Theoretical Computer Science 89 (1991) 3-32
Elsevier

Order-sorted completion: the many-sorted way

Harald Ganzinger*

Max-Planck-Institut für Informatik, Im Stadtwald, W-6600 Saarbrücken, Germany

Abstract

Ganzinger, H., Order-sorted completion: the many-sorted way, Theoretical Computer Science 89 (1991) 3-32.

Order-sorted specifications can be transformed into equivalent many-sorted ones by using injections to implement subsort relations. In this paper we improve previous results about the relation between order-sorted and many-sorted rewriting. We then apply techniques for the completion of many-sorted conditional equations to systems obtained from translating order-sorted conditional equations. Emphasis will be on ways to overcome some of the problems with non-sort-decreasing rules.

1. Introduction

1.1. Operational semantics for order-sorted specifications

Order-sorted equational logic has been introduced into algebraic specification and logic programming in order to provide for a more powerful type concept, allowing us to express partiality of functions, error handling and subtype inheritance. Order-sorted equational logic originated with [22] and, independently in the context of abstract data types, with [13]. The concept was further elaborated in [14, 12, 25, 20, 23] among others. A recent paper comparing the two main semantic variations—overloading vs. polymorphism—is [26].

In [10] an operational semantics for order-sorted specifications based on a translation scheme into many-sorted specifications is introduced. Auxiliary injection operators are added to implement the subsort relations. Therefore, standard concepts of many-sorted rewriting, completion, and theorem proving can be used in implementations of specification languages based on order-sorted logic.

In [11], a completion procedure specifically tailored to (unconditional) order-sorted equations and based on order-sorted rewriting is proposed as an alternative. The main motivation for this approach is that order-sorted rewriting can be more efficient than naive many-sorted rewriting with the translated rules [20].

The purpose of this paper is not to enter a discussion about the efficiency of rewrite relations—we believe that both approaches are of interest in their own

* This work has been partially supported by the ESPRIT-project PROSPECTRA, ref#390, at the University of Dortmund.

right—but to improve both previous approaches. There are two directions of improvement which this paper wants to contribute to.

One problem left open in the approaches of [10] and [11] is the handling of non-sort-decreasing rules. This problem had been overlooked in [10] as most of the results in this paper which relate order-sorted to many-sorted rewriting are only valid for sort-decreasing rules. Standard Knuth-Bendix completion when applied to the many-sorted translation of order-sorted equations must give up whenever a non-sort-decreasing rule is encountered. In such a case, the operationally awkward injectivity axiom for the injections has to be taken into consideration—a problem which standard Knuth-Bendix completion is not prepared to handle. For a related reason, the order-sorted completion procedure of [11] fails when a non-sort-decreasing rule is generated. Order-sorted replacement of equals by equals is not a complete proof method for order-sorted deduction in this case [25].

A second problem is to complete order-sorted specifications with *conditional* equations. Fortunately, the state-of-the-art in completion of many-sorted conditional equations, which originated with [17], has been substantially advanced during recent years, mainly by the work of Rusinowitch and Kounalis [21, 24] and by this author [8]. (Other related relevant work is described in [28] and [19]. Recently in [5] it is shown how to extend the concept of completion to the full first-order case.) Techniques which are required to make completion useful in practice are detailed in [9, 4, 3]. The method of proof orderings and proof transformations [1, 2] has been very helpful to prove completeness of these techniques. This advance in technology suggests that we can again pick up the translation idea of [10] and develop order-sorted completion the many-sorted way.

As one can immediately see, both problem areas are closely related. To handle non-sort-decreasing rules requires that we consider the effect on the equational theory of conditional equations such as the injectivity axioms for sort injections.

1.2. Summary of main results

With regard to the relation between order-sorted rewriting and many-sorted rewriting we improve the results in [10]. In particular we show that one step of order-sorted rewriting with a set of rules R is one step of many-sorted rewriting with the lowest parses $\lfloor R_S \rfloor$ of the sort specializations R_S of R modulo the axioms LP of the lowest parse. This result also holds in the presence of non-sort-decreasing rules. Moreover we show that for any sort-decreasing and convergent system of order-sorted *conditional* rewrite rules, $LP \cup \lfloor R_S \rfloor$ is an equivalent, convergent (many-sorted) rewrite system. (This result is not completely obvious as the critical pairs lemma is not true for conditional rewrite rules in general [7].) A corollary is the result in [10] which, by the way, is also only true for non-sort-decreasing rules, about the convergence of $\rightarrow_{\lfloor R_S \rfloor} \circ \xrightarrow{!}_{LP}$ in the many-sorted world.

We show that in many practical cases non-sort-decreasing rules can be replaced by sort-decreasing ones without changing the initial algebra. These replacements

are conditional rules with extra variables in the condition and in the right side. Fortunately they belong to the class of what we call *quasi-reductive rules.* Quasi-reductive rules are a generalization of reductive conditional rewrite rules and the associated rewrite process is similarly efficient.

We outline an unfailing completion procedure for conditional equations that can handle both non-reductive equations such as injectivity axioms and quasi-reductive equations as they are introduced during the replacement of non-sort-decreasing rules. It is an extension of the one which has been presented and proved correct in [8]. The correctness of the extensions (unfailing completion, quasi-reductive equations) is further addressed in [4] and proved in detail in [3]. Our techniques perform successfully on practical examples of order-sorted specifications. An implementation of the concepts as described in this paper is part of the CEC-system [15].

2. Basic notions and notations

We will only introduce the syntactic aspects of order-sorted logic and refer to [12] and [25] for the two main variants of the semantics for order-sorted specifications, and to [26] for a comparison of the two. In this paper, notions and notations mainly follow [25, 11].

Every *variable* x comes with a sort s^x which is a *sort symbol.* For every sort symbol there exist infinitely many variables having this sort.

A *subsort declaration* is an expression of form $s < s'$, where s and s' are sort symbols. A *function or operator declaration* has the form $f: s_1 \dots s_n \to s_0$, where n is the arity of f and s_i are sort symbols. An *order-sorted* signature, usually denoted Σ, is a set of sort symbols, subsort and function declarations. A *many-sorted signature,* usually denoted Σ^{ms}, is a particular case of an order-sorted signature with an empty set of subsort declarations. In a many-sorted signature we do not allow more than one declaration for any function symbol. By S^{ms} and Ω^{ms} we denote the set of sorts and operators, respectively, of a many-sorted signature Σ^{ms}.

The *subsort order* $s \leqslant s'$ is the least quasi-order on the sort symbols of Σ generated by the subsort declarations. Throughout this paper we restrict our attention to signatures for which $<$, defined as $< = \leqslant \setminus =$, is a partial order. $\leqslant$[1] extends to tuples of sort symbols of the same length by $(\underline{s}_1, \dots, \underline{s}_n) \leqslant (s_1, \dots, s_n)$, iff $\underline{s}_i \leqslant s_i$, for $1 \leqslant i \leqslant n$. We write $s \lhd s'$, if $s < s'$ and if there does not exist any s'' such that $s < s'' < s'$.

Given a set of variables X, a *Σ-term of sort* s in $\mathcal{T}_\Sigma(X)_s$ is either a variable x such that $s^x \leqslant s$, or has the form $f(t_1, \dots, t_n)$, where $f: s_1 \dots s_n \to s_0$ is a function declaration in Σ such that $s_0 \leqslant s$ and $t_i \in \mathcal{T}_\Sigma(X)_{s_i}$ is a term of sort s_i, for $1 \leqslant i \leqslant n$.

The sort of a many-sorted term t is uniquely determined and will be denoted s' in this paper. For both order-sorted and many-sorted terms t we use the notation $t: s$ to indicate that t is a term of sort s.

[1] If $<$ is a strict partial order, by $>$ we denote its inverse and by $\leqslant$ we denote $< \cup =$.

A *Σ-equation* is an ordered pair of Σ-terms u and v of the same sort written as $u \approx v$. A *conditional equation over Σ* is a formula of form $C \rightarrow u \approx v$, where $u \approx v$ is a Σ-equation and C is a finite conjunction of Σ-equations, written $u_1 \approx v_1, \ldots, u_n \approx v_n$, $n \geq 0$. If E is a set of (conditional) Σ-equations, by $E^=$ we denote the set

$$E^= = \{C \rightarrow u \approx v \mid C \rightarrow u \approx v \in E \text{ or } C \rightarrow v \approx u \in E\}.$$

For unconditional many-sorted equations E, rewriting (by applying the equations from left to right) is defined as usual. The one-step rewrite relation is denoted by $\rightarrow_E$. E-normalforms of terms t are denoted by $t\downarrow_E$.

An *order-sorted (equational) specification* consists of an order-sorted signature Σ and a set E of conditional Σ-equations.

If O is a syntactic Σ-object, i.e. a term or (conditional) equation, by $var(O)$ we denote the set of variables occurring in O. A *Σ-substitution* is a function σ from Σ-terms to Σ-terms such that (i) if u is a term of sort s, then $\sigma(u)$ is a term of sort s and (ii) $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$. We will mainly use the notation $u\sigma$ for $\sigma(u)$.

An order-sorted signature Σ is called *pre-regular*, if for any Σ-function symbol f and every string $s_1 \ldots s_n$ of Σ-sorts the set $\{\bar{s} \mid (f: \bar{s}_1 \ldots \bar{s}_n \rightarrow \bar{s}) \in \Sigma,\ s_1 \ldots s_n \leq \bar{s}_1 \ldots \bar{s}_n\}$ is either empty or has a minimum element. Σ is called *regular*, if for any string $s_1 \ldots s_n$ of Σ-sorts such that there exists a function declaration $(f: \bar{s}_1 \ldots \bar{s}_n \rightarrow \bar{s}) \in \Sigma$ with $s_1 \ldots s_n \leq \bar{s}_1 \ldots \bar{s}_n$, then there exists a least $(\underline{s}_1, \ldots, \underline{s}_n, \underline{s})$ such that $(f: \underline{s}_1 \ldots \underline{s}_n \rightarrow \underline{s}) \in \Sigma$ and $s_1 \ldots s_n \leq \underline{s}_1 \ldots \underline{s}_n$. Regularity of a signature implies pre-regularity.

Pre-regularity ensures the existence of initial algebras in the semantics of [25]. For the semantics of [12], regularity is a sufficient condition for the existence of initial algebras. Although the notion of order-sorted deduction which is used in this paper corresponds to the semantics of [12], pre-regularity will already be a sufficient condition for the syntactic properties on which our approach of order-sorted completion is based.

An order-sorted signature is called *coherent*, if each equivalence class of sorts under the equivalence closure $(\leq \cup \geq)^*$ of $\leq$ has a maximum.

From now on we will assume order-sorted signatures to be finite, pre-regular and coherent.

3. Translation of order-sorted specifications

3.1. Many-sorted representations of order-sorted terms

Definition 3.1. Let Σ be an order-sorted signature. Its *translation* into a corresponding many-sorted signature Σ^{ms} is defined as follows:

(1) the sorts in S^{ms} are the sorts in S,

(2) if $f: s_1 \ldots s_n \to s_0$ is an operator declaration in Σ, then $f_{s_1 \ldots s_n \to s_0}: s_1 \ldots s_n \to s_0 \in \Omega^{ms}$,

(3) if $s \lhd s'$ in Σ, then $\uparrow_s^{s'} \in \Omega^{ms}$. $\uparrow_s^{s'}$ is called a (basic) *injection.*

The translation disambiguates overloaded function symbols and introduces injections $\uparrow_s^{s'}$ to represent the inclusion of s in s'. Injections along chains of subsort relations can be represented by terms in elementary injections. As there may be more than one way of going from an s to an s' we have to order these paths if we want a unique representation. For that purpose we assume the partial subsort order $<$ to be extended to an arbitrary but fixed total order $\ll$ on S. The same notation is used for the *lexicographic* extension of $\ll$ to sequences of sorts. (This is different to the extension of $\leqslant$ to tuples which we have used to define the regularity properties. The latter was defined component-wise. Hence, $\leqslant$ is usually a proper subset of $\ll$.)

Now we define composite injections to proceed along minimal paths in the subsort graph.

