Combined Abstract Congruence Closure for Theories with Associativity or Commutativity*

Christophe Ringeissen and Laurent Vigneron

Université de Lorraine, CNRS, Inria, Loria, 54000 Nancy, France
Christophe.Ringeissen@loria.fr, Laurent.Vigneron@loria.fr

Abstract. Formal verification techniques, such as deductive verification, require a logic-based tool support to discharge proof obligations, in other words, satisfiability procedures. We design satisfiability procedures thanks to congruence closure methods applied to unions of axiomatized theories, targeting equational axioms such as Associativity or Commutativity. In the proposed approach, any function symbol can be uninterpreted, associative only, commutative only, but also associative and commutative. To tackle the union of these theories, we introduce a combined congruence closure procedure that can be viewed as a particular Nelson-Oppen combination method using particular congruence closure procedures for the individual theories. In this context, we consider terminating congruence closure procedures, but also non-terminating ones. Hence, we have terminating ones for Commutativity and Associativity-Commutativity, while the one for Associativity is non-terminating. We show how all the congruence closure procedures, including the combined one, can be presented in a uniform and abstract way.

Keywords: Satisfiability Procedure · Congruence Closure · Commutativity · Associativity · Union of Theories

1 Introduction

Formal verification techniques, such as deductive verification, require a logic-based tool support to discharge proof obligations, in other words, decision procedures to check whether a logic formula is satisfiable or not. Satisfiability Modulo Theories [5] (SMT, for short) is nowadays a very active research field, where many efficient SMT solvers are developed concurrently following a common specification format, the SMT-Lib. In this context, several theories are very useful in practice, such as the theory of equality, also called the empty theory or the theory of uninterpreted function symbols. This particular theory admits an efficient satisfiability procedure based on the construction of a congruence closure [14,15].

For sake of expressiveness, it is highly desirable to cope with function symbols that are interpreted in some given theory. For instance, this given theory

* Authors' version of a paper (DOI) published in *Logic-Based Program Synthesis and Transformation - 34th International Symposium, LOPSTR 2024, Milan, Italy, Sept. 9-10, 2024, Proceedings. Lecture Notes in Computer Science 14919, Springer.*

could be a large fragment of arithmetic, such as Linear Rational Arithmetic. But this theory could be also specified by some equational axioms. For example, Associativity-Commutativity is particularly interesting to reason modulo (multi)sets, Associativity allows us to reason modulo lists, and Commutativity is useful to reason modulo a symmetric relation.

Compared to Associativity-Commutativity and Commutativity where unification is finitary, Associativity is a challenging theory since associative unification is known to be infinitary. Despite the lack of an associative unification algorithm, it is possible to get useful procedures to reason modulo Associativity, as shown in [12] for the variant-based unification and satisfiability problems. Analogously, we believe that Associativity is a theory deserving to be investigated in the context of a combined congruence closure procedure.

We design rule-based congruence closure procedures modulo union of axiomatized theories, targeting equational axioms such as Associativity or Commutativity. In the proposed approach, any function symbol can be uninterpreted, associative only, flat permutative only (e.g. commutative), but also associative and commutative. To tackle the union of these theories, we introduce a combined congruence closure procedure that can be viewed as a particular Nelson-Oppen combination method [13] using particular congruence closure procedures for the individual theories. The combined congruence procedure is based on the ping-ponging of entailed equalities between (shared) constants. Actually, the congruence closure procedures used for the individual theories allow us to deduce these equalities. In this context, we consider terminating congruence closure procedures, but also non-terminating ones. Hence, we have terminating congruence closure procedures for flat permutation and Associativity-Commutativity, while the one for Associativity is non-terminating. We show how all the congruence closure procedures, including the combined one, can be presented in a uniform and abstract way along the lines of [3,8,9,10,11].

Related Work. Congruence closure modulo Associativity-Commutativity has been successfully investigated in [3,8]. It has been revisited more recently, showing how the method can be extended to take into account additional orientable axioms, for instance to handle the theory of Abelian Groups [9]. The case of flat permutative axioms, such as Commutativity, has been considered in [11]. The theory of Groups and all of its subtheories including Associativity is considered in [10], where the related congruence closure procedure is not necessarily terminating, contrarily to the one known for Associativity-Commutativity. In these papers, some particular unions of theories are studied, for instance to handle several symbols following the same equational axioms.

In our paper, we clearly focus on the combination of congruence closure procedures to cope with arbitrary unions of theories (sharing only constants). This combination of congruence closure procedures can be seen as a particular case of combination of deduction-complete satisfiability procedures, already investigated in [20]. In addition to Associativity-Commutativity, we believe that it is interesting to consider Associativity alone and flat permutation alone. On one hand, Associativity provides a significant case study of a non-terminating congruence

closure procedure. On the other hand, flat permutation (such as Commutativity) leads to a simple extension of the congruence closure procedure known for the theory of equality as done in [11].

Paper Outline. The paper is organized as follows. Section 2 introduces the main notions and notations related to terms, term rewriting and satisfiability problems. In Section 3, we describe our combination method using two kinds of processes: the orchestrator whose role is to prepare and handle a combination of theories; a theory process whose role is to complete the set of rewrite rules for a specific theory. The supported theories are detailed in Section 4. The completeness of the method is discussed in Section 5. The method is not necessarily terminating, as shown in Section 6 in the case of Associativity. In Section 7, we detail a running example involving an associative-commutative operator and a commutative one. Eventually, Section 8 concludes discussing some possible future developments.

2 Preliminaries

We assume the reader familiar with the notions of terms and term rewriting [1].

We consider n theories $\mathcal{E}_1, \ldots, \mathcal{E}_n$ such that each theory $\mathcal{E}_i$ is defined by a set of equalities over a signature Σ_i. [1] The theories $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are assumed to be pairwise signature-disjoint or to share only constants, meaning that $\Sigma_i \cap \Sigma_j$ is a set of constants which can be empty, for any $i, j \in [1, n], i \neq j$. The union of theories $\mathcal{E}_1 \cup \cdots \cup \mathcal{E}_n$ is denoted by $\mathcal{E}$ and the union of signatures $\Sigma_1 \cup \cdots \cup \Sigma_n$ is denoted by Σ. We assume a set of ground equalities Γ and a set of ground disequalities Δ, where both Γ and Δ are expressed over the signature Σ. Given any theory $\mathcal{E}'$, $=_{\mathcal{E}'}$ denotes the congruence closure of the equalities in $\mathcal{E}'$ under the law of substitutivity.

