Sort Inheritance for Order-Sorted Equational Presentations

Claus Hintermeier, Claude Kirchner, Hélène Kirchner

CRIN-CNRS & INRIA-Lorraine
BP239, 54506 Vandœuvre-lès-Nancy Cedex
France
E-mail: hinterme@loria.fr, ckirchne@loria.fr, hkirchne@loria.fr

Abstract. In an algebraic framework, where equational, membership and existence formulas can be expressed, decorated terms and rewriting provide operational semantics and decision procedures for these formulas. We focus in this work on testing sort inheritance, an undecidable property of specifications, needed for unification in this context. A test and three specific processes, based on completion of a set of rewrite rules, are proposed to check sort inheritance. They depend on the kinds of membership formulas $(t : A)$ allowed in the specifications: flat and linear, shallow and general terms t are studied.

1 Introduction

The operationalisation of order-sorted frameworks ([Obe62, SS87, SNGM89, GM92]) led to several problems in the past, due to the purely syntactic treatment of membership formulas $(t : A)$, also called term declarations. Assuming them as parsing-oriented declarations, a first pitfall is that equality in the quotient algebra is no more a congruence in general: consider for example two sorts A and B, with $A \leq B$, two constants $(a : A)$ and $(b : B)$, a unary operator $f : A \mapsto A$, and the equality $a = b$. Although $f(a) = f(b)$ is expected, $f(b)$ is not well-formed if parsing is performed using only the membership formulas $(a : A)$, $(b : B)$ and $(f(x) : A)$ for any variable x of sort A, corresponding to the operator declarations.

One possible solution to this problem is to impose sort-decreasingness and confluence of the rules associated with the equalities in a specification (see [GM92, GKK90]). However, there are examples, where sort-decreasingness is an uncomfortable restriction: for instance a definition of the square function on integers is given with two sorts Nat and Int with $Nat \leq Int$, operators $0 : \mapsto Nat$, $* : Int, Int \mapsto Int$, $sq : Int \mapsto Nat$, and for any variable $x :: Int$, $sq(x) = x * x$. However orienting this last equality is problematic since the rule $sq(x) \to x * x$ is not sort-decreasing.

Even with confluent and sort-decreasing rules, syntactic sorts are restrictive, since terms, that are not syntactically well-formed, can evaluate to a well-formed one: a well-known example is the stack of naturals, with sorts Nat (naturals), St (stacks) and $NeSt$ (non-empty stacks), such that $NeSt \leq St$, operators

$nil :\mapsto St$, $push : Nat, St \mapsto NeSt$, $pop : NeSt \mapsto St$, $top : NeSt \mapsto Nat$, variables $x :: Nat$, $z :: St$, and two rewrite rules $top(push(x,z)) \to x$ and $pop(push(x,z)) \to z$.
The term $top(pop(push(2, push(1, nil))))$ is not syntactically well-formed but clearly is semantically meaningful, since it evaluates to 1.

A solution for this problem were retracts, proposed in [GJM85, GM92]. Another solution consists in the semantic interpretation of membership formulas, i.e. equal terms belong by definition to the same sorts. Assuming sort-decreasingness, the term to be reduced is meaningful if and only if its normal form is well-formed [Wer93]. This approach also solves the congruence problem.

In order to further increase expressivity of order-sorted specifications, it makes sense to allow term declarations of the form $(t : A)$, that generalise flat and linear declarations, such as $(f(x) : B)$ for any $(x :: A)$ to declare an operator $f : A \mapsto B$. Non-linear term declarations are useful for instance in a specification involving again naturals and integers and the declarations $Nat \leq Int$, $0 :\mapsto Nat$, $suc : Nat \mapsto Nat$, $opp : Int \mapsto Int$, $* : Int, Int \mapsto Int$. Now, $(x * x : Nat)$ for any $x :: Int$ is a flat but non-linear membership formula.

Therefore, we adopt an algebraic framework called G-algebra, where membership formulas and equalities interact to compute semantic sorts of terms. However deduction in this context needs a unification which is undecidable in general, due to semantic membership and general term declarations.

In [HKK94a, HKK94b], we proposed an extended term structure, called decorated terms, where each node contains the set of sorts already proved for the corresponding subterm. Rewriting is defined on these terms and a completion process is proposed, based on the hypothesis that the axiomatisation is modularised in three parts: (i) equalities $(t = t')$, (ii) term declarations $(t : A)$ and (iii) sort inclusions $(A \leq B)$. (i) and (ii) are handled via rewrite rules (decorated and decoration rules) and are thus modified and enriched during completion. On the contrary, (iii) is stable during the *whole* completion. In particular, matchers and unifiers are computed using as usual the term structure but also the sort information given in decorations and in the *fixed* sort structure. Since matching and unification use only the sort information available in the decorated term at unification or matching time, they are correct but non complete in general for the peak reduction involved in the completion process. This completeness property will be achieved only at the end of the completion process provided it does not fail. In order to get an algorithm that computes a complete set of unifiers on decorated terms, it is necessary that the sort information given in part (iii) contains enough information to have the following property, called sort inheritance: if a term t can be proved to be of sorts A_i, $i \in [1..n]$, then there must exist a sort C with $C \leq A_i$ for all $i \in [1..n]$. Sort inheritance is in general undecidable. In [HKK94a], the completion process is performed assuming sort inheritance. When a fair completion does not fail, the resulting set of rewrite rules provides a way to prove not only equational theorems of the form $(t = t')$ but also membership theorems $(t : A)$ and existence theorems $(EX\ t)$ (cf. [Sco77]).

Thus sort inheritance is a crucial property in this framework. We propose

in this paper a test of sort inheritance based on the computation of top critical pairs between decoration rules generated during completion. It the test succeeds, it provides a counter-example for sort inheritance. Conversely, if the test never succeeds during completion, the specification is proved sort inheriting in three different situations: the first one is the case of flat and linear term declarations. This is the simplest case that corresponds to usual operator declarations. In that case, the test for sort inheritance can be postponed at the end of the completion process. The second case is when term declarations are shallow (i.e. have no variable at depth more than one). Then simplification and critical pair computation must be adapted to simultaneously reduce identical subterms at the same depth. Eventually, for general term declarations, all identical subterms have to be reduced in one step during simplification and a specific completion strategy has to be designed. This last case also forbids inter-reduction using decorated rewrite rules, which may cause divergence of the process in many cases.

This paper focusses on checking sort inheritance and is built as follows. The algebraic framework is stated in Section 2 and Section 3 explains the assumptions for the used sort structure. Then, in Section 4, the notions of decorated terms and rewriting with decorated and decoration rewrite rules are briefly recalled, with a completeness theorem stating the equivalence of replacement of equal by equal on decorated terms with deduction in G-algebra. Section 5 briefly gives the results of [HKK94a]. Section 6 provides the test for detecting non sort inheritance and successively considers the case of flat and linear term declarations, shallow declarations, and any kind of term declarations. In the conclusion, our approach is briefly compared with other related works. The full version of this paper [HKK94b] includes all proofs, examples and extended discussions of the concepts introduced here.

2 G-Algebra

A main feature of G-algebra is that function and term declarations, usually seen as part of the (static) signature in classical approaches, become formulas and are involved in proofs at the same level as equalities.

Our notations are consistent with [SNGM89, DJ90]. All notions concerning terms are defined in the same way as for classical terms.

Let $\Sigma = (\mathcal{S}, \mathcal{F})$ be a signature, providing a set of sorts $\mathcal{S}$ and of function symbols $\mathcal{F}$. $\mathcal{S}$ always contains the universal sort symbol Ω. The set of terms $\mathcal{T}(\Sigma, \mathcal{X})$ is defined as in the unsorted case, but each variable $x \in \mathcal{X}$ belongs to a unique sort, say A, denoted by $sort(x)$. This is written $(x :: A)$. With $A \in \mathcal{S}$ and $t, t' \in \mathcal{T}(\Sigma, \mathcal{X})$, existence formulas $(EX\ t)$, membership formulas $(t : A)$ and equalities $(t = t')$ are all implicitly closed by universal quantification.