Definition 3.2. Let $s \leqslant s'$. Furthermore let $s_n s_{n-1} \ldots s_0$, $n \geqslant 0$, be a sequence of sorts minimal wrt $\ll$ such that $s_0 = s$, $s_n = s'$, and $s_i \lhd s_{i+1}$, for $0 \leqslant i < n$. Then

$$\uparrow_{s_{n-1}}^{s_n} \circ \uparrow_{s_{n-2}}^{s_{n-1}} \circ \cdots \circ \uparrow_{s_0}^{s_1}$$

is called the *minimal composite injection* from s to s', denoted as $\uparrow_s^{s'}$. (If $s = s'$, $\uparrow_s^{s'}$ is empty, denoting the identity. In this case, $t\uparrow_s^{s'} = t$.)

Note that if $s \lhd s'$, the basic injection is at the same time the minimal injection from s into s'. Where no confusion can arise we will write $t\uparrow^{s'}$ for $t\uparrow_s^{s'}$.

We can now go on and define mappings between order-sorted terms and their many-sorted representations as terms over Σ^{ms}. Any many-sorted term t in Σ^{ms} represents one unique order-sorted term $\lceil t \rceil$ which is obtained by deleting injections and by collapsing the disambiguated operator symbols into the original overloaded symbol.

Definition 3.3. The *type erasing function* $\lceil \_ \rceil : T_{\Sigma^{ms}}(X) \to \mathcal{T}_\Sigma(X)$ is inductively defined as follows:

(1) if x is a variable, then $\lceil x \rceil = x$,

(2) $\lceil f_{s_1 \ldots s_n \to s_0}(t_1, \ldots, t_n) \rceil = f(\lceil t_1 \rceil, \ldots, \lceil t_n \rceil)$, for non-injections $f_{s_1 \ldots s_n \to s_0} \in \Omega^{ms}$,

(3) $\lceil t\uparrow_s^{s'} \rceil = \lceil t \rceil$, for injections $\uparrow_s^{s'} \in \Omega^{ms}$.

In the reverse direction, $\lfloor \_ \rfloor$ will compute the lowest parse of an order-sorted term.

Definition 3.4. The *lowest parse* $\lfloor \_ \rfloor : \mathcal{T}_\Sigma(X) \to T_{\Sigma^{ms}}(X)$ is inductively defined as follows:

(1) $\lfloor x \rfloor = x$, for variables $x : s \in X$,

(2) Let t_i be given and let $\lfloor t_i \rfloor \in T_{\Sigma^{ms}}(X)_{s_i'}$, $1 \leqslant i \leqslant n$. Then

$$\lfloor f(t_1, \ldots, t_n) \rfloor = f_{s_1 \ldots s_n \to s_0}(\lfloor t_1 \rfloor \uparrow_{s_1'}^{s_1}, \ldots, \lfloor t_n n \rfloor \uparrow_{s_n'}^{s_n}),$$

where $f: s_1 \ldots s_n \to s_0$ is the operator declaration in Σ for which $s_0 s_1 \ldots s_n$ is minimal with respect to $\lessdot$ such that $s_i' \leqslant s_i$, $1 \leqslant i \leqslant n$.

(3) For Σ-substitutions σ the lowest parse $\lfloor \sigma \rfloor$ is then defined to be the (many-sorted) substitution $\{x \mapsto \lfloor x\sigma \rfloor \uparrow^{s^x} \mid x \in X\}$.

As we are putting the codomain s_0 of f at the beginning of the sort sequence $s_0 s_1 \ldots s_n$ when looking for a minimal declaration for f, $\lfloor t \rfloor$ will always have a lowest possible sort with respect to $\lessdot$. Due to the pre-regularity of the signature, s_0 will also be minimal with respect to the subsort order. We observe the following properties.

Proposition 3.5. $\lceil \lfloor t \rfloor \rceil = t$ *and* $s^{\lfloor t \rfloor} \leqslant s^t$.

3.2. Computation of minimal parses by rewriting

The lowest parse of an order-sorted term is a corresponding many-sorted term of lowest possible sort and unique for pre-regular signatures. On the other hand, different many-sorted terms can represent the same order-sorted term via $\lceil \_ \rceil$. In this section we describe a canonical set of rewrite rules over Σ^{ms} which, for any given many-sorted term, computes the lowest parse of the order-sorted term $\lceil t \rceil$ it represents.

The set of rules consists of rules for computing the minimal path (with respect to $\lessdot$) between any two sorts $s < s'$ and of rules which represent the inheritance axioms for overloaded function symbols on the intersection of their domains.

Axioms (CI) for composite injections

$$s\uparrow_s^{s'}\uparrow_{s'}^{s''} \approx x\uparrow_s^{s''}, \quad \text{for } s < s' < s'', \text{ if } \uparrow_s^{s''} \neq \uparrow_{s'}^{s''} \circ \uparrow_s^{s'}.$$

Axioms (INH) for inheritance

$$f_{s_1 \ldots s_n \to s_0}(x_1\uparrow_{\tilde{s}_1}^{s_1}, \ldots, x_n\uparrow_{\tilde{s}_n}^{s_n}) \approx f_{s_1' \ldots s_n' \to s_0'}(x_1\uparrow_{\tilde{s}_1}^{s_1'}, \ldots, x_n\uparrow_{\tilde{s}_n}^{s_n'})\uparrow_{s_0'}^{s_0},$$

if $f: s_1 \ldots s_n \to s_0$ and $f: s_1' \ldots s_n' \to s_0'$ are operator declarations, $s_0' \leqslant s_0$, $s_0' s_1' \ldots s_n' \lessdot s_0 s_1 \ldots s_n$, and $\tilde{s}_i$ are maximal sorts (with respect to $<$) such that $\tilde{s}_i \leqslant s_i$ and $\tilde{s}_i \leqslant s_i'$.[2]

We will now prove that the equation system $LP = CI \cup INH$, oriented from left to right, forms a canonical system of rewrite rules. First we will define a precedence on Σ^{ms}-operators such that the induced recursive path ordering proves the termination of the system.

[2] Formally it would make no difference if one simply introduced an equation for any $\tilde{s}_i$ for which $\tilde{s}_i \leqslant s_i$ and $\tilde{s}_i \leqslant s_i'$. However, (INH)-axioms for maximal $\tilde{s}_i$ subsume (INH)-axioms for non-maximal ones.

Definition 3.6. By $>_I$ we denote the following partial order on Ω^{ms}:

(1) $\uparrow_{s_1}^{s_2} >_I \uparrow_{s_1'}^{s_2'}$, iff $s_2' < s_2$ or if $s_2' = s_2$ and $s_1' < s_1$, for injections $\uparrow_{s_1}^{s_2}$ and $\uparrow_{s_1'}^{s_2'}$.

(2) $f_{s_1 \ldots s_n \to s_0} >_I \uparrow_s^{s'}$, for any order-sorted operator f and any injection $\uparrow_s^{s'}$.

(3) $f_{s_1 \ldots s_n \to s_0} >_I f_{s_1' \ldots s_n' \to s_0'}$, iff $s_0' s_1' \ldots s_n' \lessdot s_0 s_1 \ldots s_n$, for any two declarations $f : s_1' \ldots s_n' \to s_0'$ and $f : s_1 \ldots s_n \to s_0$ of the same order-sorted operator symbol f.

By $>_I$ we also denote the recursive path ordering on $T_{\Sigma^{ms}}(X)$ induced by $>_I$.

Proposition 3.7. *For any equation $L \approx R$ in LP, $L >_I R$.*

The confluence of the system will be proved using the following proposition.

Proposition 3.8. *If*

$$J = \uparrow_{s_{n-1}}^{s_n} \circ \uparrow_{s_{n-2}}^{s_{n-1}} \circ \cdots \circ \uparrow_{s_0}^{s_1}$$

is some composite injection from s_0 to s_n, $n \geq 1$, then $J(x) \to_{CI}^* x\uparrow_{s_0}^{s_n}$.

Proof. Induction over n: If $n = 1$, $J(x) = x\uparrow_{s_0}^{s_n}$. If $n > 1$, $J(x) = J'(x)\uparrow_{s_{n-1}}^{s_n}$. By induction hypothesis, $J'(x) \to_{CI}^* x\uparrow_{s_0}^{s_{n-1}}$. Now, either

$$x\uparrow_{s_0}^{s_{n-1}}\uparrow_{s_{n-1}}^{s_n} = x\uparrow_{s_0}^{s_n} \quad \text{or} \quad x\uparrow_{s_0}^{s_{n-1}}\uparrow_{s_{n-1}}^{s_n} \approx x\uparrow_{s_0}^{s_n}$$

is a rule in (CI). □

As a consequence we have $J_1(x) \downarrow_{CI} J_2(x)$, for any two composite injections J_1 and J_2 from s to s'.

Lemma 3.9. *For any two Σ^{ms}-terms $t : s$ and $t' : s'$ such that $s \leq s_0$, and $s' \leq s_0$, we have $\lceil t \rceil = \lceil t' \rceil$, iff $t\uparrow^{s_0} =_{LP} t'\uparrow^{s_0}$ such that the $=_{LP}$-proof only involves intermediate terms smaller than $t\uparrow^{s_0}$ or $t'\uparrow^{s_0}$ with respect to $>_I$.*

Proof. The "$\Leftarrow$"-case is trivial as $\lceil N \rceil = \lceil M \rceil$, for any equation $N = M$ in *LP*.

We prove the "$\Rightarrow$"-case by induction over the structure of t.

Let $t = x : s$ be a variable. Then, $t' = J_{s,s'}(x)$ with some composite injection $J_{s,s'}$ from s to s'. Hence from Proposition 3.8,

$$t'\uparrow^{s_0} = J_{s,s'}(x)\uparrow^{s_0} \to_{CI}^* t\uparrow^{s_0}.$$

Since this proof applies *CI*-rules from left to right, the requirement about the complexity of the intermediate terms is satisfied.

If $t = t_1\uparrow^s$, then $\lceil t_1 \rceil = \lceil t' \rceil$. In this case, using Proposition 3.8,

$$t\uparrow^{s_0} = t_1\uparrow^s\uparrow^{s_0} \to_{CI}^* t_1\uparrow^{s_0} =_{LP} t'\uparrow^{s_0}.$$

Hereby, the last equivalence is the induction hypothesis, involving only intermediate terms smaller than $t_1\uparrow^{s_0}$ or $t'\uparrow^{s_0}$. Altogether, a proof of the required form has been constructed.

Now let $t = f_{s_1 \ldots s_n \to s}(t_1, \ldots, t_n)$. Then, $t' = f_{\bar{s}_1 \ldots \bar{s}_n \to \bar{s}}(\bar{t}_1, \ldots, \bar{t}_n)\uparrow^{s'}$ and $\lceil t_i \rceil = \lceil \bar{t}_i \rceil$, $1 \leq i \leq n$. Thus, $\lfloor \lceil t_i \rceil \rfloor = \lfloor \lceil \bar{t}_i \rceil \rfloor =: \tilde{t}_i$. From Proposition 3.5 we obtain $s^{\tilde{t}_i} =: \tilde{s}_i \leq s_i$ and $\tilde{s}_i \leq \bar{s}_i$. The pre-regularity of Σ implies the existence of an operator declaration $f : \underline{s}_1 \ldots \underline{s}_n \to \underline{s}$ with minimal $\underline{s}$ such that $\tilde{s}_i \leq \underline{s}_i$, $\underline{s} \leq s$ and $\underline{s} \leq \bar{s}$. Furthermore, we may choose f such that $\underline{s}\underline{s}_1 \ldots \underline{s}_n$ is minimal with respect to $\prec$. From the *INH*-equations we immediately see that

$$f_{s_1 \ldots s_n \to s}(\tilde{t}_1\uparrow^{s_1}, \ldots, \tilde{t}_n\uparrow^{s_n}) \to^*_{LP} f_{\underline{s}_1 \ldots \underline{s}_n \to \underline{s}}(\tilde{t}_1\uparrow^{\underline{s}_1}, \ldots, \tilde{t}_n\uparrow^{\underline{s}_n})\uparrow^{s}$$

and

$$f_{\bar{s}_1 \ldots \bar{s}_n \to \bar{s}}(\tilde{t}_1\uparrow^{\bar{s}_1}, \ldots, \tilde{t}_n\uparrow^{\bar{s}_n}) \to^*_{LP} f_{\underline{s}_1 \ldots \underline{s}_n \to \underline{s}}(\tilde{t}_1\uparrow^{\underline{s}_1}, \ldots, \tilde{t}_n\uparrow^{\underline{s}_n})\uparrow^{\bar{s}}.$$

