Order-Sorted Equality Enrichments Modulo Axioms*

Raúl Gutiérrez, José Meseguer, and Camilo Rocha

Department of Computer Science
University of Illinois at Urbana-Champaign

Abstract. Built-in equality and inequality predicates based on comparison of canonical forms in algebraic specifications are frequently used because they are handy and efficient. However, their use places algebraic specifications with initial algebra semantics beyond the pale of theorem proving tools based, for example, on explicit or inductionless induction techniques, and of other formal tools for checking key properties such as confluence, termination, and sufficient completeness. Such specifications would instead be amenable to formal analysis if an equationally-defined equality predicate enriching the algebraic data types were to be added to them. Furthermore, having an equationally-defined equality predicate is very useful in its own right, particularly in inductive theorem proving. Is it possible to *effectively* define a theory transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ that extends an algebraic specification $\mathcal{E}$ to a specification $\mathcal{E}^{\simeq}$ where equationally-defined equality predicates have been added? This paper answers this question in the affirmative for a broad class of order-sorted conditional specifications $\mathcal{E}$ that are sort-decreasing, ground confluent, and operationally terminating modulo axioms B and have a subsignature of constructors. The axioms B can consist of associativity, or commutativity, or associativity-commutativity axioms, so that the constructors are *free modulo* B. We prove that the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ preserves all the just-mentioned properties of $\mathcal{E}$. The transformation has been automated in Maude using reflection and it is used in Maude formal tools.

1 Introduction

It can be extremely useful, when reasoning about equational specifications with initial semantics, to have an explicit equational specification of the *equality predicate* as a binary Boolean-valued operator '$\simeq$'. For example, in *theorem proving* where the logic of universal quantifier-free formulas is automatically reduced to unconditional equational logic so that the formula $(u \neq v \vee w = r) \wedge q = t$ becomes equivalent to the equation $(not(u \simeq v)\ or\ w \simeq r)\ and\ q \simeq t = true$,

* This work has been supported in part by NSF Grant CCF 09-05584, the "Programa de Apoyo a la Investigación y Desarrollo" (PAID-02-11) of the Universitat Politècnica de València, the EU (FEDER), the spanish MICINN/MINECO under Grant TIN2010-21062-C02 and by the Generalitat Valenciana, ref. PROMETEO/2011/052.

and in *inductionless induction* where inductive proofs are reduced to proofs by consistency because any equation not holding inductively makes $true = false$.

An equationally-defined predicate can as well be useful in the *elimination of built-in equalities and inequalities* that often are introduced in algebraic specifications through built-in operators. Such built-in equalities and inequalities are not defined logically but operationally, for both expressiveness and efficiency reasons, by comparison of canonical forms. However, their non-logical character renders any formal reasoning about specifications using them impossible. In particular, the *use of formal tools* such as those checking termination, local confluence, or sufficient completeness of an algebraic specification is impossible with built-in equalities and inequalities, but becomes possible when they are replaced by an equationally axiomatized equality predicate '$\simeq$'. That is, the equality between t and t' is now expressed as $t \simeq t' = true$, and their inequality as $t \simeq t' = false$. Furthermore, the equality $t \simeq t'$ will *still be correct* when t and t' are *terms with variables*, whereas a built-in equality predicate will often give a *false negative* answer for such terms, even when the equations are confluent and terminating. For example, for natural number addition '+', defined by equations $x + 0 = x$ and $x + s(y) = s(x + y)$, the terms $x + y$ and $y + x$ are *already* in canonical form and a built-in equality predicate '$\equiv$' will evaluate $x + y \equiv y + x$ to *false*. Instead, $x + y \simeq y + x$ will remain in canonical form with '$\simeq$' and one can then inductively prove $x + y \simeq y + x = true$ using the equations defining '+' and '$\simeq$'.

In principle, the meta-theorem of Bergstra and Tucker [2] ensures that any computable data type can be axiomatized as an initial algebra defined by a finite number of Church-Rosser and terminating equations. This also means that such a computable data type *plus* its equality predicate is also finitely axiomatizable by a finite set of Church-Rosser and terminating equations. However, the Bergstra-Tucker result is *non-constructive* in the sense that it does not give an algorithm to actually obtain the equational specification of the data type with its equality predicate. Therefore, what would be highly desirable in practice is a general *constructive theory transformation* $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ that adds equationally-axiomatized equality predicates to an algebraic data type specification $\mathcal{E}$.

Such a transformation should be *as general as possible* for it to be useful in practice. For example, a transformation applicable only to "vanilla-flavored" specifications without support for types and subtypes, or that excludes conditional equations and rewriting modulo axioms would be extremely limited. The transformation should also come with *strong preservation properties*. For example, if $\mathcal{E}$ is ground confluent, ground operationally terminating, and sufficiently complete, then $\mathcal{E}^{\simeq}$ should also enjoy these same properties that are often essential both for executability and for a variety of formal reasoning forms.

These generality and property-preservation requirements on the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ are a tall order. For instance, if f is a free constructor symbol, then the equations $f(x_1, \ldots, x_n) \simeq f(y_1, \ldots, y_n) = x_1 \simeq y_1 \; and \ldots and \; x_n \simeq y_n$ and $f(x_1, \ldots, x_n) \simeq g(y_1, \ldots, y_m) = false$, for each constructor $g \neq f$ of same type, give a perfectly good and straightforward axiomatization of equality for f. But how can the equality predicate be defined when f satisfies, e.g., associativity and

commutativity axioms? Also, how should sorts and subsorts be dealt with? An even harder issue is the preservation of properties such as ground confluence, operational termination, and sufficient completeness. The difficulty is that for any given specification there are tools that can be used to prove such properties, but we need a proof that will work for *all* specifications in a very wide class. What we actually need are *metatheorems* ensuring that these properties are preserved under the transformation for *any* equational specification in the input class.

We present in this paper an effective theory transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ that satisfies the above-mentioned preservation properties. The class of equational theories $\mathcal{E}$ accepted as inputs to the transformation is quite general. Modulo mild syntactic requirements, it consists of all order-sorted theories $\mathcal{E}$ of the form $(\Sigma, E \uplus B)$ having a subsignature Ω of constructors and such that: (i) B is a set of associativity, or commutativity, or associativity-commutativity axioms[1]; (ii) the equations E can be conditional and are sort-decreasing, ground confluent, and operationally terminating; and (iii) the constructors Ω are *free modulo* B, i.e., there is an isomorphism $\mathcal{T}_{\Sigma/E \uplus B}|_{\Omega} \cong \mathcal{T}_{\Omega/B}$ of initial algebras.

Outline. Preliminaries on order-sorted equational specifications are presented in Section 2. Section 3 contains the definition and fundamental properties of an equality enrichment. Sections 4 and 5 present the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ and state its basic metatheorems. An overview on the implementation of the transformation, some of its practical consequences, and a case study are presented in Section 6. The extended version of this paper [9] contains the details and proofs of the mathematical development. The implementation of the transformation, the case study, and more examples are available at `http://camilorocha.info`.

2 Preliminaries

We assume basic knowledge on term rewriting [1] and order-sorted algebra [7].

Order-Sorted Signatures and Terms. We assume an *order-sorted signature* $\Sigma = (S, \leq, F)$ with a finite poset of sorts $(S, \leq)$ and a finite set of function symbols F. We also assume that the function symbols in F can be subsort overloaded and satisfy that if $f \in F_{w,s} \cap F_{w',s'}$ then $w \equiv_{\leq} w'$ implies $s \equiv_{\leq} s'$, where $\equiv_{\leq}$ denotes the equivalence relation generated by $\leq$ on S and $(w, s), (w', s') \in S^* \times S$. We say that $f : s_1 \cdots s_n \to s \in F$ is a *maximal typing* of f in Σ if there is no other $f : s'_1 \cdots s'_n \to s' \in F$ such that $s_i \leq s'_i$, $1 \leq i \leq n$, and $s \leq s'$. We let $X = \{X_s\}_{s \in S}$ be an S-sorted family of disjoint sets of variables with each X_s countably infinite. The set of Σ-terms of sort s is denoted by $T_{\Sigma}(X)_s$ and the set of ground terms of sort s is denoted by $T_{\Sigma,s}$, which we assume nonempty for each s. We let $\mathcal{T}_{\Sigma}(X)$ and $\mathcal{T}_{\Sigma}$ denote the corresponding order-sorted term algebras. The set of variables of a term t is written $\mathcal{V}ar(t)$ and is extended to

[1] Identity axioms are excluded from our transformation. However, by using the transformation described in [5] and subsort-overloaded operators, one can often extend our transformation to specifications that also include identity axioms.

sets of terms in the natural way. A *substitution* θ is a sorted mapping from a finite subset $\mathcal{D}om(\theta) \subseteq X$ to $T_\Sigma(X)$ and extends homomorphically in the natural way; $\mathcal{R}an(\theta)$ denotes the set of variables introduced by θ. The application of a substitution θ to a term t is denoted by $t\theta$ and the composition of two substitutions θ_1 and θ_2 is denoted by $\theta_1\theta_2$. A substitution θ is called *ground* iff $\mathcal{R}an(\theta) = \varnothing$. We assume that all order-sorted signatures are *preregular* [7], so that each Σ-term t has a *least sort* $ls(t) \in S$ such that $t \in T_\Sigma(X)_{ls(t)}$.