Presentations $\mathcal{P}$ are sets of formulas. A pair $(signature, presentation)$ is called specification. A Σ-algebra $\mathcal{A}$ is given by a domain $|\mathcal{A}|$ and interpretations for each symbol in $\mathcal{S}$ and $\mathcal{F}$: each $A \in \mathcal{S}$ is interpreted as a non-empty set $A^{\mathcal{A}}$, $\Omega^{\mathcal{A}}$ as $|\mathcal{A}|$, and each $f \in \mathcal{F}$ as a partial function $f^{\mathcal{A}} : |\mathcal{A}|^{arity(f)} \to |\mathcal{A}|$. A variable assignment α is a mapping from $\mathcal{X}$ to $\Omega^{\mathcal{A}}$ with $\alpha(x) \in A^{\mathcal{A}}$ whenever

$sort(x) = A$ (for all $x \in \mathcal{X}$). A model of $\mathcal{P}$ is a Σ-algebra s.t. for all variable assignments α, for all $(EX\ t) \in \mathcal{P}$, $\alpha^*(t) \in \Omega^{\mathcal{A}}$, for all $(t : A) \in \mathcal{P}$, $\alpha^*(t) \in A^{\mathcal{A}}$ and for all $(t = t') \in \mathcal{P}$, $\alpha^*(t) = \alpha^*(t')$, where α^* is the unique homomorphic extension of α on terms. Homomorphisms are identical to those of [SNGM89]. If some formula ϕ is true in all models of $\mathcal{P}$, we write $\mathcal{P} \models \phi$.

In [Még90] (see also [HKK94a]), a sound and complete set of deduction rules is proposed, giving a relation $\vdash$ of deductibility, s.t. $\vdash$ and $\models$ coincide. The involved substitutions σ are supposed to be *conform* with the current presentation, i.e. $(x \mapsto t) \in \sigma$ and $(x :: A) \in \mathcal{P}$ implies $\mathcal{P} \vdash (t : A)$.

Specifications with inhabited sorts (i.e. for each $A \in \mathcal{S}$, there is some $t \in \mathcal{T}(\Sigma)$, s.t. $\mathcal{P} \models (t : A)$) have an initial Σ-model whose domain is $\{t \in \mathcal{T}(\Sigma) \mid \mathcal{P} \vdash EX\ t\}$. The congruence $\equiv$ defined on $\mathcal{T}(\Sigma)$ as $\{(t, t') \mid \mathcal{P} \vdash EX\ t, EX\ t', t = t'\}$ is s.t. the quotient $\mathcal{T}(\Sigma)_{/\equiv}$ is initial in the class of models of $\mathcal{P}$.

The main difference w.r.t. purely syntactic approaches, like [GM92], relies on the typing notion. Instead of syntactic typing, where the judgement whether some t in $\mathcal{T}(\Sigma, \mathcal{X})$ belongs to a sort $A \in \mathcal{S}$ only depends on the term declarations in $\mathcal{P}$, in our approach (as in [WD89, Com92, Wit92, Wer93]), such a judgement also depends on equalities. This actually corresponds to the intuition that equal terms belong to the same sorts and is reflected by the semantic sort rule in the deduction system of G-algebra [Még90]:

$$\boxed{t : A, t = s \quad \Rightarrow \quad s : A}$$

Let us come back to the stack of naturals example, expressed in G-algebra:

$$\mathcal{P} = \left\{ \begin{array}{l|l} x :: Nat, z :: St & y :: NeSt, y : St \\ nil : St & push(x, z) : NeSt \\ pop(y) : St & top(y) : Nat \\ top(push(x, z)) = x & pop(push(x, z)) = z \end{array} \right\}$$

First remark that $(y :: NeSt), (y : St)$ is the G-algebra way to express $NeSt \leq St$. Then $\mathcal{P} \models pop(pop(push(y, push(x, nil)))) : St$ and $\mathcal{P} \not\models pop(pop(push(x, nil))) : St$. Note also, that the first membership formula is not syntactically well-formed, although the term makes sense in any model of $\mathcal{P}$.

3 The Sort Structure

In order to effectively compute in G-algebra, we have to overcome the difficulty of semantic sorts. Indeed, defining a semantic sort ordering $\leq_{\mathcal{S}}^{sem}$ in $\mathcal{P}$ as the transitive and reflexive closure of the relation: $A \leq_{\mathcal{S}}^{sem} B$ if $(x :: A) \in \mathcal{P}$ implies $\mathcal{P} \vdash (x : B)$, results in undecidability of the subsort relation. Consequently, we need to work with an approximation. We use a syntactic relation, defined as the transitive and reflexive closure of $A \leq_{\mathcal{S}}^{syn} B$ if $\{x :: A,\ x : B\} \subseteq \mathcal{P}$. Therefore, expressing a subsort relation $A \leq_{\mathcal{S}}^{syn} B$ in our framework is equivalent to give a variable definition $(x :: A)$ and a corresponding term declaration $(x : B)$.

In ordered sort structures, incomparable sorts $A_1, \ldots, A_n$ without common subsort are considered to have no term in common. If there are common subsorts

$B_1, \ldots B_m$, usually as in [GM92], only terms in these common subsorts are considered to be simultaneously in $A_1, \ldots, A_n$. This is guaranteed by a restriction called regularity, saying that each term must have a unique least sort. Regularity ensures decidability of unification and therefore feasibility of completion.

Our approach introduces new sorts in order to cope also with terms in all A_i, $i \in [1..n]$, but not in the B_j's, $j \in [1..m]$. Hence, the next step is to find a replacement for regularity, adapted to the extended sort structure, that ensures decidability of unification. We call a specification $(\Sigma, \mathcal{P})$ *sort inheriting* (SI for short) w.r.t. a subsort ordering $\leq$, if for any term $t \in \mathcal{T}(\Sigma, \mathcal{X}), \forall T \subseteq \mathcal{S}$:

$$(\forall A \in T, \mathcal{P} \vdash t : A) \Rightarrow (\exists C \in \mathcal{S}, \forall A \in T, C \leq A)$$

Fortunately, sort inheritance w.r.t. $\leq^{syn}_{\mathcal{S}}$ is stricter than $\leq^{sem}_{\mathcal{S}}$. Hence, it is sufficient to test SI w.r.t. $\leq^{syn}_{\mathcal{S}}$, the decidable relation. This motivates to write SI without precising the used relation, which implicitly means SI w.r.t. $\leq^{syn}_{\mathcal{S}}$. Sort inheritance is undecidable in general, but a constructive test computing terms that destroy SI is developed in Section 6. Simple restrictions such as syntactic regularity and sort-decreasingness of [SNGM89, GKK90, Wal92] are sufficient to ensure SI.

We assume in the following that:

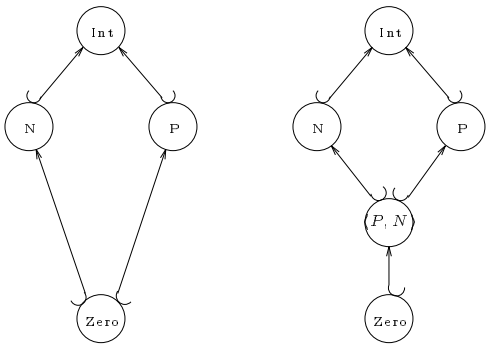
- (1) all sorts are non-empty,
- (2) $<^{syn}_{\mathcal{S}}$, the strict part of $\leq^{syn}_{\mathcal{S}}$, does not contain cycles,
- (3) the specification has bounded membership, i.e. $\{A \mid \mathcal{P} \vdash t : A\}$ is finite for all $t \in \mathcal{T}(\Sigma, \mathcal{X})$,
- (4) the set $mlb(S)$ of maximal elements of the set of lower bounds of any subset of sorts $S \subseteq \mathcal{S}$ is computable.

