Rule Based Incremental Congruence Closure with Commutative Symbols

Sylvain Conchon and Evelyne Contejean

PCRI — LRI (CNRS UMR 8623) — Inria Futurs — Université Paris Sud
Bt. 490, Université Paris-Sud, 91405 Orsay Cedex, France
{conchon,contejea}@lri.fr

Abstract. We present a rule based congruence closure algorithm that constructs a "term preserving" union-find data structure from a set of ground equations. Starting from a set of two simple inference rules, we show how our algorithm can be made incremental by adding two extra rules to the original set. Commutative symbols are also handled thanks to a slight modification of the rules. The main originality of this work rests on the description level of our framework which is high enough to enjoy rigorous (and self-contained) correctness proofs and low enough so that the rules are directly derived from our efficient OCaml implementation.

1 Introduction

The theory of equality gives the semantics of the *equality symbol* $=$. It is defined as the smallest reflexive, symmetric and transitive relation that satisfies the Leibniz's rule (also called the *congruence* axiom): $\vec{a} = \vec{b} \Rightarrow f(\vec{a}) = f(\vec{b})$, for any vectors of terms $\vec{a}$, $\vec{b}$ and any (uninterpreted) function symbol f.

The use of the equality predicate is so ineluctable in logic that the question of its automated treatment has been studied very early in computer science. In particular, the problem of deciding whether a ground equation $a = b$ logically follows from a set E of ground equalities, denoted by $a =_E b$, has been found critical in many applications, including mechanical program verification.

Algorithms to compute the *congruence closure* of a set of ground equations do exist [5, 10, 13, 2, 12]. Basically, two different approaches have been proposed: the first one aims at constructing a union-find data structure that stores the final equivalence relation on terms [5, 10, 12], while the second one aims at producing a convergent rewriting system which can then be used for checking equalities [13, 2].

Nowadays, many automated theorem provers use congruence closure algorithms to handle their built-in equality predicate [4, 3, 6]. However, because of specificities related to their application domains, these provers require some additional properties on their congruence closure module.

First, the backtracking search underlying the architecture of SAT-based theorem provers enforces an incremental treatment of the set of ground equations. Indeed, for efficiency reasons, equations are given one by one by the SAT solver to

the equality module which prevents it from realizing a global preliminary treatment on them as a set, unless restarting the congruence closure from scratch.

Secondly, theorem provers that handle quantified formulas have to find relevant instances of definitions, lemmas or axioms among the set of ground terms in the formula to be proved. While no satisfactory solution is known to this semi-decidable problem, it is clear that the pattern-matching algorithm underlying this process should benefit from the equalities discovered by the congruence closure algorithm. For instance [9], if it is assumed that $a = g(b)$ and $b = g(a)$, then $P(a, a)$ can be proved from the axiom $\forall x, P(g(g(x)), x)$ by instantiating x by a.

We make here an important difference between the two approaches described above for the construction of congruence closure. The rewriting approach has to find the instance by solving the matching problem $g(g(x)) = a \wedge x = a$. A possible solution is to use the narrowing, which terminates in that case since the convergent rewriting system contains only ground rules [7]. On the other hand, the equivalence classes of the union-fond data structure returned by the first approach should help the matcher to find that the pattern $g(g(x))$ coincide with the term a by an enumeration of the classes. Obviously, and importantly, the matching process will be more efficient if the union-find structure contains only ground terms from the original set. Furthermore, the matcher should also benefit from any extension provided to the congruence closure algorithm. For instance, if commutative symbols are handled by the equality module, it should be possible to match the pattern $1 + x$ against the term $a + 1$.

Finally, because congruence closure algorithms are at the core of theorem provers, it is crucial that their design and implementation are correct. For that, their formal description should be fine-grained enough to model most of the mechanisms currently used in the implementation and their correctness proofs should be as rigorous as possible.

Our Work. We describe a congruence closure algorithm by a set of two simple inference rules: given a set $\mathcal{T}$ of ground terms, our system aims at constructing a union-find data structure representing the congruence closure of a set E of equalities between terms of $\mathcal{T}$. The level of description of our framework is high enough to enjoy a rigorous (and self-contained) correctness proof and low enough so that the rules are directly derived from our efficient OCaml implementation.

We then show how to extend our algorithm to make it incremental. In that case, the set $\mathcal{T}$ is empty and the union-find data structure returned by the inference rules contains only ground terms contained in the processed equalities. The main originality here is that the incrementality process is clearly separated from the congruence closure part of the system: two extra rules are added to the system while keeping the original set intact. This presentation allows us to prove the correctness of the whole system without reproving most of the correctness facts relative to the closure mechanism.

Last but not least, the modularity of our algorithm allows us to extend it modulo some built-in theories. As an example, we show how commutative symbols can be handle easily with a very slight modification of the rules. As for the original algorithm, the union-find data structure returned by the frameworks

contains *only* ground terms of the initial set of equations. Taking care of this fundamental property makes the correctness proof surprisingly difficult.

Organization of the Paper. We present in section 2 a non-incremental congruence closure algorithm based on two simple inference rules and we provide a rigorous correctness proof for it. In section 3, we show how to the addition of two new rules to the original system make it incremental. We demonstrate the extension capabilities of our approach in section 4 by showing how commutative symbols can be handle very easily. We conclude in section 5.

2 A Rule Based Congruence Closure Algorithm

Let Σ be a finite signature, $\mathcal{T}$ be a finite set of ground $\Sigma-$terms closed by subterms and E be a set of equalities between terms of $\mathcal{T}$. We denote $=_E$ the equational theory induced by E on $T(\Sigma)$.

We are interested in deciding whether two terms of $\mathcal{T}$ are equal modulo E by a congruence closure algorithm that we shall formalize thanks to a set of two inference rules described figure 1. These rules handle triples $\langle \Gamma \mid \Delta \mid \Phi \rangle$ as configurations where:

- Γ is used for propagating the discovered equalities by congruence. More formally, Γ is a map, that is a partial injective function, which contains associations $u \mapsto \mathcal{C}$ where u is a term and $\mathcal{C}$ a set of terms. We denote by $\Gamma(u) = \mathcal{C}$ the fact that Γ contains $u \mapsto \mathcal{C}$ and by $\Gamma(u) = \bot$ the fact that Γ does not contain any association for u. If an equality $u = v$ is discovered it has to be propagated to terms which have u or v as subterms, namely $\Gamma(u)$ and $\Gamma(v)$.
- Δ is a union-find data structure which describes the currently known equalities; $\Delta(u)$ denotes the representative of u,
- Φ contains the ground equations which still have to be processed.

$$\textsc{Congr}\ \frac{\langle \Gamma \uplus \{\Delta(a) \mapsto \mathcal{A}, \Delta(b) \mapsto \mathcal{B}\} \mid \Delta \mid \{a = b\} \uplus \Phi \rangle}{\langle \Gamma \uplus \{\Delta'(a) \mapsto \mathcal{A} \cup \mathcal{B}\} \mid \Delta' \mid \Phi' \cup \Phi \rangle}\ \Delta(a) \neq \Delta(b)$$

$$\text{with}\ \begin{cases} \Delta' = \Delta + \{a = b\} \\ \Phi' = \{f(\vec{a}) = f(\vec{b}) \mid f(\vec{a}) \in \mathcal{A} \wedge f(\vec{b}) \in \mathcal{B} \wedge \Delta'(\vec{a}) = \Delta'(\vec{b})\} \end{cases}$$

$$\textsc{Remove}\ \frac{\langle \Gamma \mid \Delta \mid \{a = b\} \uplus \Phi \rangle}{\langle \Gamma \mid \Delta \mid \Phi \rangle}\ \Delta(a) = \Delta(b)$$

Fig. 1. A small set of inference rules for CC.

