Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT

Conditional Rewriting modulo a Built-in Algebra

Jürgen Avenhaus, Klaus Becker

SEKI Report SR-92-11 (SFB)

Conditional Rewriting modulo a Built-in Algebra[1]

Jürgen Avenhaus, Klaus Becker
Fachbereich Informatik
Universität Kaiserslautern
6750 Kaiserslautern
Germany

[1]This research was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D4-Projekt).

Abstract

We present a method to integrate built-in operations that are described by a given built-in algebra into conditional rewriting. First, equational specifications will be assigned a suitable semantics that takes into account the predefined structures. The interpretation of "semantically and syntactically mixed objects" is based on an appropriate introduction of sort hierarchies that allows to separate semantical and syntactical considerations. As a consequence of this separation a great deal of "classical rewrite theory" as for instance a critical pair lemma can be generalized to our context. Further we are able to construct appropriate well-founded orderings to guarantee the termination of the rewrite relation that allow to integrate semantical informations from the given built-in algebra. In order to illustrate the ideas several examples concerning built-in arithmetic are presented.

1 Introduction

Sets of conditional equations may be considered as the programs of a functional programming language with conditional rewriting as its computation mechanism. Conditional equations, interpreted as rewrite rules, are used to simplify terms — the basic objects to be operated on — according to the notion of "replacing equals by equals" until normal forms are possibly reached. (For a survey of conditional rewriting see [DeOk90].)

Whereas built-in operations are available in common programming languages as PASCAL, LISP or PROLOG, they usually can not be employed in the rewrite case. To yield an easy to handle programming environment it would be of great interest to be able to integrate predefined structures within the rewrite process.

Consider as an example the following equations, intended to define the greatest common divisor function over the natural numbers.

Example 1.1 *This example is refered to as 'standard example' throughout the paper.*

$$\begin{array}{lll} (1) & g(x,0) & = & x \\ (2) & g(0,y) & = & y \\ (3) & g(x+y,y) & = & g(x,y) \\ (4) & g(x,x+y) & = & g(x,y) \end{array}$$

It would be desirable to consider '+' and '0' as symbols with a predefined meaning that agrees with the natural interpretation of the symbols in the natural numbers. But in most rewrite environments equations as those above would not be "executable". Instead the specifier would have to define everything he uses himself in a bottom-up fashion.

There are at least two additional reasons that make built-in concepts attractive. First, it may become possible to use objects as for instance real numbers that cannot be specified equationally. Thus, built-ins can increase the expressive power of the specification environment. Secondly, one possibly gains efficiency when treating predefined objects by appropriate built-in algorithms.

The aim of this paper is to present a general approach of how to integrate predefined objects and operations into rewrite based equational reasoning. The problems that arise originate from the fact that syntactical and semantical methods have to be mixed when treating objects (terms, equations etc.) that consist of syntactical and semantical constituents. In particular we must generalize according to such interference effects

- matching and equivalence check, which are fundamental for conditional rewriting,
- unification, which is needed to ckeck (ground) confluence — a kind of correctness property of a set of rewrite rules,
- orderings that guarantee the termination of the rewrite relation.

To solve the problems we will separate syntactical and semantical considerations as far as possible. As a technical means for separation we switch over from "mixed terms" to "semantical congruence classes" (i.e. "mixed terms" are to be considered equivalent according to predefined equivalences). Representatives of the "semantical congruence classes" will be treated by syntactical means that have to be compatible with the semantical equivalence.

This conceptual proceeding is not new. In the case of unorientable equations a theory of rewriting modulo such equations based on the ideas just described has been developed (see [PeSt81], [JoKi86], [BaDe89]). But the applicability of this theory is rather limited. Serious difficulties arise when trying to design appropriate orderings and unification algorithms.

As we want to develop an approach to integrate — at least in principle — arbitrary built-in objects and operations without any limitation, we cannot directly adopt the existing theory. Instead we will design a theory of rewriting modulo predefined equivalences and so avoid by some moderate restrictions the interferences that cause some of the main problems. The following remaks are to make clear our concept.

Consider a built-in operation '+' that is commutative on the built-in objects, hence induces the predefined equation $x+y = y+x$. Let f be a (unary) new "syntactical" function symbol and consider the "mixed terms" $f(a+b)$ and $f(a)+f(b)$ where a and b represent built-in objects. It is reasonable to require that the equivalence of $a+b$ and $b+a$ induces the equivalence of $f(a+b)$ and $f(b+a)$. If we extended the range of validity of predefined equations in a liberal fashion, we could also demand for the equivalence of $f(a)+f(b)$ and $f(b)+f(a)$. But as such a liberal interpretation of predefined equations would cause the problems mentioned above, we do not adopt it here. This decision can be motivated not only "by need" — as just done — but also "by matter"— as follows.

When specifying by conditional equations over a built-in domain, the specifier is interested primarily in objects that are defined wrt. built-in objects (i.e. that are equivalent to built-in objects wrt. the conditional equations of the specification). Hence, if $f(a)$ or $f(b)$ is not defined wrt. built-in objects, it does not matter (concerning the intent of the specifier) whether the two terms are equivalent or not. On the other side, if both terms are defined wrt. built-in objects, then the equivalence of $f(a)+f(b)$ and $f(b)+f(a)$ results as a consequence of the commutativity of the built-in objects. Thus there is no need for requiring the liberal interpretation of predefined equations.

To realize our approach we first define an adequate semantics. As the main technical means to easily distinguish between predefined and mixed objects, we design an appropriate hierarchy of sorts: for each sort a copy is made and then the two sorts are related by a subsort declaration so that one of them — the "lower sort" — describes built-in resp. semantical objects, whereas the other — the "higher sort" — describes mixed resp. syntactical objects. This hierarchy aspect allows us to interprete equations according to the intuition that variables occuring in the equations are to range over built-in objects. As we may interprete built-in objects as constructors our approach provides a method to define a kind of "constructor semantics". Further, this approach allows us to deal with equations that define function symbols only partially wrt. the "constructors" (see also [KaMu86]).

We will demonstrate in this paper that by our concept to treat built-in structures we are able to adopt with some minor modifications a great deal of "classical rewrite theory".

Our approach to integrate built-ins into the rewrite process differs from others in various aspects. First, we do not consider implementational aspects concerning rewriting of mixed ground terms as it is done in [KaCh89] and [Wa90]. Our work is more related to that of Vorobyov [Vo89] and Kirchner, Kirchner and Rusinowitch [KKR90].

Vorobyov deals with rewriting in the special context of built-in arithmetic.

Kirchner, Kirchner and Rusinowitch develop a very general approach based on the notion of constraints. (Note that Vorobyov also uses a kind of constraints though not mentioning this expression.) As in the case of constraint logic programming (see [JaLa87a]), constraints are used to represent knowledge about predefined structures. The aim of this approach is to convert syntactical problems into semantical ones by placing them — roughly spoken — into the constraint part.

Both papers [Vo89] and [KKR90] require the conditions to be formulated in the language of built-ins, whereas we allow arbitrary conditions. Concerning some of the results of [Vo89] this restriction can be dropped (see [Ay92]).

Vorobyov avoids the interference effects between predefined equations and new syntactical function symbols by introducing a rather strong restriction: new syntactical function symbols are not allowed to have built-in domains as codomains. Note that by this restriction example 1.1 lies out of the range of his theory.

The interference problem is solved in [KKR90] in a way that is similar to ours from the methodical point of view (namely avoid the interference effects by introducing an appropriate semantics) but that differs from ours with respect to its realization.

Both approaches [Vo89] and [KKR90] differ from ours in the definition of the rewrite relation. Whereas we define it traditionally via matching, they replace matching by case splitting and a constraint satisfiability check. This difference reflects the contrary notions of explicit and implicit representation of knowledge (c. [JaLa87b]).

Our decision in favour of matching has two reasons. First, by this decision we are able — in contrast to [Vo89] and [KKR90] — to present a general method of how to guarantee termination. Secondly, we think that whenever explicit knowledge is available, then it should be employed to keep the objects to be treated by rewriting as simple as possible. Of course this decision limits rewriting in our approach to cases where matching is feasible.

The paper is organized as follows: The sections 2 and 3 introduce our basic concept of how to interprete specifications in the presence of built-ins. The division into two sections reflects the fact that the interpretation is established by syntactical as well as semantical means. In section 4 we define rewriting modulo a given built-in algebra and consider the "correctness of rewriting" (captured by the Church-Rosser property). Section 5 is devoted to a critical pair test for local confluence along the ideas of Knuth and Bendix [KnBe70]. Section 6 deals with termination problems. Finally in section 7 we make some remarks about the decidability of the rewrite relation.

We assume that the reader is familiar with the basic concepts of term rewriting, equational reasoning (see e.g. [AvMa90, DeJo90]) and mathematical logic. We define notions and notations only if they differ from standard ones.

2 Syntax

The notion of order-sorted specification is used to code some basic semantical prerequisites into the syntax. By this decision we are able to simplify the formulation of the restrictive use of variable instantiations and hence to simplify the definition of the semantics of the specifications. (For a survey of order-sorted equational computation see [SNGM89].)

The sort hierarchies to be considered in the sequel are very well-behaved, so that no conflict about which concept to treat sort hierarchies arises (see [Wa92] for a comparison of the different concepts).

Definition 2.1 *A* signature $\Sigma = (S, F, D)$ *consists of a set* S *of sort symbols, a set* F *of function symbols and a set* D *of function declarations* $f : s_1, \ldots, s_n \rightarrow s$ $(f \in F; s_i, s \in S)$ *and subsort declarations* $s_1 \lhd s_2$ $(s_1, s_2 \in S)$*, where* $\lhd$ *denotes the ordering relation between the sorts.*

A signature $\Sigma = (S, F, D)$ *is said to be* flat *iff* D *contains no subsort declarations and for any* $f \in F$ *there exists exactly one function declaration in* D*.*

Before introducing the fundamental construction of sort hierarchies used throughout this paper, we give a short motivation.

Built-in objects are described in a built-in language given by a "normal" (flat) signature Σ_0. The introduction of new "syntactical" function symbols is captured as usual by the notion

of a signature enrichment $\Sigma_0 + \Sigma_1$. Note that the resulting signature is still considered to be flat. Next this signature $\Sigma_0 + \Sigma_1$ will be suitably interpreted: By a copying process we get an order-sorted signature Σ induced by Σ_0 and Σ_1.

Definition 2.2 *A* signature enrichment $\Sigma_0 + \Sigma_1$ *consists of a flat signature* $\Sigma_0 = (S_0, F_0, D_0)$ *(for the built-in language) and a triple* $\Sigma_1 = (\emptyset, F_1, D_1)$ *such that* (S_0, F_1, D_1) *is a flat signature too with* $F_0 \cap F_1 = \emptyset$.

Note that no new sorts are introduced by Σ_1. The more general case without such a sort restriction does not cause any difficulties. However it makes necessary some additional case distinctions that we want to avoid here.

The copying process to be defined below requires F_0 to be divided into $F_0^{(=0)}$ and $F_0^{(\geq 1)}$, the set of symbols from F_0 with an arity that is equal to 0 — the constants from F_0 — resp. greater or equal than 1.