In our framework, since emptiness of sorts is related to equality, it is undecidable, but can be enforced by a decidable syntactic non-emptiness condition. In signatures with finite $\mathcal{S}$, as in G-algebra, (2) is decidable, (3) is trivial and (4) is satisfied. However, in polymorphic signatures more sophisticated properties have to be introduced (see [Smo89]).

To express unifiers of two variables using SI instead of regularity, an extended sort structure $(\mathcal{S}_\Diamond, \leq^{syn}_{\mathcal{S}_\Diamond})$ is needed. For a finite subset S of $\mathcal{S}$ s.t. $mlb(S) \neq \emptyset$ and all $A \in S$ are incomparable, the sort $\langle S \rangle$ may be understood as the intersection of all sorts A in S. Instead of $\langle \{A, \ldots\} \rangle$, we simply write $\langle A, \ldots \rangle$ and $\langle A \rangle$ is written A. $\mathcal{S}_\Diamond$ denotes the set of all $\langle S \rangle$. The sort ordering $\leq^{syn}_{\mathcal{S}}$ is conservatively extended to a new subsort relation $\leq^{syn}_{\mathcal{S}_\Diamond}$ defined by:

$$\langle S \rangle \leq^{syn}_{\mathcal{S}_\Diamond} \langle S' \rangle \text{ if } \forall B \in S', \exists A \in S, \text{ s.t. } A \leq^{syn}_{\mathcal{S}} B.$$

Example 1. Let $\mathcal{S} = \{Zero,\ P,\ N,\ Int\}$, s.t. $Zero \leq^{syn}_{\mathcal{S}} P$, $Zero \leq^{syn}_{\mathcal{S}} N$, $P \leq^{syn}_{\mathcal{S}} Int$, $N \leq^{syn}_{\mathcal{S}} Int$. Then $\mathcal{S}_\Diamond = \{Zero,\ \langle P, N \rangle,\ P,\ N,\ Int\}$. Now, $\langle P, N \rangle \leq^{syn}_{\mathcal{S}_\Diamond} N$, since $N \leq^{syn}_{\mathcal{S}} N$. Analogously, $\langle P, N \rangle \leq^{syn}_{\mathcal{S}_\Diamond} P$. Furthermore, $Zero \leq^{syn}_{\mathcal{S}_\Diamond} \langle P, N \rangle$, since $Zero \leq^{syn}_{\mathcal{S}} P$ and $Zero \leq^{syn}_{\mathcal{S}} N$. Finally, $N \leq^{syn}_{\mathcal{S}_\Diamond} Int$ and $P \leq^{syn}_{\mathcal{S}_\Diamond} Int$ hold as before. Remark that interpreting $\langle P, N \rangle$ intuitively as intersection of P and N results in the same subsort relation.



In what follows, we assume $(\Sigma, \mathcal{P})$ to be a fixed specification leading to an extended sort set $\mathcal{S}_\Diamond$ with a subsort relation $\leq^{syn}_{\mathcal{S}_\Diamond}$.

4 Decorated Terms, Rewriting and Presentations

Semantic sorts lead to difficulties when parsing terms. Clearly, as long as there is no decision procedure for typing in some presentation $\mathcal{P}$, we need to restrict the sorts of terms to those which are currently proved. Hence, we extend the term structure with typing information in each node.

Terms and Substitutions. Let $\mathcal{X}_\Diamond$ be a $\mathcal{S}_\Diamond$-sorted variable set and $\mathbf{V}$ a set of set-variables disjoint from $\mathcal{X}_\Diamond$. A decorated term is either a decorated variable $x^{:\{\langle A \rangle\}}$, where $(x :: A) \in \mathcal{X}_\Diamond$, or of the form $f(t_1{}^{:S_1}, \ldots, t_n{}^{:S_n})^{:S}$, where $t_1{}^{:S_1}, \ldots, t_n{}^{:S_n}$ are decorated terms, $f \in \mathcal{F}$ with $arity(f) = n$. Subsets S, S_i of $\mathcal{S}_\Diamond$, for $i \in [1..n]$, are called decorations. t_{nd} stands for $t^{:S}$ without decorations.

A decorated term $t^{:S}$ is said valid in a presentation $\mathcal{P}$ if all its subterms $u^{:U}$ satisfy $\forall \langle \ldots, A, \ldots \rangle \in U$, $\mathcal{P} \vdash (\alpha(u_{nd}) : A)$ for all $\mathcal{T}(\Sigma, \mathcal{X})$-instances α of u_{nd}. $\mathcal{T}_d(\mathcal{S}_\Diamond, \mathcal{F}, \mathcal{X}_\Diamond)$ or just $\mathcal{T}_d$ is the set of decorated $(\Sigma, \mathcal{X}_\Diamond)$-terms, $\mathcal{VT}_d$ the set of valid terms in $\mathcal{T}_d$. $t^{:\downarrow\emptyset}$ represents t with empty decorations everywhere, except at positions with variables, where $x^{:S}$ becomes $x^{:\{sort(x)\}}$. Syntactic equality over decorated terms is denoted by $=_d$.

To get decidable notions of matching and unification, we need to express SI on decorated terms as a property on their decorations only. Obviously, we have to avoid arbitrary decorations, since they can be interpreted incorrectly as membership formulas. Hence, it only makes sense to define SI on valid terms. Furthermore, a decidable test of SI can only be assured on subsets of $\mathcal{VT}_d$. Therefore, we need a property for arbitrary subsets $\mathcal{T}$ of $\mathcal{VT}_d$. So, a specification $(\Sigma, \mathcal{P})$ is $\mathcal{T}$-SI, if:

$$\forall t^{:T} \in \mathcal{T} \ : \ (\exists C \in \mathcal{S}, \forall A \in T \ : \ C \leq^{syn}_{\mathcal{S}_\Diamond} A).$$

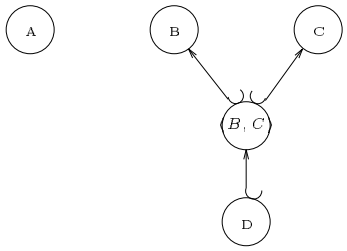
This means that we have added a sort $\langle S \rangle$ with $S = min(\{B \mid \langle \ldots, B, \ldots \rangle \in T\})$ to $\mathcal{S}_\Diamond$ w.r.t. $\mathcal{S}$, i.e. intuitively for the intersection of all sorts from $\mathcal{S}$ occurring in the decoration T. Hence, every term in $\mathcal{T}$ is *covered* by some unique, minimal

sort, just as this is the case with regularity. The difference is that SI is defined after an extension of the sort structure and furthermore SI is relative to a set of valid decorated terms. Remark that the extension of the sort structure depends only on the subsort relation $\leq_{\mathcal{S}}^{syn}$ and not on membership formulas, which are a priori undecidable.

Decorated substitutions are a subset of $\mathcal{P}$-conform substitutions. We restrict the used membership theory once more to the information already existing in the term nodes, modulo a SI closure that computes minimal sorts and performs transitive closure on them w.r.t. $\leq_{\mathcal{S}_\Diamond}^{syn}$. The SI closure $\widehat{S}$ of S is defined as:

$$\widehat{S} = \{D \in \mathcal{S}_\Diamond \mid \exists \langle T_1 \rangle, \ldots, \langle T_n \rangle \in S \ : \ \langle min(\bigcup_{i \in [1..n]} T_i) \rangle \leq_{\mathcal{S}_\Diamond}^{syn} D\}$$

Example 2. Let $\mathcal{S} = \{A, B, C, D\}$ with $D \leq_{\mathcal{S}}^{syn} B, D \leq_{\mathcal{S}}^{syn} C$ and therefore $\mathcal{S}_\Diamond = \{A, B, C, D, \langle B, C \rangle\}$ with $D \leq_{\mathcal{S}_\Diamond}^{syn} \langle B, C \rangle$, $\langle B, C \rangle \leq_{\mathcal{S}_\Diamond}^{syn} B$ and $\langle B, C \rangle \leq_{\mathcal{S}_\Diamond}^{syn} C$.



