-quantified formulae; (c) none of the rules $\approx$ELIM, $\approx$CLOSE, $\approx$ORIENT, $\approx$L, $\approx$R is applicable in $\Gamma_p \vdash \Delta_p$; (d) at least one constant c occurs on the branch; (e) for every formula $\forall x.\phi \in \Gamma_p$ and every constant c occurring on the branch, there is an instance $\phi[x/c] \in \Gamma^\infty$; and (f) for every formula $\exists x.\phi \in \Delta_p$ and every constant c there is an instance $\phi[x/c] \in \Delta^\infty$.

The ability to construct saturated branches follows directly from the observation that application of the GC-rules other than $\forall \mathrm{L}_g$ and $\exists \mathrm{L}_g$ terminates (because $\prec$ is well-founded), and that $\forall \mathrm{L}_g$ and $\exists \mathrm{L}_g$ can be managed in a fair way using a work queue. The property (ii)–(d) encodes non-emptiness of universes, and is ensured by instantiating every formula $\forall x.\phi \in \Gamma_p$ and $\exists x.\phi \in \Delta_p$ at least once on every branch (e.g., using the $\prec$-smallest constant $c_\bot$).

Lemma 11. *If a (finite or infinite) GC proof contains a saturated branch, then the root sequent $\Gamma \vdash \Delta$ has a counter-model (is invalid).*

Proof. We use persistent equations to construct a structure $S = (U, I)$. In case of a finite saturated branch, persistent formulae are the ones in the goal; without loss of generality, we assume that also finite branches contain at least one constant. U is chosen as the set of constants that do not occur as left-hand side of some persistent antecedent equation; left-hand side terms are interpreted as the right-hand side constants. In case the value of some function application $f(c_1, \ldots, c_n)$ is not determined by the equations, we set the value to some arbitrary constant $c \in U$:

$$\begin{aligned} U &= \{c \in C \mid c \text{ occurs in } \Gamma^\infty \cup \Delta^\infty\} \setminus \{c \mid c \approx d \in \Gamma_p\} \\ I(c) &= \begin{cases} d & \text{if there exists an equation } c \approx d \in \Gamma_p \\ c & \text{otherwise} \end{cases} \\ I(f)(c_1, \ldots, c_n) &= \begin{cases} d & \text{if there exists an equation } f(c_1, \ldots, c_n) \approx d \in \Gamma_p \\ c & \text{otherwise, for some arbitrary } c \in U \end{cases} \end{aligned}$$

Since no equational rule is applicable in $\Gamma_p \vdash \Delta_p$, it is clear that $\mathrm{val}_S(t \approx s) = \mathit{true}$ for every $t \approx s \in \Gamma_p$, and $\mathrm{val}_S(t \approx s) = \mathit{false}$ for every $t \approx s \in \Delta_p$.

By well-founded induction over the equations in Γ^∞, it can then be shown that in fact *all* equations in Γ^∞ evaluate to *true* under S. For this we define a well-founded order $\prec'$ over flat equations (for $c, d \in C$, $\bar{c}, \bar{c}' \in C^*$, $f, g \in F$, and $\prec_{\mathrm{lex}}$ the well-founded lexicographic order induced by $\prec$):

$$\begin{gathered} (c \approx d) \prec' (c' \approx d') \;\Leftrightarrow\; (d, c) \prec_{\mathrm{lex}} (d', c'), \qquad (c \approx d) \prec' (f(\bar{c}) \approx d'), \\ (f(\bar{c}) \approx d) \prec' (g(\bar{c}') \approx d') \;\Leftrightarrow\; f = g \text{ and } (d, \bar{c}) \prec_{\mathrm{lex}} (d', \bar{c}'). \end{gathered}$$

In particular, note that in any application of rule $\approx$L we have $(t \approx s) \prec' (t' \approx s')$ and $(t' \approx s')[t/s] \prec' (t' \approx s')$; this implies that if all equations $\prec'$-smaller than $t' \approx s'$ hold, then also $t' \approx s'$ holds. In the same way, it can be proven that all equations in Δ^∞ evaluate to *false*.

By induction over the depth of formulae we can conclude that all formulae (not only equations) in Γ^∞ evaluate to *true*, and all formulae in Δ^∞ to *false*. □

Proof (Lem. 9). Assume that an (unsuccessful) attempt was made to construct a proof P for the valid sequent $\Gamma \vdash \Delta$ by fair application of the rules in Table 1. We define a global mapping $v : V \to C$ of variables occurring in P to constants, and use v to map P to a *GC*-proof with a saturated branch. The mapping v is defined successively by depth-first traversal of P, visiting sequents closer to the root earlier than sequents further away. Note that for each branch that has not been closed by applying $\approx$CLOSE, fairness implies that $\forall$L ($\exists$R) has been applied infinitely often to every universally quantified formula in the antecedent (existentially quantified formula in the succedent).