From the induction hypothesis we infer

$$t_i =_{LP} \tilde{t}_i\uparrow^{s_i}$$

and

$$\bar{t}_i =_{LP} \tilde{t}_i\uparrow^{\bar{s}_i}$$

involving only terms smaller than $t\uparrow^{s_0}$ and $t'\uparrow^{s_0}$. Finally,

$$x\uparrow^{\bar{s}}\uparrow^{s'}\uparrow^{s_0} \to^*_{CI} x\uparrow^{s_0} \leftarrow^*_{CI} x\uparrow^{s}\uparrow^{s_0}.$$

Altogether we have constructed the following proof

$$\begin{aligned}
t\uparrow^{s_0} &= f_{s_1 \ldots s_n \to s}(t_1, \ldots, t_n)\uparrow^{s_0} \\
&=_{LP} f_{s_1 \ldots s_n \to s}(\tilde{t}_1\uparrow^{s_1}, \ldots, \tilde{t}_n\uparrow^{s^n})\uparrow^{s_0} \\
&\to^*_{LP} f_{\underline{s}_1 \ldots \underline{s}_n \to \underline{s}}(\tilde{t}_1\uparrow^{\underline{s}_1}, \ldots, \tilde{t}_n\uparrow^{\underline{s}_n})\uparrow^{s}\uparrow^{s_0} \\
&\to^*_{CI} f_{\underline{s}_1 \ldots \underline{s}_n \to \underline{s}}(\tilde{t}_1\uparrow^{\underline{s}_1}, \ldots, \tilde{t}_n\uparrow^{\underline{s}_n})\uparrow^{s_0} \\
&\leftarrow^*_{CI} f_{\underline{s}_1 \ldots \underline{s}_n \to \underline{s}}(\tilde{t}_1\uparrow^{\underline{s}_1}, \ldots, \tilde{t}_n\uparrow^{\underline{s}_n})\uparrow^{\bar{s}}\uparrow^{s'}\uparrow^{s_0} \\
&\leftarrow^*_{LP} f_{\bar{s}_1 \ldots \bar{s}_n \to \bar{s}}(\tilde{t}_1\uparrow^{\bar{s}_1}, \ldots, \tilde{t}_n\uparrow^{\bar{s}_n})\uparrow^{s'}\uparrow^{s_0} \\
&=_{LP} f_{\bar{s}_1 \ldots \bar{s}_n \to \bar{s}}(\bar{t}_1, \ldots, \bar{t}_n)\uparrow^{s'}\uparrow^{s_0} \\
&= t'\uparrow^{s_0},
\end{aligned}$$

in which the intermediate terms are all smaller than $t\uparrow^{s_0}$ or $t'\uparrow^{s_0}$. □

An immediate consequence is $\lfloor t \rfloor\uparrow^{s}\lfloor \sigma \rfloor =_{LP} \lfloor t\sigma \rfloor\uparrow^{s}$, for terms $t : s$ and substitutions σ.

Proposition 3.10. *The set of rules LP is locally confluent, hence confluent by Proposition* 3.7.

Proof. As both sides of any critical pair are equal under $\lceil \_ \rceil$, we may apply Lemma 3.9 and conclude the existence of a "subconnected" proof of $s =_{LP} t$, i.e. a proof which involves only terms smaller than or equal to s or t. Therefore, the local confluence follows from the extended critical pairs lemma of [27]. □

From Propositions 3.7 and 3.10 it follows that LP is convergent. We will now prove that the LP-normalform of a term t represents the lowest parse of the corresponding order-sorted term $\lceil t \rceil$. More precisely,

Lemma 3.11. *Let* $t \in T_{\Sigma^{ms}}(X)_s$ *and* $s^{\lfloor \lceil t \rceil \rfloor} =: s' \leqslant s'' \leqslant s$.

(i) $\lfloor \lceil t \rceil \rfloor \uparrow^{s''}$ *is irreducible under* $\rightarrow_{LP}$.

(ii) $t \geqslant_I \lfloor \lceil t \rceil \rfloor \uparrow^{s''}$.

In particular, from (i), *Lemma* 3.9 *and Propositions* 3.7 *and* 3.10, *we have that* $t\downarrow_{LP} = \lfloor \lceil t \rceil \rfloor \uparrow^{s}$.

Proof. (i) The irreducibility of $\lfloor \lceil t \rceil \rfloor \uparrow^{s''}$ follows from the fact that $\lfloor \lceil t \rceil \rfloor$ contains only minimal instances of overloaded operators which cannot occur on the left side of INH-rules. Moreover, if a (composite) injection from s_1 to s_2 occurs in $\lfloor \lceil t \rceil \rfloor \uparrow^{s''}$, it is the minimal one $\uparrow_{s_1}^{s_2}$ which is irreducible under CI-rules.

(ii) From (i), Lemma 3.9 and Propositions 3.7 and 3.10, we have that $t \rightarrow_{LP}^{*} \lfloor \lceil t \rceil \rfloor \uparrow^{s}$. Hence, $t \geqslant_I \lfloor \lceil t \rceil \rfloor \uparrow^{s}$. If $s'' = s$, we are done. Otherwise, $s'' < s$ which implies that any elementary injection in $\uparrow_{s''}^{s}$ is smaller in precedence than the topmost elementary injection in $\uparrow_{s'}^{s}$. As a consequence, $\lfloor \lceil t \rceil \rfloor \uparrow^{s} >_I \lfloor \lceil t \rceil \rfloor \uparrow^{s''}$. □

Altogether we have shown that two terms $t_1, t_2 \in T_{\Sigma^{ms}}(X)$ are representations of the same order-sorted terms, iff they are equivalent under LP. Moreover, the equivalence can be decided by rewriting the appropriately injected terms to their $\rightarrow_{LP}$-normalforms.

3.3. Order-sorted deduction and rewriting

The notion of order-sorted deduction here corresponds to the variant of order-sorted logic in [12]. To avoid the problems related to empty types, we assume that all sorts are inhabited, i.e. have ground terms of that sort. We assume X to be a fixed set of variables that has infinitely many variables for every sort. Order-sorted deduction can be described by the following set of inference rules which we have adopted from [11].

Definition 3.12 (*order-sorted deduction*). Let E be a set of order-sorted equations over Σ.

Reflexivity

$$E \vdash t \approx t, \quad \text{for any } t \in \mathcal{T}_{\Sigma}(X).$$

Symmetry

$$\frac{E \vdash t \approx t'}{E \vdash t' \approx t}.$$

Transitivity

$$\frac{E \vdash t \approx t' \qquad E \vdash t' \approx t''}{E \vdash t \approx t''}.$$

Congruence

$$\frac{E \vdash x\theta \approx x\theta', \quad \forall x \in var(t)}{E \vdash t\theta \approx t\theta'},$$

for θ, θ' substitutions and $t \in \mathcal{T}_\Sigma(X)$.

Substitutivity

$$\frac{E \vdash t_i\theta \approx t_i'\theta, \quad 1 \leq i \leq n}{E \vdash t\theta \approx t'\theta},$$

for $t_1 \approx t_1', \ldots, t_n \approx t_n' \rightarrow t \approx t' \in E$ and θ a substitution.

Let, in the following,

$$=_E = \bigcup_{n \in N} =_E^n,$$

where $=_E^0 = \emptyset$ and $t =_E^n t'$, iff $t =_E^{n-1} t'$ or if there exist u_j, u_j' such that $u_j =_E^{n-1} u_j'$ and $t \approx t'$ can be derived from $u_j \approx u_j'$ using one of the above inference rules. Clearly, $E \vdash t \approx t'$ iff $t =_E t'$. Observe also that order-sorted deduction coincides with many-sorted deduction in case Σ is actually a many-sorted signature.

We will now extend our notion of lowest parses $\lfloor \_ \rfloor$ to unconditional equations in the obvious way. Let $t_1 \approx t_2$ be an order-sorted equation, and assume that $s_i = s^{\lfloor t_i \rfloor}$. Then

$$\lfloor t_1 \approx t_2 \rfloor = \lfloor t_1 \rfloor \uparrow^s \approx \lfloor t_2 \rfloor \uparrow^s,$$

where s is some minimal sort such that $t_1, t_2 \in \mathcal{T}_\Sigma(X)_s$. (There may be more than one choice for s. This, however, is irrelevant in our context.) In particular, if $s_2 \leq s_1$, the left side of $\lfloor t_1 \approx t_2 \rfloor$ will not have an injection as the top symbol.

Now let

$$E^\# = CI \cup INH \cup IN \cup \lfloor E \rfloor,$$

where

$$\lfloor E \rfloor = \{\ldots \lfloor t_i \approx t_i' \rfloor \ldots \rightarrow \lfloor t \approx t' \rfloor \mid \ldots t_i \approx t_i' \ldots \rightarrow t \approx t' \in E\}$$

are the minimal parses of the equations in E and where

$$IN = \{x\uparrow^{s'} \approx y\uparrow^{s'} \rightarrow x \approx y \mid \uparrow^{s'} \in \Omega^{ms}\}$$

is the set of injectivity axioms for the basic injections in Σ^{ms}.

The following is the proof-theoretic equivalent of the satisfaction theorem in [10].

Theorem 3.13. *For* $t_1, t_2 \in T_{\Sigma^{ms}}(X)_s$, $t_1 =_{E^\#} t_2$, *iff* $\lceil t_1 \rceil =_E \lceil t_2 \rceil$.

Proof. The "$\Rightarrow$"-case is trivial as the images of $E^\#$-axioms under $\lceil \_ \rceil$ are either trivial equations or E-axioms.

The "$\Leftarrow$"-case proceeds by induction over n in $=_E^n$. The base case $n = 0$ is trivial. The induction step is structured according to the inference rules for order-sorted deduction.

Reflexivity: Suppose $t =_E^{n+1} t$ by reflexivity. As $t = \lceil t_1 \rceil = \lceil t_2 \rceil$, we infer from Lemma 3.9 that $t_1 =_{LP} t_2$.

Symmetry: This case follows trivially from the symmetry of $=_{E^\#}$.

Transitivity: Suppose, $\lceil t_1 \rceil =_E^n t'$ and $t' =_E^n \lceil t_2 \rceil$. This case is proved by first observing some basic relationships between the involved sorts. There exist sorts s_1 and s_2 such that $\lceil t_1 \rceil, t' \in \mathcal{T}_\Sigma(X)_{s_1}$ and $\lceil t_2 \rceil, t' \in \mathcal{T}_\Sigma(X)_{s_2}$. Furthermore let $\underline{s}_1 = s^{\lfloor \lceil t_1 \rceil \rfloor}$, $\underline{s}_2 = s^{\lfloor \lceil t_2 \rceil \rfloor}$, and $s' = s^{\lfloor t' \rfloor}$. Now let $\bar{s}$ be the maximal element of the connected component

$$s \geqslant \underline{s}_1 \leqslant s_1 \geqslant s' \leqslant s_2 \geqslant \underline{s}_2 \leqslant s.$$

The induction hypothesis provides

$$t_1 \uparrow^{\bar{s}} =_{E^\#} \lfloor t' \rfloor \uparrow^{\bar{s}} =_{E^\#} t_2 \uparrow^{\bar{s}},$$

as $\lceil t_1 \uparrow^{\bar{s}} \rceil = \lceil t_1 \rceil$, $\lceil \lfloor t' \rfloor \uparrow^{\bar{s}} \rceil = t'$, and $\lceil t_2 \uparrow^{\bar{s}} \rceil = \lceil t_2 \rceil$. Thus, $t_1 \uparrow^{\bar{s}} =_{E^\#} t_2 \uparrow^{\bar{s}}$, yielding $t_1 =_{E^\#} t_2$ by the injectivity axioms for the injections.