Let now $S_1 = \{A\}$, $S_2 = \{D\}$, $S_3 = \{B, C\}$ and $S_4 = \{A, B, C\}$ be subsets of $\mathcal{S}_\Diamond$. Then $\widehat{S_1} = \{A\}$, $\widehat{S_2} = \{B, C, D, \langle B, C \rangle\}$, $\widehat{S_3} = \{B, C, \langle B, C \rangle\}$ and $\widehat{S_4} = \{A, B, C, \langle B, C \rangle\}$.

$S \lesssim S'$ abbreviates $\widehat{S} \subseteq \widehat{S'}$, $S \approx S'$ means $S \lesssim S'$ and $S' \lesssim S$. The SI closure $\widehat{t^{:S}}$ of $t^{:S}$ is obtained by recursively applying $\widehat{\ }$ to its decorations. $\widehat{t^{:S}} = \widehat{t'^{:S'}}$ is written $t^{:S} \cong_d t'^{:S'}$ and means syntactic equality up to the SI closure of decorations at each node.

A decorated substitution σ is a mapping from decorated variables in $\mathcal{X}_\Diamond$ to valid decorated terms, such that if $\sigma(x^{:S}) =_d t^{:T}$ with $t^{:T} \neq_d x^{:S}$ and $(x :: A) \in \mathcal{P}$, then $A \in \widehat{T}$. Remark that $A \approx S$, since x is a decorated variable. We represent σ by its graph $\bigcup_{i \in [1..n]} \{x_i{}^{:S_i} \mapsto t_i{}^{:T_i}\}$ and $\mathcal{D}om(\sigma) = \{x_i \mid i \in [1..n]\}$.

Decorated term subsumption, matching and unification can be defined as for classical terms [JK91], using decorated substitutions and terms instead of the classical ones together with $\cong_d$ as term equality. The exact definitions can be found in [HKK94b] together with a matching and unification algorithm.

Rewriting. A decorated equality is a pair of decorated terms, denoted by $(p^{:S} = q^{:S'})$, $p^{:S}, q^{:S'} \in \mathcal{T}_d$. A decorated rewrite rule is an ordered pair of decorated terms, written $(p^{:S} \to q^{:S'})$, s.t. $p^{:S}, q^{:S'} \in \mathcal{T}_d$ and $\mathcal{V}ar(p^{:S}) \supseteq \mathcal{V}ar(q^{:S'})$. Applying equalities and decorated rewrite rules on valid terms is defined analogously to undecorated equalities and rewrite rules (as e.g. in [DJ90]), except

that the equality symbol ($=$) and substitutions have to be replaced by decorated equality modulo SI ($\cong_d$) and decorated substitutions.

A decorated one-step equality application in E is written $t^{:S} \longleftrightarrow^{\omega,\sigma,\phi}_{E} t'^{:S'}$, where $\phi \in E, \omega \in \mathcal{O}cc(t^{:S})$ and σ is a decorated substitution. Decorated rewriting is analogously denoted by $t^{:S} \rightarrowtail^{\omega,\sigma,\phi}_{R} t'^{:S'}$, where $\phi \in R$ and ω, σ are as before. If the used rule/equality, substitution or occurrence is not needed in the current context, we may simply omit it in our notation.

Example 3. Let $top(push(x^{:\{Nat\}}, z^{:\{St\}})^{:\{NeSt\}})^{:\{Nat\}} \rightarrow x^{:\{Nat\}}$ be a decorated rewrite rule in R. Then $top(push(0^{:\{Nat\}}, nil^{:\{St\}})^{:\emptyset})^{:\emptyset}$ cannot be rewritten, but $top(push(0^{:\{Nat\}}, nil^{:\{St\}})^{:\{NeSt\}})^{:\{Nat\}} \rightarrowtail_R 0^{:\{Nat\}}$.

Another kind of rules is introduced, whose purpose is to increase decorations. Let s be in $\mathbf{V}$. A decoration rewrite rule is of the form $(l^{:s} \rightarrow l^{:s \cup S_l}$ if $S_l \lesssim s)$, where $l^{:\emptyset} \in \mathcal{T}_d$ and $S_l \subseteq \mathcal{S}_\Diamond$.

The valid decorated term $t^{:T}$ rewrites in D to $t'^{:T'}$, written $t^{:T} \rightarrowtail^{\omega,\sigma,\phi}_{D} t'^{:T'}$, if $\phi = (l^{:s} \rightarrow l^{:s \cup S_l}$ if $S_l \lesssim s) \in D$, $t^{:T}|_\omega =_d u^{:U}$, σ is a decorated match from $l^{:U}$ to $u^{:U}$, $S_l \lesssim U$ and $t'^{:T'} =_d t^{:T}[\sigma(l^{:U})^{:U \cup S_l}]_\omega$.

Decorated and decoration rewriting are stable by context and substitution. Furthermore, they preserve validity of terms. To prove termination, the classical notion of reduction ordering can easily be extended on decorated terms (see [HKK94b] for details). A decorated or decoration rewrite step is denoted by $\rightarrowtail_{R \cup D}$, an application of $\rightarrowtail_D$ or $\longleftrightarrow_E$ is denoted by $\longleftrightarrow_{D \cup E}$. A star over a relation denotes its reflexive and transitive closure.

Example 4. Let $(0^{:s} \rightarrow 0^{:s \cup \{Nat\}}$ if $\{Nat\} \lesssim s)$ be a decoration rewrite rule. Then $0^{:\{Int\}} \rightarrowtail_D 0^{:\{Nat,Int\}}$. Remark that $\{Nat, Int\} \approx \{Nat\}$ if $Nat \leq^{syn}_{\mathcal{S}_\Diamond} Int$.

Decorated Presentations. In order to replace $\vdash$-derivations in the G-algebra deduction system [Még90] by $\longleftrightarrow_{D,E,R}$-steps, we have to extract a set of decorated equalities $E_\mathcal{P}$ and a set of decoration rules $D_\mathcal{P}$ from $\mathcal{P}$. Therefore, we take for each subterm u of a term appearing in a formula in $\mathcal{P}$ a decoration rewrite rule $((u^{:\downarrow\emptyset})^{:s} \rightarrow (u^{:\downarrow\emptyset})^{:s \cup \{\Omega\}}$ if $\{\Omega\} \lesssim s)$. For all $(u : A)$ in $\mathcal{P}$, s.t. $u \notin \mathcal{X}$, we add $((u^{:\downarrow\emptyset})^{:s} \rightarrow (u^{:\downarrow\emptyset})^{:s \cup \{A\}}$ if $\{A\} \lesssim s)$. The resulting set of decoration rewrite rules is called $D_\mathcal{P}$. Analogously, $E_\mathcal{P}$ is defined as the set of decorated equalities $(p^{:\downarrow\emptyset} = q^{:\downarrow\emptyset})$, for each equality $(p = q)$ in $\mathcal{P}$. Now, we can state:

Theorem 1. *[HKK94b] Let t, t' be two terms, $\mathcal{P}$ be a Σ-presentation. Let A be a sort and U be a set of sorts. Then:*

$(\mathcal{P} \vdash t = t') \Leftrightarrow \exists t_0, A$ *s.t.* $t^{:\downarrow\emptyset} \longleftrightarrow^*_{D_\mathcal{P} \cup E_\mathcal{P}} t_0^{:\{A\} \cup U} \longleftrightarrow^*_{D_\mathcal{P} \cup E_\mathcal{P}} t'^{:\downarrow\emptyset}$.

$(\mathcal{P} \vdash t : A) \Leftrightarrow \exists t', A' \leq^{syn}_{\mathcal{S}_\Diamond} A$ *s.t.* $t^{:\downarrow\emptyset} \longleftrightarrow^*_{D_\mathcal{P} \cup E_\mathcal{P}} t'^{:\{A'\} \cup U}$.