Terms. We denote $T(\Sigma)$ the set of *ground* (without variables) terms built over the signature Σ. A *constant* is a function symbol of Σ with arity 0. The set of positions in a term t is written $O(t)$. The subterm of a term t at position p is written $t|_p$. A subterm of t is *strict* if its position is not at the top of t. The term obtained from t by replacing $t|_p$ by a term s is written $t[s]_p$.
The process described in this paper relies on a flattening of terms. For theory $\mathcal{E}_i$ including an operator, say $+$, such that $(x + y) + z \approx x + (y + z)$ occurs in $\mathcal{E}_i$, this flattening will be performed using $+$ as a variadic operator, e.g. $a + (b + c)$ is flattened into $+(a, b, c)$.

Equalities and Rewrite Rules. The initial set of ground equalities Γ will be purified via flattening thanks to the introduction of new constants as in [7] (K denotes the set of used new constants taken from an infinite countable set U disjoint from Σ), generating pure flat rewrite rules for each theory $\mathcal{E}_i$ (denoted by the set R_i); and further deductions between those rules may generate flat

[1] The empty theory is defined by a signature, but with an empty set of equalities.

equalities in this theory (denoted by the set E_i).
The rewrite rules in R_i can have two shapes: *D-rules* denoted by $f(c_1, \ldots, c_n) \to c$, where $f \in \Sigma_i$ and $c_1, \ldots, c_n, c \in K$; *E-rules* denoted by $f(c_1, \ldots, c_m) \to f(d_1, \ldots, d_n)$, where $f \in \Sigma_i$ is a variadic operator and $c_1, \ldots, c_m, d_1, \ldots, d_n \in K$. The equalities in E_i have the same two shapes: *D-equalities* denoted by $f(c_1, \ldots, c_n) \approx c$; *E-equalities* denoted by $f(c_1, \ldots, c_m) \approx f(d_1, \ldots, d_n)$. An equality $c_1 \approx c_2$ between two constants of K is called a *C-equality*. E-rules and E-equalities will be generated only for variadic operators by the Superposition inference rule, because of the use of extended rewrite rules (see Section 3.2).
Considering a theory $\mathcal{E}_i$, if two terms t_1 and t_2 are $\mathcal{E}_i$-equal, we write $t_1 \leftrightarrow^*_{\mathcal{E}_i} t_2$. By $(R_i, \mathcal{E}_i)$ we denote the rewriting system defined by $\{u' \to v \mid u \to v \in R_i \text{ and } u' \leftrightarrow^*_{\mathcal{E}_i} u\}$. And by $(R_i^e, \mathcal{E}_i)$ we denote the rewriting system extending $(R_i, \mathcal{E}_i)$ with all possible extended rewrite rules from R_i.

Ordering. For any rewrite rule $t \to s$, t has to be greater than s ($t \succ s$); the definition of an ordering may be difficult for deduction systems modulo equational theories; but in our case the ordering is very simple as we only have to consider D-rules and E-rules: for D-rules, it suffices to assume $\forall f \in \Sigma, \forall c \in K, f \succ c$; for E-rules, we have to compare lists of constants: if of the same length, this can be done with a lexicographic or a multiset extension of an arbitrary ordering comparing two constants of K (the choice is done for each theory), and if of different length, the longest is the greatest. For example, for an associative theory the lexicographic extension will be used, and for an associative-commutative theory the multiset extension will be used.
The *lexicographic extension* of an ordering $\succ$ is defined by: $(t_1, \ldots, t_n) \succ^{lex} (s_1, \ldots, s_n)$ if $\exists i \in [1, n], \forall j < i, t_j =_{\mathcal{E}} s_j$ and $t_i \succ s_i$.
The *multiset extension* of an ordering $\succ$ is defined by: $T = \{t_1, \ldots, t_n\} \succ^{mult} \{s_1, \ldots, s_n\} = S$ if, given I the multiset $\mathcal{E}$-intersection of T and S (where the equality is checked w.r.t. $=_{\mathcal{E}}$), $\forall s_i \in S \backslash I, \exists t_j \in T \backslash I, t_j \succ s_i$.

Satisfiability Problems. In the context of this paper, a satisfiability problem is given by a finite set of ground literals, where a literal is either an equality or a disequality. Given a theory T, a satisfiability problem φ expressed over the signature of T is said to be *T-satisfiable* if $T \cup \varphi$ is consistent, i.e. $T \cup \varphi$ admits a model. We say that a model is *trivial* when its domain is of cardinality 1. A consistent theory is *trivial* when all its models are trivial. Notice that any equational theory is consistent since it admits a trivial model. The equational theories considered in this paper are non-trivial.

3 Combined Satisfiability Procedure

We describe in this section a procedure that aims at (semi-)deciding the satisfiability of any set of ground equalities Γ together with any set of ground disequalities Δ, modulo a combination of signature-disjoint equational theories $\mathcal{E}_i$ (or sharing only constants). This procedure, called `CombCC`, is based on congruence

closure and involves two kinds of processes: an orchestrator decomposing the problem to separate the different theories, and theory processes that complete rewrite rules, one process for each theory.

3.1 The Orchestrator

The role of the orchestrator is to purify and flatten the problem to be solved, to send each theory process the rewrite rules it has to handle, and to detect if any contradiction w.r.t. Δ is generated by one of the theory processes. For this purpose, several sets are handled in addition to Γ and Δ: the set of new constants K, and for each equational theory $\mathcal{E}_i$ a set of rewrite rules R_i and a set of equalities E_i. We denote RE the set of (R_i, E_i) for all theories $\mathcal{E}_i$. In the following inference rules, we will only indicate the involved sets.

The first task of the orchestrator is to transform the disequalities for "hiding" the theories involved. This is done with the following inference rule that replaces an arbitrary disequality by a disequality between two new constants together with the equalities associating each of these constants to the corresponding term:

Splitting: $$\frac{K, \Delta \cup \{t_1 \not\approx t_2\}, \Gamma}{K \cup \{c_1, c_2\}, \Delta \cup \{c_1 \not\approx c_2\}, \Gamma \cup \{t_1 \approx c_1, t_2 \approx c_2\}}$$

if $t_1, t_2 \notin K$, $c_1, c_2 \in U \setminus K$.