Definition 2.3 *Let* $\Sigma_0 + \Sigma_1 = (S_0, F_0, D_0) + (\emptyset, F_1, D_1)$ *be a (flat) signature enrichment.* $\Sigma = (S, F, D)$ *is said to be the* hierarchical signature resp. order-sorted signature induced by Σ_0 and Σ_1 — *written* $\Sigma = \Sigma_0 \oplus \Sigma_1$ — *iff*

- $S = S_0 \cup S^\wedge$ *where* $S^\wedge = \{s^\wedge \mid s \in S_0\}$ *and*
- $F = F_0 \cup F_1$ *and*
- $D = D_0 \cup D^\wedge \cup D_{sort}$ *where*
  $D^\wedge = \{f : s_1^\wedge, \ldots, s_n^\wedge \rightarrow s^\wedge \mid f \in F_0^{(\geq 1)} \cup F_1$ *and* $f : s_1, \ldots, s_n \rightarrow s \in D_0 \cup D_1\}$ *and*
  $D_{sort} = \{s \lhd s^\wedge \mid s \in S_0\}$.

When speaking roughly the elements of S_0 are called sorts of type "low" and the elements of $S^\wedge$ sorts of type "high". Note that the function symbols from the built-in language with an arity greater or equal than 1 are declared twice — a declaration for built-ins and an additional for mixed objects. This reflects the fact that these symbols have a semantical and syntactical "flavour".

Example 2.1 *We first define suitable signatures to describe natural numbers and integers. Let* $\Sigma_{nat} = (S_{nat}, F_{nat}, D_{nat})$ *resp.* $\Sigma_{int} = (S_{int}, F_{int}, D_{int})$ *with*

$$\begin{aligned}
S_{nat} &= \{nat\} \\
F_{nat} &= \{+, *, 0, 1, 2, \ldots\} \\
D_{nat} &= \{+ : nat, nat \rightarrow nat, * : nat, nat \rightarrow nat,\ 0 : nat,\ 1 : nat, \ldots\} \\
S_{int} &= \{int\} \\
F_{int} &= \{-, +, *, 0, 1, 2, \ldots\} \\
D_{int} &= \{- : int \rightarrow int, + : int, int \rightarrow int, * : int, int \rightarrow int,\ 0 : int,\ 1 : int, \ldots\}.
\end{aligned}$$

To continue our standard example let $\Sigma_0 = \Sigma_{nat}$ *and* $\Sigma_1 = (\emptyset, F_1, D_1)$ *with*

$$\begin{aligned}
F_1 &= \{g\} \\
D_1 &= \{g : nat, nat \rightarrow nat\}.
\end{aligned}$$

To obtain $\Sigma = \Sigma_0 \oplus \Sigma_1$ *let*

$$
\begin{array}{lcl}
S & = & \{nat, nat^\wedge\} \\
F & = & \{+, *, g, 0, 1, 2, \ldots\} \\
D^\wedge & = & \{+ : nat^\wedge, nat^\wedge \rightarrow nat^\wedge,\ * : nat^\wedge, nat^\wedge \rightarrow nat^\wedge,\ g : nat^\wedge, nat^\wedge \rightarrow nat^\wedge\} \\
D_{sort} & = & \{nat \lhd nat^\wedge\} \\
D & = & D_0 \cup D^\wedge \cup D_{sort}.
\end{array}
$$

For the rest of this paper let $\Sigma = \Sigma_0 \oplus \Sigma_1$. To avoid the problems occuring when empty sorts are present (see [Wa92]) we assume throughout the paper that for any sort $s_0 \in S_0$ there exists a constant $c \in F_0$ such that $c :\rightarrow s_0 \in D_0$.

Let $V = \bigcup_{s \in S_0} V_s$ be the union of disjoint infinitary sets V_s of variables for the sorts of type "low". We will not introduce variables for the sorts of type "high" as we do not need them. Σ-terms are now defined as usual.

Definition 2.4 *The set $TERM_s(\Sigma, V)$ of Σ-terms of sort s is the least set with the following properties:*

- *If $f :\rightarrow s_0 \in D$ and $s_0 \trianglelefteq s$, then $f \in TERM_s(\Sigma, V)$.*
- *If $x \in V_{s_0}$ and $s_0 \trianglelefteq s$, then $x \in TERM_s(\Sigma, V)$.*
- *If $f : s_1, \ldots, s_n \rightarrow s_0 \in D$, $s_0 \trianglelefteq s$ and $t_i \in TERM_{s_i}(\Sigma, V)$ $(i = 1, \ldots, n)$, then $f(t_1, \ldots, t_n) \in TERM_s(\Sigma, V)$.*

Let $TERM(\Sigma, V) = \bigcup_{s \in S} TERM_s(\Sigma, V)$.

The Σ-terms s and t are said to be sort compatible *iff there exists a sort $s \in S$ such that s and t are terms of sort s.*

By the construction of Σ and the restriction, that variables exist only for sorts of type "low", every (flat) $\Sigma_0 + \Sigma_1$-term can be interpreted as a (order-sorted) Σ-term (and vice versa).

As a consequence of the well-behaved sort hierarchy, every Σ-term t can be equipped with a uniquely defined sort — written $sort(t)$ — which is the wrt. $\lhd$ minimal sort s such that $t \in TERM_s(\Sigma, V)$. It is easily verified that if t contains a new symbol from F_1, then $sort(t) \in S^\wedge$, otherwise $sort(t) \in S_0$. Let $TERM_0(\Sigma, V)$ resp. $TERM^\wedge(\Sigma, V)$ denote the sets of Σ-terms t such that $sort(t) \in S_0$ resp. $sort(t) \in S^\wedge$. Notice that $TERM_0(\Sigma, V) = TERM(\Sigma_0, V)$.

Substitutions are defined in a way so that they respect the sort hierarchies:

Definition 2.5 *A (Σ-)substitution σ is an assignment from the set of variables V into the set of Σ-terms such that $sort(\sigma(x)) = sort(x)$ and $DOM(\sigma) = \{x \in V \mid \sigma(x) \not\equiv x\}$ is finite.*

We finish this section with the introduction of the fundamental notion of a specification with built-in algebra. The built-in objects and operations are described by a (flat) Σ_0-algebra $\mathcal{A}$. We assume throughout the paper that the built-in algebra $\mathcal{A}$ is *term-generated* (i.e. any element of the carrier A of $\mathcal{A}$ is the value of a Σ_0-term). Sometimes it is convenient to assume that for any element $a \in A$ there exists a constant $\underline{a} \in F_0$ to denote a.

Notice that it is possible to abstract by parameterization from the special built-in algebra. In that case a general built-in theory instead of a built-in algebra would be given in advance.

Definition 2.6 *A* conditional equation over Σ *is a formula* $\bigwedge_{i=1}^{n} u_i = v_i \Rightarrow u = v$ *such that u_i and v_i resp. u and v are sort compatible Σ-terms.*

Definition 2.7 *A* specification with built-in algebra *$(\Sigma = \Sigma_0 \oplus \Sigma_1,\ E\ ,\ \mathcal{A})$ consists of a hierarchical signature Σ induced by a (flat) signature enrichment $\Sigma_0 + \Sigma_1$, a set E of conditional equations over Σ and a (built-in) Σ_0-algebra $\mathcal{A}$.*

3 Semantics

We consider a specification $(\Sigma = \Sigma_0 \oplus \Sigma_1\ ,\ E\ ,\ \mathcal{A})$ with built-in algebra $\mathcal{A}$. The meaning of the specification will be first characterized by model-theoretical means.

The algebras of interest are intended to contain in a certain sense the built-in algebra $\mathcal{A}$ and to satisfy in addition the conditional equations from E. As a consequence of our order-sorted approach we have to consider order-sorted Σ-algebras.

Definition 3.1 *Let $\Sigma = \Sigma_0 \oplus \Sigma_1$. A Σ-algebra $\mathcal{B}$ consists of a family $\{B_s \mid s \in S\}$ of sets and functions $f^{\mathcal{B}}$ for any $f \in F$ such that*

- $B_s \subseteq B_{s^\wedge}$ *and*
- *if $f \in F_0^{(=0)}$ and $f :\rightarrow s \in D_0$, then $f^{\mathcal{B}} \in B_s$,*
  if $f \in F_0^{(\geq 1)}$ and $f : s_1, \ldots, s_n \rightarrow s \in D_0$ then $f^{\mathcal{B}} : B_{s_1^\wedge} \times \cdots \times B_{s_n^\wedge} \rightarrow B_{s^\wedge}$ such that $f^{\mathcal{B}}(b_1, \ldots, b_n) \in B_s$ for all $b_i \in B_{s_i}$,
  if $f \in F_1$ and $f : s_1, \ldots, s_n \rightarrow s \in D_1$ then $f^{\mathcal{B}} : B_{s_1^\wedge} \times \cdots \times B_{s_n^\wedge} \rightarrow B_{s^\wedge}$.

Thus, a Σ-algebra is a non-overloaded algebra in the sense of [Wa92]. Note that $B_s \neq \emptyset$ for any $s \in S$ by our assumption that sorts are not empty.

To take into account the built-in algebra $\mathcal{A}$ let $E_{\mathcal{A}} = \{u = v \mid \mathcal{A} \models u = v;\ u, v \in TERM_0(\Sigma, V)\}$ be the *set of (Σ_0-)equations induced by $\mathcal{A}$*.

As a consequence of the assumption that $\mathcal{A}$ is term-generated each model $\mathcal{B}$ of $E_{\mathcal{A}}$ has a "core" — constituted by the "base elements of type low" — that contains a uniquely defined homomorphic image of the built-in algebra $\mathcal{A}$.

The algebras that capture the model-theoretical meaning of the specification are the (order-sorted) Σ-algebras that are models of $E \cup E_{\mathcal{A}}$. Thereby, a Σ-algebra is a model of an equation $u = v$ iff it satisfies the equation wrt. any assignment that respects the sort hierarchy. Thus, by the variable restriction one only has to consider assignments that (correctly) instantiate variables by "base elements of type low".

The operational definition of the semantics of the specification uses the following inference rules depending on the set E of conditional equations and an additional set S of Σ-equations (see [Ga91], [Wa92] for similar approaches). The introduction of variable sets (see e.g. [Wa92]) can be dropped as we assume that there exist no empty sorts.

(Reflexivity)

$$\frac{}{u = u}$$

(Symmetry)

$$\frac{u = v}{v = u}$$

(Transitivity)

$$\frac{u = v,\ v = w}{u = w}$$

(Congruence)

$$\frac{u_1 = v_1, \ldots, u_n = v_n}{f(u_1, \ldots, u_n) = f(v_1, \ldots, v_n)}$$

if $f(u_1, \ldots, u_n)$ *and* $f(v_1, \ldots, v_n)$ *are both well – formed* Σ *– terms.*

(Substitutivity)

$$\frac{\sigma(u_1) = \sigma(v_1), \ldots, \sigma(u_n) = \sigma(v_n)}{\sigma(u) = \sigma(v)}$$

if σ *is a substitution,* $\bigwedge_{i=1}^{n} u_i = v_i \Rightarrow u = v \in E$ *and* $\sigma(u_i) = \sigma(v_i) \in S$ $(i = 1, \ldots, n)$.

Note that by syntactical means the applicability of the inference rule "substitutivity" is limited, as variables can only be substituted by Σ-terms of type "low".

We write $S \vdash_E u = v$ to indicate that $u = v$ can be derived from the set S of Σ-equations by the above inference rules (depending on E). One easily proves that the inference rules are sound (using the fact that there exist no empty sorts):

Lemma 3.1 *Let* S *be a set of* Σ*-equations. Then for all* $u, v \in TERM(\Sigma, V)$*, if* $S \vdash_E u = v$*, then* $S \cup E \models u = v$.