$(\mathcal{P} \vdash EX\ t) \Leftrightarrow \exists t'$ *s.t.* $t^{:\downarrow\emptyset} \longleftrightarrow^*_{D_\mathcal{P} \cup E_\mathcal{P}} t'^{:\{A\} \cup U}$.

5 Completion

In order to further motivate the need for SI, let us briefly recall the completion process in which it appears as an essential requirement. A completion procedure is described in [HKK94a] by a set $\mathcal{OSC}$ of rules that transforms triples

$P = (D, E, R)$, called decorated presentations, starting from $P_0 = (D_{\mathcal{P}}, E_{\mathcal{P}}, \emptyset)$. $D_\infty, E_\infty, R_\infty$ denote the sets of persisting rules and equalities in a derivation using $\mathcal{OSC}$. The purpose of completion is to generate a resulting decorated presentation, given by $(D_\infty, E_\infty, R_\infty)$ where $E_\infty = \emptyset$, that satisfies the following properties for $D_\infty \cup R_\infty$:

- Church-Rosser property:
  $\mathcal{P} \vdash (t = t') \Leftrightarrow \exists t'', A, S : t^{:\downarrow\emptyset} \overset{*}{\rightarrowtail}_{R_\infty \cup D_\infty} t''^{:\{A\}\cup S} \overset{*}{\leftarrowtail}_{R_\infty \cup D_\infty} t'^{:\downarrow\emptyset}$,
- Type completeness:
  $\mathcal{P} \vdash (t : A) \Leftrightarrow \exists t', S, A' \leq^{syn}_{\mathcal{S}_\Diamond} A : t^{:\downarrow\emptyset} \overset{*}{\rightarrowtail}_{R_\infty \cup D_\infty} t'^{:\{A'\}\cup S}$,
- Existential completeness:
  $\mathcal{P} \vdash (EX\ t) \Leftrightarrow \exists t', A, S : t^{:\downarrow\emptyset} \overset{*}{\rightarrowtail}_{R_\infty \cup D_\infty} t'^{:\{A\}\cup S}$,

The completion procedure essentially computes critical pairs between the two kinds of decoration/decorated rewrite rules, in order to handle critical interactions between membership formulas and rewrite rules in a convenient way. Simplification of critical pairs and inter-reduction of rules using decoration and decorated rules are allowed as in usual completion. The difference relies in the orientation process that always produces rewrite rules that increase the sort information at the replacement position, but in addition may introduce new membership formulas via decoration rewrite rules. The two orientation rules are given in Figure 1 and use an ordering $>_d$ on decorated terms defined in [HKK94b].

1. **Orient_SD**
$$\frac{D, E \cup \{p^{:S} = q^{:S'}\}, R}{D, E, R \cup \{p^{:S} \to q^{:S'}\}} \quad \text{if } p^{:S} >_d q^{:S'} \text{ and } S \subsetneq S'$$

2. **Orient_NSD**
$$\frac{D, E \cup \{p^{:S} = q^{:S'}\}, R}{D \cup \{(q^{:s} \to q^{:s \cup S \setminus S'} \text{ if } S \setminus S' \subsetneq s)\}, E, R \cup \{p^{:S} \to q^{:S \cup S'}\}}$$
if $p^{:S} >_d q^{:S\cup S'}$ and $S \subsetneq S'$
and if $q \in \mathcal{X}_\Diamond$ with $q :: A$ then $\{A\} \approx S \cup S'$

Fig. 1. Orientation rules for decorated equations.

Assuming a sort inheriting initial presentation, the main result for the completion process is stated in the next theorem.

Theorem 2. *[HKK94b] Let $\mathcal{P}$ be SI and $P_\infty = (D_\infty, E_\infty, R_\infty)$ obtained with $\mathcal{OSC}$ from $(D_{\mathcal{P}}, E_{\mathcal{P}}, \emptyset)$ s.t. $E_\infty = \emptyset$ and all critical pairs of $D_\infty \cup R_\infty$ have been computed. Then $D_\infty \cup R_\infty$ is terminating, Church-Rosser, type complete and existentially complete on $\mathcal{VT}_d$.*

In the orientation rules given in Figure 1, decorated equations of the form $(p^{:S} = x^{:S'})$ s.t. $S \cup S' \not\approx sort(x)$ are not oriented, and thus the condition $E_\infty = \emptyset$

of Theorem 2 may not be fulfilled. We add a failure rule detecting this case:

Detect_Subsort

$$\frac{D, E \cup \{p^{:S \cup \{B\}} = (x :: A)^{:S'}\}, R}{\bot, \bot, \bot} \text{ if } x \in \mathcal{X}_\Diamond \text{ and } x :: A \in \mathcal{P} \text{ and not } A \leq^{syn}_{\mathcal{S}_\Diamond} B$$

With this additional failure rule, we can improve Theorem 2 with another result more precisely stated in [HKK94a]: starting from a sort inheriting presentation, when completion succeeds (i.e. terminates after computing all the critical pairs and without any application of the **Detect_Subsort** rule), then $\leq^{syn}_{\mathcal{S}} = \leq^{sem}_{\mathcal{S}}$. When **Detect_Subsort** is applicable, then $\leq^{syn}_{\mathcal{S}} \neq \leq^{sem}_{\mathcal{S}}$. This means in particular that the unification algorithm used in the completion does not always compute a complete set of unifiers in the initial presentation. However when **Detect_Subsort** applies, it provides us with the information that $\mathcal{P} \vdash (x : B)$. Adding this term declaration to the presentation yields a conservative extension of the initial presentation $\mathcal{P}$. Consequently, the completion can be restarted, but now using more information, since $\leq^{syn}_{\mathcal{S}}$ has then been increased.

6 Testing Sort Inheritance

Since all results of the last section depend on the sort inheritance of $\mathcal{P}$, we are now left with the problem of designing a test for this undecidable property. We propose in this section a test that is applied in three kinds of completion processes according to the form of term declarations. In all three cases, the completion process including the test allows checking sort inheritance of the initial presentation.

The test for detecting non sort inheritance is realized by the **Detect_NonSI** rule given below.

Detect_NonSI

$$\frac{D \cup_{i \in [0..n]} \{(p_i^{:s} \to p_i^{:s \cup S_i} \text{ if } S_i \lneq s)\}, E, R}{\bot, \bot, \bot}$$

$$\text{if } (\exists \psi, \forall i \in [1..n], \psi(p_0^{:\emptyset}) \cong_d \psi(p_i^{:\emptyset}))$$
$$\text{and } (\exists S \subseteq \bigcup_{i \in [0..n]} S_i, \nexists C \leq^{syn}_{\mathcal{S}_\Diamond} S)$$

Clearly, if **Detect_NonSI** applies, then $\mathcal{P}$ is not SI. If $P_\infty = (\bot, \bot, \bot)$, then we can add $(\psi(p^{:\emptyset})_{nd} : C)$, since this is a conservative extension of the initial presentation $\mathcal{P}$. As in the case of **Detect_Subsort**, the completion can then be restarted.

Moreover, this test characterises SI on the set of decorated terms called D-typable, i.e. reachable with D from terms with empty decorations, provided confluence of D:

Proposition 3. *In a decorated presentation $P = (D, E, R)$, let us consider the set $\mathcal{T}_D = \{t \mid \exists t' \in \mathcal{T}_d{}^{:\downarrow\emptyset} : t' \overset{*}{\leftrightarrow}_D t\}$. Let us assume that D is confluent on $\mathcal{T}_D$. Then, the* **Detect_NonSI***-rule succeeds on D iff P is not $\mathcal{T}_D$-SI.*

The main difficulty is to extend this result from $\mathcal{T}_D$ to the set of all valid decorated terms. This extension relies on the following facts.
(1) The proof of Theorem 1 reveals that for each G-algebra proof, there is a proof using $\overset{*}{\leftrightarrow}_{D_\mathcal{P} \cup E_\mathcal{P}}$, s.t. all decorated terms $t^{:S}$ in the latter have a typing proof $(t^{:S})^{:\downarrow\emptyset} \overset{*}{\leftrightarrow}_{D_\mathcal{P}} t^{:S}$.
(2) If we check that the set of all typable terms does not contain any counter-example for SI, then the unification algorithm computes all critical pairs needed for peak reduction.
(3) Typability can be preserved until P_∞, if proof reductions and completion strategies are adapted to the form of term declarations.