When a node is visited where a new variable X is introduced by $\forall$L or $\exists$R for a quantified formula ϕ, set $v(X) = c$ for some constant $c \prec X$ that is $\prec$-minimal among the constants that have *not yet* been assigned for the same formula ϕ on this branch. If no such constant exists, an arbitrary constant $c \prec X$ is chosen. On

every infinite branch, this ensures that for every quantified formula ϕ handled via $\forall$L or $\exists$R, and every constant c occurring on the branch, there is some application of $\forall$L or $\exists$R to ϕ such that the introduced variable X is mapped to $c = v(X)$.

The function v can then be used to translate P to a GC-proof P', replacing each variable X with the constant $v(X)$, and inserting exhaustive applications of the equational rules wherever they are applicable. By Lem. 11 and since $\Gamma \vdash \Delta$ is valid, each branch in P' can be closed after finitely many steps through $\approx$CLOSE. This implies that it has to be possible to close the corresponding finite prefix of the original proof P using rule BREU, with the mapping v restricted to the variables occurring in the prefix as E-unifier. □

6 Refinements of the Calculus

The presented calculus can be refined in many practically relevant ways; in the scope of this paper, we only outline three modifications that we use in our implementation (also see Sect. 7).

General instantiation. Similar the subterm instantiation method proposed by Kanger [13], our system explicitly generates constants representing all terms possibly required for instantiation of quantified formulae, through application of $\exists$L and $\forall$R. While subterm instantiation is complete, it has been observed (e.g., in [6]) that resulting proofs can sometimes be significantly longer than the shortest proofs that can be obtained when considering arbitrary instances of quantified formulae. Instantiation with new terms can be simulated in our systems by adding a rule TOT representing the totality axiom $\forall \bar{x}.\exists y.\ f(\bar{x}) \approx y$, which iteratively increases the range of terms considered for substitution by the BREU rule. In TOT, f is a function symbol, $X_1, \ldots, X_n$ are fresh variables, and c is a fresh constant (and we set $X_i \prec c$ for all $i \in \{1, \ldots, n\}$):

$$\frac{\Gamma, f(X_1, \ldots, X_n) \approx c \ \vdash\ \Delta}{\Gamma \ \vdash\ \Delta}\ \text{TOT}$$

Local closure. The closure rule BREU can be generalised to operate not only on complete proof trees, but also on arbitrary *sub-trees,* and thus be used to guide proof expansion. For any sub-tree t, it can be checked (i) whether all goals in t contain equations that are simultaneously E-unifiable; as long as this is not the case, proof expansion can focus on t, since rules applied to branches outside of t will not be helpful; and (ii) whether the goals in t are E-unifiable with a unifier σ such that $X\sigma = X$ for all variables X that occur outside of t; in this case, t can be closed permanently and does not have to be considered again. It is also possible to define a notion of *unsatisfiable cores* for E-unification problems, which can further refine the selection of goals to be expanded.

Ground instantiation. It has also been observed that handling of quantifiers using free variables is very powerful, but is excessively expensive in case of simple

Table 2. Comparison of our prototypical implementation on TPTP benchmarks. The numbers indicate how many benchmarks in each group could be solved; the runtime per benchmark was limited to 240s (wall clock time). All experiments were done on an AMD Opteron 2220 SE machine, running 64-bit Linux, heap space limited to 1.5GB.

	FOF with eq.	**FOF w/o eq.**	**CNF with eq.**	**CNF w/o eq.**
Princess + BREU	**211**	325	**203**	252
Hyper 1.0_16112014 [2]	119	378	160	**305**
leanCoP 2.2 (CASC-J7)	153	**379**	–[1]	–[1]

quantified formulae that have to be instantiated many times, and provides little guidance for proof construction. Possible solutions include the use of connection conditions, universal variables, or simplification rules [3, 12]. In our implementation, we use a more straightforward hybrid approach that combines free variables with ground instantiation through *E-matching* [15]; in combination, free variables and e-matching can solve significantly more problems than either technique individually. E-matching can be integrated naturally in our calculus without losing completeness, following [15]; in general this requires the use of the rule TOT shown above.