Order-Sorted Equational Theories. A *Σ-equation* is an expression $t = t'$ with $t \in T_\Sigma(X)_s$, $t' \in T_\Sigma(X)_{s'}$, and $s \equiv_{\leq} s'$. A *conditional Σ-equation* is a Horn clause $t = t'$ if C with $t = t'$ a Σ-equation and $C = \bigwedge_i u_i = v_i$ a finite conjunction of Σ-equations. An *equational theory* is a tuple (Σ, E) with Σ an order-sorted signature and E a finite set of conditional Σ-equations. For φ a conditional Σ-equation, $(\Sigma, E) \vdash \varphi$ iff φ can be proved from (Σ, E) by the deduction rules in [13] iff φ is valid in all models of (Σ, E) [13]. An equational theory (Σ, E) induces the congruence relation $=_E$ on $T_\Sigma(X)$ defined for any $t, u \in T_\Sigma(X)$ by $t =_E u$ iff $(\Sigma, E) \vdash (\forall X)\, t = u$. We let $\mathcal{T}_{\Sigma/E}(X)$ and $\mathcal{T}_{\Sigma/E}$ denote the quotient algebras induced by $=_E$ on the algebras $\mathcal{T}_\Sigma(X)$ and $\mathcal{T}_\Sigma$, respectively. We call $\mathcal{T}_{\Sigma/E}$ the *initial algebra* of (Σ, E) and call a conditional Σ-equation φ an *inductive consequence* of (Σ, E) iff $\mathcal{T}_{\Sigma/E} \models \varphi$, i.e., iff $(\forall \theta : X \longrightarrow T_\Sigma)(\Sigma, E) \vdash \varphi\theta$. A theory inclusion $(\Sigma, E) \subseteq (\Sigma', E')$, with $\Sigma \subseteq \Sigma'$ and $E \subseteq E'$, is called *protecting* iff the unique Σ-homomorphism $\mathcal{T}_{\Sigma/E} \longrightarrow \mathcal{T}_{\Sigma'/E'}|_\Sigma$ of the Σ-reduct of the initial algebra $\mathcal{T}_{\Sigma'/E'}$ is a Σ-isomorphism.

Executability Conditions. We assume that the set of equations of an equational theory can be decomposed into a disjoint union $E \uplus B$, with B a collection of axioms (such as associativity, and/or commutativity, and/or identity) for which there exists a *matching algorithm modulo B* producing a finite number of B-matching substitutions, or failing otherwise. Furthermore, we assume that all axioms in B are *sort-preserving*, i.e., for each $u = v \in B$ and substitution θ we have $ls(u\theta) = ls(v\theta)$. The conditional equations E can be oriented into a set of (possibly conditional) *(ground) sort-decreasing*, *(ground) operationally terminating* [12], and *(ground) confluent conditional* rewrite rules $\overrightarrow{E}$ modulo B. We let $\rightarrow_{E/B}$ denote the one-step rewrite relation induced by $\overrightarrow{E}$ modulo B on $T_\Sigma(X)$, and let $\rightarrow^*_{E/B}$ denote its reflexive and transitive closure. A set of rewrite rules $\overrightarrow{E}$ modulo B is: (i) sort-decreasing iff for each $t = t'$ if $C \in E$ and substitution θ we have $ls(t\theta) \geq ls(t'\theta)$ if $(\Sigma, E \uplus B) \vdash C\theta$; (ii) operationally terminating iff there is no infinite well-formed proof tree modulo B in $\overrightarrow{E}$ [5]; and (iii) confluent if for all $t, t', t'' \in T_\Sigma(X)$, if $t \rightarrow^*_{E/B} t'$ and $t \rightarrow^*_{E/B} t''$, then there is $u \in T_\Sigma(X)$ such that $t' \rightarrow^*_{E/B} u$ and $t'' \rightarrow^*_{E/B} u$. A set of rewrite rules $\overrightarrow{E}$ modulo B is *ground sort-decreasing*, *ground operationally terminating*, and *ground confluent* iff it is, respectively, sort-decreasing, operationally terminating, and confluent for ground terms. We let $t{\downarrow_{E/B}} \in T_{\Sigma,s}(X)$ denote the *E-canonical form* of t modulo B, i.e., $t \rightarrow^*_{E/B} t{\downarrow_{E/B}}$ and $t{\downarrow_{E/B}}$ cannot be further rewritten. Under the above assumptions $t{\downarrow_{E/B}}$ is unique up to B-equality.

Free Constructors Modulo. Given $\mathcal{E} = (\Sigma, E \uplus B)$ ground sort-decreasing, ground confluent, and ground operationally terminating modulo B, we say that $\Omega \subseteq \Sigma$ is a subsignature of *free constructors* modulo B iff Ω has the same poset of sorts as Σ and for each sort s in Σ and ground term $t \in T_{\Sigma,s}$ there is a $u \in T_{\Omega,s}$ satisfying $t =_{E \uplus B} u$ and, moreover, $v{\downarrow_{E/B}} =_B v$ for each $v \in T_{\Omega,s}$.

3 Equality Enrichments

An *equality enrichment* [14] of an equational theory $\mathcal{E}$ is an equational theory $\mathcal{E}^{\simeq}$ extending $\mathcal{E}$ that defines equality in $\mathcal{T}_{\mathcal{E}}$ as a Boolean-valued function, as stated in Definition 1. In this section we fix an order-sorted signature $\Sigma = (S, \leq, F)$ and an order-sorted equational theory $\mathcal{E} = (\Sigma, E)$ with initial algebra $\mathcal{T}_{\mathcal{E}}$.

Definition 1 (Equality Enrichment) (generalizes [14, Definition 68]). *An equational theory $\mathcal{E}^{\simeq} = (\Sigma^{\simeq}, E^{\simeq})$ is called an* equality enrichment *of $\mathcal{E}$, with $\Sigma^{\simeq} = (S^{\simeq}, \leq^{\simeq}, F^{\simeq})$ and $\Sigma = (S, \leq, F)$, iff*

- *$\mathcal{E}^{\simeq}$ is a protecting extension of $\mathcal{E}$;*
- *the poset of sorts of $\Sigma^{\simeq}$ extends $(S, \leq)$ by adding a new sort Bool that belongs to a new connected component, with constants $\top$ and $\bot$ such that $\mathcal{T}_{\mathcal{E}^{\simeq},Bool} = \{[\top], [\bot]\}$, with $\top \neq_{E^{\simeq}} \bot$; and*
- *for each connected component in $(S, \leq)$ there is a top sort $k \in S^{\simeq}$ and a binary commutative operator $\_ \simeq \_ : k\ k \longrightarrow Bool$ in $\Sigma^{\simeq}$, such that the following holds for any* ground terms *$t, u \in T_{\Sigma,k}$:*

$$\mathcal{E} \vdash t = u \iff \mathcal{E}^{\simeq} \vdash (t \simeq u) = \top, \tag{1}$$

$$\mathcal{E} \nvdash t = u \iff \mathcal{E}^{\simeq} \vdash (t \simeq u) = \bot. \tag{2}$$

An equality enrichment $\mathcal{E}^{\simeq}$ of $\mathcal{E}$ is called Boolean *iff it contains all the function symbols and equations making the elements of $\mathcal{T}_{\mathcal{E}^{\simeq},Bool}$ a two-element Boolean algebra.*

The equality predicate $\_ \simeq \_$ in $\mathcal{E}^{\simeq}$ is sound for inferring equalities and inequalities in the initial algebra $\mathcal{T}_{\mathcal{E}}$, even for terms with variables. The precise meaning of this claim is given by Proposition 1.