Next we define the operational semantics of the specification with the aid of an appropriate congruence relation. We start with a relation that is induced by the built-in algebra $\mathcal{A}$.

Definition 3.2 *For* $s, t \in TERM(\Sigma, V)$ *let* $s \sim_{\mathcal{A}} t$ *iff*
(a) $s, t \in TERM_0(\Sigma, V)$ *and* $\mathcal{A} \models s = t$ *resp.* $s = t \in E_{\mathcal{A}}$ *or*
(b) $s, t \in TERM^{\wedge}(\Sigma, V)$, $s \equiv f(s_1, \ldots, s_n)$, $t \equiv f(t_1, \ldots, t_n)$ *and* $s_i \sim_{\mathcal{A}} t_i$ *for* $i = 1, \ldots, n$.

One easily proves that $\sim_{\mathcal{A}}$ is a congruence relation on $TERM(\Sigma, V)$, the *congruence relation induced by* $\mathcal{A}$. According to the intuition that built-in equivalences are given in advance, the inductive definition to follow starts with the congruence relation $\sim_{\mathcal{A}}$.

Definition 3.3 *Let:*

- $\sim^0_{E,\mathcal{A}} = \sim_{\mathcal{A}}$.
- *For all* $u, v \in TERM(\Sigma, V)$: $u \sim^{i+1}_{E,\mathcal{A}} v$ *iff* $u \sim^i_{E,\mathcal{A}} v$*, or there exists a set* S *of* Σ*-equations such that* $s \sim^i_{E,\mathcal{A}} t$ *for all* $s = t \in S$ *and* $S \vdash_E u = v$.
- $\sim_{E,\mathcal{A}} = \bigcup_{i \geq 0} \sim^i_{E,\mathcal{A}}$.

Obviously, $\sim_{E,\mathcal{A}}$ (as well as $\sim^i_{E,\mathcal{A}}$) is a congruence relation on $TERM(\Sigma, V)$, the *congruence relation induced by* E *and* $\mathcal{A}$.

The following "Birkhoff-theorem" states the equivalence between the model-theoretical and operational semantics of the specification.

Theorem 3.1 *Let* $(\Sigma = \Sigma_0 \oplus \Sigma_1 \ , \ E \ , \ \mathcal{A})$ *be a specification with built-in algebra* $\mathcal{A}$*. Then for any* $s, t \in TERM(\Sigma, V)$ *we have* $s \sim_{E,\mathcal{A}} t$ *iff* $E \cup E_{\mathcal{A}} \models s = t$.

Proof:
(a) For the direction from left to right we prove by induction on i: For any $s,t \in TERM(\Sigma,V)$, if $s \sim^i_{E,\mathcal{A}} t$, then $E \cup E_{\mathcal{A}} \models s = t$.

The induction base $i = 0$ follows from the (easy to prove) fact that $s \sim_{\mathcal{A}} t$ implies $E_{\mathcal{A}} \models s = t$ for all $s,t \in TERM(\Sigma,V)$.

For the induction step $i \mapsto i+1$ let $s \sim^{i+1}_{E,\mathcal{A}} t$ for $s,t \in TERM(\Sigma,V)$. The case $s \sim^{i}_{E,\mathcal{A}} t$ is covered by the induction hypothesis. Thus, let S be given such that $s' \sim^i_{E,\mathcal{A}} t'$ for all $s' = t' \in S$ and $S \vdash_E s = t$. The induction hypothesis provides $E \cup E_{\mathcal{A}} \models S$. By the soundness of the inference system we get $S \cup E \models s = t$. Consequently $E \cup E_{\mathcal{A}} \models s = t$.
(b) For the direction from right to left let $E \cup E_{\mathcal{A}} \models s = t$. Let $\mathcal{T}_{(E,\mathcal{A})}$ be the canonical term algebra with carrier $TERM(\Sigma,V)_{/\sim_{E,\mathcal{A}}}$, defined as usual. It suffices to prove that $\mathcal{T}_{(E,\mathcal{A})}$ is a model of $E \cup E_{\mathcal{A}}$ as then $\mathcal{T}_{(E,\mathcal{A})}$ is a model of $s = t$ which is equivalent to $s \sim_{E,\mathcal{A}} t$.

Let first $u = v \in E_{\mathcal{A}}$. In order to prove that $\mathcal{T}_{(E,\mathcal{A})} \models u = v$ we have to show that $\sigma(u) \sim_{E,\mathcal{A}} \sigma(v)$ for any substitution σ. Let σ be a substitution. As a consequence of our variable restriction we have $\sigma(u) = \sigma(v) \in E_{\mathcal{A}}$. Thus $\sigma(u) \sim^0_{E,\mathcal{A}} \sigma(v)$ resp. $\sigma(u) \sim_{E,\mathcal{A}} \sigma(v)$.

Now let $\bigwedge u_i = v_i \Rightarrow u = v \in E$ and let σ be a substitution such that $\mathcal{T}_{(E,\mathcal{A})} \models \sigma(u_i) = \sigma(v_i)$ resp. $\sigma(u_i) \sim_{E,\mathcal{A}} \sigma(v_i)$ for $i = 1,\ldots,n$. For an appropriate k we have $\sigma(u_i) \sim^k_{E,\mathcal{A}} \sigma(v_i)$ $(i = 1,\ldots,n)$. We get $\sigma(u) \sim^{k+1}_{E,\mathcal{A}} \sigma(v)$ by applying the related inference rule. Thus $\sigma(u) \sim_{E,\mathcal{A}} \sigma(v)$ resp. $\mathcal{T}_{(E,\mathcal{A})} \models \sigma(u) = \sigma(v)$. As σ was an arbitrary substitution, $\mathcal{T}_{(E,\mathcal{A})} \models \bigwedge u_i = v_i \Rightarrow u = v$.
□

4 Rewriting

When conditional equations are interpreted as rewrite rules, "replacement of equals by equals" is directionally limited "from left to right".

Definition 4.1 *A* conditional rewrite rule over Σ *is a (directed) conditional equation*

$$\bigwedge_{i=1}^{n} u_i = v_i \Rightarrow u = v,$$

where the left hand side u is an element of $TERM^{\wedge}(\Sigma,V)$ and all variables occuring in v, u_i, v_i also occur in u.

The extra condition $u \in TERM^{\wedge}(\Sigma,V)$, meaning that the left hand side of a rewrite rule has to contain a new symbol, is motivated by the aim that the built-in structure of the terms of a "low" sort should not be destroyed. Thus, if we interprete the terms of the "low" sorts as constructor terms, then this extra condition is a kind of constructor preserving property. In addition this condition assures that rewrite rules are always sort-decreasing, i.e. $sort(u) \trianglerighteq sort(v)$ (see [Wa92] or [SNGM89] for the relevance of this property).

Next we define conditional rewriting modulo the built-in algebra $\mathcal{A}$, which is essentially conditional rewriting of $\sim_{\mathcal{A}}$-equivalence classes.

Definition 4.2 *Let R be a set of conditional rewrite rules over Σ. A term $s \in TERM(\Sigma,V)$* rewrites modulo $\mathcal{A}$ to $t \in TERM(\Sigma,V)$ — *written* $s \longrightarrow_{R/\mathcal{A}} t$ — *iff there exist terms $s', t' \in TERM(\Sigma,V)$, an occurence $p \in O(s')$, a substitution σ and a rule $\bigwedge_{i=1}^{n} u_i = v_i \Rightarrow u = v \in R$ such that*

- $s \sim_{\mathcal{A}} s'$, $s'/p \equiv \sigma(u)$, $s'[p \leftarrow \sigma(v)] \equiv t'$, $t' \sim_{\mathcal{A}} t$ *and*
- *for any* $i \in \{1, \ldots, n\}$ *there exist* $u_i', v_i' \in TERM(\Sigma, V)$ *such that* $\sigma(u_i) \xrightarrow{*}_{R/\mathcal{A}} u_i' \sim_{\mathcal{A}} v_i' \xleftarrow{*}_{R/\mathcal{A}} \sigma(v_i)$.

The least fixpoint of this recursion defines the relation $\longrightarrow_{R/\mathcal{A}}$. *Thus:*

- $\longrightarrow^0_{R/\mathcal{A}} = \emptyset$
- $\longrightarrow^{i+1}_{R/\mathcal{A}}$ *is defined as above exept that the rewrite proofs for the conditions can be carried out in* $\bigcup_{j \leq i} \longrightarrow^j_{R/\mathcal{A}}$.
- $\longrightarrow_{R/\mathcal{A}} = \bigcup_{i \geq 0} \longrightarrow^i_{R/\mathcal{A}}$

Note that the syntactical variable restriction forces an innermost reduction strategy.

Example 4.1 *We continue our standard example.*
The chain $g(g(2,1),0) \longrightarrow_{R/\mathcal{N}} g(g(1,1),0) \longrightarrow_{R/\mathcal{N}} g(g(1,0),0) \longrightarrow_{R/\mathcal{N}} g(1,0) \longrightarrow_{R/\mathcal{N}} 1$ *is correct, whereas* $g(g(2,1),0) \longrightarrow_{R/\mathcal{N}} g(2,1)$ *is not possible.*

As in the case of rewriting modulo unorientable equations (see [BaDe89], [JoKi86]) we could switch over to a weaker rewrite relation $\longrightarrow_{R\setminus\mathcal{A}}$, defined as least fixpoint according to the following recursion:

Definition 4.3 $s \longrightarrow_{R\setminus\mathcal{A}} t$ *iff there exists an occurence* $p \in O(s)$, *a substitution* σ *and a rule* $\bigwedge_{i=1}^{n} u_i = v_i \Rightarrow u = v \in R$ *such that*

- $s/p \sim_{\mathcal{A}} \sigma(u)$, $s[p \leftarrow \sigma(v)] \equiv t$ *and*
- *for any* $i \in \{1, \ldots, n\}$ *there exist* $u_i', v_i' \in TERM(\Sigma, V)$ *such that* $\sigma(u_i) \xrightarrow{*}_{R\setminus\mathcal{A}} u_i' \sim_{\mathcal{A}} v_i' \xleftarrow{*}_{R\setminus\mathcal{A}} \sigma(v_i)$.

But this change would not have any influence on the results to come because the two rewrite relations do not differ very much in our context. To prove this we first state some technical results.

Definition 4.4 *Let* $O^\wedge(t) = \{p \in O(t) \mid sort(t/p) \in S^\wedge\}$ *be the set of* syntactical positions of $t \in TERM(\Sigma, V)$.

Lemma 4.1 *Let* $s, t, u \in TERM(\Sigma, V)$.
(a) If $s \sim_{\mathcal{A}} t$ *and* $p \in O^\wedge(s)$, *then* $p \in O^\wedge(t)$. *Further* $s/p \sim_{\mathcal{A}} t/p$ *and* $s[p \leftarrow u] \sim_{\mathcal{A}} t[p \leftarrow u]$ *if* $s[p \leftarrow u]$ *and* $t[p \leftarrow u]$ *are correct* Σ*-terms.*
(b) Let σ *be a substitution. If* $p \in O^\wedge(\sigma(s))$, *then* $p \in O^\wedge(s)$ *and* $\sigma(s/p) \equiv \sigma(s)/p$.