7 Experimental Results

We are in the process of implementing our BREU algorithm, and the calculus from Sect. 4, as an extension of the Princess theorem prover [14].[2] Our implementation uses the SAT encoding outlined in Sect. 3.3, and the Sat4j solver to solve the resulting constraints; we also include the refinements discussed in Sect. 6. Considered benchmarks were randomly selected TPTP v.6.1.0 problems with status Theorem or Unsatisfiable. To illustrate strengths and weaknesses of the compared tools, the benchmarks were categorised into FOF (first-order) problems with equality, FOF problems without equality, CNF (clause normal form) problems with equality, and CNF problems without equality. 500 benchmarks from all of TPTP were chosen in each group.

We compared our BREU implementation with the tableau provers Hyper and leanCoP from the CASC-J7 competition. Hyper uses the superposition-based equality reasoning from [2], whereas leanCoP relies on explicit equality axioms. The experimental results shown in Table 2 are still preliminary, and expected to change as further optimisations in our BREU procedure are done. However, it can be seen that even our current implementation of BREU shows performance that is comparable with the other tableau systems in all groups of benchmarks, and outperforms the other systems on benchmarks with equality.

[1] leanCoP cannot process benchmarks in the TPTP CNF dialect.

[2] `http://user.it.uu.se/~petba168/breu/`

Conclusion

We have introduced bounded rigid E-unification, a new variant of SREU, and illustrated how it can be used to construct sound and complete theorem provers for first-order logic with equality. We believe that BREU is a promising approach to handling of equality in tableaux and related calculi. Apart from improved algorithms for solving BREU, and an improved implementation, in future work we plan to consider the combination of BREU with other theories, in particular arithmetic, and integration of BREU with DPLL(T)-style clause learning.

Acknowledgements We would like to thank Christoph M. Wintersteiger for comments on this paper, and the anonymous referees for helpful feedback.

References

1. Bachmair, L., Tiwari, A., Vigneron, L.: Abstract congruence closure. J. Autom. Reasoning 31(2), 129–168 (2003)
2. Baumgartner, P., Furbach, U., Pelzer, B.: Hyper tableaux with equality. In: Pfenning, F. (ed.) CADE. LNCS, vol. 4603, pp. 492–507. Springer (2007)
3. Beckert, B.: Equality and other theories. In: D'Agostino, M., Gabbay, D., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods. Kluwer, Dordrecht (1999)
4. Degtyarev, A., Voronkov, A.: Simultaneous rigid E-Unification is undecidable. In: Büning, H.K. (ed.) CSL. LNCS, vol. 1092, pp. 178–190. Springer (1995)
5. Degtyarev, A., Voronkov, A.: What you always wanted to know about rigid E-Unification. J. Autom. Reasoning 20(1), 47–80 (1998)
6. Degtyarev, A., Voronkov, A.: Equality reasoning in sequent-based calculi. In: Handbook of Automated Reasoning (in 2 volumes). Elsevier and MIT Press (2001)
7. Degtyarev, A., Voronkov, A.: Kanger's Choices in Automated Reasoning. Springer (2001)
8. Fitting, M.C.: First-Order Logic and Automated Theorem Proving. Graduate Texts in Computer Science, Springer-Verlag, Berlin, 2nd edn. (1996)
9. Gallier, J.H., Raatz, S., Snyder, W.: Theorem proving using rigid e-unification equational matings. In: LICS. pp. 338–346. IEEE Computer Society (1987)
10. Giese, M.: Incremental closure of free variable tableaux. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR. LNCS, vol. 2083, pp. 545–560. Springer (2001)
11. Giese, M.: A model generation style completeness proof for constraint tableaux with superposition. In: Tableaux. LNCS, vol. 2381, pp. 130–144. Springer (2002)
12. Giese, M.: Simplification rules for constrained formula tableaux. In: TABLEAUX. pp. 65–80 (2003)
13. Kanger, S.: A simplified proof method for elementary logic. In: Siekmann, J., Wrightson, G. (eds.) Automation of Reasoning 1: Classical Papers on Computational Logic 1957-1966, pp. 364–371. Springer, Berlin, Heidelberg (1983), originally appeared in 1963
14. Rümmer, P.: A constraint sequent calculus for first-order logic with linear integer arithmetic. In: LPAR. LNCS, Springer (2008)
15. Rümmer, P.: E-Matching with free variables. In: LPAR. LNCS, vol. 7180, pp. 359–374. Springer (2012)
16. Tiwari, A., Bachmair, L., Rueß, H.: Rigid E-Unification revisited. In: CADE. pp. 220–234. CADE-17, Springer-Verlag, London, UK, UK (2000)