Proposition 1 (Equality Enrichment Properties). *Let $\mathcal{E}^{\simeq} = (\Sigma^{\simeq}, E^{\simeq})$ be an equality enrichment of $\mathcal{E}$. Then, for any Σ-equation $t = u$ with $X = \mathcal{V}ar(t) \cup \mathcal{V}ar(u)$:*

$$\mathcal{T}_{\mathcal{E}} \models (\forall X)\ t = u \iff \mathcal{T}_{\mathcal{E}^{\simeq}} \models (\forall X)\ (t \simeq u) = \top, \tag{3}$$

$$\mathcal{T}_{\mathcal{E}} \models (\exists X)\ \neg(t = u) \iff \mathcal{T}_{\mathcal{E}^{\simeq}} \models (\exists X)\ (t \simeq u) = \bot, \tag{4}$$

$$\mathcal{T}_{\mathcal{E}} \models (\forall X)\ \neg(t = u) \iff \mathcal{T}_{\mathcal{E}^{\simeq}} \models (\forall X)\ (t \simeq u) = \bot. \tag{5}$$

Note that by using an equality enrichment $\mathcal{E}^{\simeq}$ of $\mathcal{E}$, the problem of reasoning in $\mathcal{T}_{\mathcal{E}}$ about a universally quantified inequality $\neg(t = u)$ (abbreviated $t \neq u$) can be reduced to reasoning in $\mathcal{T}_{\mathcal{E}^{\simeq}}$ about the universally quantified equality

$(t \simeq u) = \bot$. A considerably more general reduction, not just for inequalities but for *arbitrary quantifier-free first-order formulae*, can be obtained with Boolean equality enrichments, as stated in Corollary 1.

Corollary 1. *Let $\mathcal{E}^{\simeq} = (\Sigma^{\simeq}, E^{\simeq})$ be a Boolean equational enrichment of $\mathcal{E}$. Let $\varphi = \varphi(t_1 = u_1, \ldots, t_n = u_n)$ be a quantifier-free Boolean formula whose atoms are the Σ-equations $t_i = u_i$ with variables in X, for $1 \leq i \leq n$, and with Boolean connectives in $\{\neg, \vee, \wedge\}$. Then, the following holds*

$$\mathcal{T}_{\mathcal{E}} \models (\forall X)\varphi \quad \Longleftrightarrow \quad \mathcal{T}_{\mathcal{E}^{\simeq}} \models (\forall X)\, \widehat{\varphi}(t_1 \simeq u_1, \ldots, t_n \simeq u_n) = \top, \tag{6}$$

where $\widehat{\varphi}(t_1 \simeq u_1, \ldots, t_n \simeq u_n)$ is the $\Sigma^{\simeq}$-term of sort Bool obtained from φ by replacing every occurrence of the logical connectives $\neg$, $\vee$, and $\wedge$ by, respectively, the function symbols $\multimap$, $\sqcup$, and $\sqcap$ in $\mathcal{E}^{Bool}$ (making $\mathcal{T}_{\mathcal{E},Bool}$ a Boolean algebra) and every occurrence of an atom $t_i = u_i$ by the Bool term $t_i \simeq u_i$, for $1 \leq i \leq n$.

A key property of an equality enrichment $\mathcal{E}^{\simeq}$ of $\mathcal{E}$ is that, if $\mathcal{E}^{\simeq}$ is extended with any set E' of Σ-equations that are *not* satisfiable in $\mathcal{T}_{\mathcal{E}}$, then the resulting extension is inconsistent so that one can derive the *contradiction* $\top = \bot$. Conversely, if the set E' of Σ-equations extending $\mathcal{E}^{\simeq}$ is satisfiable in $\mathcal{T}_{\mathcal{E}}$, then the resulting extension is consistent and therefore cannot yield a proof of contradiction. Statements (7) and (8) in Corollary 2 account for these two facts.

Corollary 2 (generalizes [14, Theorem 74]). *Let $\mathcal{E}^{\simeq} = (\Sigma^{\simeq}, E^{\simeq})$ be a equational enrichment of $\mathcal{E}$ and let E' be a collection of Σ-equations. Then the following holds*

$$\mathcal{T}_{\mathcal{E}} \not\models E' \quad \Longleftrightarrow \quad (\Sigma^{\simeq}, E^{\simeq} \cup E') \vdash \top = \bot, \tag{7}$$

$$\mathcal{T}_{\mathcal{E}} \models E' \quad \Longleftrightarrow \quad (\Sigma^{\simeq}, E^{\simeq} \cup E') \not\vdash \top = \bot. \tag{8}$$

4 Equality Enrichments of Theories with Free Constructors Modulo

In this section we present the effective theory transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ for enriching order-sorted equational theories having free constructors modulo structural axioms with an equality predicate. We fix an order-sorted equational theory $\mathcal{E} = (\Sigma, E \uplus B)$, with $\Sigma = (S, \leq, F)$, and assume that $\Omega \subseteq \Sigma$ is a subsignature of free constructors modulo B, where B is a union of associative (A), commutative (C), and associative-commutative (AC) axioms. Furthermore, the following convention is adopted: for x a variable and s a sort, the expression x_s indicates that x has sort s, i.e., $x \in X_s$.

The theory transformation performed by $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ consists of two main tasks or subtransformations. On input $\mathcal{E}$, it first extends $\mathcal{E}$ by adding new sorts, the equational theory $\mathcal{E}^{Bool}$ of Booleans with constructors $\top$ and $\bot$ (and with the other usual Boolean connectives equationally defined), some auxiliary functions, and the predicate $\_ \simeq \_$ for each top sort in the input theory $\mathcal{E}$. Then, it generates a set of equations defining $\_ \simeq \_$ that depend on the structural axioms of the symbols in Ω. More precisely,

Transformation 1: extends the input theory $\mathcal{E}$ by
1. generating a fresh top sort for each connected component in Σ that does not have it,
2. adding the theory $\mathcal{E}^{Bool}$ with sort $Bool$,
3. adding a Boolean-valued (binary) commutative operator $\_ \simeq \_$ for the top sort of each connected component of $\mathcal{E}$, and
4. adding for each $f \in \Omega$ with A or AC structural axioms the Boolean-valued unary operator $root^k_f$, and if f is AC adding also the Boolean-valued binary operator in^k_f.

Transformation 2: for each $f \in \Omega$, and depending on the structural axioms of f, generates a suitable set of equations defining $\_ \simeq \_$, $root^k_f$, and in^k_f.

Auxiliary Boolean-valued operators $root^k_f$ and in^k_f are useful for checking if terms are rooted by or contain the constructor symbol f, respectively. In this paper we use the Boolean theory $\mathcal{E}^{Bool}$ specified in [3, Subsection 9.1]. The theory $\mathcal{E}^{Bool}$ has free constructors modulo B^{Bool}, it is sort-decreasing, confluent, and operationally terminating modulo AC, and hence provides a Boolean decision procedure. It has signature of free constructors $\Omega^{Bool} = \{\top, \bot\}$, set of defined symbols $\Sigma^{Bool} \setminus \Omega^{Bool} = \{\, \lrcorner\, , \sqcap\, , \sqcup\, , \oplus\, , \supset \}$, and satisfies $\mathcal{T}_{\mathcal{E}^{Bool}} \models \top \neq \bot$. The choice of $\mathcal{E}^{Bool}$ is somewhat arbitrary: any equational theory implementing a Boolean decision procedure should suffice for our purposes.

Definition 2 spells out in detail Transformation 1 and prepares the ground for Transformation 2.