Lemma 4.2 *Let* R *be a set of conditional rewrite rules over* Σ. *Let* $s, t \in TERM(\Sigma, V)$. *Then* $s \xrightarrow{*}_{R/\mathcal{A}} t$ *iff there exists a term* $w \in TERM(\Sigma, V)$ *such that* $s \xrightarrow{*}_{R\setminus\mathcal{A}} w \sim_{\mathcal{A}} t$.

Proof: The direction from right to left is easy.

For the direction from left to right we prove by induction on j: If $s \xrightarrow{*}{}^{j}_{R/A} t$, then there exists a term w such that $s \xrightarrow{*}{}^{j}_{R\setminus A} w \sim_A t$.

The induction base $j = 0$ is trivial. For the induction step we first consider the one step case $s \longrightarrow^{j+1}_{R/A} t$. Let $s', t', p, \sigma, \bigwedge u_i = v_i \Rightarrow u = v, u_i', v_i'$ be given as indicated in the definition. By lemma 4.1 we get $s/p \sim_A \sigma(u)$ and $w \equiv s[p \leftarrow \sigma(v)] \sim_A s'[p \leftarrow \sigma(v)] \equiv t' \sim_A t$. By the induction hypothesis there exist $u_i'', v_i'' \in TERM(\Sigma, V)$ such that $\sigma(u_i) \xrightarrow{*}_{R\setminus A} u_i'' \sim_A u_i' \sim_A v_i' \sim_A v_i''$ ${}_{R\setminus A}\xleftarrow{*}\ \sigma(v_i)$. The iterated case $s \xrightarrow{*}{}^{j+1}_{R/A} t$ can be proved easily by an induction argument. □

Rewriting is of great interest if every equational proof can be replaced by a rewrite proof. We review some notions to capture the main ideas.

Definition 4.5 *Let R be a set of conditional rewrite rules over Σ.*
(a) A Σ-equation $s = t$ is joinable modulo $\mathcal{A}$, *written $s \downarrow_{R/A} t$, iff there exist $s', t' \in TERM(\Sigma, V)$ such that $s \xrightarrow{*}_{R/A} s' \sim_A t'$ ${}_{R/A}\xleftarrow{*}\ t$. A set S of equations is* joinable modulo $\mathcal{A}$ *iff any equation from S is joinable modulo $\mathcal{A}$.*
(b) R is said to be Church-Rosser modulo $\mathcal{A}$ *iff for any $s, t \in TERM(\Sigma, V)$:*
$s \sim_{R,A} t$ iff $s \downarrow_{R/A} t$.
(c) R is said to be confluent modulo $\mathcal{A}$ (resp. locally confluent modulo $\mathcal{A}$) *iff for any $s, s_1, s_2 \in TERM(\Sigma, V)$: if s_1 ${}_{R/A}\xleftarrow{*}\ s \xrightarrow{*}_{R/A} s_2$ (resp. s_1 ${}_{R/A}\xleftarrow{}\ s \longrightarrow_{R/A} s_2$), then $s_1 \downarrow_{R/A} s_2$.*

The following theorems are generalizations of some classical results in rewrite theory. The first one slightly generalizes the Newman-lemma.

Theorem 4.1 *Let R be a conditional rewrite system over Σ. Let $\longrightarrow_{R/A}$ be terminating. Then R is confluent modulo $\mathcal{A}$ iff R is locally confluent modulo $\mathcal{A}$.*

The proof is omited as it proceeds just analogously to the standard case.

Theorem 4.2 *Let R be a conditional rewrite system over Σ. Then R is Church-Rosser modulo $\mathcal{A}$ iff R is confluent modulo $\mathcal{A}$.*

Proof: For the direction from left to right let R be Church-Rosser modulo $\mathcal{A}$. Further let s_1 ${}_{R/A}\xleftarrow{*}\ s \xrightarrow{*}_{R/A} s_2$. One easily proves (by induction on i) that for any $u, v \in TERM(\Sigma, V)$, $u \longrightarrow^{i}_{R/A} v$ implies $u \sim_{R,A} v$. Thus we get $s_1 \sim_{R,A} s_2$. The assumption then provides $s_1 \downarrow_{R/A} s_2$.

For the direction from right to left let R be confluent modulo $\mathcal{A}$. We prove by induction on i: if $u \sim^{i}_{R,A} v$, then $u \downarrow_{R/A} v$.

The induction base is trivial. For the induction step let $u \sim^{i+1}_{R,A} v$. The case $u \sim^{i}_{R,A} v$ is covered by the induction hypothesis. Thus let S be given such that $s' \sim^{i}_{R,A} t'$ for all $s' = t' \in S$ and $S \vdash_R u = v$. By the induction hypothesis S is joinable modulo $\mathcal{A}$. We are finished if we have proved the following statement: If $S \vdash_R u = v$ and if S is joinable modulo $\mathcal{A}$, then $u = v$ is joinable modulo $\mathcal{A}$ too.

We first consider a one-step derivation. The general multi-step case is then easily shown by induction on the lenght of the derivation.

Let $S \vdash_R u = v$ by a one-step derivation and let S be joinable modulo $\mathcal{A}$. We proceed by considering the diverse inference rules. The cases "reflexivity", "symmetry" and "congruence"

are trivial. For the case "transitivity" we use the assumption that R is confluent. Finally the "substitutivity" case follows from the definition of $\longrightarrow_{R/A}$. □

We finish this section with some (easy to prove) results that will be needed below.

Lemma 4.3 *Let s, s', t, t', t_0 be Σ-terms and σ be a substitution. Then:*
(a) If $s \xrightarrow{*}_{R/A} t$, then $\sigma(s) \xrightarrow{*}_{R/A} \sigma(t)$ and $t_0[p \leftarrow s] \xrightarrow{*}_{R/A} t_0[p \leftarrow t]$ (provided that $p \in O(t_0)$, $t_0[p \leftarrow s], t_0[p \leftarrow t] \in TERM(\Sigma, V)$).
(b) If $s \downarrow_{R/A} t$, then $\sigma(s) \downarrow_{R/A} \sigma(t)$ and $t_0[p \leftarrow s] \downarrow_{R/A} t_0[p \leftarrow t]$ (provided that $p \in O(t_0)$, $t_0[p \leftarrow s], t_0[p \leftarrow t] \in TERM(\Sigma, V)$). If in addition $s' \sim_A s$ and $t \sim_A t'$, then $s' \downarrow_{R/A} t'$.

5 Critical Pair Test

In order to get a critical pair test we first generalize unification. The problem consists in determining the solutions of a Σ-equation in the presence of predefined equivalences.

Definition 5.1 *A substitution σ* satisfies a set S of Σ-equations modulo $\mathcal{A}$ resp. is an $\mathcal{A}$-solution of S *iff $\sigma(s) \sim_{\mathcal{A}} \sigma(t)$ for all $s = t \in S$.*

The aim is to determine a finite representation of all possible $\mathcal{A}$-solutions of a Σ-equation resp. a set of Σ-equations. From the theory of semantic unification (for a survey see [Si89]) we know that we cannot expect to find such a finite representation for any algebra in the form of a finite set of substitutions.

Example 5.1 *We continue the standard example. Let $\mathcal{N}$ denote the Σ_{nat}-algebra with the canonical interptetation of the symbols over the natural numbers. The equation $x + y = x' * y'$ has infinitely many $\mathcal{N}$-solutions, but no finite representation by substitutions.*

The way out of this situation is to represent the solutions not explicitly by substitutions but implicitly in the form of a constraint (see [JaLa87b]). The next definition makes precise the notion of constraint used in our context.

Definition 5.2 *A* constraint *is either a finite conjunction of Σ_0-equations or an element of $\{\top, \bot\}$.*

Constraints will be denoted by the symbols α, β, γ (possibly with an index). We identify finite sets of Σ_0-equations with the according conjunctions so that finite sets of Σ_0-equations represent constraints. Further we assume that $\top$ is true and $\bot$ is false in any Σ_0-algebra $\mathcal{A}$. Note that the equation in the example above is itself the solution representation.

Equation solving proceeds in two steps, a first conceptual one and a second one that is of more practical interest.

In the first step a constraint representation of the $\mathcal{A}$-solutions of an equation (a set of equations) is determined by mere syntactical means. In the second step this constraint may be "propagated" by a semantical built-in algorithm to yield a more explicit solution representation.

We first present an inference system $\mathcal{I}$ that allows to determine a constraint representation γ_S of the $\mathcal{A}$-solutions of an arbitrary finite set S of Σ-equations. The inference rules are designed to trace back the recursive definition of the relation $\sim_{\mathcal{A}}$.

(hierarchy fail)

$$\frac{S \cup \{s = t\}}{\{\bot\}}$$

if $sort(s) \lhd sort(t)$ *or* $sort(t) \lhd sort(s)$.

(syntactical fail)

$$\frac{S \cup \{f(s_1, \ldots, s_m) = g(t_1, \ldots, t_n)\}}{\{\bot\}}$$

if $sort(f(s_1, \ldots, s_m)) = sort(g(t_1, \ldots, t_n)) \in S^\wedge$ *and* $f \neq g$.

(decomposition)

$$\frac{S \cup \{f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)\}}{S \cup \{s_1 = t_1, \ldots, s_n = t_n\}}$$

if $sort(f(s_1, \ldots, s_n)) = sort(f(t_1, \ldots, t_n)) \in S^\wedge$.

Let $\vdash_{\mathcal{I}}$ denote the derivability relation induced by the inference system. The following properties of $\vdash_{\mathcal{I}}$ are easy to prove:

- $\vdash_{\mathcal{I}}$ is terminating.
- If a new symbol from F_1 is still present in some equation, then some inference rule is appliable.
- No inference rule is appliable if S is a constraint, i.e. if S contains only predefined symbols.
- $\vdash_{\mathcal{I}}$ is correct, i.e.: If $S \vdash_{\mathcal{I}} S'$, then for all substitutions σ, σ satisfies S modulo $\mathcal{A}$ iff σ satisfies S' modulo $\mathcal{A}$.

Example 5.2 *We continue our standard example.*
Consider the Σ*-equation* $g(x, 0) = g(x', x' + y')$. *The constraint-representation of the* $\mathcal{A}$*-solutions is* $\gamma = \{x = x', 0 = x' + y'\}$.

We now turn to a critical pair test for local confluence.

Definition 5.3 *Let* $U \Rightarrow u = v$ *and* $U' \Rightarrow u' = v'$ *be two conditional rules over* Σ *that have no variables in common. Then the conditional equation over* Σ

$$\gamma_{u/p=u'} \wedge U \wedge U' \Rightarrow u[p \leftarrow v'] = v$$

is called a (conditional) critical pair between the two rules.

In contrast to the ordinary syntactical case we do not require u/p to be no variable. There is no need for such a condition in our context, as variables are of type "low" and the left hand sides of the rules have to be of type "high". An equation with the types of the two sides being different has no solution in our context. Note that by these restrictions we do not have to consider variable overlaps in the proof of the theorem to come. Hence we do not have to demand for a termination property for R — which is necessary in the syntactical case (see [DeOk90]).