A configuration K is a $\mathcal{T}$*-configuration* when all terms occurring in K are in $\mathcal{T}$. A configuration $K = \langle \Gamma \mid \Delta \mid \Phi \rangle$ reduces to $K' = \langle \Gamma' \mid \Delta' \mid \Phi' \rangle$, denoted by

$K \to K'$, if K' can be obtained from K by applying one of the rules of figure 1 ($\to^*$ is the reflexive transitive closure of $\to$).

Contrarily to completion based congruence closure algorithms, our approach does not create new terms:

Lemma 1. *If $K \to K'$ and K is a $\mathcal{T}$-configuration then so is K'.*

Let $K_0 = \langle \Gamma_{\mathcal{T}} \mid \mathtt{id} \mid E \rangle$ be the initial $\mathcal{T}$-configuration of the algorithm where $\Gamma_{\mathcal{T}}$ is the reverted DAG[1] of the direct subterms of $\mathcal{T}$ with maximal sharing and $\mathtt{id}$ is the union-find data structure where all terms of $\mathcal{T}$ are pairwise distinct.

Example 1. If $\mathcal{T} = \{a, b, g(a,b), g(g(a,b),b)\}$ then $\Gamma_{\mathcal{T}} = \{a \mapsto \{g(a,b)\}; b \mapsto \{g(a,b), g(g(a,b),b)\}; g(a,b) \mapsto \{g(g(a,b),b)\}; g(g(a,b),b) \mapsto \{\}\}$

Theorem 1. *The relation $\to$ is terminating from any $\mathcal{T}$-configuration.*

Proof. The measure associated with a $\mathcal{T}$-configuration $\langle \Gamma \mid \Delta \mid \Phi \rangle$ is the pair (c, n) where c is the number of equivalence classes in Δ and n the number of equations in Φ. By lemma 1, $\langle \Gamma \mid \Delta \mid \Phi \rangle$ contains only $\mathcal{T}$ terms thus applying Congr on an equation $u = v$ strictly decreases c since $u, v \in \mathcal{T}$. Finally, an application of Remove does not change c and strictly decreases n.

Lemma 2. *Any irreducible configuration obtained from K_0 is of the form $\langle \Gamma \mid \Delta \mid \emptyset \rangle$.*

Theorem 2 (Correctness). *For any irreducible configuration $\langle \Gamma_\infty \mid \Delta_\infty \mid \emptyset \rangle$ obtained from K_0, for any terms $u, v \in \mathcal{T}$, $u =_E v$ iff $\Delta_\infty(u) = \Delta_\infty(v)$.*

Proof. The *if* direction is proved by the following invariant:

$$I_1(\langle \Gamma \mid \Delta \mid \Phi \rangle) = \forall u, v \in T(\Sigma), \begin{cases} \Delta(u) = \Delta(v) \Rightarrow u =_E v \\ u = v \in \Phi \Rightarrow u =_E v \end{cases}$$

- $I_1(K_0)$ is immediate.
- Let us prove that if $K \to K'$ and $I_1(K)$ then $I_1(K')$. If K' is obtained from K by Remove the result is immediate since Δ remains unchanged and the new set of equations of K' is a subset of that of K. Otherwise, $K' = \langle \Gamma' \mid \Delta' = \Delta + \{a = b\} \mid \Phi' \cup \Phi \rangle$ is obtained by Congr.
  - Let u and v such that $\Delta'(u) = \Delta'(v)$. If $\Delta(u) = \Delta(v)$ the result follows by induction hypothesis. Otherwise $\Delta(u) = \Delta(a)$ and $\Delta(v) = \Delta(b)$ (possibly by exchanging u and v). By induction hypothesis $a = b \in \Phi \cup \{a = b\}$ implies $a =_E b$; moreover (again by induction hypothesis) $u =_E a$ and $v =_E b$. We conclude by transitivity of $=_E$.
  - Let $u = v \in \Phi \cup \Phi'$. If $u = v \in \Phi$ then we conclude by induction hypothesis. Otherwise, $u = f(\vec{a})$ and $v = f(\vec{b})$ and $\Delta'(\vec{a}) = \Delta'(\vec{b})$ hence by the above argument $\vec{a} =_E \vec{b}$. We conclude by the congruence property of $=_E$.

[1] with respect to its associations

Let us now face the *only if* direction. Let $=_\Delta$ be the equational theory induced by the set of axioms $\{u = v \mid \Delta(u) = \Delta(v)\}$. We first prove the following invariants:

$$I_2(\langle \Gamma \mid \Delta \mid \Phi \rangle) = \forall t_1, \ldots, t_n \in T(\Sigma), \\ f(t_1, \ldots, t_n) \in \mathcal{T} \Rightarrow \forall i, f(t_1, \ldots, t_n) \in \Gamma(\Delta(t_i))$$
$$I_3(\langle \Gamma \mid \Delta \mid \Phi \rangle) = \forall u, v \in \mathcal{T}, u =_E v \Rightarrow (u, v) \in (=_\Phi \cup =_\Delta)^*$$

- $I_2(K_0)$ holds trivially by construction of $\Gamma_\mathcal{T}$ and since $\Delta_0 = \mathtt{id}$.
- Let us prove that if $K \to K'$ and $I_2(K)$ then $I_2(K')$. If K' is obtained from K by REMOVE the result is immediate since Γ and Δ remain unchanged. Otherwise, it is obtained by CONGR and $K = \langle \Gamma \uplus \{\Delta(a) \mapsto \mathcal{A}, \Delta(b) \mapsto \mathcal{B}\} \mid \Delta \mid \{a = b\} \uplus \Phi \rangle$ and $K' = \langle \Gamma \uplus \{\Delta'(a) \mapsto \mathcal{A} \cup \mathcal{B}\} \mid \Delta' \mid \Phi' \cup \Phi \rangle$. Let $f(t_1, \ldots, t_n)$ be a term of $\mathcal{T}$. For each t_i, we shall distinguish the two following cases:
  - If $\Delta(t_i) \neq \Delta(a)$ and $\Delta(t_i) \neq \Delta(b)$ then $\Delta'(t_i) = \Delta(t_i)$ and $\Gamma'(\Delta'(t_i)) = \Gamma(\Delta(t_i))$. We conclude by induction hypothesis.
  - If $\Delta(t_i) = \Delta(a)$, by induction hypothesis $f(t_1, \ldots, t_n) \in \mathcal{A}$. The property holds since $\Delta'(t_i) = \Delta'(a)$ and $\Gamma'(\Delta'(a)) = \mathcal{A} \cup \mathcal{B}$.
  - The case $\Delta(t_i) = \Delta(b)$ is symmetrical.
- I_3 is obvious.

Now, we shall conclude by proving that for any irreducible configuration $K_\infty = \langle \Gamma_\infty \mid \Delta_\infty \mid \emptyset \rangle$ and for all terms $u, v \in \mathcal{T}$ if $u =_{\Delta_\infty} v$ then $\Delta_\infty(u) = \Delta_\infty(v)$.

- We first prove the following congruence property of Δ_∞:

  If $f(\vec{u}) \in \mathcal{T}$, $f(\vec{v}) \in \mathcal{T}$ and $\Delta_\infty(\vec{u}) = \Delta_\infty(\vec{v})$ then $\Delta_\infty(f(\vec{u})) = \Delta_\infty(f(\vec{v}))$.

