Rewriting and Reasoning with Set-Relations II: The Non-Ground Case Completeness

Valentinas Kriaučiukas [1]⋆ *Michał Walicki* [2]⋆

[1] Matematikos ir Informatikos Institutas, Akademijos 4, 2600 Vilnius, Lithuania
valentinas.kriauciukas@mlats.mii.lt
[2] Institutt for Informatikk, Universitetet i Bergen, HiB, N-5020 Bergen, Norway
michal@ii.uib.no

Abstract. We consider reasoning and rewriting with set-relations: inclusion, nonempty intersection and singleton identity, each of which satisfies only two among the three properties of the equivalence relations. The paper presents a complete inference system which is a generalization of ordered paramodulation and superposition calculi. Notions of rewriting proof and confluent rule system are defined for such nonequivalence relations. Together with the notions of forcing and redundancy they are applied in the completeness proof. Ground completeness cannot be lifted to the nonground case because substitution for variables is restricted to deterministic terms. To overcome the problems of restricted substitutivity and hidden (in relations) existential quantification, unification is defined as a three step process: substitution of determistic terms, introduction of bindings and "on-line" skolemisation. The inference rules based on this unification derive non-ground clauses even from the ground ones, thus making an application of a standard lifting lemma impossible. The completness theorem is proved directly without use of such a lemma.

Keywords: rewriting, theorem proving, binary relations, nondeterminism, completeness, ordered superposition

1 Introduction

Reasoning with sets becomes an important issue in different areas of computer science. Its relevance can be noticed in constraint and logic programming e.g. [SD86, DO92, Jay92], in algebraic approach to nondeterminism e.g. [Hus93, Hes88, WM95b], in term rewriting e.g. [LA93, Kap88, Hus93].

The set-relations we are considering are not congruences – not even equivalences: singleton identity is not reflexive, inclusion is not symmetric, and nonempty intersection is not transitive. We study the rewriting proofs in the presence of these relations generalizing several classical notions (critical pair, confluence, rewriting proof) to the present context. Our results on rewriting extend bi-rewriting of Levy and Agustí [LA93] in that we consider three different set-relations. We also take a step beyond the framework of Bachmair and Ganzinger [BG94b] in that we study more general composition of relations than chaining of transitive relations, albeit, in a very similar way as [BG95].

⋆ Both authors gratefully acknowledge the financial support received from the Norwegian Research Council.

In an earlier paper [KW94], we proved ground-completeness of reasoning system for these three set relations. Their use originated from the study of specification of nondeterminism in [Wal93, WM95a, WM95b]. The reader is referred to these works for more detailed motivation and background. In the present paper we extend – rather than apply – these results to the non-ground case.

In a standard completeness proof one can utilize completeness of the ground case by establishing a lifting lemma. It amounts to the fact that a conclusion of an inference rule applied to ground instances of clauses is a ground instance of the conclusion of the same rule applied to the clauses. In our case, this cannot be done. For the first, some rules do not preserve groundness of clauses. The same is the case for relaxed paramodulation rule [SL91] and therefore authors constructed there a syntactic proof of completeness. Secondly, ground instances of a contradictory set of clauses may happen to be consistent. The problem is the *lack* of deterministic constants and functions, which could be used to form enough ground terms. We give a direct semantic proof of completeness not using a lifting lemma. The same method can be also applied in the case of relaxed paramodulation.

Section 2 defines the syntax and the multialgebraic semantics of the language and lists some basic properties of the set-relations. Section 3 discusses the non-standard difficulties with and defines the unification of nondeterministic terms. Section 4 introduces the reasoning system $\mathcal{I}$ and specifies the MAXIMAL LITERAL proof strategy for using $\mathcal{I}$. A sketch of the completeness proof is presented in Section 5.

Due to space limitations, only the general ideas of proofs of the main theorems are included in this version of the paper.

2 Syntax and Semantics

Specifications are written using a countable set of variables $\mathcal{V}$ and a finite non-empty set of functional symbols Σ having arity $ar : \Sigma \to \mathbb{N}$.[3] An $f \in \Sigma$ with $ar(f) = 0$ is a *constant*. Terms $\mathcal{T}$ over signature Σ are defined in the usual way.

There are only three atomic forms of formulae built using binary predicates: *equation* $s \approx t$, *inclusion* $s \prec t$ and *intersection* $s \frown t$. Atoms and their negations form the set of *literals*. An atom a is a *positive* literal, and a negated atom $\neg a$ is a *negative* literal. Negative literals are written as $s \not\approx t$, $s \not\prec t$ and $s \not\frown t$. $\overline{s \oplus t}$ is the reversed literal $t \overline{\oplus} s$, which is different from $t \oplus s$ only in the cases $\oplus \in \{\prec, \not\prec, \succ, \not\succ\}$, because these signs are not symmetric by their shape and meaning. A *clause* is a finite set of literals, a *specification* is a set of *clauses*. By *words* we will mean the union of the sets of terms, literals and clauses. $\mathcal{V}(w)$ denotes the set of variables occuring in a word w. Words are interpretated in *multialgebras* [Kap88, Hus92, Wal93] which, unlike usual algebras, allow functions to have multiple values. **T** and **F** denote the boolean values.