Definition 2 (Enrich). *Given $\mathcal{E}$, the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ generates the smallest equational theory $\mathcal{E}^{\simeq} = (\Sigma^{\simeq}, E^{\simeq} \uplus B^{\simeq})$ satisfying:*

- $\mathcal{E} \cup \mathcal{E}^{Bool} \subseteq \mathcal{E}^{\simeq}$;
- *the poset of sorts of $\mathcal{E}^{\simeq}$ extends that of $\mathcal{E}$ by adding a new connected component $\{Bool\}$, and by adding a fresh top sort to any connected component of the poset of sorts of $\mathcal{E}$ lacking a top sort;*
- *for each top sort k in $\Sigma^{\simeq}$ of a connected component of Σ, $\Sigma^{\simeq}$ contains a commutative operator:*

$$(\_ \simeq \_) : k\ k \to Bool,$$

  $B^{\simeq}$ contains the commutative structural axiom:

$$x_k \simeq y_k = y_k \simeq x_k,$$

  and $E^{\simeq}$ contains the equation:

$$x_k \simeq x_k = \top;$$

- *for each top sort k in $\Sigma^{\simeq}$ of a connected component of Σ and for each function symbol $f : s\ s' \to s'' \in \Omega$, with $s \leq k$, $s' \leq k$, and $s'' \leq k$:*
  - *if f has axioms A or AC, then $\Sigma^{\simeq}$ contains the symbol:*

$$root^k_f : k \to Bool,$$

- *if f has axioms AC, then $\Sigma^{\simeq}$ contains the symbol:*

$$in_f^k : k\ k \to Bool;$$

– *for each function symbol $f \in \Omega$, $E^{\simeq}$ contains the equations* $\mathsf{enrich}_{\mathcal{E}}(f)$ *(see the upcoming definitions).*

Function $\mathsf{enrich}_{\mathcal{E}}$ in Definition 2 formally specifies Transformation 2 and is defined by cases for each constructor symbol depending on its structural axioms. We start with the definition of $\mathsf{enrich}_{\mathcal{E}}$ for the case in which the constructor symbol has no structural axioms; we call such a symbol *absolutely free*.

Definition 3 (Absolutely Free Enrich). *Assume $f \in \Omega$ is an absolutely free symbol. Then, for each maximal typing $f : s_1 \dots s_n \to s$ of $f \in \Omega$,* $\mathsf{enrich}_{\mathcal{E}}(f)$ *adds the following equations:*

– *for each $g : s'_1 \dots s'_m \to s' \in \Omega$ a maximal typing of g such that $s \equiv_{\leq} s'$ and $f \neq g$:*

$$f(x^1_{s_1}, \dots, x^1_{s_n}) \simeq g(y^1_{s'_1}, \dots, y^m_{s'_m}) = \bot,$$

– *for f itself:*

$$f(x^1_{s_1}, \dots, x^n_{s_n}) \simeq f(y^1_{s_1}, \dots, y^n_{s_n}) = \bigsqcap_{1 \leq i \leq n} x^i_{s_i} \simeq y^i_{s_i},$$

– *for each $1 \leq i \leq n$ such that $s_i \equiv_{\leq} s$:*

$$f(x^1_{s_1}, \dots, x^n_{s_n}) \simeq x^i_{s_i} = \bot.$$

In Definition 3, some equations use the Boolean operator $\sqcap$ in $\mathcal{E}^{Bool}$ to obtain a recursive definition of $\_ \simeq \_$. Example 1 shows the results of applying Definition 2 and Definition 3 to a concrete specification.

Example 1. Consider the equational theory $\mathcal{E}^{\texttt{NATURAL}}$ in Figure 1 (left) that represents the natural numbers in Peano notation. An equality enrichment consists of $\mathcal{E}^{\texttt{NATURAL}}$ extended with the equational theory $\mathcal{E}^{Bool}$ and an equational definition of $\_ \simeq \_$. The equational theory in Figure 1 (right) is an equality enrichment of $\mathcal{E}^{\texttt{NATURAL}}$. The last equation is not essential, but it is useful for detecting a greater number of inequalities between terms with variables.

Definition 4 presents the definition or $\mathsf{enrich}_{\mathcal{E}}$ for the case in which the input symbol is commutative. For the definition of $\mathsf{enrich}_{\mathcal{E}}$ in the case of an commutative function symbol f with maximal typing of sort s', it is assumed that its two arguments have the same sort s.

Definition 4 (C-Enrich). *Assume $f \in \Omega$ is commutative and non-associative. Then, for each maximal typing $f : s\ s \to s'$ of $f \in \Omega$,* $\mathsf{enrich}_{\mathcal{E}}(f)$ *adds the following equations:*

```
fmod NATURAL is
  sort Nat .
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
endfm
```

```
fmod EQ-NATURAL is
  protecting NATURAL .
  protecting BOOL .
  op _~_ : Nat Nat
           -> Bool [comm] .
  vars N M : Nat .
  eq N ~ N    = true .
  eq 0 ~ s(N) = false .
  eq s(N) ~ s(M) = N ~ M .
  eq s(N) ~ N    = false .
endfm
```

Fig. 1. Equality Enrichment for $\mathcal{E}^{\texttt{NATURAL}}$.

- *for each* $g : s'_1 \dots s'_m \to s'' \in \Omega$ *a maximal typing of* g *such that* $s' \equiv_{\leq} s''$ *and* $f \neq g$*:*
$$f(x^1_s, x^2_s) \simeq g(y^1_{s'_1}, \dots, y^m_{s'_m}) = \bot,$$
- *for* f *itself:*
$$f(x^1_s, x^2_s) \simeq f(y^1_s, y^2_s) = (x^1_s \simeq y^1_s \sqcap x^2_s \simeq y^2_s) \sqcup (x^1_s \simeq y^2_s \sqcap x^2_s \simeq y^1_s),$$
- *if* $s \equiv_{\leq} s'$ *we add the equation:*
$$f(x^1_s, x^2_s) \simeq x^1_s = \bot.$$

For the definition of $\mathsf{enrich}_{\mathcal{E}}$ in the case of an associative function symbol f with maximal typing of sort s, it is assumed that its two arguments have also sort s. Furthermore, a *top typing* for such an f is also assumed, i.e., a typing $f : s'\ s' \to s'$ satisfying that if $f : s\ s \to s$ is another typing with $s \equiv_{<} s'$, then $s' \geq s$ (note that a top typing of f may not belong to Ω, as in Example 2 below).

Definition 5 (A-Enrich). *Assume* $f \in \Omega$ *is associative and non-commutative. Then for each maximal typing* $f : s\ s \to s$ *of* $f \in \Omega$, $\mathsf{enrich}_{\mathcal{E}}(f)$ *adds the following equations:*

- *for each* $g : s'_1 \dots s'_m \to s' \in \Omega$ *a maximal typing of* $g \in \Omega$ *such that* $s \equiv_{\leq} s'$ *and* $f \neq g$*:*
$$f(x^1_s, x^2_s) \simeq g(y_{s'_1}, \dots, y^m_{s'_m}) = \bot,$$
$$root^k_f(g(x^1_{s'_1}, \dots, x^m_{s'_m})) = \bot,$$
- *for* f *itself:*
$$root^k_f(f(x^1_s, x^2_s)) = \top,$$
$$f(x^1_s, x^2_s) \simeq f(x^1_s, y^2_s) = x^2_s \simeq y^2_s,$$
$$f(x^1_s, x^2_s) \simeq f(y^1_s, x^2_s) = x^1_s \simeq y^1_s,$$
$$f(x^1_s, x^2_s) \simeq f(y^1_s, y^2_s) = \bot \quad \textit{if} \quad \lrcorner\,(root^k_f(x^1_s)) \sqcap \lrcorner\,(root^k_f(y^1_s)) \sqcap \lrcorner\,(x^1_s \simeq y^1_s) = \top,$$

```
fmod LIST is
  protecting NATURAL .
  sorts NeNatList NatList .
  subsorts Nat < NeNatList
               < NatList .
  op nil : -> NatList [ctor] .
  op _;_ : NeNatList NeNatList
    -> NeNatList [ctor assoc] .
  op _;_ : NatList NatList
    -> NatList [assoc] .
  var L : NatList .
  eq L ; nil = L .
  eq nil ; L = L .
endfm

fmod EQ-LIST is
 protecting LIST .
 protecting BOOL .
 protecting EQ-NATURAL .
 op ;-NeNL-root : NatList
   -> Bool .
 op _~_ : NatList NatList
   -> Bool [comm] .
```

```
  vars P Q R S : NeNatList .
  var N : Nat .
  eq ;-NeNL-root(0) = false .
  eq ;-NeNL-root(s(N))
    = false .
  eq ;-NeNL-root(nil) = false .
  eq ;-NeNL-root(P ; Q)
    = true .
  eq P ~ P = true .
  eq 0 ~ nil = false .
  eq s(N) ~ nil = false .
  eq (P ; Q) ~ 0 = false .
  eq (P ; Q) ~ s(N) = false .
  eq (P ; Q) ~ nil = false .
  eq (P ; Q) ~ P = false .
  eq (P ; Q) ~ Q = false .
  eq (P ; Q) ~ (P ; R) = Q ~ R .
  eq (P ; Q) ~ (R ; Q) = P ~ R .
  ceq (P ; Q) ~ (R ; S) = false
   if (not(;-NeNL-root(P)) and
      not(;-NeNL-root(R)) and
      not(P ~ R)) = true .
endfm
```

Fig. 2. Equality Enrichment for $\mathcal{E}^{\mathtt{LIST}}$.