Once all disequalities have been decomposed, the second task of the orchestrator is to purify the equalities of Γ, by generating rewrite rules that are purely in one theory. For this purpose, it applies the following inference rules:

Flattening: $$\frac{K, \Gamma[t], R_i}{K \cup \{c\}, \Gamma[c], R_i \cup \{t \to c\}}$$

if $t \to c$ is a D-rule, $c \in U \setminus K$, t occurs in some equality in Γ that is not a D-equality, and t is Σ_i-rooted.

Orientation: $$\frac{K \cup \{c\}, \Gamma \cup \{t \approx c\}, R_i}{K \cup \{c\}, \Gamma, R_i \cup \{t \to c\}}$$ if $t \approx c$ is a D-equality and t is Σ_i-rooted.

When all equations have been transformed ($\Gamma = \emptyset$), the orchestrator runs one process per equational theory $\mathcal{E}_i$, providing it two sets of information: the set of new constants K and the set of D-rules R_i defined over Σ_i and K.

Its final task is to manage equalities between new constants, when generated by a theory process in some set E_i; there are two possibilities: if the equality contradicts a disequality of Δ then the system stops, otherwise one of the constants has to be replaced by the other one in all sets.

Contradiction: $$\frac{K \cup \{c, d\}, \Delta \cup \{c \not\approx d\}, RE \cup \{(R_i, E_i \cup \{c \approx d\})\}}{\bot}$$

Compression: $$\frac{K \cup \{c, d\}, \Delta, RE \cup \{(R_i, E_i \cup \{c \approx d\})\}}{K \cup \{d\}, \Delta\langle c \mapsto d\rangle, RE\langle c \mapsto d\rangle \cup \{(R_i\langle c \mapsto d\rangle, E_i\langle c \mapsto d\rangle)\}}$$

if $c \approx d \notin \Delta$ and $c \succ d$; the notation $\langle c \mapsto d \rangle$ denotes the homomorphic extension of the mapping σ defined as $\sigma(c) = d$ and $\sigma(x) = x$ for $x \neq c$, and $S\langle c \mapsto d \rangle$ denotes the set obtained by applying the mapping $\langle c \mapsto d \rangle$ to each term in set S.

The strategy of the orchestrator can therefore be described by: $\mathbf{Split}^* \cdot (\mathbf{Flat}^* \cdot \mathbf{Ori})^* \cdot (\mathbf{Cont}|\mathbf{Comp})^*$.

3.2 A Theory Process

A process run for an equational theory $\mathcal{E}_i$ will use inference rules to complete its term rewriting system R_i. Some inference rules are used for transforming the rewrite rules (Composition), for deducing new equalities added to a set E_i (Collapse, Superposition), or for handling those new equalities (Simplification, Orientation, Deletion). The set of new constants K is never modified, so not indicated, but it is useful in the process for checking if a constant is a new one or belongs to the theory.

For some theories, the inference system has to consider extended rewrite rules as we do not explicitly use the axioms of a theory: an extension is built w.r.t. a context defined from the theory axioms; a context is a triplet: a term $Cont$, a non variable position p of a strict subterm in $Cont$, and a set of $\mathcal{E}_i$-unification constraints UC. Let us denote $Cont_{\mathcal{E}_i}$ the set of contexts for the theory $\mathcal{E}_i$; given a D-rule or a E-rule $u \to v$, its extended version by a context $(Cont, p, UC) \in Cont_{\mathcal{E}_i}$ is written $Cont[u]_p\sigma \to Cont[v]_p\sigma$, where σ is the ground substitution in a minimal complete set of $\mathcal{E}_i$-unifiers of $Cont|_p =^?_{\mathcal{E}_i} u$ and UC. These notions of extended rewrite rules and contexts were already mentioned in [17,16], and the construction of contexts for generating extensions has been explained in [21]. The principle is to start by identifying non variable positions in the left-hand and right-hand sides of axioms of the theory, giving an initial set of contexts. Then those contexts are combined by unification from one context into the marked position of another context, and so on; each combination adds unification constraints in the new context. A notion of redundant context has been defined for keeping only useful contexts: a redundant context will create extended rewrite rules that will generate by deduction either redundant equalities/rules, or equalities/rules that can be generated using already existing (simpler) contexts.

In this paper, as we want to handle only flat rewrite rules, we will consider only theories for which extended rewrite rules have a flat form. For example, if an operator f is associative, from the axiom of this theory $f(f(x,y),z) \approx f(x,f(y,z))$, we can build three contexts:

$$(f(f(y_1,y_2),x_1),1,\emptyset),\ (f(x_2,f(y_1,y_2)),2,\emptyset) \text{ and } (f(x_3,f(y_1,y_2),x_4),2,\emptyset)$$

In each one, the left-hand side of the rewrite rule to be extended will have to unify with $f(y_1,y_2)$. So, a rewrite rule $f(a,b) \to c$ has three extensions, written in flat form: $f(a,b,x_1) \to f(c,x_1)$, $f(x_2,a,b) \to f(x_2,c)$ and $f(x_3,a,b,x_4) \to$

$f(x_3, c, x_4)$.

The inference rules used by a theory process are the following.

Simplification: $\dfrac{R_i, E_i[t]}{R_i, E_i[s]}$ where t occurs in some equality of E_i, and $t \to_{(R_i^e, \mathcal{E}_i)} s$.

Orientation: $\dfrac{R_i, E_i \cup \{t \approx s\}}{R_i \cup \{t \to s\}, E_i}$ if $t \succ s$ and $t \to s$ is a D-rule or a E-rule.

Deletion: $\dfrac{R_i, E_i \cup \{t \approx s\}}{R_i, E_i}$ if $t \leftrightarrow^*_{\mathcal{E}_i} s$.

Composition: $\dfrac{R_i \cup \{t \to s, u \to v\}, E_i}{R_i \cup \{t \to s', u \to v\}, E_i}$ if $s \to_{(\{u \to v\}^e, \mathcal{E}_i)} s'$.

Collapse: $\dfrac{R_i \cup \{t \to s, u \to v\}, E_i}{R_i \cup \{u \to v\}, E_i \cup \{t' \approx s\}}$ if $t \to_{(\{u \to v\}^e, \mathcal{E}_i)} t'$, and if $t \leftrightarrow^*_{\mathcal{E}_i} u$ then $s \succ v$.