The main problems are hence first to find a proof transformation that preserves typability of terms and second to ensure that the set of typable terms at each step of the completion does not contain a counter-example for SI.

Under these conditions, if completion does not fail, $\mathcal{P}$ is SI: indeed if $t^{:S}$ is a counter-example for SI of $\mathcal{P}$, there must exist (cf. Theorem 1) a proof

$$\Psi \; : \; (t^{:\downarrow\emptyset} \overset{*}{\leftrightarrow}_{P_0} t_1{}^{:S_1} \overset{*}{\leftrightarrow}_{P_0} t'{}^{:\downarrow\emptyset} \overset{*}{\leftrightarrow}_{P_0} \ldots \overset{*}{\leftrightarrow}_{P_0} t_n{}^{:S_n} \overset{*}{\leftrightarrow}_{P_0} t'{}^{:\downarrow\emptyset})$$

such that $S \subseteq \cup_{i\in[1..n]} S_i$. By proof reduction, we get a rewrite proof: $\Psi' \; : \; (t^{:\downarrow\emptyset} \overset{*}{\rightarrow}_{D_\infty \cup R_\infty} t'^{:S'})$ with $\cup_{i\in[1..n]} S_i \subseteq S'$ and $t'^{:S'}$ is also a counterexample for SI. Since typability is preserved, $t'^{:S'}$ has yet a typing proof using D_∞ $(t'^{:\downarrow\emptyset} \overset{*}{\leftrightarrow}_{D_\infty} t'^{:S'})$, so **Detect_NonSI** must apply to D_∞.

6.1 Flat and Linear Term Declarations

Let us call $\mathcal{SSC}$ the set of completion rules $\mathcal{OSC}$ with the two additional failure rules **Detect_Subsort** and **Detect_NonSI**.

The case of flat and linear term declarations is the most simple one. We can even postpone the relatively expensive application of the **Detect_nonSI** rule to the final decorated presentation, provided we ensure that if we simplify decoration rewrite rules with decorated rewrite rules, then the latter are decoration preserving, i.e. of the form $\phi \; : \; l^{:S} \rightarrow r^{:S}$. In this case, we call also the completion derivation decoration preserving.

Theorem 4. *[HKK94b] Let $P_\infty \neq (\bot, \bot, \bot)$ be obtained with $\mathcal{SSC}$ from $(D_\mathcal{P}, E_\mathcal{P}, \emptyset)$. Let us assume furthermore that all terms in D_∞ are flat and linear, $E_\infty = \emptyset$, all critical pairs of $D_\infty \cup R_\infty$ have been computed and the completion derivation is decoration preserving. Then the initial presentation $\mathcal{P}$ is SI on $\mathcal{VT}_d$ and $D_\infty \cup R_\infty$ is Church-Rosser, type and existentially complete.*

Intuitively, if all terms in D_∞ are flat and linear and $P_\infty \neq (\bot, \bot, \bot)$, the rewrite proofs with $D_\infty \cup R_\infty$ and all terms in proofs in intermediate decorated presentation P_k only contain D_∞-typable terms, so $\mathcal{T}_{D_\infty}$-SI is equivalent to $\mathcal{VT}_d$-SI. Therefore, it is sufficient to test SI on D_∞.

6.2 Shallow Term Declarations

A similar result to Theorem 4 for flat, possibly non-linear term declarations can be proved. It covers the class of presentations with shallow term declarations $(t : A)$, where all $x \in \mathcal{V}ar(t)$ occur either at the top or at depth one in t. These presentations can be conservatively transformed into presentations with flat, possibly non-linear term declarations.

Dropping the linearity condition forces us to simultaneously reduce identical subterms at the same depth, and to prohibit decoration rewrite rule simplification with decorated rewrite rules. Let $t^{:S}, t'^{:S'} \in \mathcal{T}_d(\mathcal{S}_\Diamond, \mathcal{F}, \mathcal{X}_\Diamond)$, $\phi \in R$ and k be a natural number. Then $t^{:S}$ *layer rewrites* to $t'^{:S'}$, written $t^{:S} \rightleftarrows_R^{\phi,\sigma,k} t'^{:S'}$, if there exists a maximal set O of positions ω in t s.t.$\forall \omega \in O$, $|\omega| = k$ and t concurrently rewrites at positions in O with the same rule ϕ and substitution σ.

Example 5. Let $\phi = (opp(opp(x^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}} \to x^{:\{Nat\}}) \in R$ be a decorated rewrite rule. Consider the decorated following terms:

$$t_1 = (opp(opp(0^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}} * opp(opp(0^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}})^{:\{Int\}}$$
$$t_2 = (opp(opp(s(0^{:\{Nat\}})^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}} * opp(opp(0^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}})^{:\{Int\}}$$
$$t_3 = (opp(opp(0^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}} * s(opp(opp(0^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}})^{:\{Int\}})^{:\{Int\}}$$

Let $\sigma = \{x^{:\{Nat\}} \mapsto 0^{:\{Nat\}}\}$. Then

$$t_1 \rightleftarrows_R^{\phi,\sigma,1} (0^{:\{Nat\}} * 0^{:\{Nat\}})^{:\{Int\}}$$
$$t_2 \rightleftarrows_R^{\phi,\sigma,1} (opp(opp(s(0^{:\{Nat\}})^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}} * 0^{:\{Nat\}})^{:\{Int\}}$$
$$t_3 \rightleftarrows_R^{\phi,\sigma,1} (0^{:\{Nat\}} * s(opp(opp(0^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}})^{:\{Int\}})^{:\{Int\}}$$

Remark that in t_2, the left subterm of $*$ does not match σ and in t_3, there are two identical redexes, but at different depth.

Let us call $\mathcal{LSC}$ the set of completion rules similar to $\mathcal{SSC}$, except that layer rewriting is used instead of standard decorated rewriting defined in section 4, which also changes the definition of critical pairs (see [HKK94b]), now called *layer critical pairs.*

Theorem 5. *[HKK94b] Let $P_\infty \neq (\bot, \bot, \bot)$ be obtained with $\mathcal{LSC}$ from $(D_\mathcal{P}, E_\mathcal{P}, \emptyset)$. Let us assume furthermore that $E_\infty = \emptyset$, all layer critical pairs of $D_\infty \cup R_\infty$ have been computed, and all terms in all generated decoration rules are flat. Then the initial presentation $\mathcal{P}$ is SI on $\mathcal{VT}_d$ and $D_\infty \cup R_\infty$ is Church-Rosser, type and existentially complete.*

Once more, one can prove that it is sufficient to test SI in D_∞. The following example illustrates the use of this proposition and provides a comparison with [Wer93]. Remark that $\rightleftarrows_R$ and $\rightarrowtail_R$ coincide here, whenever $\rightleftarrows_R$ is used.