– *for each* $1 \leq i \leq 2$*:*

$$f(x_s^1, x_s^2) \simeq x_s^i = \bot.$$

Example 2. Consider the equational theory $\mathcal{E}^{\mathtt{LIST}}$ in Figure 2 that specifies the lists of natural numbers in Peano notation. Note that $\_;\_$ is a constructor symbol only when its arguments are non-empty lists. Therefore, the signature of free constructors modulo B of the theory $\mathcal{E}^{\mathtt{LIST}}$ is:

$$\{nil : \rightarrow NatList,\ \_;\_ : NeNatList\ NeNatList \rightarrow NeNatList\}.$$

In order to have a recursive definition of equality for lists, $\mathsf{enrich}_{\mathcal{E}}(f)$ uses the auxiliary function $root_f^k$ that checks if a term of sort k is rooted by the constructor symbol f. Figure 2 presents $\mathcal{E}^{\mathtt{EQ\text{-}LIST}}$, an equality enrichment for $\mathcal{E}^{\mathtt{LIST}}$. We illustrate the use of $\mathcal{E}^{\mathtt{EQ\text{-}LIST}}$ with the following two cases:

– for $((0;0);0) \simeq ((0;0);0)$, the only applicable equations are $P \simeq P = \top$, $(P;Q) \simeq (P;R) = Q;R$, and $(P;Q) \simeq (R;Q) = P;R$ by proper associative commutations, and the result is always $\top$ independently of their application order; and

– for $((0; s(0)); 0) \simeq (s(0); 0)$ the last equation defined for $\mathcal{E}^{\texttt{EQ-LIST}}$ is applicable under substitution $\sigma(P) = \sigma(S) = 0$, $\sigma(R) = s(0)$, and $\sigma(Q) = s(0); 0$ by proper associative commutations since it is the only equation that satisfies $\lrcorner\, root_{;}^{NeNatList}(0) \sqcap \lrcorner\, root_{;}^{NeNatList}(s(0)) \sqcap \lrcorner\, 0 \simeq s(0) = \top$, thus obtaining $((0; s(0)); 0) \simeq (s(0); 0) = \bot$.

In the case in which the input symbol of $\mathsf{enrich}_{\mathcal{E}}$ with maximal typing of sort s is associative-commutative, it is assumed that its two arguments also have sort s and there is a *top typing* for f, as in the associative case.

Definition 6 (AC-Enrich). *Assume $f \in \Omega$ is associative-commutative. Then for each maximal typing $f : s\ s \to s$ of $f \in \Omega$, $\mathsf{enrich}_{\mathcal{E}}(f)$ adds the following equations:*

– *for each $g : s'_1 \dots s'_m \to s'$ a maximal typing of $g \in \Omega$ such that $s \equiv_{\leq} s'$ and $f \neq g$:*

$$f(x^1_s, x^2_s) \simeq g(y^1_{s'_1}, \dots, y^m_{s'_m}) = \bot,$$
$$root^k_f(g(x^1_{s'_1}, \dots, x^m_{s'_m})) = \bot,$$

– *for f itself:*

$$\begin{aligned}
root^k_f(f(x^1_s, x^2_s)) &= \top, \\
in^k_f(x_s, y_k) &= \bot && \text{if } root^k_f(x_s) = \top, \\
in^k_f(x_s, f(x_s, y_s)) &= \top && \text{if } \lrcorner\,(root^k_f(x_s)) = \top, \\
in^k_f(x_k, f(y^1_s, y^2_s)) &= (x_k \simeq y^1_s) \sqcup \\
&\quad\ in^k_f(x_k, y^2_s) && \text{if } \lrcorner\,(root^k_f(x_k)) \sqcap \\
&&& \quad \lrcorner\,(root^k_f(y^1_s)) = \top, \\
in^k_f(x_k, y_k) &= x_k \simeq y_k && \text{if } \lrcorner\,(root^k_f(x_k)) \sqcap \\
&&& \quad \lrcorner\,(root^k_f(y_k)) = \top, \\
f(x_s, y_s) \simeq f(x_s, z_s) &= y_s \simeq z_s, \\
f(x^1_s, x^2_s) \simeq f(y^1_s, y^2_s) &= \bot && \text{if } \lrcorner\,(root^k_f(x^1_s))) \sqcap \\
&&& \quad \lrcorner\,(in^k_f(x^1_s, f(y^1_s, y^2_s))) = \top, \\
f(x^1_s, x^2_s) \simeq x^1_s &= \bot.
\end{aligned}$$

Intuitively, if a term of sort k rooted by an associative-commutative symbol f is viewed as a multiset with union operator f, then function in^k_f in Definition 6 helps in identifying the cases in which an element (a term not rooted by f) belongs to the multiset.

Example 3. Consider the equational theory $\mathcal{E}^{\texttt{MSET}}$ in Figure 3 defining multisets of natural numbers in Peano notation. Theory $\mathcal{E}^{\texttt{EQ-MSET}}$ in Figure 3 is an equality enrichment for $\mathcal{E}^{\texttt{MSET}}$, where auxiliary functions $root^k_f$ and in^k_f are used to give a recursive comparison of equality for constructor terms rooted by AC-symbols.

Consider the following two cases:

– for $((0\ 0)\ 1) \simeq ((0\ 1)\ 0)$, the only applicable equations are $P \simeq P = \top$ and $(P\ Q) \simeq (P\ R) = Q\ R$ by proper associative-commutative commutations, and the result is always $\top$ independently of their application order; and

```
fmod MSET is
 protecting NATURAL .
 sorts NeNatMSet NatMSet .
 subsort Nat < NeNatMSet
             < NatMSet .
 op empty : -> NatMSet [ctor] .
 op __ : NeNatMSet NeNatMSet ->
  NeNatMSet [ctor assoc comm] .
 op __ : NatMSet NatMSet
  -> NatMSet [assoc comm] .
  var T : NatMSet .
  eq empty T = T .
endfm

fmod EQ-MSET is
 protecting MSET .
 protecting BOOL .
 protecting EQ-NATURAL .
 op -NeNMS-root : NatMSet
  -> Bool .
 op in--NeNMS : NatMSet NatMSet
  -> Bool .
 op _~_ : NatMSet NatMSet
  -> Bool [comm] .
 vars P Q R S : NeNatMSet .
 var N : Nat .
 vars T U : NatMSet .
 eq -NeNMS-root(0) = false .
 eq -NeNMS-root(s(N)) = false .
 eq -NeNMS-root(empty) = false .
 eq -NeNMS-root(P Q) = true .
 ceq in--NeNMS(P,Q) = false
  if -NeNMS-root(P) = true .
 ceq in--NeNMS(P, (P Q)) = true
  if not(-NeNMS-root(P)) =
     true .
 ceq in--NeNMS(T, (Q R)) =
     (T ~ Q) or in--NeNMS(T,R)
  if (not(-NeNMS-root(T)) and
     not(-NeNMS-root(Q))) =
     true .
 ceq in--NeNMS(T,U) = T ~ U
  if (not(-NeNMS-root(T)) and
     not(-NeNMS-root(U))) =
     true .
 eq P ~ P = true .
 eq 0 ~ empty = false .
 eq s(N) ~ empty = false .
 eq (P Q) ~ 0 = false .
 eq (P Q) ~ empty = false .
 eq (P Q) ~ s(N) = false .
 eq (P Q) ~ P = false .
 eq (P Q) ~ (P R) = Q ~ R .
 ceq (P Q) ~ (R S) = false
  if (not(-NeNMS-root(P)) and
     not(in--NeNMS(P, R S))) =
     true .
endfm
```

Fig. 3. Equality Enrichment for $\mathcal{E}^{\mathtt{MSET}}$.