  - The case where $\vec{u}$ is syntactically equivalent to $\vec{v}$ is immediate.
  - Otherwise, we have $K_0 \to^* K \to K' \to^* K_\infty$, where K is the last configuration such that $\Delta(\vec{u}) \neq \Delta(\vec{v})$, and K' is first one such that $\Delta'(\vec{u}) = \Delta'(\vec{v})$. K' is obtained from K by applying the rule CONGR and there exists an index i such that $\Delta(u_i) \neq \Delta(v_i)$, $\Delta(u_j) = \Delta(v_j)$ for $j \neq i$, $a = b \in \Phi$, $\Delta(u_i) = \Delta(a)$, $\Delta(v_i) = \Delta(b)$ and $\Delta' = \Delta + \{a = b\}$. Furthermore, $\Delta(a) \mapsto \mathcal{A} \in \Gamma$ and $\Delta(b) \mapsto \mathcal{B} \in \Gamma$. By I_2 on K we have $f(\vec{u}) \in \mathcal{A}$ and $f(\vec{v}) \in \mathcal{B}$ so the CONGR rule will add $f(\vec{u}) = f(\vec{v})$ in Φ' and this equation will eventually be part of Δ_∞.
- Finally, we proceed by induction on the size of the proof of $u =_{\Delta_\infty} v$, where the size of a proof, seen as a sequence of equational steps, is the total sum of the terms' size which occur in it. The result is immediate when u is syntactically equivalent to v. Otherwise, we distinguish the two following cases:
  - If $u \equiv f(\vec{u}) =_{\Delta_\infty} f(\vec{v}) \equiv v$ has no equational step at the root then $\vec{u} =_{\Delta_\infty} \vec{v}$ and all the subproofs $u_i =_{\Delta_\infty} v_i$ are strictly smaller than $u =_{\Delta_\infty} v$. By induction hypothesis, and since $\mathcal{T}$ is closed by subterms, we have $u_i, v_i \in \mathcal{T}$ and $\Delta_\infty(u_i) = \Delta_\infty(v_i)$. We then conclude by the above property.

- If there is at least an equational step at the root in the proof, the proof has the following shape $u =_{\Delta_\infty} u' \leftrightarrow^{\Lambda}_{\Delta_\infty} v' =_{\Delta_\infty} v$. By definition of $=_{\Delta_\infty}$, $\Delta_\infty(u') = \Delta_\infty(v')$ hence $u', v' \in \mathcal{T}$. Applying the induction hypothesis on the subproofs $u =_{\Delta_\infty} u'$ and $v =_{\Delta_\infty} v'$ yields $\Delta_\infty(u) = \Delta_\infty(u')$ and $\Delta_\infty(v) = \Delta_\infty(v')$. We conclude by transitivity.

Example 2. From $K_0 = \langle \Gamma_{\mathcal{T}} \mid \mathtt{id} \mid \{g(a,b) = a\}\rangle$ we can get the following configurations:

$$\begin{aligned} K_0 &\equiv \langle \Gamma \uplus \{a \mapsto \{g(a,b)\}; g(a,b) \mapsto \{g(g(a,b),b)\}\} \mid \mathtt{id} \mid \{g(a,b) = a\}\rangle \\ &\to \langle \Gamma \uplus \{a \mapsto \{g(a,b); g(g(a,b),b)\} \mid \mathtt{id} + \{g(a,b) = a\} \mid \{g(a,b) = g(g(a,b),a)\}\rangle \\ &\to \langle \Gamma_\infty \mid \mathtt{id} + \{g(a,b) = a; g(a,b) = g(g(a,b),a)\} \mid \emptyset\rangle \\ &\equiv \langle \Gamma_\infty \mid \Delta_\infty \mid \emptyset\rangle \end{aligned}$$

So, we have $\Delta_\infty(g(g(a,b),b)) = \Delta_\infty(a)$ which proves that $g(a,b) = a$ implies $g(g(a,b),b) = a$.

3 Adding Incrementality

We present in this section an incremental version of our algorithm where the set E is now considered as a sequence of equations and queries between closed terms. A query $u \stackrel{?}{=} v$ of E is valid if and only if $u =_{E'} v$ where E' is the set of equations of E occurring *before* the query.

Taking the sequential aspect of E into account amounts to replace the union of sets ($\cup$ and $\uplus$) by a sequence operator ; for the third component of the configurations in the rules Congr and Remove of figure 1.

In the sequential case, $\mathcal{T}$ is not known at the beginning of the algorithm. Hence Γ_0 is empty and Γ has to be constructed step by step from the sequence E. However, it's not sufficient!

For instance, if E contains the sequence $a = b; f(a) = t; f(b) = u$, the non-incremental algorithm will fail to prove that $t =_E u$ since the equality $a = b$ is processed too early, when $f(a)$ and $f(b)$ are not yet in the structure Γ.

This problem is fixed by the rule AddTerm, described in figure 2, which determines the new equalities that can be propagated by congruence when processing a new term. For example, processing the term $f(b)$ in $f(b) = u$ will update Γ and add $f(a) = f(b)$ to Φ which will eventually trigger the Congr rule. We also add an extra rule Query to validate queries.

Theorem 3. *The relation $\to$ is terminating from any configuration $\langle \emptyset \mid \mathtt{id} \mid \Phi\rangle$ where Φ is a finite sequence.*

Proof. We define the set $\mathcal{T}$ as the set of terms occurring in Φ and closed by subterms. Since Φ is finite, $\mathcal{T}$ is finite. The measure associated with a $\mathcal{T}$-configuration $\langle \Gamma \mid \Delta \mid \Phi\rangle$ is the triple (c, g, n) where c and n are defined as in theorem 1. g is the number of terms u in $\mathcal{T}$ such that $\Gamma(\Delta(u))$ is not defined. Congr strictly decreases c. Remove and Query leave c and g unchanged and strictly decreases n. AddTerm leaves c unchanged and strictly decreases g.

$$\textsc{AddTerm}\ \frac{\langle \Gamma \uplus \cup_{v\in\vec{a}}\{\Delta(v) \mapsto \mathcal{C}_v\} \mid \Delta \mid C[f(\vec{a})]; \Phi\rangle}{\langle \Gamma \uplus \Gamma' \mid \Delta \mid \Phi'; C[f(\vec{a})]; \Phi\rangle}\ \Gamma(f(\vec{a})) = \bot$$

where $C[f(\vec{a})]$ denotes an equation or a query containing the term $f(\vec{a})$

$$\text{with}\ \begin{cases} \Gamma' = (f(\vec{a}) \mapsto \{\}) + \{\Delta(v) \mapsto \mathcal{C}_v + f(\vec{a}) \mid v \in \vec{a}\} \\ \Phi' = \left\{ f(\vec{a}) = f(\vec{b}) \,\middle|\, v \in \vec{a}, f(\vec{b}) \in C_v \wedge \Delta(\vec{a}) = \Delta(\vec{b}) \right\} \end{cases}$$

$$\textsc{Query}\ \frac{\langle \Gamma \cup \{\Delta(a) \mapsto \mathcal{A}, \Delta(b) \mapsto \mathcal{B}\} \mid \Delta \mid a \stackrel{?}{=} b; \Phi\rangle}{\langle \Gamma \cup \{\Delta(a) \mapsto \mathcal{A}, \Delta(b) \mapsto \mathcal{B}\} \mid \Delta \mid \Phi\rangle}\ \Delta(a) = \Delta(b)$$

Fig. 2. Incremental Congruence Closure Algorithm

Lemma 3. *Any irreducible configuration obtained from K_0 is either of the form $\langle \Gamma \mid \Delta \mid \emptyset\rangle$ or $\langle \Gamma \mid \Delta \mid u \stackrel{?}{=} v; \Phi\rangle$ with $\Delta(u) \neq \Delta(v)$.*

Theorem 4 (Correctness). *For any ground terms u, v, the equation $u =_E v$ holds iff there is a configuration $\langle \Gamma_\infty \mid \Delta_\infty \mid \emptyset\rangle$ reachable from $\langle \emptyset \mid \mathtt{id} \mid E; u \stackrel{?}{=} v\rangle$.*

Proof. The *if* direction is proved by the following invariant:

$$I_1(\langle \Gamma \mid \Delta \mid \Phi\rangle) = \forall u, v \in T(\Sigma)\ \begin{cases} \Delta(u) = \Delta(v) \Rightarrow u =_E v \\ u = v \in \Phi \Rightarrow u =_E v \end{cases}$$