[3] To simplify the notation we are treating only the unsorted case. Extensions to many sorts are straightforward.

Definition 1. A Σ-*multialgebra* A is a pair $\langle S^A, \Sigma^A \rangle$ where S^A is a non empty *carrier* set, and Σ^A is a set of set-valued functions $f^A : (S^A)^{ar(f)} \to \mathcal{P}^+(S^A)$, where $f \in \Sigma$, and $\mathcal{P}^+(S^A)$ is the power-set of S^A with the empty set excluded.

We are dealing with total multialgebras and therefore exclude the empty set. Its admission, for instance for modelling partiality as in [BK95], would require modification of the inference system we are introducing in this paper.

Definition 2. Let A be a Σ-multialgebra, $v : \mathcal{V} \to S^A$ be an interpretation of variables, then value $[\![w]\!]^v$ of a word w is defined for :

1. a variable $x \in \mathcal{V}$, $[\![x]\!]^v \stackrel{\text{def}}{=} v(x)$;
2. a constant $c \in \Sigma$, $[\![c]\!]^v \stackrel{\text{def}}{=} c^A$;
3. a term $t = f(t_1, \ldots, t_n) \in \mathcal{T}$, $[\![t]\!]^v \stackrel{\text{def}}{=} \bigcup_{\alpha_i \in [\![t_i]\!]^v} f^A(\alpha_1, \ldots, \alpha_n)$;
4. a literal $l = s \oplus t$, $[\![l]\!]^v \stackrel{\text{def}}{=} F_\oplus([\![s]\!]^v, [\![t]\!]^v)$, where
   $$F_\approx(U,V) \equiv \forall\alpha \in U\ \forall\beta \in V\ \alpha = \beta, \qquad F_\prec(U,V) \equiv \forall\alpha \in U\ \exists\beta \in V\ \alpha = \beta,$$
   $$F_\frown(U,V) \equiv \exists\alpha \in U\ \exists\beta \in V\ \alpha = \beta, \quad F_{\neg\otimes}(U,V) \equiv \neg F_\otimes(U,V);$$
5. a clause C, $[\![C]\!]^v \stackrel{\text{def}}{=} \bigvee_{l \in C} [\![l]\!]^v$ if $C \neq \emptyset$, and $[\![C]\!]^v \stackrel{\text{def}}{=} \mathbf{F}$ if $C = \emptyset$.

The multialgebra A *satisfies* a clause C if $[\![C]\!]^v = \mathbf{T}$ for every interpretation $v : \mathcal{V} \to S^A$. It *satisfies* a literal l iff it satisfies the clause $\{l\}$, and *satisfies* a specification $\mathcal{S}$ iff it satisfies all the clauses in $\mathcal{S}$.

Term types $\exists$ and $\forall$ in literals. The point 4 of Definition 2 involved existential quantification in some predicates. The meaning of atoms is defined according to the following schema: $[\![s \oplus t]\!]^v = Q_1^\oplus \alpha \in [\![s]\!]^v\, Q_2^\oplus \beta \in [\![t]\!]^v\ \alpha = \beta$, where $Q_i^\oplus \in \{\forall, \exists\}$. We thus say that (the occurrence of) the term s *has type* $Q_1^\oplus$ and of t the type $Q_2^\oplus$. There are four possibilities of arranging the quantifiers $Q_1^\oplus Q_2^\oplus$: $\forall\forall$ corresponds to $\oplus = \approx$, $\forall\exists$ to $\prec$ and $\exists\exists$ to $\frown$. The fourth $\exists\forall$, say $\gg$, has not been used but it can be defined as $s \gg t \Leftrightarrow s \succ t \wedge t \approx t$. It is the exact counterpart of the relation ':' used in *unified algebras* [Mos89].

Basic properties of atoms. Set equality can be defined as $s \asymp t \stackrel{\text{def}}{=} s \prec t \wedge s \succ t$. The positive, resp. negative, relations are totally ordered by strength:

$$u \approx v \Rightarrow u \asymp v \Rightarrow u \prec v \Rightarrow u \frown v \qquad \textit{and} \qquad u \not\approx v \Leftarrow u \not\asymp v \Leftarrow u \not\prec v \Leftarrow u \not\frown v. \quad (1)$$

Derivations and lemmas below refer always to the strongest possible relation.

Replacement of terms – "equals by equals" – is possible only in equations, nevertheless the following lemmas will allow later to develop techniques of term-rewriting. ($u[t]_p$ denotes term u with term t substituted at the position p.)