Definition 5.4 *A conditional equation $S \Rightarrow s = t$ over Σ is said to be* joinable modulo $\mathcal{A}$ *iff for any substitution σ, if $\sigma(S)$ is joinable modulo $\mathcal{A}$, then $\sigma(s) = \sigma(t)$ is joinable modulo $\mathcal{A}$.*

Theorem 5.1 *Let R be a conditional rewrite system over Σ. If all conditional critical pairs that can be built from the rules of R are joinable modulo $\mathcal{A}$, then R is locally confluent modulo $\mathcal{A}$.*

Proof: Let $t, t_1, t_2 \in TERM(\Sigma, V)$ be such that $t_1 \ {}_{R/\mathcal{A}}\!\longleftarrow t \longrightarrow_{R/\mathcal{A}} t_2$ with the rules $U \Rightarrow u = v$ and $U' \Rightarrow u' = v'$. Hence for appropriate $s, s', s_1, s_2 \in TERM(\Sigma, V)$, positions p, q and substitution τ:

- $t \sim_{\mathcal{A}} s$, $s/p \equiv \tau(u)$, $s[p \leftarrow \tau(v)] \equiv s_1$, $s_1 \sim_{\mathcal{A}} t_1$, $\tau(U)$ is joinable modulo $\mathcal{A}$.
- $t \sim_{\mathcal{A}} s'$, $s'/q \equiv \tau(u')$, $s'[q \leftarrow \tau(v')] \equiv s_2$, $s_2 \sim_{\mathcal{A}} t_2$, $\tau(U')$ is joinable modulo $\mathcal{A}$.

As $u, u' \in TERM^{\wedge}(\Sigma, V)$ we have $p \in O^{\wedge}(s)$ and $q \in O^{\wedge}(s')$. By lemma 4.1 we get $p, q \in O^{\wedge}(t)$. Then either $p \mid q$ or $q = pq'$ or $p = qp'$ for appropriate p', q'.
Case 1: $p \mid q$. We have $q \in O^{\wedge}(s)$ and $s/q \sim_{\mathcal{A}} t/q \sim_{\mathcal{A}} s'/q$. Further $q \in O^{\wedge}(s[p \leftarrow \tau(v)])$. Hence $s[p \leftarrow \tau(v)]/q \equiv s/q \sim_{\mathcal{A}} s'/q \equiv \tau(u')$. Analogously $s'[q \leftarrow \tau(v')]/p \equiv s'/p \sim_{\mathcal{A}} s/p \equiv \tau(u)$. Let $w_1 \equiv s[p \leftarrow \tau(v)][q \leftarrow \tau(v')]$ and $w_2 \equiv s'[p \leftarrow \tau(v)][q \leftarrow \tau(v')]$. Then:
$t_1 \longrightarrow_{R/\mathcal{A}} w_1 \sim_{\mathcal{A}} w_2 \ {}_{R/\mathcal{A}}\!\longleftarrow t_2$.
Case 2: $q = pq'$. (The case $p = qp'$ proceeds analogously.) We have $q = pq' \in O^{\wedge}(s')$, hence $pq' \in O^{\wedge}(s)$, hence $q' \in O^{\wedge}(s/p)$, hence $q' \in O^{\wedge}(\tau(u))$, hence $q' \in O^{\wedge}(u)$ (by use of lemma 4.1). Then: $\tau(u/q') \equiv \tau(u)/q' \equiv s/pq' \sim_{\mathcal{A}} t/pq' \equiv t/q \sim_{\mathcal{A}} s'/q \equiv \tau(u')$. Thus τ satisfies $u/q' = u'$ resp. $\gamma_{u/q'=u'}$ modulo $\mathcal{A}$. It follows that $\tau(\gamma_{u/q'=u'}) \wedge \tau(U) \wedge \tau(U')$ is joinable modulo $\mathcal{A}$.

As the critical pair $\gamma_{u/q'=u'} \wedge U \wedge U' \Rightarrow u[q' \leftarrow v'] = v$ is joinable modulo $\mathcal{A}$ by assumption we get $\tau(u)[q' \leftarrow \tau(v')] \downarrow_{R/\mathcal{A}} \tau(v)$. Lemma 4.3 then provides $t_1 \downarrow_{R/\mathcal{A}} t_2$. □

The second step in the equation solving process uses a built-in constraint propagation algorithm to make the solution representation more explicit. We are here interested only in the results that such a propagation algorithm may output and not in the details concerning the algorithm itself.

To describe solutions that are partially explicit and partially implicit the notion of a constrained substitution is introduced.

Definition 5.5 *A* constrained substitution *(γ, σ) is a pair consisting of a constraint γ and a substitution σ.*

In practice σ will be idempotent. Further no variable occuring in γ will belong to the domain of σ. As usual $\tau \mid_X$ denotes the substitution that has the same values as τ on the set of variables X and that is the identical mapping on $V \backslash X$.

Definition 5.6 *Let γ be a constraint and X be a (finite) set of variables such that $VAR(\gamma) \subseteq X$. The set $\{(\gamma_1, \sigma_1), \ldots, (\gamma_k, \sigma_k)\}$ of constrained substitutions is said to be a* partially solved representation of γ wrt X (modulo $\mathcal{A}$) *iff the following items hold:*
(a) For any $i \in \{1, \ldots, k\}$ and any substitution μ, if μ is an $\mathcal{A}$-solution of γ_i, then $\mu \circ \sigma_i$ is an $\mathcal{A}$-solution of γ.
(b) For any substitution τ, if τ is an $\mathcal{A}$-solution of γ, then there exists an $i \in \{1, \ldots, k\}$ and a substitution μ such that μ is an $\mathcal{A}$-solution of γ_i and $\tau \mid_X \sim_{\mathcal{A}} \mu \circ \sigma_i \mid_X$.

A constraint propagation algorithm is intended to receive a constraint γ as input and to produce a partially solved representation of γ as output. The next lemma states without proof (that is straightforward) that constraint propagation does not effect the joinability of conditional equations.

Lemma 5.1 *Let $\gamma \wedge U \Rightarrow s = t$ be a conditional equation. Let X be the set of variables occuring in $\gamma \wedge U \Rightarrow s = t$. Let $\{(\gamma_1, \sigma_1), \ldots, (\gamma_k, \sigma_k)\}$ be a partially solved representation of γ wrt X (modulo $\mathcal{A}$). Then $\gamma \wedge U \Rightarrow s = t$ is joinable modulo $\mathcal{A}$ iff $\{\gamma_i \wedge \sigma_i(U) \Rightarrow \sigma_i(s = t) \mid i = 1, \ldots, k\}$ is joinable modulo $\mathcal{A}$.*

Thus we may propagate the constraint part of a critical pair before testing for joinability. Note that constraint propagation strongly depends on the built-in algebra $\mathcal{A}$.

Example 5.3 *We continue the standard example. Consider the rules from example 1.1. All critical pairs are joinable modulo $\mathcal{N}$. We consider only three cases.*

(a) overlap (1),(1)
equation to be soved: $g(x, 0) = g(x', 0)$
constraint solution: $\{x = x', 0 = 0\}$
propagated solution: $(\top, \{x \leftarrow x'\})$
propagated critical pair: $\top \Rightarrow x = x$

(b) overlap: (1),(4)
equation to be solved: $g(x, 0) = g(x', x' + y')$
constraint solution: $\{x = x', 0 = x' + y'\}$
propagated solution: $(\top, \{x \leftarrow 0, x' \leftarrow 0, y' \leftarrow 0\})$
propagated critical pair: $\top \Rightarrow 0 = g(0, 0)$
joining reductions: $g(0, 0) \longrightarrow_{R/\mathcal{N}} 0$

(c) overlap (3),(4)
equation to be solved: $g(x + y, y) = g(x', x' + y')$
constraint solution: $\{x + y = x', y = x' + y'\}$
propagated solution: $(\top, \{x \leftarrow 0, y \leftarrow x', y' \leftarrow 0\})$
propagated critical pair: $\top \Rightarrow g(0, x') = g(x', 0)$
joining reductions: $g(0, x') \longrightarrow_{R/\mathcal{N}} x'$ *and* $g(x', 0) \longrightarrow_{R/\mathcal{N}} x'$

Hence: R is locally confluent modulo the natural number interpretation $\mathcal{N}$.

6 Termination

In this section we investigate the termination of the rewrite relation $\longrightarrow_{R/\mathcal{A}}$. After developing some general results we consider recursive path orderings that integrate semantic information from the built-in algebra $\mathcal{A}$.

Example 6.1 *The equations from example 1.1 do not induce a terminating rewrite relation modulo the natural number interpretation $\mathcal{N}$, as $g(x, 0) \sim_{\mathcal{N}} g(x + 0, 0)$ and hence $g(x, 0) \longrightarrow_{R/\mathcal{N}} g(x, 0)$. We change the rewrite rules by adding "semantical information". Let boole be an additional built-in sort and let $\succ, true$ be additional built-in symbols which are interpretated in (an extended version of) $\mathcal{N}$ in the natural way. Let R now be the new rewrite system:*

$$
\begin{array}{llllllll}
(1) & & & & & g(x,0) & = & x \\
(2) & & & & & g(0,y) & = & y \\
(3) & y \succ 0 & = & true & \Rightarrow & g(x+y,y) & = & g(x,y) \\
(4) & x \succ 0 & = & true & \Rightarrow & g(x,x+y) & = & g(x,y)
\end{array}
$$

Now we may use the well-foundedness of the algebra $\mathcal{N}$ wrt $\succ$ and true and the fact that $\mathcal{N} \models y \succ 0 = true \Rightarrow x + y \succ x = true$ and $\mathcal{N} \models x \succ 0 = true \Rightarrow x + y \succ y = true$ to conclude that $\longrightarrow_{R/\mathcal{N}}$ is terminating.

First the notion of decreasingness (see [DeOk90]) is generalized in order to guarantee the well-foundedness of $\longrightarrow_{R/\mathcal{A}}$ and to avoid an infinite regress in the recursive condition check.

For reasons that will become clear below we split the condition U of a conditional rewrite rule into two parts, a constraint γ and an additional part C. Thus a conditional rule will be written in the form $\gamma \wedge C \Rightarrow u = v$.

Definition 6.1 *Let R be a conditional rewrite system over Σ. R is said to be* decreasing modulo *$\mathcal{A}$ iff there exists a well-founded extension $>$ of $\longrightarrow_{R/\mathcal{A}}$ that satisfies the following items:*

- *$>$ contains the subterm relation $>_{st/\mathcal{A}}$, where for $s,t \in TERM(\Sigma,V)$ we have: $s >_{st/\mathcal{A}} t$ iff there exist $s' \in TERM^{\wedge}(\Sigma,V)$ and $t' \in TERM(\Sigma,V)$ such that $s \sim_{\mathcal{A}} s' >_{st} t' \sim_{\mathcal{A}} t$ ($>_{st}$ denotes the normal proper subterm ordering).*
- *For any rule $\gamma \wedge C \Rightarrow u = v \in R$ and any substitution σ that satisfies γ modulo $\mathcal{A}$, $\sigma(u) > \sigma(u_i), \sigma(v_i)$ for all $u_i = v_i \in C$.*

As in the "normal syntactical case" one easily proves that there cannot exist any infinite descent when recursively checking conditions for rule application if the rewrite relation is decreasing modulo $\mathcal{A}$. Hence in that case the rewrite relation $\longrightarrow_{R/\mathcal{A}}$ is terminating and decidable (provided that $\mathcal{A}$-equivalence and $\mathcal{A}$-matching are decidable; see also section 7). In order to prove decreasingness modulo $\mathcal{A}$ we next introduce a suitable notion of reduction ordering that takes into account the signature hierarchy and the built-in algebra. Compatbility with such a reduction ordering that in addition satisfies a subterm property then provides the desired decreasingness.