Superposition: $\dfrac{R_i \cup \{t_1 \to s_1, t_2 \to s_2\}, E_i}{R_i \cup \{t_1 \to s_1, t_2 \to s_2\}, E_i \cup \{Cont_1[s_1]_{p_1}\sigma \approx Cont_2[s_2]_{p_2}\sigma\}}$

if $(Cont_1, p_1, UC_1), (Cont_2, p_2, UC_2) \in Cont_{\mathcal{E}_i}$, [2] and the substitution σ is the ground substitution in a minimal complete set of $\mathcal{E}_i$-unifiers of $Cont_1[t_1]_{p_1} =^?_{\mathcal{E}_i} Cont_2[t_2]_{p_2}$, $Cont_1|_{p_1} =^?_{\mathcal{E}_i} t_1$, $Cont_2|_{p_2} =^?_{\mathcal{E}_i} t_2$, UC_1 and UC_2; the resulting equality will be written in flat form.

A strategy for combining all those inference rules is: $(\mathbf{Com}^* \cdot (\mathbf{Col}|\mathbf{Sup}) \cdot \mathbf{Sim}^* \cdot (\mathbf{Del}|\mathbf{Ori}))^*$
So this process handles a pair (R_i, E_i): it starts with $(R_i, \emptyset)$ and, if terminating, it ends with $(R_i^\infty, \emptyset)$ where there is no more possible inference rule involving rules of R_i^∞. If an equality between two constants of K is generated, it will be handled by the orchestrator.

For applying inference rules, this theory process has to use a $\mathcal{E}_i$-matching algorithm for applying rewriting steps w.r.t. $(R_i^e, \mathcal{E}_i)$. It also needs a $\mathcal{E}_i$-unification algorithm, but which will have to solve only simple unification problems, of the shape $Cont_1[t_1]_{p_1} =^?_{\mathcal{E}_i} Cont_2[t_2]_{p_2}$, where t_1 and t_2 are ground; if there is a solution, it will be the unique most general unifier since the variables occurring in $Cont_i[\cdot]_{p_i}$ will be instantiated by subterms of the ground term t_{3-i}.

4 Supported Theories

We detail now some of the input theories $\mathcal{E}_i$ that allow us to get a complete combined satisfiability procedure. We consider three kinds of theories (in addition to the empty theory of course), and discuss also about additional ones.

[2] Note that according to the theory $\mathcal{E}_i$, the contexts may be selected to guarantee a useful ground new equality (see Section 4.2).

4.1 Flat Permutative Theories

A flat permutative theory is represented by a set of axioms

$$\{f(x_1, \ldots, x_k) \approx f(x_{\sigma(1)}, \ldots, x_{\sigma(k)}) \mid \sigma \text{ is a permutation of } \{1, \ldots, k\}\}$$

where x_i are variables that do not need to be all distinct.
With such a theory, there is no extension of rewrite rules to be considered, so the Superposition inference rule cannot apply. For the ordering, we can compare the arguments of two f-terms with a lexicographic extension of the ordering between constants of K, and always use the biggest term of the equivalence class w.r.t. this theory.
Commutative theories are a particular case of flat permutative theories where all x_i are distinct variables and any permutation is possible; for the ordering we can use a multiset extension of the ordering between constants of K.

4.2 Associative Theories

An associative theory is represented by the axiom

$$f(f(x_1, x_2), x_3) \approx f(x_1, f(x_2, x_3))$$

This axiom generates three possible extensions of rewrite rules whose left-hand side is a f-term, with the contexts $(f(f(y_1, y_2), x_1), 1, \emptyset)$, $(f(x_2, f(y_1, y_2)), 2, \emptyset)$ and $(f(x_3, f(y_1, y_2), x_4), 2, \emptyset)$. Those three contexts can be used for applying term rewriting steps w.r.t. $(R_i^e, \mathcal{E}_i)$. But for the Superposition inference rule between two rules $t_1 \to s_1$ and $t_2 \to s_2$, we only need to consider their extensions $f(t_1, x_1) \to f(s_1, x_1)$ and $f(x_2, t_2) \to f(x_2, s_2)$ because this is the only combination of contexts for which the unification problem $(f(t_1, x_1) =^?_{\mathcal{E}_i} f(x_2, t_2))$ can generate a ground most general unifier, and any other combination of contexts would generate a redundant equation. For the ordering, the arguments of an associative operator are compared with a lexicographic extension of the ordering between constants of K.

4.3 Associative-Commutative Theories

An associative-commutative theory is represented by axioms

$$f(x_1, x_2) \approx f(x_2, x_1) \text{ and } f(f(x_1, x_2), x_3) \approx f(x_1, f(x_2, x_3))$$

It generates only one possible extension of rewrite rules whose left-hand side is a f-term, with the context $(f(f(y_1, y_2), x_1), 1, \emptyset)$, used for applying term rewriting steps w.r.t. $(R_i^e, \mathcal{E}_i)$, and the Superposition inference rule. For the ordering, the arguments of an associative-commutative operator are compared with a multiset extension of the ordering between constants of K.

4.4 Additional Theories

The theory process defined in the previous section is very general and can apply to many theories. We have given above three examples of theories that are often necessary to theorem provers or SMT solvers.

It is possible to define some extensions to those three theories by considering additional axioms that will be represented by a set of *flat collapsing* rewrite rules $R_{\mathcal{E}_i}$ which is confluent and terminating, and so that those axioms do not introduce new contexts w.r.t. $Cont_{\mathcal{E}_i}$, and extended rules from $R_{\mathcal{E}_i}$ are redundant: if $t \to^*_{(R^e_{\mathcal{E}_i}, \mathcal{E}_i)} s$ and s is in normal form w.r.t. $(R^e_{\mathcal{E}_i}, \mathcal{E}_i)$, then $t \to^*_{(R_{\mathcal{E}_i}, \mathcal{E}_i)} s$. The right-hand side of a rewrite rule of $R_{\mathcal{E}_i}$ is either a subterm of the left-hand side, or a constant. So their orientation from left to right does not need a specific ordering.

The building of $R_{\mathcal{E}_i}$ from the added axioms uses the Flattening and Orientation inference rules of the orchestrator; so, non-variable subterms in $R_{\mathcal{E}_i}$ are new constants of K and they may be concerned by the Compression inference rule. The use of $R_{\mathcal{E}_i}$ in the theory process consists in normalizing w.r.t. $(R_{\mathcal{E}_i}, \mathcal{E}_i)$ every equation or rule deduced.