Lemma 3. *The following term replacement properties hold:*

$$s \approx t \Rightarrow u[s]_p \asymp u[t]_p, \qquad s \prec t \Rightarrow u[s]_p \prec u[t]_p, \qquad s \frown t \Rightarrow u[s]_p \frown u[t]_p. \quad (2)$$

Rules for replacement of terms in literals, related to the appearance of *critical peaks* [DJ90] and generation of *critical pairs*, are described in

Lemma 4. *The defined predicates satisfy the rules given in Table 1.*

$\mathsf{Repl}(\_,\_)$	$u[s] \approx v$	$u[s] \prec v$	$u[s] \succ v$	$u[s] \frown v$	$u[s] \not\approx v$	$u[s] \not\prec v$	$u[s] \not\succ v$	$u[s] \not\frown v$
$s \approx t$	$u[t] \approx v$	$u[t] \prec v$	$u[t] \succ v$	$u[t] \frown v$	$u[t] \not\approx v$	$u[t] \not\prec v$	$u[t] \not\succ v$	$u[t] \not\frown v$
$s \prec t$	$u[t] \succ v$	$u[t] \frown v$	$u[t] \succ v$	$u[t] \frown v$	$u[t] \not\approx v$	$u[t] \not\prec v$	$u[t] \not\approx v$	$u[t] \not\prec v$
$s \succ t$	$u[t] \approx v$	$u[t] \prec v$	–	–	–	–	$u[t] \not\succ v$	$u[t] \not\frown v$
$s \frown t$	$u[t] \succ v$	$u[t] \frown v$	–	–	–	–	$u[t] \not\approx v$	$u[t] \not\prec v$

Table 1. Rules for subterm replacement

The table may be encoded as a partial function $\mathsf{Repl}(\_,\_)$: $(s \oplus t \wedge u[s]_p \otimes v \Rightarrow u[t]_p \mathsf{Repl}(\oplus, \otimes) v)$ for any terms s, t, u, v and position p at u.

When $u[s] = s$, we may use a similar table for *chaining* or *composing* relations, e.g., $s {\approx} t {\wedge} t {\prec} u \Rightarrow s {\prec} u$. We write it as $\approx \diamond \prec\; = \;\prec$. Notice that in (2) the predicate '$\approx$' was not inherited after substitution unlike the other two. Thus, sometimes, composition may produce stronger result than replacement. For instance, $s \approx t \wedge t \succ u \Rightarrow s \approx u$ (the table would yield $s \succ u$). The differences occur in row 1, columns 3 through 6, where the result of chaining will be

$$\approx \diamond \succ \; = \; \approx, \qquad \approx \diamond \frown \; = \; \prec, \qquad \approx \diamond \not\approx \; = \; \not\succ, \qquad \approx \diamond \not\prec \; = \; \not\frown. \tag{3}$$

It is easy to see that $\_ \diamond \_$ is transitive (so the first relation is reversed comparing with entries of Table 1). We join the two operations into one:

$$\mathsf{Sup}(p, \oplus, \otimes) = \begin{cases} \overline{\oplus} \diamond \otimes, & \text{if } p \text{ is the top position,} \\ \mathsf{Repl}(\oplus, \otimes), & \text{otherwise.} \end{cases}$$

Sup(erposition) of negative and positive atoms is symmetric to the superposition of the positive and the negative ones given in the table. Superposition of two negative atoms does not allow one to draw any specific conclusion and therefore is not mentioned at all.

3 Unification of nondeterministic terms

There are several things hindering us from the application of the usual unification techniques and we begin here with a brief example illustrating these difficulties.

1) The essential feature of calculus of nondeterministic operations is unsoundness of unrestricted substitution. Terms and variables denote objects of different kind: variables always mean single elements, while terms mean sets of possible values even if values of their variables are fixed. This excludes unification of terms by substitutions. Therefore, we apply rules like *relaxed paramodulation* [SL91], which make terms identical by "cutting out" some subterms, but which introduce new literals into derived clauses, like in the following derivation:

$$\frac{y \not\prec g(c); \quad h(x,x) \prec g(x)}{y \not\prec h(x,x), x \not\frown c}$$

(Comma between literals means disjunction and x, y are the only variables.) The variable x can not be replaced by c in $h(x,x) \prec g(x)$ because $h(c,c) \prec g(c)$ is equivalent to $x \frown c \wedge y \frown c \Rightarrow h(x,y) \prec g(c)$, but not to $x \frown c \Rightarrow h(x,x) \prec g(c)$, when c has more than one possible value. Fortunately, $y \not\prec g(c)$ is equivalent to $x \frown c \Rightarrow y \not\prec g(x)$, what is used in the derivation. Literals $x \not\frown c$, where x is a variable, are called *bindings*.

2) Another complicating circumstance is that if some nondeterministic term occured instead of y in the left premise, the derivation, although correct, would in some cases yield too weak a conclusion (Example 1).