– for $((0\ s(0))\ 0) \simeq (s(0)\ 0)$, we can apply the following equations:
  - $(P\ Q) \simeq P = \bot$ under substitution $\sigma(P) = 0\ s(0)$ and $\sigma(Q) = 0$ by proper associative-commutative commutations;
  - $(P\ Q) \simeq (P\ R) = Q\ R$ under substitutions:
    * $\sigma(P) = 0$, $\sigma(Q) = 0\ s(0)$, and $\sigma(R) = s(0)$, obtaining $s(0)\ 0 \simeq s(0)$, and hence the only applicable equations now are:
      · $P\ Q \simeq s(N) = \bot$ using the substitution $\sigma(P) = s(0)$, $\sigma(Q) = 0$ (or vice versa), and $\sigma(N) = 0$
      · $P\ Q \simeq P = \bot$ under substitution $\sigma(P) = s(0)$ and $\sigma(Q) = 0$,
    * $\sigma(P) = s(0)$, $\sigma(Q) = 0\ 0$, and $\sigma(R) = 0$, obtaining $0\ 0 \simeq 0$, and hence the only applicable equations now are:
      · $P\ Q \simeq 0 = \bot$ using the substitution $\sigma(P) = 0$ and $\sigma(Q) = 0$,
      · $P\ Q \simeq P = \bot$ under substitution $\sigma(P) = 0$ and $\sigma(Q) = 0$.

5 Executability Properties of $\mathcal{E}^{\simeq}$

It would be enormously useful, both from a theoretical and a practical point of view, that if a theory $\mathcal{E}$ satisfies some executability properties, then that the equality enrichment $\mathcal{E}^{\simeq}$ of $\mathcal{E}$ obtained from the transformation in Section 4 could inherit these properties. In particular, if the original theory $\mathcal{E}$ is sort-decreasing (resp., ground sort-decreasing), confluent (resp., ground confluent), and operationally terminating (resp., ground operationally terminating), then $\mathcal{E}^{\simeq}$ must be so. Also, the subsignature of free constructors of $\mathcal{E}^{\simeq}$ must be an extension of the subsignature of free constructors of $\mathcal{E}$ (modulo the structural axioms). In this way, full agreement between mathematical and operational semantics is preserved in the equality enrichment $\mathcal{E}^{\simeq}$ of $\mathcal{E}$.

Note that the domain of the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ includes exactly those equational theories whose structural axioms are any combination of A and/or C axioms for some of its symbols. However, if the input theory $\mathcal{E}$ has symbols with identity axioms, one could use the results in [5] to remove them and instead add them as equations, provided that the constructors remain free after the transformation. Note that, as illustrated by the `LIST` and `MSET` examples, where identities for lists and multisets are specified as oriented equations and not as axioms, this is often possible in practice.

In what follows, $\mathcal{E} = (\Sigma, E \uplus B)$ is an order-sorted equational theory with signature of free constructors $\Omega \subseteq \Sigma$ modulo B and $\mathcal{E}^{\simeq} = (\Sigma^{\simeq}, E^{\simeq} \uplus B^{\simeq})$ is the Boolean equality enrichment $\mathcal{E}^{\simeq}$ obtained by using the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$.

5.1 Preservation of Executability Properties and Free Constructors

Recall from Section 2 that the equational theory $\mathcal{E} = (\Sigma, E \uplus B)$ is sort-decreasing (resp., ground sort-decreasing) iff for each $t = t'$ if $C \in E$, and substitution (resp., ground substitution) θ with $(\Sigma, E \uplus B) \vdash C\theta$, we have $ls(t\theta) \geq ls(t'\theta)$. The key observation is that since *Bool* is a fresh sort in a new connected component of $\mathcal{E}^{\simeq}$ and all equations in $\bigcup_{f \in \Omega} \mathsf{enrich}_{\mathcal{E}}(f)$ are of sort *Bool*, it is impossible that the equations in $\mathcal{E}^{Bool}$ or in $\bigcup_{f \in \Omega} \mathsf{enrich}_{\mathcal{E}}(f)$ can be applied to terms in T_{Σ}.

Theorem 1. *If $\mathcal{E}$ is sort-decreasing (resp., ground sort-decreasing), then $\mathcal{E}^{\simeq}$ is sort-decreasing (resp., ground sort-decreasing).*

The notion of reductive theory is used in proving $\mathcal{E}^{\simeq}$ operationally terminating.

Definition 7 (Reductive Theory Modulo Axioms). *Let $\rhd$ be the strict subterm relation on terms. An equational theory $\mathcal{E} = (\Sigma, E \uplus B)$ is* reductive modulo B *iff there exists a reduction ordering $\succ$ and a symmetric, stable, and monotonic relation $\sim$ such that:*

1. *$l \notin X$ for each equation $l = r$ if $\bigwedge_{i=1..n} t_i = u_i \in E$.*
2. *$l \succ r$ for each equation $l = r$ if $\bigwedge_{i=1..n} t_i = u_i \in E$.*

3. $l(\succ \cup \rhd)^+ t_i$ *and* $l(\succ \cup \rhd)^+ u_i$ *for each equation* $l = r$ *if* $\bigwedge_{i=1..n} t_i = u_i \in E$.
4. $u \sim v$ *for each equation* $u = v \in B$
5. $\sim ; \succ \subseteq \succ$.

Lemma 1. *If an equational theory* $\mathcal{E} = (\Sigma, E \uplus B)$ *is* reductive modulo B*, then it is operationally terminating modulo* B*.*

In general, the union of two operationally terminating theories may not be operationally terminating. Furthermore, the fact of dealing with arbitrary theories whose rules are unknown makes the task of proving operational termination of the union more involved. However, sort information is used to obtain a proof of operational termination in the following way:

1. first, we prove that $(\Sigma^{\simeq}, E \uplus B^{\simeq})$ is operationally terminating (resp., ground operationally terminating) and confluent (resp., ground confluent)[2],
2. then, we prove that $(\Sigma^{\simeq}, (E^{\simeq} \setminus E) \uplus B^{\simeq})$ is operationally terminating (resp., ground operationally terminating),
3. finally, we prove that the union of both theories is operationally terminating (resp., ground operationally terminating).

Theorem 2. *If* $\mathcal{E}$ *is sort-decreasing (resp., ground sort-decreasing), confluent (resp., ground confluent) and operationally terminating (resp., ground operationally terminating) in a* Σ*-extensible way, then* $\mathcal{E}^{\simeq}$ *is operationally terminating (resp., ground operationally terminating).*

Since $\mathcal{E}^{\simeq}$ is sort-decreasing by Theorem 1 and operationally terminating by Theorem 2, the confluence of $\mathcal{E}^{\simeq}$ follows from its local confluence. Similarly, ground local confluence follows for the ground case.

Theorem 3. *If* $\mathcal{E}$ *is sort-decreasing (resp., ground sort-decreasing), operationally terminating (resp., ground operationally terminating) in a* Σ*-extensible way, and confluent (resp., ground confluent), then* $\mathcal{E}^{\simeq}$ *is confluent (resp., ground confluent).*

The proof of Theorem 3 is obtained by case analysis. It considers the conditional critical pairs of E that are joinable by assumption, the critical pairs of E^{Bool} that are also joinable by the choice of $\mathcal{E}^{Bool}$, and the conditional critical pairs of $E^{\simeq} \setminus (E \cup E^{Bool})$. Note that, since we may have $B^{\simeq}$ containing associative axioms, $B^{\simeq}$-unification is infinitary in general. Hence, we have to reason about the possible form of any $B^{\simeq}$-unifier that can involve a critical pair between two oriented equations with A symbols to conclude the local confluence of $\mathcal{E}^{\simeq}$.

Lemma 2 is used for reasoning about the signature of constructors of $\mathcal{E}^{\simeq}$.

[2] We assume that operational termination (resp., ground operational termination) of $(\Sigma, E \uplus B)$ is Σ-extensible, i.e., if $(\Sigma, E \uplus B)$ is operationally terminating (resp., ground operationally terminating) then $(\Sigma \cup \Delta, E \uplus B)$ is so too. This is not a strong restriction in practice, since all the actual existing tools for proving termination properties on rewriting theories generate Σ-extensible orderings.

Lemma 2. *Let $\mathcal{E}^{\simeq}$ be obtained by using the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$, where $\mathcal{E}$ is ground sort-decreasing, ground confluent, and ground operationally terminating in a Σ-extensible way, and Ω is the signature of free constructors modulo B of $\mathcal{E}$. Then, if $t, t' \in T_\Omega$, then $t \simeq t' \rightarrow^{+}_{E^{\simeq}/B^{\simeq}} \top$ iff $t =_B t'$.*

Intuitively, Lemma 2 states that $\mathcal{E}^{\simeq}$ is a conservative extension of $\mathcal{E}$ for ground terms in Σ and the equationally defined equality predicate in $\mathcal{E}^{\simeq}$ is well-defined.