The conditions defining those extensions are very restrictive. The reason is that therefore they imply only minor modifications in our procedure. Without such conditions, we would have either to handle additional contexts together with extended matching and unification algorithms, or to add many rewrite rules for each deduced rule as done in [9] for considering idempotency or nilpotency in addition to associativity-commutativity. But even with those conditions, some interesting examples can be considered. For theories generating contexts (such as those including associativity of an operator f), extensions can include axioms such as $f(x, 1) \approx x$ or $f(x, 0) \approx 0$. For theories that do not generate contexts (such as the commutativity of an operator f), extensions can include axioms like $f(x, x) \approx x$ or $f(x, x) \approx 0$.

5 Completeness of `CombCC`

The `CombCC` procedure is refutationally complete, provided that deductions are fairly applied. Moreover, if the `CombCC` procedure terminates without finding a contradiction with disequalities of Δ, it generates a terminating confluent term rewriting system for the equational theory $\mathcal{E} \cup \Gamma$.

Theorem 1. *Let $\mathcal{E}$ be any disjoint union of empty, flat permutative, associative, and associative-commutative theories over the combined signature Σ which may include uninterpreted function symbols and constants. Consider Γ is any set of ground Σ-equalities and Δ is any set of ground Σ-disequalities. Given the input $\Gamma \cup \Delta$, the `CombCC` procedure halts on $\bot$ if $\Gamma \cup \Delta$ is $\mathcal{E}$-unsatisfiable. If the `CombCC` procedure halts on an output distinct from $\bot$, then $\Gamma \cup \Delta$ is $\mathcal{E}$-satisfiable, and the output provides a rewriting system R such that (1) R is terminating and confluent modulo $\mathcal{E}$ on $T(\Sigma \cup K)$, and (2) any two ground terms in $T(\Sigma)$ are*

$\mathcal{E} \cup R$-equal iff they are $\mathcal{E} \cup \Gamma$-equal. Moreover, the `CombCC` *procedure is necessarily terminating if $\mathcal{E}$ does not involve associative theories.*

To prove the completeness of `CombCC`, we can rely on a Nelson-Oppen combination method [13] based on the ping-ponging of entailed equalities between (shared) constants. This combination method is applicable without loss of completeness because one can rely on a union of convex and stably infinite theories. The same proof idea as the one initiated in [2] can be reused, by considering two cases. In the simple case of the trivial model, a set of literals is satisfiable if and only if it is a set of equalities. In the general case excluding the trivial model, the underlying theories are stably infinite.

Lemma 1 ([2]). *Let $\mathcal{E}_i$ be any non-trivial equational Σ_i-theory, Γ any finite set of ground Σ_i-equalities, Δ any finite set of ground Σ_i-disequalities, and a, b two distinct fresh constants not occurring in $\mathcal{E}_i$, Γ and Δ. Then, we have:*

(i) $\mathcal{E}_i \cup \{a \not\approx b\}$ is convex and stably infinite,
(ii) $\Gamma \cup \Delta$ is $(\mathcal{E}_i \cup \{a \not\approx b\})$-unsatisfiable iff $\Gamma \cup \Delta \cup \{a \not\approx b\}$ is $\mathcal{E}_i$-unsatisfiable.

Proof. The statement (ii) is straightforward, so let us focus on (i): $\mathcal{E}_i \cup \{a \not\approx b\}$ is convex since it is a Horn theory and any Horn theory is known to be convex [19]. Moreover, $\mathcal{E}_i \cup \{a \not\approx b\}$ is stably infinite since any convex theory with no trivial models is known to be stably infinite [4].

The convexity induces a particular way to decide the satisfiability of equalities plus a conjunction of disequalities: it allows us to consider each disequality separately. This is a direct rewording of the definition of convexity.

Lemma 2. *Let T be any convex theory. For any T-satisfiable finite set Γ of ground equalities, and any finite set Δ of ground disequalities, both expressed over the signature of T, we have that $\Gamma \cup \Delta$ is T-unsatisfiable iff there exists some disequality $s \not\approx t$ in Δ such that $\Gamma \cup \{s \not\approx t\}$ is T-unsatisfiable.*

In `CombCC`, the input satisfiability problem is transformed via Splitting and Flattening into an equisatisfiable problem including only flat literals, meaning that all the disequalities in Δ are of the form $c \not\approx d$ where c and d are constants. Thus, we are looking for inference systems with the property of being deduction-complete [20], in order to derive each equality $c \approx d$ such that $\Gamma \Rightarrow c \approx d$ is valid in the underlying theory. This is exactly the purpose of a congruence closure procedure when it applies to an input set of flat equalities Γ. It generates all the equalities between constants that are logically entailed by Γ.

Compared to a classical application of the Nelson-Oppen combination method, we have to accommodate congruence closure procedures that are not necessarily terminating since $\mathcal{E}$ may include associative theories for which non-terminating derivations are shown in Section 6. Let us shortly explain why `CombCC` is refutationally complete. According to the completeness of the Nelson-Oppen method, the satisfiability problem in any disjoint union of stably infinite theories is reducible to the satisfiability problems in the component theories, provided that

all possible arrangements are guessed. Consequently, given any disjoint union of stably infinite theories, using refutationally complete procedures for the satisfiability problems in the component theories allows us to get a refutationally complete procedure for the satisfiability problem in the union. In our context, stably infinite theories are also convex and so the guessing of all possible arrangements can be replaced by a ping-ponging of entailed equalities between constants. Then, we use the property that all the entailed equalities between constants are eventually generated since our congruence closure procedures are deduction-complete.

Remark 1 (About theories sharing constants). Our combination procedure permits theories to share constants, as exemplified in Section 4.4 by the addition of axioms. Let us briefly explain how those shared constants are handled. Assume that a constant a is shared by two theories $\mathcal{E}_1$ and $\mathcal{E}_2$. The orchestrator will replace this constant by a new one by Flattening, generating a D-rule $a \to c$ ($c \in K$). Then it will include this D-rule in the set of rewrite rules of each theory (R_1 and R_2). The theory process of $\mathcal{E}_i$ may have to use this D-rule for transforming rules of R_i and equations in E_i, eliminating all other occurrences of constant a. And if later on an equation $c \approx d$ is generated, it will be propagated in all other theory processes by the Compression inference rule of the orchestrator. Thus, considering theories with shared constants preserves the way our `CombCC` procedure works and its refutational completeness.