Definition 6.2 *A partial ordering $>$ on $TERM(\Sigma,V)$ is said to be a* reduction ordering wrt *$(\Sigma = \Sigma_0 \oplus \Sigma_1, \mathcal{A})$ iff the following items hold:*

(a) $>$ is compatible with the built-in algebra $\mathcal{A}$: for any $s,t,s',t' \in TERM(\Sigma,V)$, if $s \sim_{\mathcal{A}} s' > t' \sim_{\mathcal{A}} t$, then $s > t$.

(b) $>$ is monotonic wrt syntactical replacement: for any $s,t \in TERM(\Sigma,V)$ and any symbol f, if $s > t$ and $f(\ldots,s,\ldots) \in TERM^{\wedge}(\Sigma,V)$, then $f(\ldots,s,\ldots) > f(\ldots,t,\ldots)$.

(c) $>$ is well-founded.

In the sequel we write $\gamma : u > v$ iff for any substitution σ, if σ satisfies γ modulo $\mathcal{A}$, then $\sigma(u) > \sigma(v)$.

Definition 6.3 *Let R be a conditional rewrite system. A partial ordering $>$ is* compatible with *R iff for any $\gamma \wedge C \Rightarrow u = v \in R$ we have $\gamma : u > v, u_i, v_i$, where $u_i = v_i \in C$.*

Note that this kind of compatibility requires a kind of stability too.

Lemma 6.1 *Let $>$ be a reduction ordering wrt $(\Sigma = \Sigma_0 \oplus \Sigma_1, \mathcal{A})$ that is compatible with the conditional rewrite system R. Let further $>$ contain $>_{st/\mathcal{A}}$. Then R is decreasing modulo $\mathcal{A}$.*

Proof: First, $>$ is well-founded by definition. Next we prove that $\longrightarrow_{R/\mathcal{A}} \subseteq >$.

Let $s \longrightarrow_{R/\mathcal{A}} t$ with $s', t', p, \sigma, \gamma \wedge C \Rightarrow u = v$ as indicated in the definition of the rewrite relation. By assumption $\gamma : u > v$. As $\mathcal{A} \models \sigma(\gamma)$ this implies $\sigma(u) > \sigma(v)$. As $u \in TERM^{\wedge}(\Sigma, V)$ we get $s' > t'$ by the monotonicity property of $>$. Finally, $s > t$ as $>$ is compatible with $\mathcal{A}$.

Now let $\gamma \wedge C \Rightarrow u = v \in R$ and let σ be a substitution that satisfies γ modulo $\mathcal{A}$. The assumption $\gamma : u > u_i, v_i$ yields $\sigma(u) > \sigma(u_i), \sigma(v_i)$ for any $u_i = v_i \in C$. □

As usual the assumption that $>$ contains $>_{st/\mathcal{A}}$ can be dropped if $>$ is sort compatible (i.e. if $s > t$, then s and t are sort compatible). The transitive closure of $>$ and $>_{st/\mathcal{A}}$ satisfies the desired properties.

As the compatibility property $\gamma : u > v, u_i, v_i$ requires possibly an infinite number of tests we next integrate the constraints into the notion of reduction ordering by parameterizing orderings with constraints.

Definition 6.4 *The set $\{>^{(\gamma)} | \ \gamma$ is a constraint$\}$ of partial orderings on $TERM(\Sigma, V)$ is said to be a* constrained reduction ordering system wrt *$(\Sigma = \Sigma_0 \oplus \Sigma_1, \mathcal{A})$ iff the following items hold:*

(a) $>^{(\top)}$ is a reduction ordering wrt $(\Sigma = \Sigma_0 \oplus \Sigma_1, \mathcal{A})$.

(b) The system is stable (wrt semantical substitutions): for all $u, v \in TERM(\Sigma, V)$, all constraints γ and any substitution σ, if $u >^{(\gamma)} v$, then $\sigma(u) >^{(\sigma(\gamma))} \sigma(v)$.

(c) The system is compatible with constraint satisfaction: If $\mathcal{A} \models \gamma$, then $>^{(\gamma)} = >^{(\top)}$.

Definition 6.5 *Let $\{>^{(\gamma)} | \ \gamma$ is a constraint$\}$ be a constrained reduction ordering system wrt $(\Sigma = \Sigma_0 \oplus \Sigma_1, \mathcal{A})$. The system is* compatible with the conditional rewrite system *R iff for any $\gamma \wedge C \Rightarrow u = v \in R$ we have $u >^{(\gamma)} v, u_i, v_i$, where $u_i = v_i \in C$.*

Lemma 6.2 *Let $\{>^{(\gamma)} | \ \gamma$ is a constraint$\}$ be a constrained reduction ordering system wrt $(\Sigma = \Sigma_0 \oplus \Sigma_1, \mathcal{A})$ that is compatible with R. Then $>^{(\top)}$ is a reduction ordering that is compatible with R (in the sense of definition 6.3).*

Proof: Let $\gamma \wedge C \Rightarrow u = v \in R$ and $u_i = v_i \in C$. We prove $\gamma : u >^{(\top)} v, u_i, v_i$. Let $\mathcal{A} \models \sigma(\gamma)$ for a substitution γ. As $u >^{(\gamma)} v, u_i, v_i$ we get $\sigma(u) >^{(\sigma(\gamma))} \sigma(v), \sigma(u_i), \sigma(v_i)$ by the stability property. Compatibility with constraint satisfaction finally yields $\sigma(u) >^{(\top)} \sigma(v), \sigma(u_i), \sigma(v_i)$. □

To give an example of a constrained reduction ordering system we define suitable recursive path orderings. The knowledge about the built-in algebra $\mathcal{A}$ intended to be integrated into the construction of the recursive path ordering is given by a so-called constrained base ordering system. As a method of integration we let the base orderings be part of the precedences that induce the recursive path orderings.

We first generalize $\mathcal{A}$-equivalence by integrating constraints:

Definition 6.6 *For $s, t \in TERM(\Sigma, V)$ let $s \sim_{\mathcal{A}}^{(\gamma)} t$ iff*
(a) $s, t \in TERM_0(\Sigma, V)$ and $\mathcal{A} \models \gamma \Rightarrow s = t$ or
(b) $s, t \in TERM^{\wedge}(\Sigma, V)$, $s \equiv f(s_1, \ldots, s_n)$, $t \equiv f(t_1, \ldots, t_n)$ and $s_i \sim_{\mathcal{A}}^{(\gamma)} t_i$ for $i = 1, \ldots, n$.

One easily proves that $\sim_{\mathcal{A}}^{(\gamma)}$ is stable wrt substitutions, i.e. for all $s,t \in TERM(\Sigma,V)$ and any substitution γ, if $s \sim_{\mathcal{A}}^{(\gamma)} t$, then $\sigma(s) \sim_{\mathcal{A}}^{(\sigma(\gamma))} \sigma(t)$.

Definition 6.7 *A set $\{>_0^{(\gamma)} | \ \gamma$ is a constraint$\}$ of partial orderings on $TERM_0(\Sigma,V)$ is said to be a* constrained base ordering system modulo $\mathcal{A}$ *iff the following items hold:*

(a) $>_0^{(\top)}$ *is wellfounded.*

(b) $>_0^{(\gamma)}$ *is compatible with $\mathcal{A}$, i.e. for any $s,t,s',t' \in TERM(\Sigma,V)$ and any constraint γ, if $s \sim_{\mathcal{A}}^{(\gamma)} s' >_0^{(\gamma)} t' \sim_{\mathcal{A}}^{(\gamma)} t$, then $s >_0^{(\gamma)} t$.*

(c) The system is stable wrt substitutions.

(d) The system is compatible with constraint satisfaction.

For an example of such a constrained base ordering system modulo $\mathcal{A}$ let $\mathcal{A}$ be *well-founded wrt.* $\succ, true \in F_0$ (i.e. the relation $\{(a,b) \in A \times A \mid \succ^{\mathcal{A}} (a,b) = true^{\mathcal{A}}\}$ is well-founded). Now let for $u,v \in TERM_0(\Sigma,V)$

$$u >_0^{(\gamma)} v \ iff \ \mathcal{A} \models \gamma \Rightarrow u \succ v = true.$$

Then $\{>_0^{(\gamma)} | \ \gamma$ *is a constraint*$\}$ is a constrained base ordering system modulo $\mathcal{A}$ — the *constrained base ordering system induced by* $\mathcal{A}, \succ, true$.

For the rest of this section let $\{>_0^{(\gamma)} | \ \gamma$ *is a constraint*$\}$ be a constrained base ordering system modulo $\mathcal{A}$.

Hierarchical Σ-terms will be converted into flat terms by abstracting from the internal structure of predefined terms in order to use the syntactical recursive path ordering construction. For that purpose let $[u]$ be a new constant symbol for any term $u \in TERM_0(\Sigma,V)$ and $C_0 = \{[u] \mid u \in TERM_0(\Sigma,V)\}$ be the set of these new constants. Further let $F^* = F^{\wedge} \cup C_0$ where $F^{\wedge} = F_0^{(\geq 1)} \cup F_1$. The mapping α converts (hierarchical) Σ-terms into (flat) F^*-terms. For $u \in TERM(\Sigma,V)$ let

$$\alpha(u) = \begin{cases} [u] & if \ u \in TERM_0(\Sigma,V) \\ f(\alpha(u_1),\ldots,\alpha(u_n)) & if \ u \in TERM^{\wedge}(\Sigma,V) \ and \ u = f(u_1,\ldots,u_n). \end{cases}$$

By abuse of notation let for $s \in TERM(F^*)$ and any substitution σ

$$\sigma(s) = \begin{cases} [\sigma(u)] & if \ s = [u] \\ f(\sigma(s_1),\ldots,\sigma(s_n)) & if \ s = f(s_1,\ldots,s_n). \end{cases}$$

Let also by abuse of notation $[u] >_0^{(\gamma)} [v]$ iff $u >_0^{(\gamma)} v$ and $\alpha(s) \sim_{\mathcal{A}}^{(\gamma)} \alpha(t)$ iff $s \sim_{\mathcal{A}}^{(\gamma)} t$.

We now define the recursive path ordering $>_{rpo,\mathcal{A}}^{(\gamma)}$ modulo $\mathcal{A}$ on the F^*-terms. We first review the usual definition of the recursive path ordering (see [De87]).

Definition 6.8 *Let $\gtrsim$ be a quasi-ordering (the precedence) with $\sim$ denoting its equivalence part and $>$ its strict part.*

(a) The permutative congruence $\sim_{perm}$ induced by $\gtrsim$ *is defined by: For $s \equiv f(s_1,\ldots,s_n), t \equiv g(t_1,\ldots,t_n) \in TERM(F)$ let $s \sim_{perm} t$ iff $f \sim g$ and there exists a permutation π on $\{1,\ldots,n\}$ such that $s_i \sim_{perm} t_{\pi(i)}$ for $i = 1,\ldots,n$.*

(b) *The* recursive path ordering $>_{rpo}$ induced by $\gtrsim$ *is defined by: For* $s \equiv f(s_1, \ldots, s_m), t \equiv g(t_1, \ldots, t_n) \in TERM(F)$ *let* $s >_{rpo} t$ *iff one of the following items holds:*

(α) $s_i >_{rpo} t$ *or* $s_i \sim_{perm} t$ *for some* $i \in \{1, \ldots, m\}$.

(β) $f > g$ *and* $s >_{rpo} t_j$ *for all* $j \in \{1, \ldots, n\}$.