3) Yet another problem is illustrated by an attempt to apply transitivity of $\prec$ to atoms $f(y) \prec g(c,y)$ and $g(x, h(x,x)) \prec e(x)$. The term $h(x,x)$ may be moved into a binding, but c cannot because $f(y) \prec g(c,y)$ means $\exists x(x \prec c \wedge f(y) \prec g(x,y))$ but not $\forall x(x \prec c \Rightarrow f(y) \prec g(x,y))$. Bindings do not help in this situation, because the occurrence of the term $g(c,y)$ in the literal $f(y) \prec g(c,y)$ is, as we call it, of type $\exists$. Following Skolem, we know that there exists a function α satisfying $\alpha(y) \prec c$ and $y \frown h(\alpha(y), \alpha(y)) \Rightarrow f(y) \prec e(\alpha(y))$. The function α is a semantical object, and we will introduce notation rules for such functions since they will be needed in the derivations in our inference system.

Unification is used during the proof process but, in order to solve the above problems, it will not only unify two terms but also produce some additional assumptions to be included in the processed clauses. We consider *refutational* proofs with a strategy analogous to *ordered resolution* and *ordered superposition* [BG94a, PP91], in which term ordering is used to restrict the proof search space. According to this strategy, only maximal terms may be involved in the applications of the inference rules. Our results are valid for any *simplification* ordering [DJ90] of terms. In the following example the maximal terms are underlined.

Example 1.

$$\underline{f(x)} \prec h(x,x); \tag{4}$$

$$\underline{g(x)} \succ h(x,x); \tag{5}$$

$$\underline{g(c)} \not\succ f(c); \tag{6}$$

This set of clauses has no model [Hus92, Wal93]. The ordering of the functional symbols, $g > f > h > c$, gives rise to the term ordering used here. There is only one possibility to start: to unify terms with g. But if c were moved from $g(c)$ into a binding just now, the clause $x \not\frown c, g(x) \not\succ f(c)$ would be obtained, which does not contradict the first two clauses. First, the other side of the literal must be made deterministic. A new deterministic constant d denotes an element of $f(c)$ which is not an element of $g(c)$:

$$\underline{f(c)} \succ d; \qquad (6) \tag{7}$$

$$\underline{g(c)} \not\frown d; \qquad (6) \tag{8}$$

Only now c may be moved into a binding and transitivity of $\succ$ applied:

$$c \not\frown x,\ \underline{h(x,x)} \not\succ d\ (8,5) \tag{9}$$

$$\underline{c} \succ e; \qquad (7) \tag{10}$$

$$\underline{f(e)} \succ d; \qquad (7) \tag{11}$$

In the similar way clause 7 was prepared to be resolved with clause 4, because deterministic terms can be substituted without any restrictions:

$$\underline{h(e,e)} \succ d; \qquad (11,4) \tag{12}$$

$$\underline{c} \not\succ e,\, d \not\succ d; \qquad (12,9) \tag{13}$$

$$\underline{e} \not\succ e,\, d \not\succ d; \qquad (10,13) \tag{14}$$

$$\underline{d} \not\succ d; \qquad (14) \tag{15}$$

$$\square \tag{16}$$

The clauses 8 and 11 are *assumptions* about new deterministic terms (constants in this example) d, e, which are some kind of *Skolem* functions, introduced to break down existential binding present inside of some terms. Their introduction, like introduction of variable bindings, is an effect of term unification, and can not be avoided in this strategy.

The example shows how complicated unification is in the case of nondeterministic operations. The definition of this unification was the main problem to be solved on the way to the completeness result.

3.1 Substitutions are deterministic

In proofs we extend syntax by additional functional symbols from an infinite set Φ called *f-variables.* They are always interpreted as deterministic functions, therefore terms constructed completely of variables and f-variables are called *d-terms* (shorthand for *deterministic terms*), their set is denoted $\mathcal{D}$, so $\mathcal{T} \cap \mathcal{D} = \mathcal{V}$. Terms of this kind are used to construct a model in the completeness proof. Only d-terms are allowed to be substituted into variables.

Definition 5. Call a *substitution* any function $\sigma : \mathcal{V} \to \mathcal{D}$. The domain $\mathcal{D}om(\sigma)$ of σ is the set $\{x : \sigma(x) \neq x\}$ of variables on which σ is non-trivial. As a set the substitution σ is considered as the set of pairs $\{\langle x, \sigma(x)\rangle : x \in \mathcal{D}om(\sigma)\}$.

Variables are the only d-terms in the set $\mathcal{T}$. After instantiation of some variables by d-terms, the obtained terms become divided into two parts: the top is nondeterministic, and bottom is deterministic. The natural way to present this division is to write such terms in the form $t\sigma$, where t does not contain f-variables and σ is a substitution. The same applies to other words, like literals and clauses. Words in such presentation remind of *closures* $w \cdot \sigma$ from [BGLS95]. We sometimes use this form of presentation. In our case, unlike in [BGLS95], this form is derivable from the word structure because of non-intersection of classes $\mathcal{T} \setminus \mathcal{V}$ and $\mathcal{D} \setminus \mathcal{V}$. We borrow some terminology from [BGLS95]: w is called a *skeleton* and $\mathcal{V}ar(w)$ is called the *frontier* of a word $w\sigma$, also denoted $\lfloor w\sigma \rfloor$. It is supposed that all variables in any skeleton are different, therefore all skeletons of the word $w\sigma$ are equal up to renaming of variables. (This is relevant to introduction of f-variables discussed below.) In spite of non-uniqueness of skeletons, we will write $w\sigma = w \cdot \sigma$, as if interpreting the sign '$\cdot$' as an application of substitution to the word w.