- $I_1(K_0)$ is immediate.
- Let us prove that if $K \to K'$ and $I_1(K)$ then $I_1(K')$. By case on the last rule applied.
  - If K' is obtained by Remove or Congr then the proof of the non-incremental system applies verbatim.
  - If K' is obtained from K by Query the result is immediate since Δ remains unchanged and the new set of equations of K' is equal to that of K.
  - If $K' = \langle \Gamma' \mid \Delta \mid \Phi'; C[f(\vec{a})]; \Phi\rangle$ is obtained by AddTerm. If $\Delta(u) = \Delta(v)$ the result is immediate by induction hypothesis. Let $u = v \in \Phi'; C[f(\vec{a})]; \Phi$. If $u = v \in C[f(\vec{a})]; \Phi$ then we conclude by induction hypothesis. Otherwise, $u = v \in \Phi'$, $u = f(\vec{a})$, $v = f(\vec{b})$ and $\Delta(\vec{a}) = \Delta(\vec{b})$ hence by the above argument $\vec{a} =_E \vec{b}$. We conclude by the congruence property of $=_E$.

Hence, if $(\Gamma_\infty, \Delta_\infty, \emptyset)$ is reachable from $(\emptyset, \mathtt{id}, E; u \stackrel{?}{=} v)$, the last step has to be an application of the Query rule on $u \stackrel{?}{=} v$, which means that $\Delta_\infty(u) = \Delta_\infty(v)$. We conclude by the invariant I_1 that $u =_E v$.

Let us now face the *only if* direction. The equalities associated with a configuration $K = \langle \Gamma \mid \Delta \mid \Phi\rangle$ are defined as

$$\mathcal{E}q(K) = \{u = v \mid \Delta(u) = \Delta(v)\} \cup \{u = v \mid u = v \in \Phi\}$$

It should be noticed that $\mathcal{E}q(K)$ does not contain the queries occurring in Φ. We first prove the following invariants:

$$I_2(\langle \Gamma \mid \Delta \mid \Phi \rangle) = \forall t_1, \ldots, t_n \in T(\Sigma), \Gamma(\Delta(f(t_1, \ldots, t_n))) \neq \bot \Rightarrow \forall i, \Gamma(\Delta(t_i)) \neq \bot \wedge f(t_1, \ldots, t_n) \in \Gamma(\Delta(t_i))$$
$$I_3(K) = \forall u, v \in T(\Sigma), u =_E v \Rightarrow u =_{\mathcal{E}q(K)} v$$

- $I_2(K_0)$ holds trivially since Γ_0 is undefined for all terms.
- Let us prove that if $K \rightarrow K'$ and $I_2(K)$ then $I_2(K')$. By case on the last rule applied.
  - REMOVE: immediate since Γ and Δ are unchanged.
  - CONGR: We shall first prove that for all terms v, if $\Gamma'(\Delta'(v))$ is defined then so is $\Gamma(\Delta(v))$.
    Let us assume that CONGR has been applied on the equation $a = b$. We distinguish the two following cases: if $\Delta(v) \neq \Delta(a)$ and $\Delta(v) \neq \Delta(b)$ then $\Gamma'(\Delta'(v)) = \Gamma(\Delta(v))$ else $\Delta(v) = \Delta(a)$ or $\Delta(v) = \Delta(b)$ hence $\Gamma(\Delta(v)))$ is defined since $\Gamma(\Delta(a)))$ and $\Gamma(\Delta(b)))$ have to be defined in order to apply CONGR.
    Using the above property, we can apply the induction hypothesis and then conclude as in the proof of I_2 in the standard case.
  - ADDTERM. Let us first notice the immediate property P that for all terms v, if $\Gamma(\Delta(v))$ is defined then $\Gamma'(\Delta'(v))$ remains also defined and $\Gamma(\Delta(v)) \subseteq \Gamma'(\Delta'(v))$. Now, if $\Gamma'(\Delta'(f(t_1, \ldots, t_n)))$ is defined this means that either $\Gamma(\Delta(f(t_1, \ldots, t_n)))$ is defined or $f(t_1, \ldots, t_n) = f(\vec{a})$. We distinguish these two cases:
    * If $\Gamma(\Delta(f(t_1, \ldots, t_n))) \neq \bot$ then we can apply the induction hypothesis and get that $\Gamma(\Delta(t_i))$ is defined and contains $f(t_1, \ldots, t_n)$. We conclude by $P(t_i)$.
    * If $f(t_1, \ldots, t_n) = f(\vec{a})$ then $\Gamma(\Delta(t_i)) \neq \bot$ since ADDTERM applies. By construction $f(t_1, \ldots, t_n)$ is in $\Gamma'(\Delta(t_i))$ for each t_i.
- I_3 is immediate since $\mathcal{E}q(K_0)$ contains E, and if $K \rightarrow K'$, then $\mathcal{E}q(K) \subseteq \mathcal{E}q(K')$.

By the termination property, there exists an irreducible configuration K_∞ reachable from $\langle \emptyset \mid \mathtt{id} \mid E; u \stackrel{?}{=} v \rangle$ which is either of the form $\langle \Gamma_\infty \mid \Delta_\infty \mid \emptyset \rangle$ or $\langle \Gamma_\infty \mid \Delta_\infty \mid u \stackrel{?}{=} v \rangle$ with $\Delta_\infty(u) \neq \Delta_\infty(v)$. The first case is immediate. In the last case, since $u =_E v$ and $\Phi_\infty = u \stackrel{?}{=} v$, by I_3 we have $u =_{\Delta_\infty} v$. Furthermore, since ADDTERM does not apply, $\Gamma_\infty(\Delta_\infty(u))$ and $\Gamma_\infty(\Delta_\infty(v))$ are defined. We shall conclude (*ad absurdum*) by proving that for any irreducible configuration $K_\infty = \langle \Gamma_\infty \mid \Delta_\infty \mid \Phi_\infty \rangle$ and for all terms u and v in $T(\Sigma)$ such that $\Gamma_\infty(\Delta_\infty(u))$ and $\Gamma_\infty(\Delta_\infty(v))$ are defined and $u =_{\Delta_\infty} v$ then $\Delta_\infty(u) = \Delta_\infty(v)$.

- We first prove the following congruence property of Δ_∞:

  For all $f(\vec{u})$ and $f(\vec{v})$ in $T(\Sigma)$ such that $\Gamma_\infty(\Delta_\infty(f(\vec{u})))$ and $\Gamma_\infty(\Delta_\infty(f(\vec{v})))$ are defined and $\Delta_\infty(\vec{u}) = \Delta_\infty(\vec{v})$ then $\Delta_\infty(f(\vec{u})) = \Delta_\infty(f(\vec{v}))$.

- The case where $\vec{u}$ is syntactically equivalent to $\vec{v}$ is immediate.
- Otherwise, there exists along the reduction path from K_0 to K_∞ a configuration $\langle \Gamma \mid \Delta \mid \Phi \rangle$ such that $\Delta(\vec{u}) = \Delta(\vec{v})$ holds for the first time. We then distinguish two cases. Either $\Gamma(\Delta(\vec{u}))$ and $\Gamma(\Delta(\vec{v}))$ are both defined and we can conclude as in the standard case, or at least one of them is undefined. Let us assume without loss of generality that in that case $f(\vec{v})$ is added after $f(\vec{u})$: this means that there exists a configuration $K' = \langle \Gamma' \mid \Delta' \mid \Phi' \rangle$ such that $\Delta'(\vec{u}) = \Delta'(\vec{v})$, $\Gamma'(\Delta'(f(\vec{u})))$ is defined, $\Gamma'(\Delta'(f(\vec{v})))$ is undefined, and $K' \to K'' = \langle \Gamma'' \mid \Delta'' \mid \Phi'' \rangle$ where $\Gamma''(\Delta''(f(\vec{v})))$ is defined. K'' is necessarily obtained by an application of AddTerm on $f(\vec{v})$. Since $\Gamma'(\Delta'(f(\vec{u})))$ is defined and $I_2(K')$ holds, we have $f(\vec{u}) \in \Gamma'(\Delta'(a))$ for all a in $\vec{v}$. The rule AddTerm has thus to add the equation $f(\vec{u}) = f(\vec{v})$ to Φ'. The result follows.