(γ) $f \sim g$ *and for appropriate* A, B, a_i, b_i *we have* $\{s_1, \ldots, s_m\} = A \cup \{a_1, \ldots, a_k\}$, $\{t_1, \ldots, t_n\} = B \cup \{b_1, \ldots, b_k\}$, $A >_{rpo} >_{rpo} B$ *and* $a_i \sim_{perm} b_i$ $(i = 1, \ldots, k)$.

Definition 6.9 *Let* $\gtrsim_{F^\wedge}$ *be a well-founded partial ordering on* $F^\wedge$ *and* $\{>_0^{(\gamma)} | \gamma$ *is a constraint*$\}$ *be a constrained base ordering system modulo* $\mathcal{A}$. *Let* γ *be a constraint. Then*

$$\gtrsim^{(\gamma)} = \sim_{\mathcal{A}}^{(\gamma)} \cup \gtrsim_{F^\wedge} \cup >_0^{(\gamma)} \cup \{(f, [u]) \mid f \in F^\wedge; [u] \in C_0\}$$

is the precedence modulo $\mathcal{A}$ induced by $\gtrsim_{F^\wedge}$ and $>_0^{(\gamma)}$. *This precedence induces the* permutative equivalence $\sim_{perm,\mathcal{A}}^{(\gamma)}$ modulo $\mathcal{A}$ wrt. $\gtrsim_{F^\wedge}$ and $>_0^{(\gamma)}$ *and the* recursive path ordering $>_{rpo,\mathcal{A}}^{(\gamma)}$ modulo $\mathcal{A}$ wrt. $\gtrsim_{F^\wedge}$ and $>_0^{(\gamma)}$ *(on the* F^**-terms).*

Lemma 6.3 *Let* $\gtrsim_{F^\wedge}$ *be a well-founded partial ordering on* $F^\wedge$ *and* $\{>_0^{(\gamma)} | \gamma$ *is a constraint*$\}$ *be a constrained base ordering system modulo* $\mathcal{A}$. *Let* $>_{rpo,\mathcal{A}}^{(\gamma)}$ *be the recursive path ordering modulo* $\mathcal{A}$ *wrt.* $\gtrsim_{F^\wedge}$ *and* $>_0^{(\gamma)}$. *Let further* $s, s', t, t' \in TERM(F^*)$, γ *be a constraint and* σ *be a substitution. Then the following items hold:*

(a) *If* $s \sim_{\mathcal{A}}^{(\gamma)} s' \sim_{perm,\mathcal{A}}^{(\gamma)} t' \sim_{\mathcal{A}}^{(\gamma)} t$, *then* $s \sim_{perm,\mathcal{A}}^{(\gamma)} t$.

(b) *If* $s \sim_{\mathcal{A}}^{(\gamma)} s' >_{rpo,\mathcal{A}}^{(\gamma)} t' \sim_{\mathcal{A}}^{(\gamma)} t$, *then* $s >_{rpo,\mathcal{A}}^{(\gamma)} t$.

(c) *If* $s \sim_{perm,\mathcal{A}}^{(\gamma)} t$, *then* $\sigma(s) \sim_{perm,\mathcal{A}}^{(\sigma(\gamma))} \sigma(t)$.

(d) *If* $s >_{rpo,\mathcal{A}}^{(\gamma)} t$, *then* $\sigma(s) \, {}^{(\sigma(\gamma))}_{rpo,\mathcal{A}} \sigma(t)$.

(e) *If* $\mathcal{A} \models \gamma$, *then* $>_{rpo,\mathcal{A}}^{(\gamma)} = >_{rpo,\mathcal{A}}^{(\top)}$.

(f) $>_{rpo,\mathcal{A}}^{(\top)}$ *is wellfounded.*

Proof: The proof of (a), (b) resp. (c), (d) proceeds by induction on the sum of the lengths of s' and t' resp. s and t. Below we sketch only the proof of part (b), the other cases are shown analogously. Statement (e) follows from the fact that by the related property of the constrained base ordering system the two precedences are identical. To see (f) note that $>_0^{(\top)}$ is assumed to be wellfounded. Hence the precedence $\gtrsim^{(\top)}$ is wellfounded too. It follows (see [De87]) that $>_{rpo,\mathcal{A}}^{(\top)}$ is wellfounded.

Proof of part (b): For the induction base let s' and t' be constants from F^*.
Case 1: $s' = [u']$, $t' = [v']$ for appropriate $u', v' \in TERM(\Sigma_0, V)$. Then there exist $u, v \in TERM(\Sigma_0, V)$ such that $s = [u]$, $t = [v]$. By inspecting the definition of the precedence we get $u \sim_{\mathcal{A}}^{(\gamma)} u' >_0^{(\gamma)} v' \sim_{\mathcal{A}}^{(\gamma)} v$. The assumption that $\{>_0^{(\gamma)} | \gamma$ *is a constraint*$\}$ is a constrained base ordering system modulo $\mathcal{A}$ provides $u >_0^{(\gamma)} v$. Thus $s >_{rpo,\mathcal{A}}^{(\gamma)} t$.
Case 2: $s' \in F^\wedge$ and $t' = [v']$ for $v' \in TERM(\Sigma_0, V)$. Then $s = s'$ and $t = [v]$ for some $v \in TERM(\Sigma_0, V)$. Thus $s >^{(\gamma)} t$ and consequently $s >_{rpo,\mathcal{A}}^{(\gamma)} t$.

Case 3: $s', t' \in F^{\wedge}$. Then $s = s'$ and $t = t'$, consequently $s >_{rpo,\mathcal{A}}^{(\gamma)} t$.
Case 4: $s' = [u']$ for some $u' \in TERM(\Sigma_0, V)$ and $t' \in F^{\wedge}$. Note that this case is impossible by the definition of the precedence.

For the induction step let the sum of the lengths of s' and t' be greater than 2. Notice that the claim follows immediately if $t' \in C_0$ and that the case $s' \in C_0$ is impossible by the definition of the precedence. It remains the "syntactical case" $s' \equiv f(s'_1, \ldots, s'_m)$ and $t' \equiv g(t'_1, \ldots, t'_n)$ where $f, g \in F^{\wedge}$. Then we have $s \equiv f(s_1, \ldots, s_m)$ and $t \equiv g(t_1, \ldots, t_n)$ with appropriate F^*-terms s_i, t_j. The proof now proceeds straightforward by case analysis using the induction hypothesis and statement (a). □

The recursive path ordering modulo $\mathcal{A}$ on $TERM(\Sigma, V)$ wrt. $\gtrsim_{F^{\wedge}}$ and $>_0^{(\gamma)}$ is defined in the obvious way:

$$s >_{rpo,\mathcal{A}}^{(\gamma)} t \; iff \; \alpha(s) >_{rpo,\mathcal{A}}^{(\gamma)} \alpha(t).$$

With the aid of lemma 6.3 the following theorem is easily proved:

Theorem 6.1 *Let $\gtrsim_{F^{\wedge}}$ be a well-founded partial ordering on $F^{\wedge}$ and $\{>_0^{(\gamma)} | \; \gamma$ is a constraint$\}$ be a constrained base ordering system modulo $\mathcal{A}$. Let $>_{rpo,\mathcal{A}}^{(\gamma)}$ be the recursive path ordering modulo $\mathcal{A}$ on $TERM(\Sigma, V)$ wrt. $\gtrsim_{F^{\wedge}}$ and $>_0^{(\gamma)}$. Then he set $\{>_{rpo,\mathcal{A}}^{(\gamma)} | \; \gamma$ is a constraint$\}$ is a constrained reduction ordering system wrt $(\Sigma = \Sigma_0 \oplus \Sigma_1, \mathcal{A})$. In addition $>_{rpo,\mathcal{A}}^{(\top)}$ contains $>_{st/\mathcal{A}}$.*

Corollary 6.1 *Let $\gtrsim_{F^{\wedge}}$ be a well-founded partial ordering on $F^{\wedge}$ and $\{>_0^{(\gamma)} | \; \gamma$ is a constraint$\}$ be a constrained base ordering system modulo $\mathcal{A}$. Let $>_{rpo,\mathcal{A}}^{(\gamma)}$ be the recursive path ordering modulo $\mathcal{A}$ on $TERM(\Sigma, V)$ wrt. $\gtrsim_{F^{\wedge}}$ and $>_0^{(\gamma)}$. Let further the system $\{>_{rpo,\mathcal{A}}^{(\gamma)} | \; \gamma$ is a constraint$\}$ be compatible with a conditional rewrite system R. Then R is decreasing modulo $\mathcal{A}$.*

Example 6.2 *We continue example 7.1*
Let $\mathcal{N}$ be the natural number interpretation (enriched by a standard boolean interpretation). We have $F^{\wedge} = \{+, \succ, g\}$. Let $>_{F^{\wedge}} = \emptyset$. Let $\{>_0^{(\gamma)} | \; \gamma$ is a constraint$\}$ be induced by $\mathcal{N}, \succ, true$. Then

$$\begin{array}{lll} g(x,0) & >_{rpo}^{(\top)} & x \\ g(0,y) & >_{rpo}^{(\top)} & y \\ g(x+y,y) & >_{rpo}^{(y \succ 0 = true)} & g(x,y) \\ g(x,x+y) & >_{rpo}^{(x \succ 0 = true)} & g(x,y) \end{array}$$

Thus R is decreasing modulo $\mathcal{N}$ and especially, $\longrightarrow_{R/\mathcal{N}}$ is terminating.

7 Decidability of $\mathcal{A}$-equivalence and $\mathcal{A}$-match

In order to get a decidable rewrite relation modulo the built-in algebra $\mathcal{A}$-equivalence and $\mathcal{A}$-match have to be decidable. We finish the paper with some remarks about decidability results.

We first consider the decidability of $\mathcal{A}$-eqivalence. In general $\mathcal{A}$-equivalence is not decidable. This follows from the fact, that there exists a signature Σ_0 and a Σ_0-algebra $\mathcal{A}$ such that the equivalence relation $\sim_{\mathcal{A}}$ on $TERM_0(\Sigma, V)$ is not decidable (we thus consider a case where no new symbols are present). For a justification let E be a set of Σ_0-equations (for an appropriate

Σ_0) such that the inductive theory of E is not recursively enumerable. Let $\mathcal{A}$ denote the initial model of E. Now, if we could decide $u \sim_{\mathcal{A}} v$ resp. $\mathcal{A} \models u = v$ for arbitrary $u, v \in TERM(\Sigma_0, V)$, then we obviously could enumerate the theory of $\mathcal{A}$ resp. the inductive theory of E.

In many special cases however $\mathcal{A}$-equivalence can be proved to be decidable. First note that $s \sim_{\mathcal{A}} t$ iff $\mathcal{A} \models \gamma_{s=t}$ $(s, t \in TERM(\Sigma, V))$, so that we can concentrate on Σ_0-equations.

In the "mere syntactical case" we assume that $\mathcal{A} \models u = v$ iff $u \equiv v$ for all $u, v \in TERM(\Sigma_0, V)$. This case is available if $\mathcal{A}$ is the free term algebra induced by Σ_0 and V. In this case $\mathcal{A}$-equivalence is obviously decidable.

Example 7.1 *Let $S_0 = \{nat\}$, $F_0 = \{0, s\}$ and $D_0 = \{0 :\rightarrow nat, s : nat \rightarrow nat\}$. Let $\mathcal{A}$ be the canonical term algebra induced by Σ_0 and V.*

Next the special Σ_{nat}-algebra $\mathcal{N}$ and Σ_{int}-algebra $\mathcal{Z}$ are considered. $\mathcal{A}$-equivalence turns out to be decidable for these two algebras. For a proof we first introduce the notion of a polynomial over Σ_{nat} resp. Σ_{int}.