The restriction to substitute only d-terms restricts the possibility to unify terms. As a kind of compensation for that, it is allowed to *replace* some subterms by d-terms. Soundness of this replacement is based on special properties of f-variables formulated in Lemma 7.

3.2 Introduction of f-variables

They are related with some literals and *positions* in them. We include the functional symbols in positions in order to be able to relate positions directly with specific terms.

Definition 6. A *position* is any finite sequence $f_1 n_1 \dots f_k n_k$, where f_i are functional sybols and n_i are natural numbers such that $0 < n_i \leq ar(f_i)$. The empty sequence, the *top position*, is denoted Λ.

Concatenation of positions p, q is denoted by juxtaposition pq, with unit Λ. For a set of positions Q, pQ denotes $\{pq : q \in Q\}$. Positions form upper semilattice with Λ as the top *wrt.* the *prefix order*: position pq is *below* p for $q \neq \Lambda$. For a set of positions P, $\min(P)$ and $\max(P)$ denote, respectively, the sets of minimal and maximal positions in P. The set of positions in t, $\mathcal{P}os(t)$, is the smallest set of positions such that $\Lambda \in \mathcal{P}os(t)$ and, if $t = f(t_1, \dots, t_n)$, then $\mathcal{P}os(t) \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} fi \cdot \mathcal{P}os(t_i)$. $t|_p$ denotes a subterm of t which *occurs at the position* p : $t|_\Lambda = t$ and $t|_{qfi} = t_i$ if $t|_q = f(t_1, \dots, t_n)$, otherwise $t|_{qfi}$ is undefined. $t[s]_p$ denotes the term t with the subterm $t|_p$ replaced by s (the case $s = t|_p$ is possible). For a set of positions P, $t|_P$ denotes the set of subterms $\{t|_p : p \in P \cap \mathcal{P}os(t)\}$. $\mathcal{V}ar(t)$ denotes the set of *variable positions* in a term t, *i.e.*, $\{p \in \mathcal{P}os(t) : t|_p \text{ is a variable}\}$. $\mathcal{V}ar(t)$ is a subset of $\min(\mathcal{P}os(t))$, the remaining minimal positions (if any) are occupied by constants.

When are f-variables introduced? The f-variables correspond to *Skolem* functions and are needed in unification of the terms of type $\exists$. There are four general forms of literals whith a non-variable term s of this type in which new f-variables may be introduced by unification:

$$s \frown t, \quad s \not\approx t, \quad s \not\succ t, \quad s \succ t, \tag{17}$$

$s, t \in \mathcal{T}$. To describe all possible appearences of f-variables in any proof, we establish a bijection $\delta(\_, \_)$ between the following sets:

- the set of all pairs $\langle l, p \rangle$, where a literal l has one of the forms presented in (17), t is a variable in the case $l = s \succ t$, p is a non-variable position in s,
- the set of d-terms with exactly one f-variable.

For $e(\mathbf{x}) = \delta(l, p)$, $\mathbf{x}$ is the list of all variables occuring in l, in the same order from left to right and with all repetitions. Thanks to this requirement, $\delta(l \cdot \sigma, p) = \delta(l, p)\sigma$ for any substitution σ. The inverse bijection $\delta^{-1}(\_)$ is a pair of functions $\langle \ell(\_), \pi(\_) \rangle$, *i.e.*, $\ell(\delta(l, p)) = l$, $\pi(\delta(l, p)) = p$ and $\delta(\ell(d), \pi(d)) = d$. We also denote by $\tau(e(\mathbf{x})) \stackrel{\text{def}}{=} s|_p$, the subterm of l which can be safely replaced in it by $e(\mathbf{x})$ in the sense of Lemma 7.

Semantics for f-variables. For a Σ-multialgebra A, each f-variable e is interpretated as a deterministic function $\phi(e) : (S^A)^{ar(e)} \to S^A$. Let ϕ be such an interpretation of f-variables. It extends A to a $\Sigma \cup \Phi$-multialgebra, denoted A_ϕ. Values of d-terms in the multialgebra A_ϕ are evaluated according to the usual rules of (deterministic) algebras. The interpretations of f-variables must satisfy the additional conditions given in:

Lemma 7. *Any Σ-multialgebra A can be extended with an intepretation ϕ of f-variables in such a way that for any d-term d :*

- *A_ϕ satisfies the atom $d \prec \tau(d)$;*
- *A_ϕ satisfies $\ell(d)$ iff it satisfies $\ell(d)[d]_{\pi(d)}$*

The last condition says that the subterm $\tau(d)$ can be replaced by d in $\ell(d)$ without changing the meaning of $\ell(d)$.