Congruence: Let $\lceil t_1 \rceil = t\theta_1$, $\lceil t_2 \rceil = t\theta_2$, and $x\theta_1 =_E^n x\theta_2$, for any $x \in var(t)$. Define, for $i = 1, 2$, $\sigma_i = \lfloor \theta_i \rfloor$. Then, $\lceil \lfloor t \rfloor \sigma_i \rceil = \lceil t_i \rceil$. Hence, $t_i =_{LP} \lfloor t \rfloor \sigma_i \uparrow^s$. $x\sigma_1 =_{E^\#} x\sigma_2$ is the induction hypothesis. This gives us $t_1 =_{E^\#} t_2$ from the congruence properties of $=_{E^\#}$.

Substitutivity: Suppose, $\ldots v_i \approx v_i' \ldots \to u \approx u' \in E$, $v_i\theta =_E^n v_i'\theta$, and $u\theta = \lceil t_1 \rceil$, $u'\theta = \lceil t_2 \rceil$. Again let $\sigma = \lfloor \theta \rfloor$. From the construction of $\lfloor E \rfloor$ there exists an equation $\ldots b_i \approx b_i' \ldots \to a \approx a' \in \lfloor E \rfloor$ such that $\lceil b_i\theta \rceil = v_i\theta$, $\lceil b_i'\sigma \rceil = v_i'\theta$, $\lceil a\sigma \rceil = u\theta$ and $\lceil a'\sigma \rceil = u'\theta$. Hence, $a\sigma =_E a'\sigma$, using this equation and the induction hypothesis. As $\lceil a\sigma \rceil = \lceil t_1 \rceil$ and $\lceil a'\sigma \rceil = \lceil t_2 \rceil$, the remaining problem is to coerce the sorts. Let $s^{a\sigma} = s^{a'\sigma} = s'$ and $\underline{s} = s^{\lfloor \lceil t_1 \rceil \rfloor}$. Then, $\underline{s} \leqslant s$, s', hence there exists $\bar{s}$ such that s, $s' \leqslant \bar{s}$. Therefore,

$$t_1 \uparrow^{\bar{s}} =_{LP} a\sigma \uparrow^{\bar{s}} =_{E^\#} a'\sigma \uparrow^{\bar{s}} =_{LP} t_2 \uparrow^{\bar{s}}.$$

Hence, by the injectivity axioms, $t_1 =_{E^\#} t_2$. □

This theorem proves the equivalence of order-sorted deduction in E with standard many-sorted equational logic in $E^\#$.

We now go on and compare order-sorted rewriting to many-sorted rewriting. As an order-sorted (conditional) rewrite rule we admit any order-sorted conditional equation $C \to l \approx r$.[3] We call an equation a rule whenever we want to emphasize its use from left to right in replacements of equals by equals. Formally, rules and equations are the same in this paper.

[3] One often finds additional restrictions about the variables such as $(var(C) \cup var(r)) \subset var(l)$. These conditions matter only if one is concerned with the decidability of the rewrite relation.

Definition 3.14. Let R be a set of order-sorted rules and A a set of order-sorted equations. A term $t \in \mathcal{T}_\Sigma(X)$ *rewrites* to t' *modulo* A with a rewrite rule $\rho = C \to l \approx r$ in R, which is denoted $t \to_{R/A} t'$, whenever

(1) $t =_A w$, σ is a substitution such that $w/o = l\sigma$, $w' = s[r\sigma]_o$, and $w' =_A t'$,

(2) there is a sort s such that, for x a variable of sort s, $w[x]_o$ is a well-formed term and $l\sigma$, $r\sigma \in \mathcal{T}_\Sigma(X)_s$,

(3) for any $u \approx v \in C$ there exist terms u' and v' such that $u\sigma \to^*_{R/A} u'$, $v\sigma \to^*_{R/A} v'$, and $u' =_A v'$, with $\to^*_{R/A}$ the reflexive and transitive closure of $\to_{R/A}$. In this case we also write $u\sigma \downarrow_{R/A} v\sigma$.

The least fixpoint of this recursion defines $\to_{R/A}$. That is, $\to_{R/A} = \bigcup \to_{R/A,n}$, where $\to_{R/A,0} = \emptyset$ and where $\to_{R/A,n+1}$ is the set of one-step rewritings in which the rewrite proofs for the conditions can be carried out in $\bigcup_{j \leq n} \to_{R/A,j}$. Where A is empty, we will simply write $\to_R$ and $\to^*_R$ for $\to_{R/A}$ and $\to^*_{R/A}$, respectively. In this case, $\to_R$ is just ordinary conditional rewriting. For many-sorted rewriting, which is just order-sorted rewriting for many-sorted signatures, the second condition of the previous definition becomes trivial.

Order-sorted rewriting with R corresponds to many-sorted rewriting modulo LP, using as rules the lowest parses of all *specializations* of R. To formally introduce the notion of specialization it is useful to define the notion of a sort assignment. A sort assignment is a map $\alpha : \bar{X} \to S$, where $\bar{X}$ is the set of names of variables in X. Hence, a sorted set of variables is a pair $(\bar{X}, \alpha)$, denoted X_α. Sort assignments inherit the subsort ordering such that $\alpha \leq \alpha'$, iff $\alpha(x) \leq \alpha'(x)$, for any $x \in \bar{X}$. A specialization is a substitution $\rho : X_\alpha \to X_{\alpha'}$, where $\alpha' \leq \alpha$, sending $x : \alpha(x)$ to $x : \alpha'(x)$. To *specialize* an order-sorted term of formula ϕ means to apply a specialization to ϕ. If Φ is a set of order-sorted terms of formulas, by Φ_S we denote the set of all specializations of terms or formulas in Φ.

Theorem 3.15. *For any* $n \geq 0$, $u \to_{\lfloor R_S \rfloor / LP,n} v$ *iff* $\lceil u \rceil \to_{R,n} \lceil v \rceil$; *hence* $u \to_{\lfloor R_S \rfloor / LP} v$ *iff* $\lceil u \rceil \to_R \lceil v \rceil$.

Proof. The "only if" case is obvious. For the converse, the proof will be by induction over n. The base case $n = 0$ is trivial. Suppose now that $\lceil u \rceil \to_{R,n+1} \lceil v \rceil$. Let $X = (\bar{X}, \alpha)$. According to the definition of order-sorted rewriting there exists a term N and an occurrence o in $\lceil u \rceil$ such that $\lceil u \rceil = N[l\sigma]_o$, $\lceil v \rceil = N[r\sigma]_o$, $N = \lceil u \rceil[x]_o$, where x is a variable of sort s such that $l\sigma$, $r\sigma \in \mathcal{T}_\Sigma(X)_s$, and where $T = l_1 \approx r_1, \ldots, l_n \approx r_n \to l \approx r \in R$ is the rule that is applied in this rewrite step. As $l\sigma$, $r\sigma$ both have sort s, there exists a specialization $\rho : (\bar{X}, \alpha) \to (\bar{X}, \alpha')$ of T such that $l\rho$ and $r\rho$ are terms of sort s, and such that, for any x in X, $\lfloor x\sigma \rfloor$ has the form $\lfloor x\sigma \rfloor = t_x \uparrow^{\alpha(x)} =_{LP} t_x \uparrow^{\alpha'(x)} \uparrow^{\alpha(x)}$. Let τ be the substitution that maps any $x : \alpha'(x)$ to $t_x \uparrow^{\alpha'(x)}$. The substituted condition equations of T are satisfied, i.e. $l_i\sigma \to^*_{R,n} t_i$ and $r_i\sigma \to^*_{R,n} t_i$. From the induction hypthesis we infer that

$$\lfloor l_i\rho \rfloor \uparrow^{s_i} \tau \downarrow_{\lfloor R_S \rfloor / LP,n} \lfloor r_i\rho \rfloor \uparrow^{s_i} \tau,$$

if $\lfloor T\rho \rfloor$ has the form $\lfloor l_1\rho \rfloor\uparrow^{s_1} \approx \lfloor r_1\rho \rfloor\uparrow^{s_1}, \ldots, \lfloor l_n\rho \rfloor\uparrow^{s_n} \approx \lfloor r_n\rho \rfloor\uparrow^{s_n} \to \lfloor l\rho \rfloor\uparrow^{s} \approx \lfloor r\rho \rfloor\uparrow^{s}$.

Obviously, $s^{\lfloor l\sigma \rfloor} \leqslant s^{\lfloor l\rho \rfloor} \leqslant s$ and $s^{\lfloor r\sigma \rfloor} \leqslant s^{\lfloor r\rho \rfloor} \leqslant s$. According to Lemma 3.9 we obtain the LP-congruences

$$\begin{aligned}
\lfloor l\sigma \rfloor\uparrow^{s} &=_{LP} \lfloor l\sigma \rfloor\uparrow^{s^{\lfloor l\rho \rfloor}}\uparrow^{s} \\
&=_{LP} \lfloor l\rho \rfloor\tau\uparrow^{s} \\
&= \lfloor l\rho \rfloor\uparrow^{s}\tau \\
&\to_{\lfloor R_S \rfloor/LP,n+1} \lfloor r\rho \rfloor\uparrow^{s}\tau \\
&= \lfloor r\rho \rfloor\tau\uparrow^{s} \\
&=_{LP} \lfloor r\sigma \rfloor\uparrow^{s^{\lfloor r\rho \rfloor}}\uparrow^{s} \\
&=_{LP} \lfloor r\sigma \rfloor\uparrow^{s}.
\end{aligned}$$

If we now denote by o' that occurrence of x in $\lfloor N \rfloor$ which corresponds to the occurrence o in N, we have shown that

$$u' = \lfloor N \rfloor[\lfloor l\sigma \rfloor\uparrow^{s}]_{o'} \to_{\lfloor R_S \rfloor/LP,n+1} \lfloor N \rfloor[\lfloor r\sigma \rfloor\uparrow^{s}]_{o'} = v'.$$

Then, for an appropriate injection J, $u =_{LP} J(u') \to_{\lfloor R_S \rfloor/LP,n+1} J(v') =_{LP} v$, from which $u \to_{\lfloor R_S \rfloor/LP,n+1} v$ is inferred. □

This theorem proves that order-sorted rewriting is equivalent to rewriting the LP-equivalence classes of the many-sorted representations of terms, using the lowest parses of the specializations of the order-sorted rules as rewrite rules. If $\to_R$ is convergent, $\lfloor R_S \rfloor/LP$ is also convergent. (A rewrite system is called *convergent* if it is confluent and terminating.) Convergence of R is by itself not sufficient to ensure convergence of the somewhat more practical rewrite system $\lfloor R_S \rfloor \cup LP$. Suppose we have subsort relations $s < s_1 < s'$, $s < s_2 < s'$, constants $a : s$, $b : s_1$, and $a \approx b$ as the only rewrite rule in R. If $s_2 \not< s_1$, CI consists of $x\uparrow^{s_1}\uparrow^{s'} \approx x\uparrow^{s_2}\uparrow^{s'}$. Therefore,

$$a\uparrow^{s_2}\uparrow^{s'} \leftarrow_{\lfloor R \rfloor \cup LP} a\uparrow^{s_1}\uparrow^{s'} \to_{\lfloor R \rfloor \cup LP} b\uparrow^{s'}$$

is a non-convergent peak in $\lfloor R \rfloor \cup LP$. However, $R^{\#} = \lfloor R_S \rfloor \cup LP$ is convergent if R is convergent and *sort-decreasing*, as we shall see below.

Definition 3.16. An order-sorted rule $C \to s \approx t$ is called *sort-decreasing*, iff for any specialization ρ, $s\rho \approx t\rho$ has a lowest parse such that $s^{\lfloor s\rho \rfloor} \geqslant s^{\lfloor t\rho \rfloor}$. A many-sorted rule $C \to s \approx t$ is called *sort-decreasing*, iff the left side s does not carry an injection at its top. A set of rules is sort-decreasing if each of its members is sort-decreasing.