Example 6. $\mathcal{P} = \{x :: N,\ y :: Z,\ z :: Z,\ x : Z,\ 0 : N,\ suc(x) : N,\ opp(y) : Z,\ sq(y) : N,\ sqrt(x) : N,\ |y| : N,\ y * z : Z,\ |x| = x,\ sq(y) = y * y,\ opp(y) * opp(y) = y * y\}$ The initial decoration rules are:

$$\begin{array}{rll}
0^{:s} & \to 0^{:s\cup\{N\}} & \text{if } \{N\} \not\subseteq s \\
suc(x^{:\{N\}})^{:s} & \to suc(x^{:\{N\}})^{:s\cup\{N\}} & \text{if } \{N\} \not\subseteq s' \\
opp(y^{:\{Z\}})^{:s} & \to opp(y^{:\{Z\}})^{:s\cup\{Z\}} & \text{if } \{Z\} \not\subseteq s \\
sq(y^{:\{Z\}})^{:s} & \to sq(y^{:\{Z\}})^{:s\cup\{N\}} & \text{if } \{N\} \not\subseteq s \\
sqrt(x^{:\{N\}})^{:s} & \to sqrt(x^{:\{N\}})^{:s\cup\{N\}} & \text{if } \{N\} \not\subseteq s \\
|y^{:\{Z\}}|^{:s} & \to |y^{:\{Z\}}|^{:s\cup\{N\}} & \text{if } \{N\} \not\subseteq s \\
(y^{:\{Z\}} * z^{:\{Z\}})^{:s} & \to (y^{:\{Z\}} * z^{:\{Z\}})^{:s\cup\{Z\}} & \text{if } \{Z\} \not\subseteq s
\end{array}$$

The initial decorated equalities:

$$\begin{array}{rl}
|x^{:\{N\}}|^{:\emptyset} & = x^{:\{N\}} \\
sq(y^{:\{Z\}})^{:\emptyset} & = (y^{:\{Z\}} * y^{:\{Z\}})^{:\emptyset} \\
(opp(y^{:\{Z\}})^{:\emptyset} * opp(y^{:\{Z\}})^{:\emptyset})^{:\emptyset} & = (y^{:\{Z\}} * y^{:\{Z\}})^{:\emptyset}
\end{array}$$

After simplification with decoration rewrite rules, we get:

$$\begin{array}{rl}
|x^{:\{N\}}|^{:\{N\}} & = x^{:\{N\}} \\
sq(y^{:\{Z\}})^{:\{N\}} & = (y^{:\{Z\}} * y^{:\{Z\}})^{:\{Z\}} \\
(opp(y^{:\{Z\}})^{:\{Z\}} * opp(y^{:\{Z\}})^{:\{Z\}})^{:\{Z\}} & = (y^{:\{Z\}} * y^{:\{Z\}})^{:\{Z\}}
\end{array}$$

Now, decorating and orienting the equalities yields:

$$\begin{array}{rll}
|x^{:\{N\}}|^{:\{N\}} & \to x^{:\{N\}} & \\
sq(y^{:\{Z\}})^{:\{N\}} & \to (y^{:\{Z\}} * y^{:\{Z\}})^{:\{N,Z\}} & \\
(y^{:\{Z\}} * y^{:\{Z\}})^{:s} & \to (y^{:\{Z\}} * y^{:\{Z\}})^{:s\cup\{N\}} & \text{if } \{N\} \not\subseteq s \\
(opp(y^{:\{Z\}})^{:\{Z\}} * opp(y^{:\{Z\}})^{:\{Z\}})^{:\{Z\}} & \to (y^{:\{Z\}} * y^{:\{Z\}})^{:\{N,Z\}} &
\end{array}$$

This is already the final presentation, since no more completion rule is applicable.

The equality $sqrt(sq(y)) = sqrt(sq(opp(y)))$ can be proved as follows:

$$\begin{array}{rl}
sqrt(sq(y))^{:\downarrow\emptyset}\downarrow_{D\cup R} =_d & sqrt((y^{:\{Z\}} * y^{:\{Z\}})^{:\{N,Z\}})^{:\{N\}} \\
=_d & sqrt(sq(opp(y)))^{:\downarrow\emptyset}\downarrow_{D\cup R}.
\end{array}$$

6.3 Arbitrary Term Declarations

Further extension for non-linear arbitrary term declarations needs a different proof reduction [HKK94b] and a rewrite relation in which all identical redexes are reduced simultaneously. A decorated term $t^{:S}$ *rewrites in a maximally subterm sharing way* into $t'^{:S'}$ using a decorated rewrite rule $\phi : l^{:S_l} \to r^{:S_r}$ and a decorated substitution σ if there exists a maximal set of positions $O = \{\omega \in \mathcal{O}cc(t^{:S}) \mid t^{:S}|_\omega \cong_d \sigma(l^{:S_l})\}$ s.t. t concurrently rewrites at all positions in O. This is written $t^{:S} \rightarrowtail\!\!\!\!\!\to^{\sigma,\phi}_R t'^{:S'}$. Clearly, this changes once again the definition of critical pairs.

Example 7. Consider once more ϕ, t_1, t_2, t_3 and σ from example 5. Then

$$
\begin{aligned}
t_1 &\rightarrowtail\!\!\!\!\!\rightarrow_R^{\sigma,\phi} (0^{:\{Nat\}} * 0^{:\{Nat\}})^{:\{Int\}} \\
t_2 &\rightarrowtail\!\!\!\!\!\rightarrow_R^{\sigma,\phi} (opp(opp(s(0^{:\{Nat\}})^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}} * 0^{:\{Nat\}})^{:\{Int\}} \\
t_3 &\rightarrowtail\!\!\!\!\!\rightarrow_R^{\sigma,\phi} (0^{:\{Nat\}} * s(0^{:\{Nat\}})^{:\{Int\}})^{:\{Int\}}
\end{aligned}
$$

Remark that in t_2, again the left subterm of $*$ does not match σ, but in t_3, the two identical redexes are reduced simultaneously this time.

The set of completion rules called $\mathcal{MSSC}$ is obtained from $\mathcal{SSC}$ by dropping any simplification by decorated rewrite rules and using adequate critical pairs, called $\mathcal{MSS}$-critical pairs. The completion rules have to be applied with a strategy that essentially gives a higher priority to computation of critical pairs between decoration rewrite rules (see [HKK94b]).

Theorem 6. *Let $P_\infty \neq (\bot, \bot, \bot)$ be obtained with $\mathcal{MSSC}$ from $(D_\mathcal{P}, E_\mathcal{P}, \emptyset)$ s.t. the strategy restrictions are fulfilled. Let us assume furthermore that $E_\infty = \emptyset$ and all $\mathcal{MSS}$-critical pairs of $D_\infty \cup R_\infty$ have been computed. Then the initial presentation $\mathcal{P}$ is SI on $\mathcal{VT}_d$ and $D_\infty \cup R_\infty$ is Church-Rosser, type and existentially complete.*

Remark that under the conditions of this theorem, $\mathcal{P}$ is SI and hence full $\mathcal{OSC}$ from [HKK94a] can be applied to continue completion, i.e. essentially interreduce the rules in $D_\infty \cup R_\infty$. Note that every $\rightrightarrows_R^{\phi,\sigma,k}$ and $\rightarrowtail\!\!\!\!\!\rightarrow_R^{\sigma,\phi}$ step can be replaced by a sequence of $\rightarrowtail_R^{\sigma,\phi,\omega}$ steps, i.e. all proofs using the former relations can be transformed into ones using $\rightarrowtail_R^{\sigma,\phi,\omega}$ only.

In order to illustrate the difficulties arising with non-flat, non-linear term declarations, consider the following example:

Example 8. Let $nil \rightarrow List$ and $cons : NatList \rightarrow List$ be the usual operator declarations for lists of natural numbers. Now, if we want to distinguish lists, where two identical numbers follow each other (let us call them ML, for multi-lists), we need to say something like, $cons(x, cons(x, l)) : ML$, where x is of sort Nat and l of sort $List$.

Remark that ML should now be declared as subsort of $List$, i.e. $ML \leq_\mathcal{S}^{syn} List$. The non-linear, non-flat term declaration becomes in the decorated term framework the following self-overlapping decoration rewrite rule ϕ:

$$
\begin{aligned}
&cons(x^{:\{Nat\}}, cons(x^{:\{Nat\}}, l^{:\{List\}})^{:\{List\}})^{:s} \\
&\quad \rightarrow cons(x^{:\{Nat\}}, cons(x^{:\{Nat\}}, l^{:\{List\}})^{:\{List\}})^{:s \cup \{ML\}} \text{ if } \{ML\} \not\subseteq s
\end{aligned}
$$

Now, overlapping ϕ at position 2 with itself or the result of the overlap can be repeated ad infinitum, resulting in a non-terminating completion of the set of decoration rewrite rules. Hence, the SI test is not complete either, since we cannot guarantee confluence. This situation may be encountered with practically relevant examples – for instance multisets realized as ordered lists.