6 On Non-termination for Associative Theories

Our procedure may not terminate (if no contradiction exists) with associative theories. For example, we consider the combination of two theories, one with an associative operator f, and the empty theory with constants a, b, c and d, and we are given the equalities

$$\Gamma = \{f(a,b) \approx c, f(d,a) \approx c, f(a,c) \approx f(c,a)\}$$

For any precedence ordering between the four constants, an infinite number of rewrite rules is generated as illustrated below by the following case study, where traces are rebuilt from the output of our implementation. [3]

If $d \prec c$, the E-rules $f(c, d^n, c) \to f(d^n, c, c)$ are generated as follows:

```
> Equality: f(a,b) = c
  D-rule f(a,b) → c generated (Orientation)
> Equality: f(d,a) = c
  D-rule f(d,a) → c generated (Orientation)
> Equality: f(a,c) = f(c,a)
  E-rule f(a,c) → f(c,a) generated (Orientation)
> Equality f(c,b) = f(d,c) generated
```

[3] For clarity, constants a, b, c, d have not been replaced by new constants in the traces.

```
  (Superposition between f(d,a) → c and f(a,b) → c)
> Equality: f(c,b) = f(d,c)
  E-rule f(c,b) → f(d,c) generated (Orientation)
> Equality f(c,c) = f(d,c,a) generated
  (Superposition between f(d,a) → c and f(a,c) → f(c,a))
> Equality: f(c,c) = f(d,c,a)
  E-rule f(d,c,a) → f(c,c) generated (Orientation)
> Equality f(c,c,b) = f(d,c,c) generated
  (Superposition between f(d,c,a) → f(c,c) and f(a,b) → c)
> Equality: f(c,c,b) = f(d,c,c)
  Subterm f(c,c,b) simplified into f(c,d,c) (with f(c,b) → f(d,c))
  E-rule f(c,d,c) → f(d,c,c) generated (Orientation)
> Equality f(d,c,c,b) = f(c,d,d,c) generated
  (Superposition between f(c,d,c) → f(d,c,c) and f(c,b) → f(d,c))
> Equality: f(d,c,c,b) = f(c,d,d,c)
  Subterm f(d,c,c,b) simplified into f(d,c,d,c) (with f(c,b) → f(d,c))
  Subterm f(d,c,d,c) simplified into f(d,d,c,c) (with f(c,d,c) → f(d,c,c))
  E-rule f(c,d,d,c) → f(d,d,c,c) generated (Orientation)
...
> Equality f(d^n,c,c,b) = f(c,d^{n+1},c) generated
  (Superposition between f(c,d^n,c) → f(d^n,c,c) and f(c,b) → f(d,c))
  %% extended term f(c,d^n,c,b) generates f(d^n,c,c,b) = f(c,d^n,d,c)
> Equality f(d^n,c,c,b) = f(c,d^{n+1},c)
  Subterm f(d^n,c,c,b) simplified into f(d^n,c,d,c) (with f(c,b) → f(d,c))
  Subterm f(d^n,c,d,c) simplified into f(d^{n+1},c,c) (with f(c,d,c) → f(d,c,c))
  E-rule f(c,d^{n+1},c) → f(d^{n+1},c,c) generated (Orientation)
```

Else ($d \succ c$), if $c \prec b$, the E-rules $f(c, b^n, c) \to f(c, c, b^n)$ are generated as follows:

```
> Equality: f(a,b) = c
  D-rule f(a,b) → c generated (Orientation)
> Equality: f(d,a) = c
  D-rule f(d,a) → c generated (Orientation)
> Equality: f(a,c) = f(c,a)
  E-rule f(a,c) → f(c,a) generated (Orientation)
> Equality f(c,b) = f(d,c) generated
  (Superposition between f(d,a) → c and f(a,b) → c)
> Equality: f(c,b) = f(d,c)
  E-rule f(d,c) → f(c,b) generated (Orientation)
> Equality f(c,c) = f(d,c,a) generated
  (Superposition between f(d,a) → c and f(a,c) → f(c,a))
> Equality: f(c,c) = f(d,c,a)
  Subterm f(d,c,a) simplified into f(c,b,a) (with f(d,c) → f(c,b))
  E-rule f(c,b,a) → f(c,c) generated (Orientation)
> Equality f(c,c,b) = f(c,b,c) generated
  (Superposition between f(c,b,a) → f(c,c) and f(a,b) → c)
> Equality: f(c,c,b) = f(c,b,c)
  E-rule f(c,b,c) → f(c,c,b) generated (Orientation)
> Equality f(c,b,b,c) = f(d,c,c,b) generated
  (Superposition between f(d,c) → f(c,b) and f(c,b,c) → f(c,c,b))
> Equality: f(c,b,b,c) = f(d,c,c,b)
```

```
  Subterm f(d,c,c,b) simplified into f(c,b,c,b) (with f(d,c) → f(c,b))
  Subterm f(c,b,c,b) simplified into f(c,c,b,b) (with f(c,b,c) → f(c,c,b))
  E-rule f(c,b,b,c) → f(c,c,b,b) generated (Orientation)
...
> Equality f(c,b^{n+1},c) = f(d,c,c,b^n) generated
  (Superposition between f(d,c) → f(c,b) and f(c,b^n,c) → f(c,c,b^n))
  %% extended term f(d,c,b^n,c) generates f(c,b,b^n,c) = f(d,c,c,b^n)
> Equality: f(c,b^{n+1},c) = f(d,c,c,b^n)
  Subterm f(d,c,c,b^n) simplified into f(c,b,c,b^n) (with f(d,c) → f(c,b))
  Subterm f(c,b,c,b^n) simplified into f(c,c,b^{n+1}) (with f(c,b,c) → f(c,c,b))
  E-rule f(c,b^{n+1},c) → f(c,c,b^{n+1}) generated (Orientation)
```