If we are given a reduction ordering $>$ on $\mathcal{T}_\Sigma(X)$, it can be extended to a reduction ordering $>^{ms}$ on $T_{\Sigma^{ms}}(X)$ which is compatible with $=_{LP}$, simply by defining $t >^{ms} t'$ iff $\lceil t \rceil > \lceil t' \rceil$. Another reduction ordering $\rightsquigarrow$ on $T_{\Sigma^{ms}}(X)$ is obtained by defining $t \rightsquigarrow t'$, if (i) $\lceil t \rceil > \lceil t' \rceil$ or (ii) $\lceil t \rceil = \lceil t' \rceil$ and $t >_I t'$, where $>_I$ is the recursive path ordering that we have introduced to order the LP axioms. This ordering proves termination of both $\to_{\lfloor R_S \rfloor/LP}$ and $\to_{R^{\#}}$, if R is contained in $>$.

Theorem 3.17. *Let R be a set of order-sorted rules.*

(i) *R is sort-decreasing if and only if $R^{\#}$ is sort-decreasing.*

(ii) *Let R be sort-decreasing. R is convergent if and only if $R^{\#}$ is convergent and $\rightarrow_{\lfloor R_S \rfloor / LP}$ is terminating.*

(iii) *If $R^{\#}$ is convergent and sort-decreasing, then, $\downarrow_{R^{\#}} = =_{E^{\#}}$, i.e. for any two terms $u, v \in T_{\Sigma^{ms}}(X)_s$ we have $u =_{E^{\#}} v$, iff $u \downarrow_{R^{\#}} v$.*

Proof. (i) R is sort-decreasing, iff for any specialization ρ and any rule $C \rightarrow s \approx t$ in R, $s\rho \approx t\rho$ has a lowest parse such that $s^{\lfloor s\rho \rfloor} \geqslant s^{\lfloor t\rho \rfloor}$. According to the definition of lowest parses for rules, the latter is equivalent to $\lfloor s\rho \approx t\rho \rfloor = \lfloor s\rho \rfloor \approx \lfloor t\rho \rfloor \uparrow^{s^{\lfloor s\rho \rfloor}}$.

(ii) Suppose, $R^{\#}$ is convergent and $\rightarrow_{\lfloor R_S \rfloor / LP}$ is terminating. From Theorem 3.15 and the sort-decreasingness of R, $u \rightarrow_R v$ implies $\lfloor u \rfloor \rightarrow_{\lfloor R \rfloor / LP} \lfloor v \rfloor \uparrow^{s^{\lfloor u \rfloor}}$. Hence R is terminating. The local confluence of $\rightarrow_R$ follows from Theorem 3.15 and from the confluence of $R^{\#}$.

If, conversely, R is convergent, then both $\rightarrow_{\lfloor R_S \rfloor / LP}$ and $R^{\#}$ are contained in $\rightsquigarrow$ and hence are terminating. It remains to prove the local confluence of $R^{\#}$. We will show that for any peak $v \leftarrow_{R^{\#}} u \rightarrow_{R^{\#}} t$ there exists a proof of $v \overset{*}{\leftrightarrow}_{R^{\#}} t$ which only involves intermediate terms smaller than u wrt $\rightsquigarrow$. We have to distinguish four cases, depending on which kind of rule (either in LP or $\lfloor R_S \rfloor$) has been applied in any of the two rewrite steps.

The case in which both rules are in LP is proved by Proposition 3.10.

Let us now assume that both rules are in $\lfloor R_S \rfloor$. Then, from Theorem 3.15, $\lceil v \rceil \leftarrow_R \lceil u \rceil \rightarrow_R \lceil t \rceil$, hence $\lceil v \rceil \rightarrow_R^* r \leftarrow_R^* \lceil t \rceil$, by confluence of R. Applying Theorem 3.15, we obtain $v \rightarrow^*_{\lfloor R \rfloor / LP} \lfloor r \rfloor \uparrow^{s^{u}} \leftarrow^*_{\lfloor R \rfloor / LP} t$. (The lowest sort of r is less or equal to the lowest sorts of v and t as R is sort-decreasing.) This proof of $v =_{R^{\#}} u$ involves applications of LP-congruences and applications of rules in $\lfloor R \rfloor$ in terms smaller than u wrt $>^{ms}$. As $>^{ms}$ is compatible with LP, all intermediate terms in the complete proof are smaller than u wrt $\rightsquigarrow$.

For the mixed cases assume that $v \leftarrow_{R^{\#}} u$ by applying a rule $l \approx r \in LP$ and $u \rightarrow_{R^{\#}} t$ by applying a rule $C \rightarrow a \approx b \in \lfloor R_S \rfloor$. As the critical pairs lemma can only be proved for *reductive* conditional rewrite rules [16], it is not sufficient to just consider the case in which both rule applications overlap at a non-variable occurrence in a. Let us, however, start with this particular case by noting that there cannot be an overlap of a at a non-variable occurrence in l below the top of l. The top symbol in a is not an injection. Hence, overlaps are only created by unification of l with a non-variable subterm of a (including a). They yield mgus τ which send any variable in a to a composite injection. (l is either a composite injection itself, or it is a linear term of form $f(t_1, \ldots, t_n)$, with a non-injection f at the top and composite injections t_i as subterms.) Hence $\lceil (C \rightarrow a \approx b)\tau \rceil$ is a specialization of $C \rightarrow a \approx b$. Now, rewriting the r-part in v to LP-normalform makes $\lfloor \lceil a\tau \approx b\tau \rceil \rfloor$ applicable and rewriting with this rule produces a term t' which is LP-congruent to t. This completes the construction of the subconnected proof of $u = v$, except for one detail. We need to show in addition that the proof of convergence of the condition instances

in C upon $u \to_{R^\#} t$ carries over to a proof of convergence of the condition instances in $\lfloor\lceil C\tau\rceil\rfloor$. If C is empty, this is trivial. Otherwise, supposee, $u \to_{R^\#,n+1} t$. Then, the proofs for the C-instances are in $\overset{*}{\leftrightarrow}_{R^\#,n}$ and $n \geq 1$. As $\lceil\lfloor\lceil C\tau\rceil\rfloor\rceil = \lceil C\tau\rceil$, we obtain from Theorem 3.15 $\lfloor\lceil C\tau\rceil\rfloor \subset \overset{*}{\leftrightarrow}_{\lfloor R_S\rfloor/LP,n}$. As the LP-rules are all unconditional (i.e. all LP-steps occur on level 1), and $\leftarrow_{R^\#,n} \circ \to_{R^\#,n} \subset \downarrow_{R^\#}$ by the induction hypothesis, $\overset{*}{\leftrightarrow}_{\lfloor R_S\rfloor/LP,n} \subseteq \downarrow_{R^\#}$. This provides us with the required proof of convergence for the conditions $\lfloor\lceil C\tau\rceil\rfloor$.

The remaining non-trivial subcase is the one in which the two rewrite steps occur one above the other. If the LP-step is above the $\lfloor R_S\rfloor$-step, there is no problem, as the equations in LP are unconditional. The converse case is proved again by induction over the recursion level in rewriting. Rewrite proofs for condition instances allow us to construct rewrite proofs for LP-rewritten condition instances as both the original and the rewritten condition instances are proved on the same (smaller) recursion level.

(iii) We have to prove that the injectivity axioms for the injections are logical consequences of the equational theory $=_{R^\#}$, if $R^\#$ is sort-decreasing and convergent. Let $x\uparrow^{s'} \approx y\uparrow^{s'} \to x \approx y$ be an injectivity axiom and σ a substitution such that $x\uparrow^{s'}\sigma =_{R^\#} y\uparrow^{s'}\sigma$. Then there exists a rewrite proof of form

$$x\sigma\uparrow^{s'} \to^*_{R^\#} x'\uparrow^{s'} \to^*_{R^\#} u \leftarrow^*_{R^\#} y'\uparrow^{s'} \leftarrow^*_{R^\#} y\sigma\uparrow^{s'}$$

where x' and y' are the normalforms of $x\sigma$ and $y\sigma$, respectively. The proof $x'\uparrow^{s'} \to^*_{R^\#} u \leftarrow^*_{R^\#} y'\uparrow^{s'}$ can only involve the rules for the composite injections in CI which rewrite injections to injections. As both x' and y' are irreducible, $x' = u'\uparrow^s$ and $y' = u'\uparrow^s$, with u' some term that does not have an injection at the top. Therefore, $x' = y'$ and hence $x\sigma =_{R^\#} y\sigma$, which was to be shown. □

3.4. Eliminaton of non-sort-decreasing rules

Theorem 3.17 requires order-sorted rules to be sort-decreasing for the construction of an equivalent convergent system of many-sorted rules. Likewise, order-sorted completion as proposed in [11] requires rules to be sort-decreasing and fails, if non-sort-decreasing rules are generated. We shall see in Section 5.1 that translating into many-sorted specifications and applying conditional equation completion (to deal with the injectivity axiom of injections) is successful in simple cases of non-sort-decreasing rules. In many interesting cases, as in the subsequent example, the completion procedure which we will describe in Section 4 will not terminate.

Example 3.18

```
sort nzNat < nat, nat < int, nzNat < nzInt, nzInt < int
op
 0 : nat
 s : nat → nzNat
```

```
 +:int*int→int, nat*nat→nat, nat*nzNat→nat
   nzNat*nat→nat, nzNat*nzNat→nzNat
 -:nat→int, nzNat→nzInt, int→int, nzInt→nzInt
 *:int*int→int, nat*nat→nat
 square:int*int→nat
var i:int, j:int, n:nat
axioms
 -(0)=0
 -(-i)=i
 i+0=i
 0+i=1
 k+s(m)=s(k+m)
 (-s(k))+s(m)=(-k)+m
 i+(-j)=-((-i)+j)
 i*0=0
 0*i=0
 i*s(n)=i*n+i
 i*(-j)=-(i*j)
 (-i)*j=-(i*j)
 square(i)=i*i
```

The last axiom, when oriented from left to right, is clearly not sort-decreasing. A specification with the same initial algebra would be the one in which this equation is replaced by

$$i * i \approx n \rightarrow square(i) \approx n,$$

with n a variable of sort *nat*. This equation, when oriented from left to right, is sort-decreasing. However, it has the extra variable n in its condition and right side. The lowest parse of this equation would be

$$i * i \approx n\uparrow^{int} \rightarrow square(i) \approx n.$$

Equations of this kind are usually not admitted as rewrite rules. In fact, we plan to associate a specific operational semantics with it. $square(i)$ may be replaced by any n which can be obtained from normalizing $i * i$ and type checking the result by matching $n\uparrow^{int}$ with the normalform. If the normalform is unique, this process of finding the substitution for i and n at rewrite-time is completely backtrack-free. Deterministic oriented goal solving is not a complete goal solving method in general. Fortunately, an adequately designed completion procedure can make it become complete.

This idea of replacing non-sort-decreasing equations by sort-decreasing ones should be obvious, not requiring any further formalization. However, we should be saying something about whether this replacement preserves the initial algebra of a specification. We assume to be given a set E of order-sorted equations, as well as its many-sorted equivalent $E^{\#}$.

Definition 3.19. Let C be a set of unconditional equations and let $t \in \mathcal{T}_\Sigma(X)_s$ be a term of sort s, $var(C) \subset X$, and $s' \leq s$. We say that t *is of* E'*-type* s' *in context* C, where $E' \subseteq E$, if for any ground substitution σ of the variables in X (i) $E \vdash C\sigma$ iff $E' \vdash C\sigma$, and (ii) if $E \vdash C\sigma$, then there exists a term $u_\sigma \in \mathcal{T}_\Sigma(X)_{s'}$ such that $E' \vdash t\sigma \approx u_\sigma$.

In our example above we have $i * i$ of type *nat* in the empty context as, for any ground substitution, $i * i$ is equal to $s^{i*i}(0)$, a term of sort *nat*.

Proposition 3.20. *Let* $e = C \to l \approx r$ *be a conditional equation in* E *and let* r *be of* $E \backslash \{e\}$*-type* s *in context* C. *Then, replacing* $C \to l \approx r$ *by*

$$C, r \approx x \to l \approx x,$$

with x *a new variable of sort* s, *preserves* $=_E$ *on ground terms, and hence the intial algebra of the specification.*

4. Completion of many-sorted conditional equations

In this section, we will assume to be given a fixed many-sorted signature Σ^{ms}. Equations, terms, substitutions, etc. will be taken over this signature, unless specified otherwise. Furthermore, we assume a reduction ordering $>$ to be given on $T_{\Sigma^{ms}}(X)$. $>_{st}$ will denote the transitive closure of $> \cup st$, with st the strict subterm ordering. $>_{st}$ is well-founded and stable under substitutions.