3.3 Unification

Success in unifying terms depends not only on terms, but also on the literals in which they occur. In usual unification, one tries to make terms identical by some unifying substitution. In our case, only d-subterms can be substituted for variables. If a variable should be replaced by a non-deterministic subterm, then the inverse action is made — the subterm is replaced by the variable, we say that the subterm is *ejected* and put into a new literal, called a *binding*. In clauses, this kind of replacement is legal only inside of terms of type $\forall$. In terms of type $\exists$, the ejected subterms must be replaced by new f-variables, which are then bound by *assumptions*, the special kind of clauses. To be shorter in some places below, we call *unifying sets* collections consisting of a substitution (presented as a set), a set of bindings and a set of assumptions. In general, the process of unification can be presented as a sequence of three phases: 1) ejection of some subterms, 2) formation of the unifying sets, and 3) usual unification.

Ejection. To perform an ejection from a term, it is sufficient to know the frontier of the other term, so we formulate ejection relatively to some given set Q of positions. Let $l = s \oplus t$ and $l' = l \cdot \sigma$ be literals, l' be the one where ejection should occur, and let $P \stackrel{\text{def}}{=} \max(Q \cap \mathcal{P}os(s)) \setminus \mathcal{V}ar(s)$ be the set of non-variable positions of s which are maximal in Q. If P is empty, then there are no subterms to be ejected. If not, then we proceed as follows. All subterms $s|_P$ are ejected and replaced by new, neither from $\mathcal{D}om(\sigma)$ nor from $\mathcal{V}(l')$, all distinct variables. Let s_Q be the term obtained from s by this replacement.

The rest depends on the form of the literal l and is presented in the next phases.

Formation of unifying sets. Let $B = \{s_Q|_p \not\prec s|_p : p \in P\}$ be a set of bindings, $A = \{\delta(l,p) \prec s|_p : p \in P\}$ be a set of atoms, and $S = \{\langle s_Q|_p, \delta(l',p)\rangle : p \in P\}$ be a substitution. All these sets are obtained by the replacement of $s|_P$.

Different cases to consider are presented in Table 2. The second term of l (*i.e.*, t) can be changed to a new variable y, so the final forms of l and t are denoted l_Q and t_Q, while $\sigma(l',Q)$ and $\mathsf{A}(l',Q)$ denote the obtained substitution and the set of assumptions, respectively. The first line of the table describes the trivial case $P = \emptyset$.

for Q and $l' = (s \oplus t) \cdot \sigma$	$\mathsf{B}(l', Q)$	l_Q	$\sigma(l', Q)$	$\mathsf{A}(l', Q)$
$(Q \cap \mathcal{P}os(s)) \setminus \mathcal{V}ar(s) = \emptyset$	$\emptyset$	l	$\emptyset$	$\emptyset$
$\oplus = \succ \ \wedge \ t \notin \mathcal{V}$	$\{y \not\frown t\}$	$s_Q \succ y$	$\sigma \cup S$	A
$\oplus \in \{\succ, \frown, \not\approx, \not\prec\} \wedge$ $(\oplus = \succ \ \Rightarrow \ t \in \mathcal{V})$	$\emptyset$	$s_Q \oplus t$	$\sigma \cup S$	A
$\oplus = \not\succ \wedge t \notin \mathcal{V}$	B	$s_Q \not\frown y$	$\sigma \cup \{\langle y, \delta(t \not\prec s, \Lambda)\rangle\}$	$\{\delta(t \not\prec s, \Lambda) \prec t\}$
$\oplus \in \{\approx, \prec, \not\succ, \not\frown\} \wedge$ $(\oplus = \not\succ \ \Rightarrow \ t \in \mathcal{V})$	B	$s_Q \oplus t$	σ	$\emptyset$

Table 2. Ejection cases

Deterministic unification. The ejection and formation of unifying sets were formulated relatively to some unspecified set of positions Q. In unification of two terms s', s'', Q is the union of their frontiers, $\lfloor s' \rfloor \cup \lfloor s'' \rfloor$. In the first step, every non-variable term from $s'|_Q \cup s''|_Q$ was ejected and replaced by a new variable, in the second step, the literals $l' \cdot \sigma'$ and $l'' \cdot \sigma''$ containing s', s'' were considered and, if necessary, transformed. s', s'' became now $s'_Q\sigma(l', Q)$ and $s''_Q\sigma(l'', Q)$. The whole question is reduced now to unification of the latter two terms by a substitution, say ρ. The previous transformations were needed only to ensure that no nondeterministic term appears in ρ. Practical unification algorithms could, of course, proceed in other way, but this is another story.

4 Inference system

Unification is used in inference rules, as the case of *literal overlapping*. $\mathsf{mgu}(s, t)$ denotes the usual *most general unifier* of terms s, t.

