Equational Completion in Order-sorted Algebras

Extended Abstract

Isabelle Gnaedig Claude Kirchner Hélène Kirchner
INRIA CRIN/INRIA CRIN/INRIA
BP239, 54506 Vandœuvre les Nancy,
France
E-Mail: {gnaedig, ckirchner, hkirchner}@crin.crin.fr

Abstract

We describe and prove completion procedures for equational term rewriting systems in order-sorted algebras. Problems specific to order-sorted equational logic are emphasized.

1 Introduction

Order-sorted algebras have been introduced in the context of logical programming languages. Type structure supports conceptual clarity and detection of many errors at program entry time. However, strong typing in a many-sorted style can be too rigid and lacks the expressive power needed to deal with errors and with partiality [18]. Many of these problems are overcome by an order-sorted type structure that brings inheritance, operator overloading, and error handling into the realm of equational logic; many seemingly partial or problematic functions are total and well-defined on the right subsort [16,17]. An order-sorted algebra is like a many-sorted one, except that there is a partial ordering on the set of sorts that is interpreted as set inclusion. For instance, we can have a subset relation \( \text{Nat} < \text{Int} \), interpreted as the inclusion \( N \subseteq \mathbb{Z} \) of the naturals into the integers in the standard model. In addition, function symbols such as \(_+:_\) may be overloaded as \(_+:_\text{Nat Nat} \to \text{Nat}\), \(_+:_\text{Int Int} \to \text{Int}\), and are required to agree in their results when restricted to arguments in the same subsorts.

Programming languages based on equational logic have been developed by several authors including [19,50,5,20,11]. We will take as example OBJ, a logical programming language based on order-sorted equational logic. In OBJ 3, version developed at SRI International programs are equational specifications and computation is an efficient form of equational deduction by order-sorted rewriting [36].

Nicely, all the basic results of unsorted equational logic generalize to the order-sorted case [16]. However, order-sorted deduction is more subtle, and its correspondence with concepts such as replacement of equals for equals and term rewriting requires a careful analysis that was initiated in [14] and is further developed in [36]. In programming languages based on equational logic, a computation is the application of rewrite rules until getting a normal form to which no more rules apply. In order to get a unique result, the rewriting system must be terminating and confluent.

For example, natural numbers and integers are defined by the following OBJ-like specification:
obj NAT is
    sorts Nat .
    op 0, 1 : Nat .
    eq : Nat, N + 0 = N .

obj INT is
    sorts Int .
    op - : Int -> Int .
    eq : Int, I + 0 = I .
    eq : Int, I + (- I) = 0 .

jbo

However, this specification does not provide enough rules for computation, since for instance, the term 0 + (-0) reduces either to 0 using the second rule and to -0 using the first one. A new oriented equation -0 = 0 must be added. Eventually the completion procedure generates for INT the following set of (oriented) equations:

obj INT is
    protecting NAT .
    sorts Int .
    subsorts Nat < Int .
    op -: Int -> Int .

    eq : - 0 --> 0 .
    eq : I : Int, I + (- I) --> 0 .
    eq : I : Int, - - I --> I .
    eq : I : Int, I + 0 --> I .
    eq : I,J : Int, (- I) + (- J) --> - (I + J) .

jbo

The goal of a completion method is to construct a terminating and confluent rewrite system R from a given set of equations. The Knuth-Bendix completion procedure is based on using equations as rewrite rules and computing critical pairs when left-hand sides of rules overlap. If a critical pair has distinct irreducible forms, a new rule must be added and the procedure recursively applies until it maybe stops. This procedure requires the termination property of the set of rules, which can be proved by various tools [7,8].

The method was extended to handle the case of an equational term rewriting system, i.e. a set of axioms split into a first subset R whose axioms are used as rules and a second subset E whose axioms are used as equations. This allows the inclusion of axioms, such as commutativity, that cannot be used as rules without losing the termination property. A first approach by Lankford and Ballantyne [42,41,40] handles the case of commutative, or more generally permutative, axioms that generate finite E-congruence classes. The case of infinite E-congruence classes is studied in [47,26,23]. Huet's approach [23] is restricted to sets of left-linear rules, while Peterson and Stickel's one [47] is restricted to linear theories E for which a finite and complete unification algorithm is known. These results have been unified in [28]. Finally a new technique, orderings for equational proofs, introduced in [3,2] allowed simpler and more intuitive proofs of improved completion procedures.

Completion procedures have been applied to a wide class of problems including the word problem in universal algebra [39], theorem proving in first-order logic [21], proofs of inductive properties in abstract data types [24] and computing with rewrite programs [9,20].

Although criteria of confluence and completion procedures are well-known in the case of unsorted equational logics, the case of order-sorted logic has been approached only in the unequational case by [6]. The main goal of this paper is to carefully design a completion procedure in the framework of equational order-sorted theories and to point out the differences with the standard case.