4.1. Annotated equations and reductive rewriting

At completion-time we do not put any restrictions on the syntactic form of conditional equations. In particular, conditions and right sides may have extra variables. However, the application of equations at rewrite-time should be restricted to achieve decidability of the rewrite relation. Completion, if it terminates, will guarantee that this restricted application is complete. Formally, application restrictions can be modelled by considering a given set E of equations as a *generator* for rewrite rules.[4] In particular, the set E^r of *reductive* instances of the equations in E is of interest:

$$E^r = \{C\sigma \to s\sigma \approx t\sigma \mid C \to s \approx t \in E^{=},\ s\sigma > t\sigma,\ s\sigma >_{st} u\sigma,\ s\sigma >_{st} v\sigma, \text{ for any } u \approx v \in C\}.$$

In the general case, $\to_{E^r}$ is undecidable and requires (restricted) paramodulation to solve conditions of equations in E. Furthermore, the computed solutions have to be tested for reductivity. In order to make rewriting decidable and not too inefficient, our goal is to be able to appropriately restrict application of equations at rewrite-time.

[4] In [4] we develop a more general concept of application restrictions based on a notion of relevant substitutions.

We will annotate equations to specify in which way their use at rewrite-time should be restricted. For the purposes of this paper, an equation can be annotated as *operational* or *non-operational.* The intuitive meaning is that a non-operational equation should not contribute at all to the equational theory. Injectivity axioms, for example, should be irrelevant at rewrite-time.

In *operational* equations $C \to s \approx t$, condition equations $u \approx v \in C$ will be annotated as either *oriented* or *unoriented.* We will use the notation $u \stackrel{\cdot}{\approx} v$ to indicate the annotation "oriented". For an oriented condition, oriented goal solving is wanted. Altogether:

Definition 4.1. Let E be a set of annotated equations. E is viewed to generate the set E^a of rewrite rules $C\sigma \to s\sigma \approx t\sigma$ such that

(1) $C \to s \approx t \in E^=$ is annotated as operational and $C\sigma \to s\sigma \approx t\sigma \in E^r$, i.e. the instance is reductive,

(2) if $u \stackrel{\cdot}{\approx} v \in C$, then $v\sigma\sigma'$ is $\to_{E^a}$-irreducible for any $\to_{E^a}$-irreducible substitution σ'.

Clearly, $\to_{E^a} \subseteq \to_{E^r} \subseteq \to_E$, where the subset inclusions are in general proper, hence $\stackrel{*}{\leftrightarrow}_{E^a} \neq =_E$ in general. E is called *complete*, iff $\stackrel{*}{\leftrightarrow}_{E^a} = =_E$ and if $\to_{E^a}$ is convergent. A completion procedure attempts to complete an initally given E_0, i.e. attempts to find an equivalent and complete E.

In many practical cases, a final system E obtained by completion will have additional properties which makes $\to_{E^a}$ efficiently computable. For example, if

$$i * i \stackrel{\cdot}{\approx} n \uparrow^{int}_{nat} \to square(i) \approx n.$$

with the condition annotated as oriented, is in a complete E, $square(i)$ needs only to be rewritten for those instances of i for which the $\to_{E^a}$-normalform of $i * i$ is of the form $n \uparrow^{int}_{nat}$. Moreover, if $i * i$ is smaller in the reduction order than square(i), the replacement n will also always be smaller than $square(i)$, making any application of the equation reductive. No reductivity tests are required at rewrite-time. Equations which have this property will be called *quasi-reductive.* Note that let-expressions with patterns in functional programming languages such as MIRANDA are another example of equations with oriented conditions (cf. definition of quicksort below).

4.2. Quasi-reductive equations

To simplify the formal treatment in this section, we can assume that operational equations have oriented conditions only. (If an equation has an unoriented condition $u \approx v$, we can replace the latter by the two oriented conditions $u \stackrel{\cdot}{\approx} x$ and $v \stackrel{\cdot}{\approx} x$, where x is a new variable.)

In the classical case of unoriented conditions, the class of reductive equations [17, 16] allows for efficient rewriting [18]. In particular, conditions of equations are easily proved or disproved, and no goal solving is required. Moreover, there are no

reductivity tests required at rewrite-time, as any instance of a reductive equation is reductive.

In the case of oriented goal solving there exists a similarly efficient class of equations. Oriented goal solving $u\sigma \to_E^* v\sigma$ boils down to normalizing $u\sigma$ and then matching $v\sigma$ with the normal form, if any of the variables of u is already bound by the matching of the redex, or by the solution of some other condition equation. To formalize this idea, we will have to look at how variables are bound within an equation. We call a conditional equation $u_1 \approx v_1, \ldots, u_n \approx v_n \to s \approx t$ (with oriented conditions) *deterministic*, if, after appropriately changing the orientation of the consequent and choosing the order of the condition equations, the following holds true:

$$var(u_i) \subset var(s) \cup \bigcup_{1 \leq j \leq i} (var(u_j) \cup var(v_j)),$$

and

$$var(t) \subset var(s) \cup \bigcup_{j=1}^{n} (var(u_j) \cup var(v_j)).$$

Definition 4.2. A deterministic equation $u_1 \approx v_1, \ldots, u_n \approx v_n \to s \approx t$, $n > 0$, is called *quasi-reductive*, if for any substitution σ the following is satisfied:

(i) for any $0 \leq i < n$, if $u_j\sigma \geqslant v_j\sigma$, for $1 \leq j \leq i$, then $s\sigma >_{st} u_{i+1}\sigma$, and

(ii) if $u_j\sigma \geqslant v_j\sigma$, for $1 \leq j \leq n$, then $s\sigma > v\sigma$.

An unconditional equation $s \approx t$ is quasi-reductive if $s > t$.

The equation

$$i * i \approx n\uparrow^{int} \to square(i) \approx n$$

becomes quasi-reductive under a recursive path ordering, if $square > *$ in precedence. Also quasi-reductive is

$$split(x, l) \approx (l_1, l_2) \to sort(cons(x, l)) \approx append(sort(l_1), cos(x, sort(l_2))).$$

The termination proofs can be given by an appropriately chosen polynomial interpretation.

The following method appears to be useful for checking quasi-reductivity. Let us assume the existence of some enrichment $\Sigma' \supseteq \Sigma^{ms}$ of the signature such that the given reduction ordering on $T_{\Sigma^{ms}}(X)$ can be extended to a reduction ordering on $T_{\Sigma'}(X)$.

Proposition 4.3. *A deterministic equation $u_1 \approx v_1, \ldots, u_n \approx v_n \to s \approx t$, $n > 0$, is quasi-reductive, if there exists a sequence $h_i(\xi)$ of terms in $T_{\Sigma'}(X)$, $\xi \in X$, such that $s > h_1(u_1)$, $h_i(v_i) \geqslant h_{i+1}(u_{i+1})$, $1 \leq i < n$, and $h_n(v_n) \geqslant t$.*

Quasi-reductivity is a proper generalization of reductivity.

Proposition 4.4. *If the equation* $u_1 \approx u_{n+1}, \ldots, u_n \approx u_{2n} \to s \approx t$ *is reductive, then the equation*

$$u_1 \approx x_1, u_{n+1} \approx x_1, \ldots, u_n \approx x_n, u_{2n} \approx x_n \to s \approx t,$$

is quasi-reductive, if the x_i *are new, pairwise distinct variables.*

Lemma 4.5. *Let* E *be finite and* $u_1 \approx v_1, \ldots, u_n \approx v_n \to s \approx t \in E$ *a quasi-reductive equation.*

(1) *If* σ *is a substitution such that* $u_i\sigma \to_E^* v_i\sigma$, $1 \leq i \leq n$, *then* $s\sigma > t\sigma$.

(2) *If* N *is a term and* $N' \to_{E^a} N''$ *is decidable for all terms* N' *such that* $N >_{st} N'$, *then the applicability of the equation* $u_1 \approx v_1, \ldots, u_n \approx v_n \to s \approx t$ *in* N *is decidable.*

Proof. The proof of (1) follows immediately from the assumptions and the fact that $u_i\sigma \geqslant v_i\sigma$.

For the proof of (2), we note that because the given equation is deterministic, any σ which solves the condition is obtained by rewriting the $u_i\sigma_i$ and matching the rewrites against $v_i\sigma_i$, where σ_i is that part of σ which has been obtained after having matched s against a subterm of N and after having solved the condition equations up to index $i-1$. Because of termination of $\to_{E^a}$ there are only finitely many irreducible r with $u_i\sigma \to_{E^a}^* r$ that need to be matched against $v_i\sigma_i$, and, hence, finitely many σ_{i+1} which need to be considered for the next condition. As $u_j\sigma \to_{E^a}^* v_j\sigma$, and hence $u_j\sigma \geqslant v_j\sigma$, for $1 \leq j < i$ the quasi-reductivity implies that $N \geqslant_{st} s\sigma = s\sigma_i >_{st} u_i\sigma_i$. Hence any of these r can be effectively computed, as we have assumed the decidability of $\to_{E^a}$ in terms smaller than N. □

Corollary 4.6. *Let* E *be a set of annotated equations in which any operational equation is quasi-reductive. Then* $\to_{E^a}$ *is decidable.*

For confluent $\to_{E^a}$, the applicability of a quasi-reductive equation can be decided by matching the left side and, then, for $1 \leq i \leq n$, matching the v_i against the normal forms of the substituted u_i to obtain another part of the substitution. As quasi-reductive equations are deterministic, each variable in u_i is bound at the time when the ith condition is to be checked. Computing the substitution σ is completely backtrack-free in this case. Moreover, no termination proofs are required at rewrite-time.

4.3. Completion inference rules

Standard completion CC in the conditional case according to the concepts in [8] and further refined in [4] consists of three inference rules for adding consequences and of powerful techniques for eliminating redundant clauses and inferences. The availability of the latter is crucial for an acceptable termination behaviour of the completion procedure in practice.

To explain the underlying theory in detail would go beyond the scope of this paper. We will only briefly describe the basic inferences of completion and state some major results. For detailed proofs of these results using proof orderings we refer the reader to [3]. Proof techniques based on semantic arguments can be found in [6, 5]. These three papers also contain a detailed description of techniques for eliminating redundant clauses.

Adding a critical pair

Definition 4.7. Let $D \to u \approx v$ and $C \to s \approx t$ in $E^{=}$ both be operational. Furthermore, assume that their variables have been renamed such that they do not have any common variables. Assume, moreover, that o is a non-variable occurrence in s such that u/o and s can be unified with a mgu σ. Then,

$$C\sigma, D\sigma \to u[t]_o\sigma \approx v\sigma$$

is a *contextual critical pair* with superposition term $u\sigma$, if σ is potentially reductive both for $C \to s \approx t$ and $D \to u \approx v$. (We call σ *potientially reductive* for $C \to s \approx t$, if $s\sigma$ is a strictly maximal term in $C\sigma \to s\sigma \approx t\sigma$ with respect to $\succ$.)

Critical pairs are computed to replace peaks in rewriting. This definition allows us to superpose both sides of the consequent of any two operational equations to form critical pairs. However we have restricted attention to such critical pairs which can possibly appear in a peak of reductive applications. If σ is not possibly reductive for one of the overlapping equations, it cannot be further instantiated to a reductive application of the equation, and hence does not contribute to E^a.

Adding a superposition instance

If a non-operational or non-reductive equation has a non-empty condition part, superposition on a condition may be required to achieve that the equation will in fact be irrelevant for the equational theory of the final system.

Definition 4.8. Let D, $u \approx v \to l \approx r$ in E and let $C \to s \approx t$ in $E^{=}$ be an operational equation. Furthermore, assume that variables have been appropriately renamed. Let o be a non-variable occurrence in $u \approx v$ such that $(u \approx v)/o$ and s can be unified with a mgu σ. Then,

$$C\sigma, D\sigma, (u\sigma \approx v\alpha)[t\sigma]_o \to l\sigma \approx r\sigma$$

is called *an instance of* D, $u \approx v \to l \approx r$ *by superposing* $C \to s \approx t$ *on the condition* $u \approx v$, if σ is potentially reductive for $C \to s \approx t$, and if, in case $v\sigma$ and $u\sigma$ are comparable under the reduction order, o is inside the bigger of the two terms.