Else ($d \succ c \succ b$), the E-rules $f(c, b^n, c, b) \to f(c, b, b^n, c)$ are generated as follows:

```
> Equality: f(a,b) = c
  D-rule f(a,b) → c generated (Orientation)
> Equality: f(d,a) = c
  D-rule f(d,a) → c generated (Orientation)
> Equality: f(a,c) = f(c,a)
  E-rule f(a,c) → f(c,a) generated (Orientation)
> Equality f(c,b) = f(d,c) generated
  (Superposition between f(d,a) → c and f(a,b) → c)
> Equality: f(c,b) = f(d,c)
  E-rule f(d,c) → f(c,b) generated (Orientation)
> Equality f(c,c) = f(d,c,a) generated
  (Superposition between f(d,a) → c and f(a,c) → f(c,a))
> Equality: f(c,c) = f(d,c,a)
  Subterm f(d,c,a) simplified into f(c,b,a) (with f(d,c) → f(c,b))
  E-rule f(c,b,a) → f(c,c) generated (Orientation)
> Equality f(c,c,b) = f(c,b,c) generated
  (Superposition between f(c,b,a) → f(c,c) and f(a,b) → c)
> Equality: f(c,c,b) = f(c,b,c)
  E-rule f(c,c,b) → f(c,b,c) generated (Orientation)
> Equality f(c,b,c,b) = f(d,c,b,c) generated
  (Superposition between f(d,c) → f(c,b) and f(c,c,b) → f(c,b,c))
> Equality: f(c,b,c,b) = f(d,c,b,c)
  Subterm f(d,c,b,c) simplified into f(c,b,b,c) (with f(d,c) → f(c,b))
  E-rule f(c,b,c,b) → f(c,b,b,c) generated (Orientation)
> Equality f(c,b,b,c,b) = f(d,c,b,b,c) generated
  (Superposition between f(d,c) → f(c,b) and f(c,b,c,b) → f(c,b,b,c))
> Equality: f(c,b,b,c,b) = f(d,c,b,b,c)
  Subterm f(d,c,b,b,c) simplified into f(c,b,b,b,c) (with f(d,c) → f(c,b))
  E-rule f(c,b,b,c,b) → f(c,b,b,b,c) generated (Orientation)
...
> Equality f(c,b^{n+1},c,b) = f(d,c,b^{n+1},c) generated
  (Superposition between f(d,c) → f(c,b) and f(c,b^n,c,b) → f(c,b,b^n,c))
  %% extended term f(d,c,b^n,c,b) generates f(c,b,b^n,c,b) = f(d,c,b,b^n,c)
> Equality: f(c,b^{n+1},c,b) = f(d,c,b^{n+1},c)
  Subterm f(d,c,b^{n+1},c) simplified into f(c,b,b^{n+1},c) (with f(d,c) → f(c,b))
  E-rule f(c,b^{n+1},c,b) → f(c,b,b^{n+1},c) generated (Orientation)
```

There are also shorter examples, for instance with two associative operators f and g and the equalities

$$\Gamma = \{f(a,b) \approx c, f(a,c) \approx f(c,a), g(b,a) \approx c, g(a,c) \approx g(c,a)\}$$

Either the theory process of f, or the one of g, will generate an infinite number of rewrite rules, depending on the chosen ordering between constants a and c deciding of the orientation of the second and fourth equalities.

For associativity, it is well-known that unification is infinitary. But with those examples we illustrate that it is possible to generate infinite derivations, even if we consider only ground equalities and unification problems with finitely many most general unifiers.

7 Running Example

We have implemented the combination procedure described in this paper for several theories: the empty theory, commutative theories, associative theories, associative-commutative theories. It is written in C (6000 lines of code).

We detail below the trace obtained in 6 *ms* for an example combining three theories: one where the operator *cp* (a short-cut for *compatible*) is commutative, one where the operator *and* is associative-commutative, and the empty theory with all the constants and the *owns* operator. The initial set of equalities Γ is:

$$\begin{cases} owns(Ali, and(cp(boat, engine), cp(engine, captain), cp(captain, boat))) \approx true \\ and(cp(boat, captain), cp(boat, engine)) \approx and(boat, cp(captain, engine)) \\ and(cp(captain, engine), cp(captain, engine)) \approx cp(captain, engine) \\ and(cp(engine, captain), boat) \approx ready\_boat \\ owns(Ali, ready\_boat) \approx false \end{cases}$$

and there is one disequality: $\Delta = \{true \not\approx false\}$.

A contradiction is derived by the following deductions; [4] new constants (of K) are written $\_i$, where i is an integer.

> Equality: $owns(Ali, rdy) = false$
  D-rule $owns(Ali, rdy) \to false$ generated (Orientation)
> Equality: $and(cp(eng, cap), boat) = rdy$
  D-rule $cp(cap, eng) \to \_1$ generated (Extension)
  D-rule $and(boat, \_1) \to rdy$ generated (Orientation)
> Equality: $and(cp(cap, eng), cp(cap, eng)) = cp(cap, eng)$
  Subterm $cp(cap, eng)$ simplified into _1 (with $cp(cap, eng) \to \_1$)
  Subterm $cp(cap, eng)$ simplified into _1 (with $cp(cap, eng) \to \_1$)
  Subterm $cp(cap, eng)$ simplified into _1 (with $cp(cap, eng) \to \_1$)
  D-rule $and(\_1, \_1) \to \_1$ generated (Orientation)
> Equality: $and(cp(boat, cap), cp(boat, eng)) = and(boat, cp(cap, eng))$
  D-rule $cp(boat, cap) \to \_2$ generated (Extension)
  D-rule $cp(boat, eng) \to \_3$ generated (Extension)

[4] For clarity, in the trace the initial constants have been shortened: "*eng*" stands for "*engine*", "*cap*" stands for "*captain*", "*rdy*" stands for "*ready_boat*".

```
  Subterm cp(cap, eng) simplified into _1 (with cp(cap, eng) → _1)
  Subterm and(boat, _1) simplified into rdy (with and(boat, _1) → rdy)
  D-rule and(_3, _2) → rdy generated (Orientation)
> Equality: owns(Ali, and(cp(boat, eng), cp(eng, cap), cp(cap, boat))) = true
  Subterm cp(boat, eng) simplified into _3 (with cp(boat, eng) → _3)
  Subterm cp(cap, eng) simplified into _1 (with cp(cap, eng) → _1)
  Subterm cp(boat, cap) simplified into _2 (with cp(boat, cap) → _2)
  Subterm and(_3, _2, _1) simplified into and(_1, rdy) (with and(_3, _2) → rdy)
  D-rule and(_1, rdy) → _4 generated (Extension)
  D-rule owns(Ali, _4) → true generated (Orientation)
> Equality and(boat, _1) = and(_1, rdy) generated
  (Superposition between and(boat, _1) → rdy and and(_1, _1) → _1)
> Equality: and(boat, _1) = and(_1, rdy)
  Subterm and(boat, _1) simplified into rdy (with and(boat, _1) → rdy)
  Subterm and(_1, rdy) simplified into _4 (with and(_1, rdy) → _4)
  C-rule _4 → rdy generated (Orientation)
> Compression with _4 → rdy
  D-rule and(_1, rdy) → _4 replaced by and(_1, rdy) → rdy
  (Composition with _4 → rdy)
  D-rule owns(Ali, _4) → true replaced by owns(Ali, rdy) → true
  (Collapse with _4 → rdy)
  D-rule owns(Ali, rdy) → false replaced by false → true
  (Collapse with owns(Ali, rdy) → true)
> Compression with false → true
  Disequality: true ≠ false simplified into true ≠ true (with false → true)
*** A contradiction has been found! ***
```