We identify $\Omega^{\simeq} = \Omega \uplus \{\top, \bot\} \subseteq \Sigma^{\simeq}$ as a signature of constructors for $\mathcal{E}^{\simeq}$ and prove that the constructors in $\Omega^{\simeq}$ are free modulo $B^{\simeq}$.

Theorem 4. *If $\mathcal{E}$ is ground sort-decreasing, ground confluent, and ground operationally terminating in a Σ-extensible way, and Ω is the signature of free constructors modulo B of $\mathcal{E}$, then $\mathcal{E}^{\simeq}$ has $\Omega^{\simeq} = \Omega \uplus \{\top, \bot\} \subseteq \Sigma^{\simeq}$ as a signature of free constructors modulo $B^{\simeq}$.*

5.2 $\mathcal{E}^{\simeq}$ is an Equality Enrichment

The properties of $\mathcal{E}^{\simeq}$ inherited from $\mathcal{E}$ are very useful in proving that $\mathcal{E}^{\simeq}$ is indeed an equality enrichment.

Theorem 5. *Let $\mathcal{E} = (\Sigma, E \uplus B)$ be an order-sorted equational theory with signature $\Omega \subseteq \Sigma$ of free constructors modulo B and let $\mathcal{E}^{\simeq} = (\Sigma^{\simeq}, E^{\simeq} \uplus B^{\simeq})$ be the equational theory obtained by using $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$. If $\mathcal{E}$ is ground sort-decreasing, ground operationally terminating in a Σ-extensible way, and ground confluent modulo B, then $\mathcal{E}^{\simeq}$ is a Boolean equality enrichment of $\mathcal{E}$.*

6 Automation and Applications of $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$

The transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ is obviously *constructive* and has been automated in Maude using its reflective features: it takes the meta-representation of $\mathcal{E}$ in Maude as input and constructs a meta-representation of $\mathcal{E}^{\simeq}$ as output. The transformation itself has already been incorporated into Maude formal tools, including the Maude Church-Rosser and Coherence Checker [6] (CRC-ChC), and the Maude Invariant Analyzer tool [17].

6.1 A Case Study

We present a case study in which the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ is used in the Maude Invariant Analyzer (InvA) tool [17]. The InvA tool mechanizes an inference system for deductively proving safety properties of rewrite theories: it transforms all formal temporal reasoning about safety properties of concurrent transitions to purely equational inductive reasoning. The InvA tool provides a substantial degree of mechanization and can automatically discharge many proof obligations without user intervention. In this section, we illustrate how equality enrichments can be used to support the deductive verification task in the InvA tool for a mutual exclusion property of processes in the `QLOCK` protocol.

The mutual exclusion protocol `QLOCK` uses a global queue as follows:

- each process that participates in the protocol does the following:
  - if the process wants to use the critical resource and its name is not in the global queue, it places its name in the queue;
  - if the process wants to use the critical resource and its name is in the global queue, if its name is at the top of the queue then the process gains access to the critical resource; otherwise it waits; and
  - if the process finishes the critical resource, it removes its name from the top of the global queue;
- the protocol should start from a state where the queue is empty; and
- it is assumed that each process can use the critical resource any number of times.

Consider the following equational theory $\mathcal{E}^{\texttt{QLOCK-STATE}}$, which represents the states of QLOCK with terms of sort *State*. It protects the equational theory $\mathcal{E}^{\texttt{MSET}}$ presented in Section 4. Processes and names of processes are modeled with natural numbers of sort *Nat* in Peano notation. A term $Pi \mid Pw \mid Pc \mid Q$ of sort *State* describes the state in which Pi is the collection of processes whose name is not in the global queue (or *idle* processes), Pw is the collection of processes whose names that are waiting to gain access to the critical resource (or *waiting* processes), Pc is the collection of processes that are using the critical resource (or *critical* processes), and Q is the global queue of the system. Sorts *MSet* and *Queue* are used to represent collections of processes and queues of processes' names, respectively.

```
fmod QLOCK-STATE is
  protecting MSET .


  sort Queue .
  op nil : -> Queue [ctor] .
  op _@_ : Nat Queue -> Queue [ctor] .
  op _;_ : Queue Queue -> Queue .
  eq nil ; Q:Queue = Q:Queue .
  eq (N:Nat @ Q1:Queue) ; Q2:Queue = N:Nat @ (Q1:Queue ; Q2:Queue) .

  sort State .
  op _|_|_|_ : MSet MSet MSet Queue -> State [ctor] .
endfm
```

The behavior of a concurrent system in rewriting logic is specified by rewrite rules that define how the individual transitions change the state of the system. The specification of all transitions of QLOCK is described by six rewrite rules in the rewrite theory $\mathcal{R}^{\texttt{QLOCK}}$ as follows.

```
mod QLOCK is
  protecting QLOCK-STATE .

  vars Pi Pw Pc : MSet .   var Q : Queue .   vars N N' N'' : Nat .

  rl [to-wait-1] : N | Pw | Pc | Q  =>  empty | Pw N | Pc
```

```
                | Q ; (N @ nil ) .
  rl [to-wait-2] : N Pi | Pw | Pc | Q  =>  Pi | Pw N | Pc
                | Q ; (N @ nil ) .
  rl [to-crit-1] : Pi | N | Pc | N @ Q  =>  Pi | empty | Pc N | N @ Q .
  rl [to-crit-2] : Pi | Pw N | Pc | N @ Q  =>  Pi | Pw | Pc N | N @ Q .
  rl [to-idle-1] : Pi | Pw | N | N' @ Q  =>  Pi N | Pw | empty | Q .
  rl [to-idle-2] : Pi | Pw | Pc N | N' @ Q  =>  Pi N | Pw | Pc | Q .
endm
```

Rewrite rules `to-idle-1` and `to-idle-2` specify the behavior of a process that finishes using the critical resource: it goes to state idle and the name on top of the global queue is removed. Similarly, rewrite rules `to-wait-1` and `to-wait-2`, and `to-crit-1` and `to-crit-2`, specify the behavior of a process that wants to use the critical resource and of a process that is granted access to the critical resource, respectively.

We want to verify that the `QLOCK` system satisfies the following safety properties. It is key that: (i) it satisfies the mutual exclusion property, namely, that at any point of execution there is at most one process using the critical resource. We also want to verify that: (ii) the name on top of the global queue coincides with the name of the process using the critical resource, if any. Finally, we want to verify that: (iii) the global queue only contains the names of all waiting and critical processes. State predicates *mutex*, *priority*, and *cqueue*, respectively, specify properties (i), (ii), and (iii) in the following equational theory $\mathcal{E}^{\texttt{QLOCK-PREDS}}$. State predicate *init* specifies the set of initial states of `QLOCK`, with auxiliary function *set*? that characterizes multisets having no repeated elements. State predicate *unique* is a strengthening of *mutex* and *priority*. Auxiliary function $to - soup$ on input Q of sort *Queue* computes the multiset of natural numbers appearing in Q.

```
fmod QLOCK-PREDS is
  protecting QLOCK-STATE .
  protecting EQ-MSET .

  vars N N'      : Nat .   var Q   : Queue .
  vars Pi Pw Pc  : MSet .  var NeS  : NeMSet .

  ops init mutex unique priority cqueue : State -> [Bool] .

  eq init( Pi | empty | empty | nil ) = set?(Pi) .
  eq mutex( Pi | Pw | empty | Q ) = true .
  eq mutex( Pi | Pw | N | Q ) = true .
  eq mutex( Pi | Pw | N NeS | Q ) = false .
  eq unique( Pi | Pw | empty | Q ) = set?(Pi Pw) .
  eq unique( Pi | Pw | N | N @ Q ) = set?(Pi Pw N) .
  eq unique( Pi | Pw | N NeS | Q ) = false .
  eq priority( Pi | Pw | empty | Q ) = true .
  eq priority( Pi | Pw | N | N' @ Q ) = N ~ N' .
  eq priority( Pi | Pw | N Pc | N' @ Q ) = (N ~ N') and
```

```
                                          (Pc ~ empty) .
  eq cqueue( Pi | Pw | Pc | Q ) = Pw Pc ~ to-soup(Q) .
....
endfm
```

Observe that $\mathcal{E}^{\texttt{QLOCK-PREDS}}$ protects the equality enrichment $\mathcal{E}^{\texttt{EQ-MSET}}$, in Section 4, for the connected component of sort *MSet* that defines the equality enrichment for sorts *Nat*, *MSet*, and *NeMSet*. The equality enrichments for these sorts are key in the specification of the state predicates. For instance, predicates *priority* and *cqueue* are directly defined in terms of the equality predicate for sorts *Nat* and *MSet*, and also use the Boolean connective for conjunction *and* that comes with the Boolean equality enrichment. Auxiliary function *set*? also makes use of the equality enrichment for sort *Nat*. Note that, in general, defining from scratch the equality enrichment for an AC-symbol such as the multiset union in $\mathcal{E}^{\texttt{MSET}}$, can be a daunting task. Instead, in $\mathcal{E}^{\texttt{QLOCK-PREDS}}$, the definition of the state predicate *cqueue* was straightforward with the help of the equality enrichment for multisets of natural numbers.