Adding a resolution instance

Definition 4.9. Let D, $u \approx v \to s \approx t \in E$ such that u and v can be unified with an mgu σ. Then

$$D\sigma \to s\sigma \approx t\sigma$$

is an instance of D, $u \approx v \to s \approx t$ obtained by *resolving* the condition $u \approx v$ (with $x \approx x$).

Superposition and resolution instances are computed to achieve $\overset{*}{\leftrightarrow}_{E^a} = =_E$ in the limit. Only potentially reductive instances of *operational* equations need to be superposed on conditions of equations. To improve the termination behaviour, one should additionally avoid, as much as possible, the computation of superpositions on conditions of instances of equations which are rewrite rules in E^a. In particular, superposition on the left side conditions of quasi-reductive equations such as

$$i * i \overset{*}{\approx} n\uparrow^{int} \to square(i) \approx n.$$

are redundant. Additionally, for non-quasi-reductive equations one may, as in [21], adopt the strategy of superposing on maximal literals. Here, as in [8] and [4], we propose to superpose on one specifically but arbitrarily selected condition. In many practical examples this will lead to far less superpositions. Hence, for non-operational equations and non-quasi-reductive equations which have conditions, we assume that exactly one of the conditions is annotated as being *selected for superposition.* As with any other annotation, this selection must not be changed throughout the life-time of the equation. These considerations lead to the following formal definition.

Definition 4.10. A resolution is called *critical*, if (i) or (ii) below apply. A superposition instance of D, $u \approx v \to l \approx r$ is called *critical* if any of the following three cases applies:

(i) D, $u \approx v \to l \approx r$ is annotated as non-operational and $u \approx v$ is the condition annotated as being selected for superposition.

(ii) D, $u \approx v \to l \approx r$ is annotated as operational, $u \approx v$ is the condition annotated as being selected for superposition, and neither $D\sigma$, $u\sigma \approx v\sigma \to r\sigma \approx l\sigma$ nor $D\sigma$, $u\sigma \approx v\sigma \to l\sigma \approx r\sigma$ is quasi-reductive.

(iii) D, $u \approx v \to l \approx r$ is annotated as operational, $u \approx v$ is annotated as oriented, σ is potentially reductive for either D, $u \approx v \to r \approx l$ *or* D, $u \approx v \to l \approx r$, *and* o, *the superposition occurrence, is inside* v.

With these definitions, only paramodulation on the right side of any of their oriented conditions is critical for quasi-reductive equations.

4.4. Fair completion procedures

A completion procedure is a mechanism which applies these inference rules, or removes redundant clauses. Starting from an initial set of equations E_0, it produces a sequence $E_0, E_1, \ldots$ of sets of equations, called CC-derivation, such that E_{i+1} results from E_i either by adding a conditional equation which follows from E_i, or by deleting an equation which is redundant in E_i. (An equation is redundant in a set E of equations, if it follows from "smaller" equations in E. We refer the reader to [6] or [5] for a formal definition of redundancy.) Let $E_\infty = \bigcup_j \bigcap_{k \geq j} E_k$ denote the limit system of this process. The required annotations of initial as well as generated equations and conditions may be given *in an arbitrary way* by some kind of "expert system", e.g. the human user. Some annotations may result in a fair completion and a successful terminaton of the process, whereas other annotations may cause failure or non-termination. A completion process is called *fair* if it is successful in the limit, i.e. if E_∞ is complete. Annotations also may result in a final set of equations for which $\rightarrow_{E_\infty^a}$ can be efficiently computed, particularly if all final equations are either non-operational or quasi-reductive, (cf. Corollary 4.6).

The following fairness result has been proved in [8] (with the extension to unfailing completion and oriented goals in [4]).

Theorem 4.11. *A CC-derivation $E_0, E_1, \ldots$ is fair, if the following holds true:*

(1) *E_∞ does not contain any unconditional equation annotated as non-operational;*

(2) *$\bigcup E_k$ contains all critical paramodulation instances of final equations by final equations and all critical resolution instances of final equations in E_∞;*

(3) *$\bigcup E_k$ contains all critical pairs between final (operational) equations in E_∞.*

Note that because of the first fairness constraint, completion may fail if a non-operational equation without conditions cannot be reduced or eliminated during completion. Completion becomes unfailing and, hence, refutationally complete for the equational theory, if the given reduction order is total on ground terms and if no unconditional equation will be annotated as non-operational (cf. [3]).

5. Order-sorted completion: the many-sorted way

In this section we illustrate by means of examples that our techniques of completion for conditional equations can be successfully applied to order-sorted specifications. In the examples non-operational equations will be labelled by a "-". Moreover, we rearrange conditions of non-operational equations such that the first condition is always the one which is selected for superposition. Operational equations will all be reductive or quasi-reductive with the given orientation of literals and ordering of conditions.

5.1. Smolka's example

Our first example is due to Smolka and shows the incompleteness of order-sorted replacement of equals by equals (cf. [25] or [20]) in the case of non-sort-decreasing rules.

Example 5.1

```
sort s1 < s2
op a:s1, b:s1, d:s2, f : s1 → s1
axioms
 a=d
 b=d
```

In this example, $f(a) \approx f(b)$ can be derived by order-sorted deduction, however it cannot be proven by replacement of equals by equals. The many-sorted equivalent $E^{\#}$ consists of the following equations.

Example 5.2

```
1    i(a)=d
2    i(b)=d
3—   i(x)=i(y) ⇒ x=y
```

where $i : s1 \to s2$ is the injection $\uparrow_{s1}^{s2}$. Axiom 3 is the injectivity property of i. Orienting 1 and 2 from left to right creates the following final system of equations.

Example 5.3

```
1    i(a)=d
3—   i(x)=i(y) ⇒ x=y
4—   i(x)=d    ⇒ x=a
5    b=a
```

Equation 4 is generated from superposing equation 1 on the condition of the non-operational injectivity axiom 3 (cf. fairness constraint 2). Here we have decided to classify 4 as non-operational although it becomes a quasi-reductive equation when orienting its literals from right to left. After this, 4 generates equation 5 from superposition with equation 2. If $b > a$ in precedence, equation 5 is reductive, allowing us to eliminate equation 2 by reduction. Any other superposition on the condition of 3 or 4 only generates redundant equations.

5.2. Squares of integers

We return to the specification of integers as given in Example 3.18. The corresponding set of many-sorted equations $E^{\#}$ is given below. Here, injections $\uparrow_s^{s'}$ are ambiguously denoted by their codomain s', e.g. *int* denotes both $\uparrow_{nzInt}^{int}$ and $\uparrow_{nat}^{int}$. The order-sorted rather than the many-sorted function symbols are used. $x : s$ denotes a variable x of sort s.

Example 5.4

```
 1      -(0)=int(0)
 2      -(-(i:int))=i
 3      (i:int)+int(0)=i
 4      int(0)+(i:int)=i
 5      (k:nat)+nat(s(m:nat))=nat(s((k:nat)+(m:nat)))
 6      int(-s(k:nat))+int(nat(s(m:nat)))=-(k:nat)+int(m:nat)
 7      (i:int)+(-(j:int))=-(-(i:int)+(j:int))
 8      (i:int)*int(0)=int(0)
 9      int(0)*(i:int)=int(0)
10      (i:int)*int(nzInt(s(m:nat)))=(i:int)*int(m:nat)+(i:int)
11      (i:int)*(-(j:int))=-(i:int)*(j:int)
12      (-(i:int))*(j:int)=-(i:int)*(j:int)
13      (i:int)*(i:int)≈int(k:nat) ⇒ square(i:int)=k
-----
I1      -int(X1:nat)=-(X1:nat)
I2      -int(X1:nzInt)=int(-(X1:nzInt))
I3      -nat(X1:nzNat)=int(-nzInt(X1:nzNat))
I4      -nat(X1:nzNat)=int(-(X1:nzNat))
I5      -nzInt(X1:nzNat)=-(X1:nzNat)
I6      int(X2:nat)+int(X1:nat)=int((X2:nat)+(X1:nat))
I7      nat(X2:nzNat)+(X1:nat)=(X2:nzNat)+(X1:nat)
I8      int(X2:nat)*int(X1:nat)=int((X2:nat)*(X1:nat))
I9      int(nat(X:nzNat))=int(nzInt(X:nzNat))
-----
i1-     nat(X:nzNat)=nat(Y:nzNat) ⇒ X=Y
i2-     int(X:nat)=int(Y:nat) ⇒ X=Y
i3-     nzInt(X:nzNat)=nzInt(Y:nzNat) ⇒ X=Y
i4-     int(X:nzInt)=int(Y:nzInt) ⇒ X=Y
```

The (INH)-equations have a number $I1$ to $I8$, equation $I9$ is the only (CI)-axiom, the non-operational injectivity axioms are the equations $i1$-$i4$. We have only introduced (INH)-equations between any two neighboring operators (wrt $<$) as the remaining ones are generated as critical pairs. The (INH)- and (CI)-equations and any other equation $u \approx v$ for which $\lceil u \rceil = \lceil v \rceil$ will be oriented in the many-sorted

world according to $>_I$. Any equation $u \approx v$, for which $\lceil u \rceil \neq \lceil v \rceil$ will be ordered by a reduction order on the order-sorted terms. As $>_I$ is predefined (cf. Definition 3.6), the user need not be bothered with precedences between the many-sorted operators. The result of completing this system is the following.

Example 5.5

```
 1    -(0)=int(0)
 2    -(-(i:int))=i
 2a   -(-(X1:nzInt))=X1
 2b   -(-(X1:nat))=int(X1:nat)
 2c   -(-(X1:nzNat))=nzInt(X1:nzNat)
 3    (i:int)+int(0)=i
 3a   (X2:nat)+0=X2
 3b   (X:nzNat)+0=nat(X:nzNat)
 4    int(0)+(i:int)=i
 4a   0+(X1:nat)=X1
 5    (k:nat)+nat(s(m:nat))=nat(s((k:nat)+(m:nat)))
 5a   (X2:nzNat)+nat(s(m:nat))=nat(s((X2:nzNat)+(m:nat)))
 6    int(-s(k:nat))+int(nzInt(s(m:nat)))=-(k:nat)+int(m:nat)
 7    (i:int)+(-(j:int))=-(-(i:int)+(j:int))
 7a   (i:int)+int(-(X1:nzInt))=-(-(i:int)+int(X1:nzInt))
 7b   (i:int)+int(-(X1:nzNat))=-(-(i:int)+int(nzInt(X1:nzNat)))
 7c   (i:int)+(-(X1:nat))=-(-(i:int)+int(X1:nat))
 8    (i:int)*int(0)=int(0)
 8a   (X2:nat)*0=0
 9    int(0)*(i:int)=int(0)
 9a   0*(X1:nat)=0
10    (i:int)*int(nzInt(s(m:nat)))=(i:int)*int(m:nat)+(i:int)
10a   (X2:nat)*nat(s(m:nat))=(X2:nat)*(m:nat)+(X2:nat)
11    (i:int)*(-(j:int))=-(i:int)*(j:int)
11a   (i:int)*int(-(X1:nzInt))=-(i:int)*int(X1:nzInt)
11b   (i:int)*(-(X1:nat))=-(i:int)*int(X1:nat)
11c   (i:int)*int(-(X1:nzNat))=-(i:int)*int(nzInt(X1:nzNat))
12    (-(i:int))*(j:int)=-(i:int)*(j:int)
12a   int(-(X1:nzInt))*(j:int)=-int(X1:nzInt)*(j:int)
12b   (-(X1:nat))*(j:int)=-int(X1:nat)*(j:int)
12c   int(-(X1:nzNat))*(j:int)=-int(nzInt(X1:nzNat))*(j:int)
13    (i:int)*(i:int)≈int(k:nat) ⇒ square(i:int)=k
13a   (i:int)*(i:int)≈int(nzInt(X:nzNat))
        ⇒ square(i:int)=nat(X:nzNar)
-----
 I1    -int(X1:nat)=-(X1:nat)
```

```
 I2    -int(X1:nzInt)=int(-(X1:nzInt))
 I3    -nat(X1:nzNat)=int(-(X1:nzNat))
 I5    -nzInt(X1:nzNat)=-(X1:nzNat)
 I6    int(X2:nat)+int(X1:nat)=int((X2:nat)+(X1:nat))
 I7    nat(X2:nzNat)+(X1:nat)=(X2:nzNat)+(X1:nat)
 I8    int(X2:nat)*int(X1:nat)=int((X2:nat)*(X1:nat))
 I9    int(nat(X:nzNat))=int(nzInt(X:nzNat))
I10    int(X2:nat)+int(nzInt(X:nzNat))=int((X2:nat)+nat(X:nzNat))
I11    int(nzInt(X:nzNat))+int(X1:nat)=int((X:nzNat)+(X1:nat))
I12    int(nzInt(X:nzNat))+int(nzInt(Y:nzNat))
         =int((X:nzNat)+nat(Y:nzNat))
I13    int(X2:nat)*int(nzInt(X:nzNat))
         =int((X2:nat)*nat(X:nzNat))
I14    int(nzInt(X:nzNat))*int(X1:nat)=int(nat(X:nzNat)*(X1:nat))
I15    int(nzInt(X:nzNat))*int(nzInt(Y:nzNat))
         =int(nat(X:nzNat)*nat(Y:nzNat))
-----
i1-    nat(X:nzNat)=nat(Y:nzNat) ⇒ X=Y
i2-    int(X:nat)=int(Y:nat) ⇒ X=Y
i3-    nzInt(X:nzNat)=nzInt(Y:nzzNat) ⇒ X=Y
i4-    int(X:nzInt)=int(Y:nzInt) ⇒ X=Y
i5-    int(nzInt(X:nzNat))=int(Y:nat) ⇒ nat(X:Nat)=Y
i6-    int(nzInt(X1:nzNat))=int(nzInt(X:nzNat))
        and int(nzInt(X:nzNat))=(i1:int)*(i1:int) ⇒ X=X1
```