Definition 7.1 *A* polynomial $p(x_1, \ldots, x_k)$ over Σ_{nat} resp. Σ_{int} *is a term of the form*

$$p(x_1, \ldots, x_k) = \sum_{i=0}^{d} \sum_{i_1 + \ldots + i_k = i} c_{(i_1 \ldots i_k)} x_1^{i_1} \cdots x_k^{i_k}$$

where the so-called coefficients $c_{(i_1 \ldots i_k)}$ *are elements of the "basic" Σ_{nat}-terms $\{0, 1, 2, \ldots\}$ resp. of the "basic" Σ_{int}-terms $\{0, 1, -(1), \ldots\}$ and d is a natural number called* maximal exponent.

The degree of p, *written $deg(p)$, is the greatest number $n \leq d$ such that there exists a coefficient $c_{(i_1 \ldots i_k)}$ with $i_1 + \ldots + i_k = n$ and $c_{(i_1 \ldots i_k)} \not\sim_{\mathcal{A}} 0$ $(\mathcal{A} = \mathcal{N}, \mathcal{Z})$.*

Note that we omit brackets and use the usual priority and abbreviation conventions.

It is easily verified that for any $s \in TERM(\Sigma_0, V)$ one can effectively construct a polynomial p_s over Σ_0 such that $p_s \sim_{\mathcal{A}} s$ (where $\Sigma_0 = \Sigma_{nat}, \Sigma_{int}$ and $\mathcal{A} = \mathcal{N}, \mathcal{Z}$). The next lemma states a well known result and provides a method to decide $\mathcal{A}$-equivalence in our special cases.

Lemma 7.1 *Let $p(x_1, \ldots, x_k)$ and $p'(x_1, \ldots, x_k)$ be polynomials over Σ_{nat} resp. Σ_{int} with (w.l.o.g) the same maximal exponent d and with coefficients $c_{(i_1, \ldots, i_k)}$ and $c'_{(i_1, \ldots, i_k)}$. Then for $\mathcal{A} = \mathcal{N}$ and $\mathcal{A} = \mathcal{Z}$,*

$$p \sim_{\mathcal{A}} p' \; iff \; c_{(i_1, \ldots, i_k)} \equiv c'_{(i_1, \ldots, i_k)} \; for \; all \; coefficients \; (i.e. \; p \; and \; p' \; are \; identical).$$

Corollary 7.1 *$\mathcal{N}$-equivalence and $\mathcal{Z}$-equivalence are decidable.*

We turn to $\mathcal{A}$-matching. In general $\mathcal{A}$-matching is not decidable. As a counterexample consider $\mathcal{Z}$-matching. If $\mathcal{Z}$-matching were decidable, then Hilbert's 10th problem were too, in contrast to the well-known result in [Ma70].

The "mere syntactical case" (see above) is trivial as in this case $\mathcal{A}$-matching reduces to ordinary syntactical matching.

Whereas $\mathcal{Z}$-matching is undecidable, $\mathcal{N}$-matching is decidable. The decision procedure presented below is far away from being of practical interest. Our only interest here is to make the search space for the matching substitution finite. Let u, s be Σ_{nat}-terms. The decision to be made is whether there exists a substitution σ such that $\sigma(u) \sim_{\mathcal{N}} s$ or not. Again the problem can be simplified by switching over to polynomial representations.

Definition 7.2 *A substitution σ is called a* polynomial substitution *iff $\sigma(x_i)$ is a polynomial $q_i(y_1, \ldots, y_l)$ over Σ_{nat} such that the maximal exponent of q_i coincides with the degree of q_i.*

It suffices to decide for arbitrary polynomials $p(x_1, \ldots, x_k)$ and $q(y_1, \ldots, y_l)$ over Σ_{nat} whether there exists a polynomial substitution σ that satisfies $\sigma(p) \sim_{\mathcal{N}} q$.

Definition 7.3 *Let $p(x_1, \ldots, x_k)$ be a polynomial over Σ_{nat} and let σ be a substitution. The variable x_j* contributes to *p wrt σ (and $\mathcal{N}$) iff there exists a constituent $c_{(i_1 \ldots i_k)} x_1^{i_1} \ldots x_k^{i_k}$ of p such that $i_j \neq 0$ and $c_{(i_1 \ldots i_k)} \not\sim_{\mathcal{N}} 0$.*

Now let $p(x_1, \ldots, x_k)$ and $q(y_1, \ldots, y_l)$ be given. Let d be the degree of q and let c be the (wrt $\mathcal{N}$) maximal coefficient of q. Suppose that σ is a polynomial substitution with $\sigma(p) \sim_{\mathcal{N}} q$. It is easily verified that if x_j contributes to p wrt σ, then the degree of the polynomials $\sigma(x_j)$ is less or equal than d and the coefficients of $\sigma(x_j)$ are less or equal than c. If x_j does not contribute to p wrt σ, then the value $\sigma(x_j)$ can be changed into 0 without changing $\sigma(p)$ wrt $\sim_{\mathcal{N}}$. Thus, if there exists a polynomial substitution σ with $\sigma(p) \sim_{\mathcal{N}} q$, then there exists one so that $\sigma(x_j)$ is a polynomial with degree $\leq d$ and coefficients $\leq c$ wrt $\mathcal{N}$. As there exist only a finite number of such "test-substitutions" we get the following result:

Lemma 7.2 *The $\mathcal{N}$-match is decidable.*

Let us finally return to $\mathcal{Z}$-matching again. Though undecidable in general there can be made some positive statements. Fortunately matching is usually needed only for a finite number of patterns — the Σ_0-terms introduced by the left hand sides of the rewrite rules from R. Thus, if we keep the syntax of R rather simple, we can possibly decide $\mathcal{Z}$-matching wrt the relevant patterns.

Definition 7.4 *A Σ_{int}-term is said to be* linear *iff it is a polynomial over Σ_{int} of a degree ≤ 1.*
R is said to be linear *iff every Σ_{int}-term occuring as subterm in a left hand side of a rule from R is linear.*

Now the problem is to decide for a linear term $p \equiv a_1 x_1 + \ldots a_k x_k + c$ and an arbitrary polynomial $q'(y_1, \ldots, y_l)$ over Σ_{int} whether there exists a substitution σ with $\sigma(p) \sim_{\mathcal{Z}} q'$. First we transform the problem into an equivalent one by "bringing c on the other side". Let $q(y_1, \ldots, y_l)$ be the resulting polynomial. It is easily verified that if there exists a matching substitution for p and q, then there exists such a substitution σ with

$$\sigma(x_j) = \sum_{i=0}^{d} \sum_{i_1 + \ldots + i_k = i} c_{j,(i_1 \ldots i_k)} x_1^{i_1} \ldots x_k^{i_k}$$

with appropriate "basic" integer terms $c_{j,(i_1 \ldots i_k)}$. Now such a substitution satisfies $\sigma(p) \sim_{\mathcal{Z}} q$ iff the following system of equations with the unknown $c_{j,(i_1 \ldots i_k)}$ has a solution in $\mathcal{Z}$:

$$\{a_1 c_{1,(i_1 \ldots i_k)} + \ldots + a_k c_{k,(i_1 \ldots i_k)} = c_{(i_1 \ldots i_k)} \mid i = 0, \ldots, d \text{ and } i_1 + \ldots + i_k = i\}.$$

As such a system of equations can be solved we get:

Lemma 7.3 *$\mathcal{Z}$-matching wrt a linear rewrite system R is decidable.*

References

[AvMa90] J. Avenhaus and K. Madlener, Term rewriting and equational reasoning, in: R. B. Banerji, ed., *Formal Techniques in Artificial Intelligence* (North-Holland, Amsterdam, 1990) pp. 1-43.

[Ay92] M. Ayala Rincon, Built-in conditional rewrite systems, to appear as internal report at the university of Kaiserslautern.

[BaDe89] L. Bachmair and N. Dershowitz, Completion for rewriting modulo a congruence, *Theoretical Computer Science* 67 (1989) pp. 173-201.

[De87] N. Dershowitz, Termination of rewriting, *J. Symbolic Computation* 3 (1987) pp. 69-116.

[DeJo90] N. Dershowitz and J.P. Jouannaud, Rewriting systems, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B* (Elsevier, Amsterdam, 1990) pp. 241-320.

[DeOk90] N. Dershowitz and M. Okada, A rationale for conditional equational programming, *Theoretical Computer Science* 75 (1990) pp. 111-138.

[Ga91] H. Ganzinger, Order-sorted completion: the many-sorted way, *Theoretical Computer Science* 89 (1991) pp. 3-32.

[JaLa87a] J. Jaffar and J.-L. Lassez, Constraint logic programming, in *Proc. of ACM Symp. on Principles of Programming Languages '87* (1987) pp. 111-119.

[JaLa87b] J. Jaffar and J.-L. Lassez, From unification to constraints, in: K. Furukawa et al., eds., *Logic Programming '87, Proc. of the 6th Conference*, LNCS 315, (Springer, Berlin, 1988) pp. 1-18.

[JoKi86] J.-P. Jouannaud and H. Kirchner, Completion of a set of rules modulo a set of equations, *SIAM J. on Computing* 15 (1986) pp. 1155-1194.

[KaCh89] S. Kaplan and C. Choppy, Abstract rewriting with concrete operators, in: *3rd RTA '89*, LNCS 355, (Springer, Berlin, 1989) pp. 178-185.

[KaMu86] D. Kapur, D.R. Musser, Inductive reasoning for incomplete specifications, in: *Proc. IEEE Symposium on Logic in Computer Science* (Cambridge MA, 1986) pp. 367-377.

[KKR90] C. Kirchner, H. Kirchner and M. Rusinowitch, Deduction with symbolic constraints, *Revue d'Intelligence Artificielle* 4(3) (1990) pp. 9-52.

[KnBe70] D.E. Knuth and P.B. Bendix, Simple word problems in universal algebra, in: J. Leech, ed., *Computational Problems in Abstract Algebra* (Pergamon Press, Oxford, 1970) pp. 342-376.

[Ma70] J.V. Matijasevic, Enumerable sets are Diophantine, *Soviet Math. (Dokl.)* 11 (1970) pp. 354-357.

[PeSt81] G.E. Peterson and M.E. Stickel, Complete sets of reductions for some equational theories, *J. of the Association for Computing Machinery* 28 (1981) pp. 233-264.

[Si89] J.H. Siekmann, Unification theory *J. of Symbolic Computation* 7 (1989) pp. 207-274.

[SNGM89] G. Smolka, W. Nutt, J.A. Goguen and J. Meseguer, Order-sorted equational computation, in: H. Ait-Kaci and M. Nivat, eds., *Resolution of Equations in Algebraic Structures, Vol. 2* (Academic Press, San Diego CA, 1989) pp. 297-367.

[Vo89] S.G. Vorobyov, Conditional rewrite rule systems with built-in arithmetic and induction, in: *3rd RTA '89*, LNCS 355, (Springer, Berlin, 1989) pp. 492-512.

[Wa92] U. Waldmann, Semantics of order-sorted specifications, *Theoretical Computer Science* 94 (1992) pp. 1-35.

[Wa90] H.R. Walters, Hybrid implementations of algebraic specifications, in: *Algebraic and Logic Programming, Proc. 2nd Int. Conf. 1990*, LNCS 463, (Springer, Berlin, 1990) pp. 40-54.