The way out of this dilemma seems to be the use of more sophisticated decoration rewrite rules, like the following ϕ':

$$cons(x^{:\{Nat\}}, cons(x^{:\{Nat\}}, l^{:\{List\}})^{:\{ML\}})^{:s} \\ \rightarrow cons(x^{:\{Nat\}}, cons(x^{:\{Nat\}}, l^{:\{List\}})^{:\{ML\}})^{:s \cup \{ML\}} \text{ if } \{ML\} \not\subseteq s$$

Remark that the only difference to ϕ consists in the decoration at position 2, which became ML instead of $List$. Using ϕ and ϕ' does not prevent ϕ from being self-overlapping but makes the result of the overlap being subsumed by ϕ'. Hence, the completion of ϕ and ϕ' only terminates.

ϕ' does not correspond to any formula in G-algebra. However, allowing for conditional rules for membership formulas, similar to the sort constraints in [GJM85], gives an extension of G-algebra where ϕ' is translated into the following formula:

$$cons(x, cons(x, l)) : ML \quad \textbf{if} \quad cons(x, l) : ML$$

We extended G-algebras lately to an equational Horn clause logics, called G^n-logics [HKM94], where sets of nesting depth up to n can be specified. Sets of depth 1 correspond with sorts. G^n-logics share the useful properties of existence of a sound and complete deduction system and initial models with G-algebras. We hope to extend the results given in this paper to a fragment of G^n-logics covering the problems illustrated by the last example.

7 Related Work and Conclusion

Completion procedures for order-sorted algebraic specifications have already been proposed, but either fail by non-sort decreasingness or do not handle term declarations and semantic sorts. The completion using "syntactic sorts" [GKK90] is subsumed by our completion, i.e. for every completion in that framework, we can do a similar one using our decorated completion.

The tree automata approach of [Com92] produces rewrite rule schemas using second order variables instead of critical pairs between decoration rewrite rules and decorated rewrite rules. In [HKK94b], we give an example for a specification, that can be completed in a finite number of steps using our approach, but which does not terminate with the approach described in [Com92].

Another related approach is the signature extension method [CH91], which introduces new function symbols in order to solve the problem of equalities that cannot be oriented into sort-decreasing rules. However, this technique does not seem to be well-adapted to functional programming, since evaluation may result in a term involving a new function symbol that has no interpretation in the initial specification.

The T-contact method ([Wer93]) uses variable overlaps in order to cope with non-sort-decreasing rules. This results in a high number of new equations and may cause the completion to diverge.

The works of L. With (see [Wit92]) or Watson and Dick [WD89]) are very close to our approach, but do not really contain solutions concerning the undecidability problems for unification. More detailed comparisons with all these approaches can be found in [HKK94b]. We currently investigate relations with unified algebras [Mos89] or many-sorted algebras with semantic sorts and sort operations [Mei92].

To summarize, the main contribution of this paper relies in the elaboration of a test for sort inheritance in presentations with term declarations. Even when term declarations are not explicitely used in specifications, they may occur as a consequence of equality orientation when the semantic sort approach is adopted (see Example 6) and it is thus crucial to handle them in completion.

Decorations have been successfully used to formalize typing and to compute with semantic sorts. While keeping the interesting notion of sorts as constraints, they provide an adequate tool for testing sort inheritance.

Acknowledgements: We thank Uwe Waldmann and Andreas Werner for their comments on earlier drafts of this work. This work is partially supported by the Esprit Basic Research working group 6112, COMPASS.

References

[CH91] H. Chen and J. Hsiang. Order-sorted equational specification and completion. Technical report, State University of New York at Stony Brook, November 1991.

[Com92] H. Comon. Completion of rewrite systems with membership constraints. In W. Kuich, editor, *Proceedings of ICALP 92*, volume 623 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 6, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990.

[GJM85] J. A. Goguen, J.-P. Jouannaud, and J. Meseguer. Operational semantics for order-sorted algebra. In W. Brauer, editor, *Proceeding of the 12th International Colloquium on Automata, Languages and Programming, Nafplion (Greece)*, volume 194 of *Lecture Notes in Computer Science*, pages 221–231. Springer-Verlag, 1985.

[GKK90] I. Gnaedig, C. Kirchner, and H. Kirchner. Equational completion in order-sorted algebras. *Theoretical Computer Science*, 72:169–202, 1990.

[GM92] J. A. Goguen and J. Meseguer. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 2(105):217–273, 1992.

[HKK94a] C. Hintermeier, C. Kirchner, and H. Kirchner. Dynamically-typed computations for order-sorted equational presentations –extended abstract–. In S. Abiteboul and E. Shamir, editors, *Proc. 21st International Colloquium on Automata, Languages, and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 450–461. Springer-Verlag, 1994.

[HKK94b] C. Hintermeier, C. Kirchner, and H. Kirchner. Dynamically-typed computations for order-sorted equational presentations. research report 2208, INRIA, Inria Lorraine, March 1994. 114 p., also as CRIN report 93-R-309.

[HKM94] C. Hintermeier, H. Kirchner, and P. Mosses. R^n- and G^n-logics. Technical report, Centre de Recherche en Informatique de Nancy, 1994.

[JK91] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: a rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic. Essays in honor of Alan Robinson*, chapter 8, pages 257–321. The MIT press, Cambridge (MA, USA), 1991.

[Még90] A. Mégrelis. *Algèbre galactique — Un procédé de calcul formel, relatif aux semi-fonctions, à l'inclusion et à l'égalité.* Thèse de Doctorat d'Université, Université de Nancy 1, 1990.

[Mei92] K. Meinke. Algebraic semantics of rewriting terms and types. In M. Rusinowitch and J. Rémy, editors, *Proceedings 3rd International Workshop on Conditional Term Rewriting Systems, Pont-à-Mousson (France)*, volume 656 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 1992.

[Mos89] P. D. Mosses. Unified algebras and institutions. In *Proceedings 4th IEEE Symposium on Logic in Computer Science, Pacific Grove*, pages 304–312, 1989.

[Obe62] A. Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Math. Annalen*, 145(1):297–333, 1962.

[Sco77] D. Scott. Identity and existence in intuitionistic logic. In M. P. Fourman and C. J. Mulvey, editors, *Applications of Sheaves*, volume 753 of *Lecture Notes in Mathematics*, pages 660–696. Springer-Verlag, 1977.

[Smo89] G. Smolka. *Logic Programming over Polymorphically Order-Sorted Types.* PhD thesis, FB Informatik, Universität Kaiserslautern, Germany, 1989.

[SNGM89] G. Smolka, W. Nutt, J. A. Goguen, and J. Meseguer. Order-sorted equational computation. In H. Aït-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures, Volume 2: Rewriting Techniques*, pages 297–367. Academic Press, 1989.

[SS87] M. Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations.* PhD thesis, Universität Kaiserslautern (Germany), 1987.

[Wal92] U. Waldmann. Semantics of order-sorted specifications. *Theoretical Computer Science*, 94(1):1–33, 1992.

[WD89] P. Watson and J. Dick. Least sorts in order-sorted term rewriting. Technical report, Royal Holloway and Bedford New College, University of London, 1989.

[Wer93] A. Werner. A semantic approach to order-sorted rewriting. In C. Kirchner, editor, *Proceedings 5th Conference on Rewriting Techniques and Applications, Montreal (Canada)*, volume 690 of *Lecture Notes in Computer Science*, pages 47–61. Springer-Verlag, 1993.

[Wit92] L. With. Completeness and confluence of order-sorted term rewriting. In M. Rusinowitch and J.-L. Rémy, editors, *Proceedings 3rd International Workshop on Conditional Term Rewriting Systems, Pont-à-Mousson (France)*, number 656 in Lecture Notes in Computer Science, pages 393–407. Springer-Verlag, July 1992.

This article was processed using the LaTeX macro package with LLNCS style