The final system also contains the lowest parses of many specializations (indicated by letters $a, b, c, \ldots$) of the initial order-sorted rules. This is in accordance with Theorem 3.17. In a *reduced* final system like the one above, however, not all specializations need to be present. The remaining (INH)-equations $I10$–$I15$ have been added. Equation 13a has been generated from superposing $I9$ on the right side of the condition of 13 (cf. Theorem 4.11). In this example, completion has just verified the completeness of the initial system. No new order-sorted equation has been generated. The new equations on the many-sorted level serve to synchronize the application of order-sorted rules with the computation of lowest parses.

5.3. Maximum of integers

In this final example completion restructures in a non-trivial way the initial given set of conditional equations. Here, the subsort structure on integers is as in the previous example.

Example 5.6

```
sort bool, nzNat<nat, nat<int, nzNat<nzInt, nzInt<int
op
```

```
0 : nat
s : nat → nzNat
- : nat→int, nzNat→nzInt, int→int, nzInt→nzInt
true : bool
false : bool
=< : int*int→bool
max : int*int→int
var i, j, l:int, k, m:nat, n:nzNat
axioms
-(0)=0
-(-i)=i
-----
(0=< n)=true
(-m=< n)=true
(m=<-n)=false
(s(m)=< s(k))=(m=< k)
(m=< k)=((-k)=< (-m))
(i=< j)=true and (j=< l)=true ⇒ (i=< l)=true
-----
(i=< j)=true ⇒ max(i, j)=j
(i=< j)=false ⇒ max(i, j)=i
max(i, i)=i
(i=< max(i, j))=true
(i=< max(j, i))=true
```

The subset of operational equations of the final system which is generated from $E^{\#}$ in this case is the following.

Example 5.7

```
-int(X1:nat)=-(X1:nat)
-int(X1:nzInt)=int(-(X1:nzInt))
-nat(X1:nzNat)=int(-(X1:nzNat))
-nzInt(X1:nzNat)=-(X1:nzNat)
-(0)=int(0)
-(-(i:int))=i
int(nat(X:nzNat))=int(nzInt(X:nzzNat))
-(-(X1:nzInt))=X1
-(-(X1:nat))=int(X1:nat)
-(-(X1:nzNat))=nzInt(X1:nzNat)
-(k:nat)=<-(m:nat)=int(m:nat)=<int(k:nat)
-(m:nat)=<int(nzInt(nn:nzNat))=true
int(m:nat)=<int(-(n:nzNat))=false
-(k:nat)=<int(0)=int(0)=<int(k:nat)
```

```
int(0)=<-(m:nat)=int(m:nat)=<int(0)
int(nzInt(X1:nzNat))=<int(0)=false
int(0)=<int(nzInt(n:nzNat))=true
-(k:nat)=<int(-(X1:nzNat))=int(nzInt(X1:nzNat))=<int(k:nat)
int(-(X1:nzNat))=<-(m:nat)=int(m:nat)=<int(nzInt(X1:nzNat))
int(-(X1:nzNat))=<int(nzInt(n:nzNat))=true
int(nzInt(X:nzNat))=<(-(n:nzNat))=false
int(-(T:nzNat))=<int(-(X:nzNat))
 =int(nzInt(X:nzNat))=<int(nzInt(Y:nzNat))
int(nzInt(s(m:nat)))=<int(nzInt(s(k:nat)))
 =int(m:nat)=<int(k:nat)
max(i:int, i:int)=i
i:int=<j:int=true⇒max(i:int, j:int)=j
i:int=<j:int=false⇒max(i:int, j:int)=i
j:int=<j:int=true
int(-(n:nzNat))=<int(m:nat)=true
i1:int=<j1:int=true⇒i1:int=<max(j1:int, j:int)=true
i1:int=<j1:int=true⇒i1:int=<max(j:int, j1:int)=true
```

For example, the last two equations do not directly correspond to equations in the initial system. They are generated through superposition on the transitivity axion for $=<$.

Acknowledgment

The author is grateful to Hubert Bertling for many discussions on the subject of this paper.

References

[1] L. Bachmair, Proof methods for equational theories, PhD thesis, University of Illinois at Urbana-Champaign, 1987.

[2] L. Bachmair, N. Dershowitz and J. Hsiang, Orderings for equational proofs, in: *Proc. 1st Symp. on Logic in Computer Science*, Boston, MA, (1986) 346-357.

[3] H. Bertling, Knuth-Bendix completion of Horn clause programs for restricted linear resolution and paramodulation, PhD thesis, FB Informatik, Universität Dortmund, 1990.

[4] H. Bertling and H. Ganzinger, Completion-time optimization of rewrite-time goal solving, in: *Proc. 3rd Internat. Conf. on Rewriting Techniques and Applications*, Chapel Hill, Lecture Notes in Computer Science **355** (Springer, Berlin, 1989) 45-58.

[5] L. Bachmair and H. Ganzinger, Completion of first-order clauses with equality by strict superposition, in: *Proc. 2nd Internat. Workshop on Conditional and Typed Rewriting*, Montreal, Lecture Notes in Computer Science (Springer, Berlin, 1991).

[6] L. Bachmair and H. Ganzinger, On restrictions of ordered paramodulation with simplification, in: *Proc. 10th Internat. Conf. on Automated Deduction*, Kaiserslautern, Lecture Notes in Computer Science **449** (Springer, Berlin, 1990) 427-441.

[7] N. Dershowitz, M. Okada and D.A. Plaisted, Confluence of conditional rewrite systems, in: *Proc. 1st Internat. Workshop on Conditional Term Rewriting*, Orsay, 1987, Lecture Notes in Computer Science **308** (Springer, Berlin, 1988) 31-44.

[8] H. Ganzinger, A completion procedure for conditional equations, in: *Proc. 1st Intenat. Workshop on Conditional Term Rewriting*, Orsay, 1987, Lecture Notes in Computer Science **308** (Springer, Berlin, 1988) 62-83; revised version *J. Symb. Comput.* **11** (1991) 51-81.

[9] H. Ganzinger, Completion with history-dependent complexities for generated equations, in: D.T. Sannella and A. Tarlecki, eds., *Recent Trends in Data Type Specifications*, Lecture Notes in Computer Science **332** (Springer, Berlin, 1988) 73-91.

[10] J.A. Goguen, J.P. Jouannaud and J. Meseguer, Operational semantics of order-sorted algebra, in: *Proc. 12th Internat. Conf. on Automata, Languages and Programming, Nafplion,* Lecture Notes in Computer Science **194** (Springer, Berlin, 1985) 221-231.

[11] I. Gnaedig, C. Kirchner and H. Kirchner, Equational completion in order-sorted algebras, in: *Proc. CAAP'88*, Lecture Notes in Computer Science **299** (Springer, Berlin, 1988) 165-184.

[12] J.A. Goguen and J. Meseguer, Order-sorted algebra I: partial and overloaded operations, errors and inheritance, Technical Report, SRI International, Computer Science Laboratory, 1987.

[13] J.A. Goguen, Order-sorted algebra. Semantics and theory of computation, Report No. 14, UCLA Computer Science Department, 1978.

[14] M. Gogolla, Partiell geordnete Sortenmengen und deren Anwendung zur Fehlerbehandlung in abstrakten Datentypen, Dissertation, Technische Universität Braunschweig, West Germany, 1986.

[15] H. Ganzinger and R. Schäfers, System support for modular order-sorted Horn clause specifications, in: *Proc. 12th Internat. Conf. on Software Engineering*, Nice (1990) 150-163.

[16] J.P. Jouannaud and B. Waldman, Reductive conditional term rewriting systems, in: *Proc. 3rd TC2 Working Conf. on the Formal Description of Prog. Concepts*, Ebberup, Denmark (North-Holland, Amsterdam, 1986).

[17] S. Kaplan, Fair conditional term rewriting systems: unification, termination and confluence, in: *Recent Trends in Data Type Specification*, IFB 116 (Springer, Berlin, 1985).

[18] S. Kaplan, A compiler for conditional term rewriting systems, in: *Proc. 2nd Internat. Conf. on Rewriting Techniques and Applications*, Bordeaux, Lecture Notes in Computer Science **256** (Springer, Berlin, 1987) 25-41.

[19] S. Kaplan and J.-L. Remy, Completion algorithms for conditional rewriting systems, in: *MCC Workshop on Resoluton of Equations in Algebraic Structures*, Austin (1987).

[20] C. Kirchner, H. Kirchner and J. Meseguer, Operational semantics of OBJ3, in: *Proc. 15th Internat. Coll. on Automata, Languages and Programming,* Tampere (1988).

[21] E. Kounalis and M. Rusinowitch, On word problems in Horn logic, in: N. Dershowitz and S. Kaplan, eds., *Proc. 1st Workshop on Conditional Rewriting Systems*, Lecture Notes in Computer Science **308** (Springer, Berlin, 1988) 144-160.

[22] A. Oberschelp, Untersuchungen zur mehrsortigen Quantorenlogik, *Math. Ann.* **145** (1962) 297-333.

[23] A. Poigne, Partial algebras, subsorting, and dependent types, in: D.T. Sannella and A. Tarlecki, eds., *Recent Trends in Data Type Specifications*, Lecture Notes in Computer Science **332** (Springer, Berlin, 1988) 208-234.

[24] M. Rusinowitch, Theorem-proving with resolution and superposition: an extension of Knuth and Bendix procedure as a complete set of inference rules, Report 87-R-128, CRIN, Nancy, 1987.

[25] G. Smolka, W. Nutt, J.A. Goguen and J. Meseguer, Order sorted equational computation, in: *Proc. Coll. on Resolution of Equations in Algebraic Structures*, Austin 1(987).

[26] U. Waldmann, Semantics of ordersorted-specifications, Forschungsbericht 297, FB Informatik, Universität Dortmund, 1989 *Theoret. Comput. Sci.* (to appear).

[27] F. Winkler and B. Buchberger, A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm, in: *Proc. Coll. on Algebra, Combinatorics and Logic in Computer Science*, Györ, Hungary (1983).

[28] Hantao Zhang and J.-L. Rémy, Contextual rewriting, in: *Proc. 1st Internat. Conf. on Rewriting Techniques and Applicatons*, Dijon, Lecture Notes in Computer Science **202** (Springer, Berlin, 1985) 46-52.