Definition 8. Let $l' = s \oplus t$ and $l'' = u \otimes v$ be literals, p be a position in u above the frontier, $Q = \max(\lfloor s \rfloor \cup \lfloor u|_p \rfloor)$ be a set of positions, $s' \oplus' t' = l'_Q \cdot \sigma(l', Q)$, $u' \otimes' v' = l''_{pQ} \cdot \sigma(l'', pQ)$. Then the literal l' *overlaps* the literal l'' at a position p, if the substitution $\sigma(l', l'', p) = \mathsf{mgu}(s\sigma(l', Q), u|_p\sigma(l'', Q))$ called the *unifying substitution* exists.

Literal overlapping is not sufficient to derive new literal from l' and l'', the additional condition being that the relation $\ominus = \mathsf{Sup}(p, \oplus', \otimes')$ is non-trivial. In this case,

- the literal $\mathsf{L}(l', l'', p) \stackrel{\text{def}}{=} u'[t']_p \ominus v$ is called the *critical literal* formed by l' and l'';
- the literal set $\mathsf{B}(l', l'', p) \stackrel{\text{def}}{=} \mathsf{B}(l', Q) \cdot \sigma(l', Q) \cup \mathsf{B}(l'', pQ) \cdot \sigma(l'', pQ)$ is called the *binding set*;
- the clause $\mathsf{CC}(l', l'', p) \stackrel{\text{def}}{=} \mathsf{B}(l', l'', p) \cup \{\mathsf{L}(l, l', p)\}$, is called the *critical clause* formed by the l' and l'';
- the set of single clauses $\mathsf{A}(l', l'', p) \stackrel{\text{def}}{=} \{\{a \cdot \sigma(l', Q)\} : a \in \mathsf{A}(l', Q)\} \cup \{\{a \cdot \sigma(l'', pQ)\} : a \in \mathsf{A}(l'', pQ)\}$ is called the *assumption set.*

In the ground case [KW94] we only had the critical literal without any bindings or assumptions. The critical clause $\mathsf{CC}(l', l'', p)$, the assumption set $\mathsf{A}(l', l'', p)$ and the unifying substitution $\sigma(l', l'', p)$ are now used in the inference system $\mathcal{I}$, with the following rules:

Reflexivity resolution $\dfrac{C, s \oplus s'}{\{(B, C)\sigma\} \cup \mathcal{A}}$ where $\oplus$ is one of $\not\prec$, $\not\frown$ or $\not\succ$,

$\mathcal{A} = \mathsf{A}(l, \bar{l}, \Lambda)$, $B = \mathsf{B}(l, \bar{l}, \Lambda)$ and σ is a substitution $\sigma(l, \bar{l}, \Lambda)$ for $l = s \oplus s'$.

Superposition $\dfrac{C, a \qquad D, l}{(C, D, \mathsf{CC}(a, l, p))\sigma(a, l, p)}$ atom a overlaps literal l at p.

Compositionality resolution $\dfrac{C, s \oplus t \qquad D, s' \odot u, s'' \ominus w}{(C, t \otimes u, s \odot u)\sigma}$

where $\odot = \oplus \diamond \neg\otimes$ and $\sigma = \mathsf{mgu}\{s, s', s''\}$.

Theorem 9. *The inference system $\mathcal{I}$ is sound.*

Proof. To prove soundness of inference rules, which use so complicated unification, is not a trivial task. It consists of two subtask: 1) to prove soundness of unification on which the rules are based, 2) for each rule, to demonstrate particular property of predicates which is applied in the rule.

4.1 Ordering of words and the proof strategy

We assume the existence of a *simplification ordering* '<' on the ground terms [DJ90] to define a more specific proof strategy for the system $\mathcal{I}$, to study the possibility of rewriting wrt. the introduced predicates and, finally, to define the model in the completeness proof. The partial ordering of non-ground terms is derived from the ordering of ground ones according to the rule: $u > v \Leftrightarrow u\sigma > v\sigma$ for any ground substitution σ.

Our specific assumption about the orderings is that any deterministic term (from $\mathcal{D}$) is strictly smaller than any non-variable non-deterministic term (from $\mathcal{T}$). Thus any variable is smaller than non-variable term from $\mathcal{T}$. But in the set $\mathcal{D}$ of d-terms we have usual picture of term ordering.

Literals and clauses are identified with multisets and their ordering is defined by the *multiset extension* [DM79] of the term ordering.[4] A literal $s \oplus t$ is represented by the multiset $\{\{s, \oplus\}, \{t, \overline{\oplus}\}\}$. We assume that any term is bigger than any predicate symbol. A stronger positive predicate is bigger than a weaker one, the order between negative predicates is reversed, and all negative predicates are bigger than the positive ones:

$$\not\approx \;>\; \not\prec \;>\; \not\succ \;>\; \not\frown \;>\; \approx \;>\; \prec \;>\; \succ \;>\; \frown. \tag{18}$$

[4] It is possible to use sets instead of multisets but this would require definition of different new orderings on sets. For instance, if $t < s$ we want $t \approx s < s \approx s$. This is obtained directly using the multiset extension but not using extension to sets. Furthermore, multisets work uniformely when extending the ordering to the level of literals and then clauses. It is easier to work with such uniform extensions than with possibly different extensions to sets.