8 Conclusion

We have defined `CombCC` as an orchestrator that does not need to handle specific algorithms related to theories $\mathcal{E}_i$. It could be more efficient using some inference rules of theory processes, like Simplification and Deletion. But we did this on purpose for the clarity of the paper and to show that the orchestrator can be defined independently from the theories. We are considering several extensions of our procedure, to apply it to any theory having a deduction system preserving the groundness of generated rules/equalities. This applies to flat permutative theories, an extension of commutative theories. It also applies to extensions of associative and/or commutative theories with axioms that can be used as flat collapsing rewrite rules.

Our new implementation allows us to experiment extensions of Associativity and/or Commutativity with orientable equational axioms specifying for instance an idempotent or nilpotent operator, or an operator with a neutral or absorbent element. This implementation is very helpful to identify counter-examples and non-terminating examples.

In the future, we plan to investigate congruence closure procedures for the unions of theories possibly sharing constructor symbols with associative and/or commutative equational properties. It is clearly challenging to try to go beyond the simple case of shared constants.

Currently, we consider theories where classical equational completion techniques are applicable. An interesting future work would be to study other completion techniques, such as unfailing completion, in the context of congruence closure. In order to integrate various completion techniques within a uniform framework, we could envision to reuse the notion of normalizing mapping initially introduced for the (combined) word problem [6], or study the combination of extended canonizers [18] to go beyond the classical rewrite-based normalization. Applying this notion to congruence closure remains to be studied. This would pave the way of extending our combined abstract framework for congruence closure to new equational theories.

References

1. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
2. Baader, F., Tinelli, C.: A new approach for combining decision procedure for the word problem, and its connection to the Nelson-Oppen combination method. In: McCune, W. (ed.) 14th Int. Conference on Automated Deduction, CADE, Townsville, North Queensland, Australia, July 1997, Proceedings. LNCS, vol. 1249, pp. 19–33. Springer (1997)
3. Bachmair, L., Tiwari, A., Vigneron, L.: Abstract congruence closure. Journal of Automated Reasoning **31**(2), 129–168 (2003)
4. Barrett, C.W., Dill, D.L., Stump, A.: A generalization of Shostak's method for combining decision procedures. In: Armando, A. (ed.) 4th Int. Workshop on Frontiers of Combining Systems, FroCoS, Santa Margherita Ligure, Italy, April 2002, Proceedings. LNCS, vol. 2309, pp. 132–146. Springer (2002)
5. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability - Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 1267–1329. IOS Press (2021)
6. Erbatur, S., Marshall, A.M., Ringeissen, C.: Combined hierarchical matching: the regular case. In: Felty, A.P. (ed.) 7th Int. Conference on Formal Structures for Computation and Deduction, FSCD, Haifa, Israel, August 2022. LIPIcs, vol. 228, pp. 6:1–6:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
7. Kapur, D.: Shostak's congruence closure as completion. In: Comon, H. (ed.) 8th Int. Conference on Rewriting Techniques and Applications, RTA, Sitges, Spain, June 1997, Proceedings. LNCS, vol. 1232, pp. 23–37. Springer (1997)
8. Kapur, D.: A modular associative commutative (AC) congruence closure algorithm. In: Kobayashi, N. (ed.) 6th Int. Conference on Formal Structures for Computation and Deduction, FSCD, Buenos Aires, Argentina, July 2021 (Virtual Conference). LIPIcs, vol. 195, pp. 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
9. Kapur, D.: Modularity and combination of associative commutative congruence closure algorithms enriched with semantic properties. Logical Methods in Computer Science **19**(1) (2023)
10. Kim, D.: Congruence closure modulo groups. CoRR **abs/2310.05014** (2023), https://doi.org/10.48550/arXiv.2310.05014

11. Kim, D., Lynch, C.: Congruence closure modulo permutation equations. In: Kutsia, T. (ed.) 9th Int. Symposium on Symbolic Computation in Software Science, SCSS, Hagenberg, Austria, September 2021, Proceedings. EPTCS, vol. 342, pp. 86–98 (2021)
12. Meseguer, J.: Variants and satisfiability in the infinitary unification wonderland. Journal of Logical and Algebraic Methods in Programming **134**, 100877 (2023)
13. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Transactions on Programming Languages and Systems **1**(2), 245–257 (1979)
14. Nelson, G., Oppen, D.C.: Fast decision procedures based on congruence closure. Journal of the ACM **27**(2), 356–364 (1980)
15. Nieuwenhuis, R., Oliveras, A.: Fast congruence closure and extensions. Information and Computation **205**(4), 557–580 (2007)
16. Peterson, G.E., Stickel, M.E.: Complete sets of reductions for some equational theories. Journal of the ACM **28**(2), 233–264 (1981)
17. Plotkin, G.: Building-in equational theories. Machine Intelligence **7**, 73–90 (1972)
18. Ranise, S., Ringeissen, C., Tran, D.: Nelson-Oppen, Shostak and the extended canonizer: A family picture with a newborn. In: Liu, Z., Araki, K. (eds.) Theoretical Aspects of Computing - ICTAC 2004, First International Colloquium, Guiyang, China, September 2004, Revised Selected Papers. LNCS, vol. 3407, pp. 372–386. Springer (2004)
19. Tinelli, C.: Cooperation of background reasoners in theory reasoning by residue sharing. Journal of Automated Reasoning **30**(1), 1–31 (2003)
20. Tran, D., Ringeissen, C., Ranise, S., Kirchner, H.: Combination of convex theories: Modularity, deduction completeness, and explanation. Journal of Symbolic Computation **45**(2), 261–286 (2010)
21. Vigneron, L.: Positive deduction modulo regular theories. In: Kleine Büning, H. (ed.) 9th Int. Workshop on Computer Science Logic, CSL, Annual Conference of the EACSL, Paderborn, Germany, September 1995, Selected Papers. LNCS, vol. 1092, pp. 468–485. Springer (1995)