By using the InvA tool we are able to automatically prove that predicates *mutex* and *priority* are invariants of $\mathcal{R}^{\texttt{QLOCK}}$ for any initial state that satisfies predicate *init*. For predicate *cqueue* some proof obligations cannot be automatically discharged. In general terms, 22 out of 26 proof obligations were automatically discharged. However, this is an encouraging result, given that the current version of the InvA tool does not yet have dedicated inference support for Boolean equality enrichments, which could further improve the degree of automation.

7 Related Work and Conclusion

In [8], the author generalizes and simplifies the technique given in [15] for proving induction hypothesis without induction (so-called *inductionless induction*) using enriched theories with equality. The notion of *s-taut* related to a sort s can be seen as a initial approximation of what we called in this paper an equality enrichment. The technique described in the paper is based in the result stated in Corollary 2.

In [14], the authors define the notion of *equality enrichment* (without axioms) as an explicit subrepresentation of an *equational equality presentation*. Our work extends this notion of equality enrichment with subsorts and axioms and also presents an automatic way to generate this equality enrichment modulo axioms. As the authors of [14] also remark, an equality enrichment can be used for inductionless induction theorem proving.

In [16], the authors propose an equality predicate for algebraic specifications. Unlike our work, the authors do not consider axioms and sufficient completeness in their theories, hence they have to manage terms with defined symbols. In the positive cases, their equality predicate is equivalent to ours, but in the negative cases, a *false* answer in [16] does not mean that both terms are distinct for any possible instantiation (as we state in our work), because the negative rules are

based on a check of convergence between terms. The goal of this behavior is to avoid false positives instead of capturing negative cases.

In conclusion, this paper solves an important open problem: how to make the addition of equationally defined equality predicates effective and automatic for a very wide class of equational specifications with initial algebra semantics. That such a transformation should exists is suggested by the Bergstra-Tucker meta-theorem [2], but such a meta-result is not constructive and gives no insight as to how the transformation could be defined. We have shown that it can be defined for a very wide class of algebraic specifications with highly expressive features such as order-sorted types, conditional equations, and rewriting modulo commonly occurring axioms. We have also shown that all the expected good properties of the input theory $\mathcal{E}$ are preserved by the transformation $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$.

Using reflection, the transformation has been implemented in Maude and has already been integrated into the Maude Church-Rosser and Coherence Checker [6] (CRC-ChC), and the Maude Invariant Analyzer tool [17]. In the near future it should be added to other tools such as the Maude Termination Tool [4] (MTT) and the Maude Sufficient Completeness Checker [11] (SCC). One obvious advantage of these additions is the possibility of systematically transforming specifications making use of built-in equalities and inequalities, which cannot be handled by formal tools, into specifications where such built-in equalities and inequalities are systematically replaced by equationally-defined equalities, so that formal tools can be applied. But this is not the only possible application by any means. For example, the case study in Subsection 6.1 shows how the addition of equationally-defined equality predicates also makes the specification and verification of safety properties in the InvA tool considerably easier.

It is also clear that adding an equationally-defined equality to Maude's Inductive Theorem Prover [10] (ITP) would make this tool more effective in many ways, and would also greatly reduce the complexities of dealing with arbitrary universal formulas as goals, since all such formulas could be reduced to unconditional equality goals. It would also be very useful to explore the use of the $\mathcal{E} \mapsto \mathcal{E}^{\simeq}$ transformation in *inductionless induction* theorem proving. Yet another very useful field of application would be *early failure detection* in narrowing-based unification. The idea is that $E \uplus B$-unification goals can be viewed as equality goals, which can be detected to have already *failed* if they can be rewritten to *false* with $E^{\simeq}$ modulo $B^{\simeq}$.

In general, the contribution presented in this work opens many useful applications to improve the state of the art in formal verification of algebraic specifications using, in particular, the Maude formal environment.

References

1. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
2. Bergstra, J., Tucker, J.: Characterization of Computable Data Types by Means of a Finite Equational Specification Method. In: de Bakker, J.W., van Leeuwen,

J. (eds.) Proc. of the 7th International Colloquium on Automata, Languages and Programming, ICALP'80. LNCS, vol. 81, pp. 76–90. Springer-Verlag (1980)
3. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude – A High-Performance Logical Framework, LNCS, vol. 4350. Springer-Verlag (2007)
4. Durán, F., Lucas, S., Marché, C., Meseguer, J., Urbain, X.: Proving Operational Termination of Membership Equational Programs. Higher Order Symbolic Computation 21(1-2), 59–88 (2008)
5. Durán, F., Lucas, S., Meseguer, J.: Termination Modulo Combinations of Equational Theories. In: Ghilardi, S., Sebastiani, R. (eds.) Proc. of the 7th International Conference on Frontiers of Combining Systems, FroCoS'09. LNCS, vol. 5749, pp. 246–262. Springer-Verlag (2009)
6. Durán, F., Meseguer, J.: On the Church-Rosser and Coherence Properties of Conditional Order-Sorted Rewrite Theories. Journal of Logic and Algebraic Programming to appear (2011)
7. Goguen, J., Meseguer, J.: Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations. Theoretical Computer Science 105, 217–273 (1992)
8. Goguen, J.A.: How to Prove Algebraic Inductive Hypotheses Without Induction. In: Bibel, W., Kowalski, R. (eds.) Proc. of the 5th Conference on Automated Deduction, CADE'80. LNCS, vol. 87, pp. 356–373. Springer-Verlag (1980)
9. Gutiérrez, R., Meseguer, J., Rocha, C.: Order-Sorted Equality Enrichments Modulo Axioms (Extended Version). Tech. rep., University of Illinois at Urbana-Champaing (December 2011), available at `http://hdl.handle.net/2142/28597`
10. Hendrix, J.: Decision Procedures for Equationally Based Reasoning. Ph.D. thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA (2008)
11. Hendrix, J., Clavel, M., Meseguer, J.: A Sufficient Completeness Reasoning Tool for Partial Specifications. In: Giesl, J. (ed.) Proc. of the 16th International Conference on Rewriting Techniques and Applications, RTA'05. LNCS, vol. 3467, pp. 165–174. Springer-Verlag (2005)
12. Lucas, S., Marché, C., Meseguer, J.: Operational Termination of Conditional Term Rewriting Systems. Information Processing Letters 95(4), 446–453 (2005)
13. Meseguer, J.: Membership Algebra as a Logical Framework for Equational Specification. In: Parisi-Presicce, F. (ed.) Recent Trends in Algebraic Development Techniques, Proc. of the 12th International Workshop on Workshop on Algebraic Development Techniques, WADT'97. LNCS, vol. 1376, pp. 18–61. Springer-Verlag (1997)
14. Meseguer, J., Goguen, J.A.: Initially, Induction and Computability. Algebraic Methods in Semantics (1986)
15. Musser, D.R.: On Proving Inductive Properties of Abstract Data Types. In: Proc. of the 7th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'80. pp. 154–162. ACM Press (1980)
16. Nakamura, M., Futatsugi, K.: On Equality Predicates in Algebraic Specification Languages. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) Proc. of the 4th International Conference on Theoretical Aspects of Computing, ICTAC'07. LNCS, vol. 4711, pp. 381–395. Springer-Verlag (2007)
17. Rocha, C., Meseguer, J.: Proving safety properties of rewrite theories. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) Proc. of 4th International Conference on Algebra and Coalgebra in Computer Science, CALCO'11. LNCS, vol. 6859, pp. 314–328. Springer-Verlag (2011)