– Finally, we proceed by induction on the size of the proof of $u =_{\Delta_\infty} v$. The result is immediate when u is syntactically equivalent to v. Otherwise, we distinguish the two following cases:
  - If $u \equiv f(\vec{u}) =_{\Delta_\infty} f(\vec{v}) \equiv v$ has no equational step at the root then $\vec{u} =_{\Delta_\infty} \vec{v}$ and all the subproofs $u_i =_{\Delta_\infty} v_i$ are strictly smaller than $u =_{\Delta_\infty} v$. Since I_2 holds on K_∞, $\Gamma_\infty(\Delta_\infty(a))$ is defined for all a in $\vec{u}$ or $\vec{v}$. By induction hypothesis $\Delta_\infty(u_i) = \Delta_\infty(v_i)$. We then conclude by the above property.
  - If there is an equational step at the root in the proof, the proof has the following shape $u =_{\Delta_\infty} u' \leftrightarrow^{\Lambda}_{\Delta_\infty} v' =_{\Delta_\infty} v$. By definition of $=_{\Delta_\infty}$, $\Delta_\infty(u') = \Delta_\infty(v')$ hence there must exist two configurations $K = \langle \Gamma \mid \Delta \mid \Phi \rangle$ and $K' = \langle \Gamma' \mid \Delta' \mid \Phi' \rangle$ such that $K \to K'$, $\Delta(u') \neq \Delta(v')$ and $\Delta'(u') = \Delta'(v')$. In that case, K' is obtained from K by an application of Congr on an equation $u'' = v''$ where $\Delta(u') = \Delta(u'')$, $\Delta(v') = \Delta(v'')$ and $\Gamma(\Delta(u''))$ and $\Gamma(\Delta(v''))$ are defined. Therefore, by P, $\Gamma_\infty(\Delta_\infty(u'))$ and $\Gamma_\infty(\Delta_\infty(v'))$ are also defined. Applying the induction hypothesis on the subproofs $u =_{\Delta_\infty} u'$ and $v =_{\Delta_\infty} v'$ yields $\Delta_\infty(u) = \Delta_\infty(u')$ and $\Delta_\infty(v) = \Delta_\infty(v')$. We conclude by transitivity.

Example 3. From $K_0 = \langle \emptyset \mid \mathtt{id} \mid \Phi_0 \rangle$ where Φ_0 is $a = b; f(a) = t; f(b) = u; t \stackrel{?}{=} u$ we can get the following configurations:

K_i	$\langle \Gamma \mid \Delta \mid \Phi \rangle$	Rule
K_0	$\langle \emptyset \mid \mathtt{id} \mid a=b; f(a)=t; f(b)=u; t \stackrel{?}{=} u \rangle$	AddTerm* on a, b
K_1	$\langle a \mapsto \{\}, b \mapsto \{\} \mid \mathtt{id} \mid a=b; f(a)=t; f(b)=u; t \stackrel{?}{=} u \rangle$	Congr
K_2	$\langle \Delta_2(b) \mapsto \{\} \mid \{a=b\} + \mathtt{id} \mid f(a)=t; f(b)=u; t \stackrel{?}{=} u \rangle$	AddTerm* on $f(a), t$
K_3	$\langle \Delta_2(b) \mapsto \{f(a)\}, \ldots \mid \Delta_2 \mid \Phi_2 \rangle$	Congr
K_4	$\langle \Delta_4(b) \mapsto \{f(a)\}, \ldots \mid \{f(a)=t\} + \Delta_2 \mid f(b)=u; t \stackrel{?}{=} u \rangle$	AddTerm on $f(b)$
K_5	$\langle \ldots \mid \Delta_4 \mid f(a)=f(b); f(b)=u; t \stackrel{?}{=} u \rangle$	Congr* AddTerm*
K_6	$\langle \ldots \mid \{t=u\} + \ldots \mid t \stackrel{?}{=} u \rangle$	Query
K_7	$\langle \ldots \mid \{t=u\} + \ldots \mid \emptyset \rangle$	

4 Handling Commutative Symbols

Let Σ_C be the subset of Σ corresponding to the commutative symbols. We denote by $=_{E,C}$ the equational theory induced by E and the commutativity of the symbols of Σ_C.

We suppose given $\leq_{T(\Sigma)}$ a total ordering on the terms of $T(\Sigma)$ and we define a function which sorts vectors of $T(\Sigma)$ only if its extra parameter is a commutative symbol. More formally,

$$\mathtt{sort}(f, \vec{u}) = \text{if } f \in \Sigma \setminus \Sigma_C \text{ then } \vec{u} \text{ else } (\vec{u} \text{ sorted by } \leq_{T(\Sigma)})$$

In order to handle commutative symbols in the incremental algorithm[2], we only have to modify the rules Congr and AddTerm by changing their definitions of the set Φ' as follows:

Congr:
$\Phi' = \{f(\vec{a}) = f(\vec{b}) \mid \ f(\vec{a}) \in \mathcal{A} \wedge f(\vec{b}) \in \mathcal{B} \wedge \mathtt{sort}(f, \Delta'(\vec{a})) = \mathtt{sort}(f, \Delta'(\vec{b}))\}$

AddTerm:
$\Phi' = \{f(\vec{a}) = f(\vec{b}) \mid v \in \vec{a}, f(\vec{b}) \in C_v \wedge \mathtt{sort}(f, \Delta(\vec{a})) = \mathtt{sort}(f, \Delta(\vec{b}))\}$

It is obvious that the termination of $\rightarrow$ is preserved since sorting does not affect the number of equivalence classes of Δ and does not create new terms.

Surprisingly, the correctness proof is made difficult by the fact that in order to keep the "term preserving" property of Δ, we restrict the new equalities introduced in Φ to terms defined in Γ. For instance, if the symbol $+$ is commutative, the proof of $a_1 + (a_2 + a_3) =_C (a_3 + a_2) + a_1$ needs a middle term (either $a_1 + (a_3 + a_2)$ or $(a_2 + a_3) + a_1$) which is not in the original set of terms. We thus have to prove that our algorithm can detect that the query $a_1 + (a_2 + a_3) \stackrel{?}{=} (a_3 + a_2) + a_1$ is valid without using this middle term.

Theorem 5 (Correctness). *For any ground terms u, v, the equation $u =_{E,C} v$ holds iff there is a configuration $\langle \Gamma_\infty \mid \Delta_\infty \mid \emptyset \rangle$ reachable from $\langle \emptyset \mid \mathtt{id} \mid E; u \stackrel{?}{=} v \rangle$.*

Proof. The proof has the same structure as in section 3. We prove the invariants I_1 and $I2$ defined as in the incremental case by replacing $=_E$ by $=_{E,C}$. I_3 has to be slightly modified by adding some more equations to the set $\mathcal{E}q(K)$ in order to take care of commutativity.

The proof of the *if* direction applies almost verbatim. The only difference in the proof of I_1 is in the induction step where the new equation $f(u_1, u_2) = f(v_1, v_2)$ is added in Φ' by Congr or AddTerm because f is a commutative symbol and $\mathtt{sort}(f, \Delta'(u_1, u_2)) = \mathtt{sort}(f, \Delta'(v_1, v_2))$. This means that either $\Delta'(u_1) = \Delta'(v_1) \wedge \Delta'(u_2) = \Delta'(v_2)$ or $\Delta'(u_1) = \Delta'(v_2) \wedge \Delta'(u_2) = \Delta'(v_1)$. In

[2] Handling commutative symbols in the non-incremental case would require to modify the algorithm in such way that it would amount to add a kind of incrementality in the Congr rule. Incrementality is needed when a proof $u =_{E,C} v$ uses only C steps.

the first case, we conclude as in the non-commutative case. In the second case, we get that $u_1 =_{E,C} v_2$ and $u_2 =_{E,C} v_1$, hence

$$f(u_1, u_2) =_{E,C} f(v_2, v_1) =_{E,C} f(v_1, v_2)$$

since the equality $f(x, y) = f(y, x) \in C \subset E \cup C$.

Let us now face the *only if* direction. The set of equations associated to a configuration $K = \langle \Gamma \mid \Delta \mid \Phi \rangle$ with respect to a set of ground terms $\mathcal{G}$ is

$$\begin{aligned} \mathcal{E}q_{\mathcal{G}}(K) = & \{u = v \mid \Delta(u) = \Delta(v)\} \ \cup \\ & \{u = v \mid u = v \in \Phi\} \ \cup \\ & \left\{ f(u_1, u_2) = f(u_2, u_1) \;\middle|\; \begin{array}{l} f \in \Sigma_C, \\ \left(\begin{array}{l} \Gamma(\Delta(u_i)) \neq \bot \ \vee \\ u_i \text{ is a subterm of a term in } \mathcal{G} \ \vee \\ u_i \text{ occurs as a subterm in a} \\ \quad \text{non-trivial equation of } \Phi \end{array} \right) \end{array} \right\} \end{aligned}$$

It should be noticed that the queries and the equations $u = u$ of Φ do not affect the definition of $\mathcal{E}q$. I_2 is defined as in the incremental case, and

$$I_3(K) = \forall u, v \in T(\Sigma), u =_{E,C} v \Rightarrow u =_{\mathcal{E}q_{\{u,v\}}(K)} v$$

The proof of I_2 is exactly the same as in the standard case, since the modification does not affect the first two components of the configurations. However, the invariant I_3 is no longer obvious as in section 3.

– $I_3(K_0)$ is $\forall u, v \in \mathcal{T} \ u =_{E,C} v \Rightarrow u =_{\mathcal{E}q_{\{u,v\}}(K_0)} v$, where

$$\begin{aligned} \mathcal{E}q_{\{u,v\}}(K_0) = & \{a = b \mid \Delta(a) = \Delta(b)\} \ \cup \\ & \{a = b \mid a = b \in E\} \ \cup \\ & \left\{ f(u_1, u_2) = f(u_2, u_1) \;\middle|\; \begin{array}{l} f \in \Sigma_C, \\ \left(\begin{array}{l} u_i \text{ is a subterm of } u \text{ or } v \ \vee \\ u_i \text{ occurs as a subterm in a} \\ \quad \text{non-trivial equation of } E \end{array} \right) \end{array} \right\} \end{aligned}$$

$I_3(K_0)$ is proved by induction on the size of the sequence of equational steps π of $u =_{E,C} v$: If π has length 0 then u and v are syntactically equal and the result is immediate. Otherwise,

- If there is no equational step at the root then $u = f(u_1, \ldots, u_n)$, $v = f(v_1, \ldots, v_n)$ and for all $i \in \{1..n\}$ there is a proof π_i of $u_i =_{E,C} v_i$ obtained by projection of π. Hence each π_i is strictly smaller than π. So, by induction hypothesis, $u_i =_{\mathcal{E}q_{\{u_i,v_i\}}(K_0)} v_i$. Since u_i and v_i are respectively subterms of u and v, it is clear that $\mathcal{E}q_{\{u_i,v_i\}}(K_0) \subseteq \mathcal{E}q_{\{u,v\}}(K_0)$. Hence $u_i =_{\mathcal{E}q_{\{u,v\}}(K_0)} v_i$ and by the congruence property of equational theories, we get that $u =_{\mathcal{E}q_{\{u,v\}}(K_0)} v$.
- If there is at least an equational step at the root using an equation of E, the proof π has the following shape $u \leftrightarrow^*_{E,C} u' \leftrightarrow^{\Lambda}_{E} v' \leftrightarrow^*_{E,C} v$. The

sub-proofs π_1 of $u =_{E,C} u'$ and π_2 of $v' =_{E,C} v$ are smaller than π: by induction hypothesis, $u =_{\mathcal{E}q_{\{u,u'\}}(K_0)} u'$ and $v' =_{\mathcal{E}q_{\{v',v\}}(K_0)} v$. Since the equation $u' = v'$ belongs to E, the sets $\{u' = v'\}$, $\mathcal{E}q_{\{u,u'\}}(K_0)$ and $\mathcal{E}q_{\{v',v\}}(K_0)$ are all included in $\mathcal{E}q_{\{u,v\}}(K_0)$. We conclude by transitivity.

- If there is exactly one equational step at the root, which is a C-step, then the proof π is of the form:

$$u \equiv f(u_1, u_2) \leftrightarrow_{E,C}^{(\neq\Lambda)^*} f(u_1', u_2') \leftrightarrow_{E,C}^{\Lambda} f(u_2', u_1') \leftrightarrow_{E,C}^{(\neq\Lambda)^*} v \equiv f(v_1, v_2)$$

and there exist $\pi_1 : u_1 =_{E,C} u_1' =_{E,C} v_2$ and $\pi_2 : u_2 =_{E,C} u_2' =_{E,C} v_1$ obtained by projection of π. By induction hypothesis on π_1 and π_2, $u_1 =_{\mathcal{E}q_{\{u_1,v_2\}}(K_0)} v_2$ and $u_2 =_{\mathcal{E}q_{\{u_1,v_2\}}(K_0)} v_1$. Since u_1, u_2, v_1 and v_2 are subterms of u or v, $\mathcal{E}q_{\{u_1,v_2\}}(K_0)$ and $\mathcal{E}q_{\{u_2,v_1\}}(K_0)$ are included in $\mathcal{E}q_{\{u,v\}}(K_0)$. Moreover by construction, $\mathcal{E}q_{\{u,v\}}(K_0)$ contains the equation $f(u_1, u_2) = f(u_2, u_1)$. u and v have the following proof of $\mathcal{E}q_{\{u,v\}}(K_0)$-equality:

$$u \equiv f(u_1, u_2) \leftrightarrow_{\mathcal{E}q_{\{u,v\}}(K_0)}^{\Lambda} f(u_2, u_1) \leftrightarrow^{\pi_1} f(u_2, v_2) \leftrightarrow^{\pi_2} f(v_1, v_2) \equiv v$$

- If there are at least two equational C-steps at the root, then the proof π is of the form:

$$u \equiv f(u_1, u_2) \overset{(\neq\Lambda)^*}{\underset{E,C}{\longleftrightarrow}} f(u_1', u_2') \overset{\Lambda}{\underset{C}{\leftrightarrow}} f(u_2', u_1') \overset{(\neq\Lambda)^*}{\underset{E,C}{\longleftrightarrow}} f(u_2'', u_1'') \overset{\Lambda}{\underset{C}{\leftrightarrow}} f(u_1'', u_2'') \overset{*}{\underset{E,C}{\longleftrightarrow}} v$$

where $u_1 =_{E,C} u_1''$ and $u_2 =_{E,C} u_2''$. We can rebuild a strictly smaller middle proof $u \equiv f(u_1, u_2) =_{E,C} f(u_1'', u_2'') =_{E,C} v$ to which we can apply the induction hypothesis.

– The induction step showing that if $K \to K'$ and $I_3(K)$ then $I_3(K')$ follows immediately from the fact that $\mathcal{E}q_{\mathcal{G}}(K) \subseteq \mathcal{E}q_{\mathcal{G}}(K')$:

- Query : obviously, a query in Φ does not affect the definition of $\mathcal{E}q_{\mathcal{G}}$, hence $\mathcal{E}q_{\mathcal{G}}(K) = \mathcal{E}q_{\mathcal{G}}(K')$.
- Remove : If this rule removes an equation $u = v$ in Φ, it was already in Δ. If u and v are syntactically equal, the equation $u = v$ is trivial, hence does not induce any commutative equation in the third part of $\mathcal{E}q_{\mathcal{G}}(K)$. Otherwise, the induced equations are still in $\mathcal{E}q_{\mathcal{G}}(K')$ since u and v have to be added by the rule AddTerm before the rule Congr added an equation equivalent to $u = v$ to Δ, hence $\Gamma(\Delta(u))$ and $\Gamma(\Delta(v))$ are defined and by I_2, the subterms of u and v enjoy the same property. As a conclusion, $\mathcal{E}q_{\mathcal{G}}(K) = \mathcal{E}q_{\mathcal{G}}(K')$.
- Congr : If this rule moves an equation $u = v$ from Φ to Δ, this means that $\Gamma(\Delta(u))$ and $\Gamma(\Delta(v))$ are defined; we can conclude as above.
- AddTerm : This rule increases Φ and makes $\Gamma(\Delta())$ defined on a larger set of terms, hence it is obvious that $\mathcal{E}q_{\mathcal{G}}(K) \subseteq \mathcal{E}q_{\mathcal{G}}(K')$.

By the termination property, there exists an irreducible configuration K_∞ reachable from $\langle \emptyset \mid \mathtt{id} \mid E; u \stackrel{?}{=} v \rangle$ which is either of the form $\langle \Gamma_\infty \mid \Delta_\infty \mid \emptyset \rangle$ or

$\langle \Gamma_\infty \mid \Delta_\infty \mid u \stackrel{?}{=} v \rangle$ with $\Delta_\infty(u) \neq \Delta_\infty(v)$. The first case is immediate. In the last case, since $u =_{E,C} v$ and $\Phi_\infty = u \stackrel{?}{=} v$, by I_3 we have $u =_{\mathcal{E}q_{\{u,v\}}(K_\infty)} v$.

Furthermore, since AddTerm does not apply $\Gamma_\infty(\Delta_\infty(u))$ and $\Gamma_\infty(\Delta_\infty(v))$ are defined and by I_2, this is also the case for the subterms of u and v.

$$\begin{aligned}\mathcal{E}q_{\{u,v\}}(K_\infty) &= \{a = b \mid \Delta_\infty(a) = \Delta_\infty(b)\} \ \cup \\ &\quad \left\{ f(u_1,u_2) = f(u_2,u_1) \,\middle|\, f \in \Sigma_C, \left(\Gamma_\infty(\Delta_\infty(u_i)) \neq \bot\right) \right\} \\ &= \mathcal{E}q_\emptyset(K_\infty)\end{aligned}$$

We shall conclude (*ad absurdum*) by proving that for any irreducible configuration $K_\infty = \langle \Gamma_\infty \mid \Delta_\infty \mid \Phi_\infty \rangle$, for all terms u and v in $T(\Sigma)$ such that $\Gamma_\infty(\Delta_\infty(u))$ and $\Gamma_\infty(\Delta_\infty(v))$ are defined, if $u =_{\mathcal{E}q_\emptyset(K_\infty)} v$ then $\Delta_\infty(u) = \Delta_\infty(v)$.

- As in the standard case, we first need a congruence property on Δ_∞:

  For all $f(\vec{u})$ and $f(\vec{v})$ in $T(\Sigma)$ such that $\Gamma_\infty(\Delta_\infty(f(\vec{u})))$ and $\Gamma_\infty(\Delta_\infty(f(\vec{v})))$ are defined and $\mathtt{sort}(f, \Delta_\infty(\vec{u})) = \mathtt{sort}(f, \Delta_\infty(\vec{v}))$ then $\Delta_\infty(f(\vec{u})) = \Delta_\infty(f(\vec{v}))$.

  - The case where $\vec{u}$ is syntactically equivalent to $\vec{v}$ is immediate.
  - Otherwise, there exists along the reduction path from K_0 to K_∞ a configuration $\langle \Gamma \mid \Delta \mid \Phi \rangle$ such that $\mathtt{sort}(f, \Delta(\vec{u})) = \mathtt{sort}(f, \Delta(\vec{v}))$ holds for the first time. We then distinguish two cases. Either $\Gamma(\Delta(\vec{u}))$ and $\Gamma(\Delta(\vec{v}))$ are both defined or at least one of them is undefined. In the first case, the rule Congr will add the equation $f(\vec{u}) = f(\vec{v})$ to Φ, and this equation will eventually become a part of Δ_∞. In the second case, let us assume without loss of generality that $f(\vec{v})$ is added after $f(\vec{u})$: this means that there exists a configuration $K' = \langle \Gamma' \mid \Delta' \mid \Phi' \rangle$ such that $\mathtt{sort}(f, \Delta'(\vec{u})) = \mathtt{sort}(f, \Delta'(\vec{v}))$, $\Gamma'(\Delta'(f(\vec{u})))$ is defined, $\Gamma'(\Delta'(f(\vec{v})))$ is undefined, and $K' \to K'' = \langle \Gamma'' \mid \Delta'' \mid \Phi'' \rangle$ where $\Gamma''(\Delta''(f(\vec{v})))$ is defined. K'' is necessarily obtained by an application of AddTerm on $f(\vec{v})$. Since $\Gamma'(\Delta'(f(\vec{u})))$ is defined and $I_2(K')$ holds, we have $f(\vec{u}) \in \Gamma'(\Delta'(a))$ for all a in $\vec{v}$. The rule AddTerm has thus to add the equation $f(\vec{u}) = f(\vec{v})$ to Φ'. The result follows.
- Finally, we proceed by induction on the size of the proof of $u =_{\mathcal{E}q_\emptyset(K_\infty)} v$. The result is immediate when u is syntactically equivalent to v. Otherwise, we distinguish the two following cases:
  - If $u \equiv f(\vec{u}) =_{\mathcal{E}q_\emptyset(K_\infty)} f(\vec{v}) \equiv v$ has no equational step at the root then $\vec{u} =_{\mathcal{E}q_\emptyset(K_\infty)} \vec{v}$ and all the subproofs $u_i =_{\mathcal{E}q_\emptyset(K_\infty)} v_i$ are strictly smaller than $u =_{\mathcal{E}q_\emptyset(K_\infty)} v$. Since I_2 holds on K_∞, $\Gamma_\infty(\Delta_\infty(a))$ is defined for all a in $\vec{u}$ or $\vec{v}$. By induction hypothesis $\Delta_\infty(u_i) = \Delta_\infty(v_i)$. We then conclude by the above property.
  - If there is at least an equational step at the root using an equation of Δ_∞, the proof π has the following shape $u \leftrightarrow^*_{\mathcal{E}q_\emptyset(K_\infty)} u' \leftrightarrow^{\Lambda}_{\Delta_\infty} v' \leftrightarrow^*_{\mathcal{E}q_\emptyset(K_\infty)} v$, where $\Delta_\infty(u') = \Delta_\infty(v')$. The sub-proofs π_1 of $u =_{\mathcal{E}q_\emptyset(K_\infty)} u'$ and π_2 of $v' =_{\mathcal{E}q_\emptyset(K_\infty)} v$ are smaller than π and $\Gamma_\infty(\Delta_\infty(u'))$ and $\Gamma_\infty(\Delta_\infty(v'))$ are defined (indeed the rule Congr cannot add to Δ an equation equivalent to $u' = v'$ if $\Gamma_\infty(\Delta_\infty(u'))$ or $\Gamma_\infty(\Delta_\infty(v'))$ are not defined): by induction

hypothesis, $\Delta_\infty(u) = \Delta_\infty(u')$ and $\Delta_\infty(v') = \Delta_\infty(v)$. We conclude by transitivity.

- If there is exactly one equational step at the root, which is a C-step, then the proof π is of the form:

$$u \equiv f(u_1, u_2) \leftrightarrow^{(\neq\Lambda)^*}_{\mathcal{E}_{q_\emptyset}(K_\infty)} f(u'_1, u'_2) \leftrightarrow^{\Lambda}_{C} f(u'_2, u'_1) \leftrightarrow^{(\neq\Lambda)^*}_{\mathcal{E}_{q_\emptyset}(K_\infty)} v \equiv f(v_1, v_2)$$

  and there exist $\pi_1 : u_1 =_{\mathcal{E}_{q_\emptyset}(K_\infty)} u'_1 =_{\mathcal{E}_{q_\emptyset}(K_\infty)} v_2$ and $\pi_2 : u_2 =_{\mathcal{E}_{q_\emptyset}(K_\infty)} u'_2 =_{\mathcal{E}_{q_\emptyset}(K_\infty)} v_1$ obtained by projection of π. By $I_2(K_\infty)$, since u_1, u_2, v_1 and v_2 are subterms of u or v, $\Gamma_\infty(\Delta_\infty())$ is defined on them, so we can apply the induction hypothesis and get that $\Delta_\infty(u_1) = \Delta_\infty(v_2)$ and $\Delta_\infty(u_2) = \Delta_\infty(v_1)$. Hence $\mathtt{sort}(f, \Delta_\infty(u_1, u_2)) = \mathtt{sort}(f, \Delta_\infty(v_1, v_2))$ and we conclude by the above property.
- If there are at least two equational C-steps at the root, then the proof π is of the form:

$$\begin{aligned} u \equiv f(u_1, u_2) &\longleftrightarrow^{(\neq\Lambda)^*}_{\mathcal{E}_{q_\emptyset}(K_\infty)} f(u'_1, u'_2) \leftrightarrow^{\Lambda}_{C} f(u'_2, u'_1) \\ &\longleftrightarrow^{(\neq\Lambda)^*}_{\mathcal{E}_{q_\emptyset}(K_\infty)} f(u''_2, u''_1) \leftrightarrow^{\Lambda}_{C} f(u''_1, u''_2) \longleftrightarrow^{*}_{\mathcal{E}_{q_\emptyset}(K_\infty)} v \end{aligned}$$

  where $u_1 =_{\mathcal{E}_{q_\emptyset}(K_\infty)} u''_1$ and $u_2 =_{\mathcal{E}_{q_\emptyset}(K_\infty)} u''_2$. We can rebuild a strictly smaller middle proof $u \equiv f(u_1, u_2) =_{\mathcal{E}_{q_\emptyset}(K_\infty)} f(u''_1, u''_2) =_{\mathcal{E}_{q_\emptyset}(K_\infty)} v$ to which we can apply the induction hypothesis.

Example 4. From $K_0 = \langle \emptyset \mid \mathtt{id} \mid \Phi_0 \rangle$ where Φ_0 is $a_1 + (a_2 + a_3) \stackrel{?}{=} (a_3 + a_2) + a_1$ we can get the following configurations:

K_i	$\langle \Gamma \mid \Delta \mid \Phi \rangle$	Rule
K_0	$\langle \emptyset \mid \mathtt{id} \mid \Phi_0[a_1 + (a_2 + a_3)] \rangle$	AddTerm*
K_1	$\langle a_2 \mapsto \{a_2 + a_3\}, \dots \mid \mathtt{id} \mid \Phi_0[a_3 + a_2] \rangle$	AddTerm
K_2	$\langle a_2 \mapsto \{a_2 + a_3, a_3 + a_2\}, \dots \mid \mathtt{id} \mid a_2 + a_3 = a_3 + a_2; \Phi_0 \rangle$	Congr
K_3	$\langle a_1 \mapsto \{\dots\}, \dots \mid \{a_2 + a_3 = a_3 + a_2\} + \mathtt{id} \mid \Phi_0[(a_3 + a_2) + a_1] \rangle$	AddTerm
K_4	$\langle \Gamma_4 \mid \Delta_3 \mid a_1 + (a_2 + a_3) = (a_3 + a_2) + a_1; \Phi_0 \rangle$	Congr
K_5	$\langle \Gamma_5 \mid \{a_1 + (a_2 + a_3) = (a_3 + a_2) + a_1\} + \Delta_3 \mid \Phi_0 \rangle$	Query
K_6	$\langle \Gamma_5 \mid \Delta_5 \mid \emptyset \rangle$	

5 Conclusion and Related Works

We have presented an incremental rule based congruence closure algorithm for which rigorous correctness proofs are given. Following the original works on this algorithm, our inference system constructs a union-find data structure that contains only ground terms from the initial set of equalities. The way our framework has been easily extended to handle commutative symbols is promising and we leave for future works a generalized approach for handling others theories.

Related Works. Original papers [5, 10, 13] have presented non-incremental congruence closure algorithms using pseudo-code notations. Our first inference system can be seen as a clean reformulation of these algorithms for which a rigorous correctness proof is given.

Completion like methods [2, 1, 8] demystified congruence closure algorithms using the framework of ground completion. While this approach allows for rigorous correctness proofs, it fails to produce a union-find data structure used by the pattern-matching algorithms underlying automated theorem provers.

Recent works [12, 11] have presented an incremental congruence closure algorithm using pseudo-code notations which is not modular with respect to incrementality and for which it is then more difficult to produce proofs. Furthermore, the initial *Currifying* transformation applied to the terms makes difficult the treatment of commutative symbols.

References

1. L. Bachmair, I. V. Ramakrishnan, A. Tiwari, and L. Vigneron. Congruence closure modulo associativity and commutativity. In H. Kirchner and C. Ringeissen, editors, *Proceedings of FroCoS 2000 Nancy (France)*, volume 1794, pages 245–259. Springer-Verlag, 2000.
2. L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *Journal of Automated Reasoning*, 31(2):129–168, 2003.
3. C. Barrett and S. Berezin. CVC Lite: A new implementation of the cooperating validity checker. In R. Alur and D. A. Peled, editors, *16th International Conference on Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 515–518, Boston, MA, USA, July 2004. Springer-Verlag.
4. D. Detlefs, G. Nelson, and J. B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.
5. P. J. Downey, R. Sethi, and R. E. Tarjan. Variations on the common subexpressions problem. *J. ACM*, 27(4):771–785, 1980.
6. J.-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. ICS: Integrated Canonization and Solving (Tool presentation). In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of CAV'2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 246–249. Springer-Verlag, 2001.
7. J.-M. Hullot. Canonical forms and unification. In *Proc. 5th Conf. on Automated Deduction, Les Arcs, France, LNCS 87*. Springer-Verlag, July 1980.
8. D. Kapur. Shostak's congruence closure as completion. In H. Comon, editor, *Proceedings of the 8th International Conference on Rewriting Techniques and Applications*, volume 1232. Springer-Verlag, 1997.
9. G. Nelson. *Techniques for Program Verification*. PhD thesis, Stanford University, 1980. available from University Microfilms International.
10. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27:356–364, 1980.
11. R. Nieuwenhuis and A. Oliveras. Congruence closure with integer offsets. 2003.
12. R. Nieuwenhuis and A. Oliveras. Proof-Producing Congruence Closure. In J. Giesl, editor, *Proceedings of RTA'05 (Nara, Japan)*, volume 3467 of *Lecture Notes in Computer Science*, pages 453–468. Springer, 2005.
13. Robert E. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21(2):583–585, july 1978.