The ordering of the predicates will make the negated form of an atom bigger than the atom itself. Whenever possible, we suppose in a written literal $s \oplus t$ that $s < t$ is not the case. It explains why both signs '$\prec$' and '$\succ$' are used. This rule, of course, is not applied to the conclusions of the proof rules. The ordering of literals is the twofold extension of '$<$' because each literal is a multiset of two multisets. Clauses are compared as multisets of literals, so their ordering is the multiset extension of the ordering of literals (threefold multiset extension of '$<$'). Although we have here three different orderings, we will use the same symbol '$<$' to denote any of them. This should not introduce any confusion as the sets of terms, literals and clauses are disjoint.

The MAXIMAL LITERAL proof strategy. This proof strategy is known in the equational case as *ordered paramodulation*. The literals mentioned explicitly in the premises of the proof rules are called *active*. Various ways of selecting the active literals will lead to different proof strategies. The MAXIMAL LITERAL strategy requires that the active literals in the premise clauses are the ones which are maximal *wrt.* the ordering defined above. Stated explicitly the strategy amounts to the following restrictions on the application of the rules:

Reflexivity resolution: $(s \oplus s')\sigma$ is maximal in the clause $(C, s \oplus s')\sigma$.

Superposition: the atom $a\sigma$ and the literal $l\sigma$, where $\sigma = \sigma(a, l, p)$, are maximal in the respective clauses $(C, a)\sigma$ and $(D, l)\sigma$.

Compositionality resolution: the atom $(s \oplus t)\sigma$ is maximal in the clause $(C, s \oplus t)\sigma$. The maximal atom in the clause $(D, s' \odot u, s'' \ominus w)\sigma$ is $(s'' \ominus w)\sigma$, and $s' \odot u$ is an atom.

The important observation for our proof of completeness concerns the ordering of clauses in premisses and conclusions of the proof rules. The nondeterministic terms from bindings that appear in the conclusions are subterms of the active literals, and therefore binding literals are smaller than the active literals from premisses. So, if other new ("not contained in premisses") literals are smaller than the (maximal) active literals then the conclusion clause is smaller than the maximal of premisses clauses. Furthermore, new variables may be introduced instead of non-deterministic terms. But then the assumption that $\mathcal{D}$ (including variables!) are smaller than non-variable $\mathcal{T}$ makes the conclusion smaller than the respective premisses.

5 Completeness

In this section $\mathcal{L}$ denotes a set of ground literals and $\mathcal{A}$ a set of ground atoms. Such sets can be viewed as rewriting systems.

Definition 10. Let $\mathcal{L}$ be a set of ground literals, and r, r_1, r_2 be ground literals.

1. r is a *rewriting step* in $\mathcal{L}$ if either $r \in \mathcal{L}$, or r is an atom $u[s]_p \oplus u[t]_p$ for some term u, a position p in u, and an atom $s \otimes t \in \mathcal{L}$, where $\oplus \in \{\prec, \succ\}$, if $\otimes = \approx$, or $\oplus = \otimes$, otherwise.
2. A sequence of rewriting steps $\langle s \oplus_1 t_1,\ t_1 \oplus_2 t_2,\ \dots,\ t_n \oplus_n t\rangle$ is a *rewriting sequence* – the derived result is $s \oplus t$ with $\oplus = \oplus_1 \diamond \oplus_2 \diamond \cdots \diamond \oplus_n$.

3. A rewriting sequence is a *rewriting-proof* if it does not contain a *peak* (a pair of consecutive rewriting steps $s \oplus t$, $t \otimes u$ such that $s \leq t \geq u$.)
4. The *rewriting closure* of $\mathcal{L}$ is the set of ground literals, $\mathcal{L}^*$, defined as follows:
   - all ground atoms $s \prec s$ and $s \frown s$ belong to $\mathcal{L}^*$;
   - if an atom $s \oplus t \in \mathcal{L}$ and a literal $u[s]_p \otimes v \in \mathcal{A}^*$, then the literal $u[t]_p \odot v \in \mathcal{L}^*$, if $\odot = \mathsf{Sup}(p, \oplus, \otimes)$;
5. An atom $r_1 = s \oplus t$ *overlaps* $r_2 = u[s]_p \otimes v$ if $\odot = \mathsf{Sup}(p, \oplus, \otimes)$ is not trivial and $u[t]_p \odot v$, called a *critical literal*, is different from r_1 and r_2.
6. $\mathcal{L}$ is *confluent* if $\mathcal{L}^*$ contains all critical literals formed by all overlapping literals from $\mathcal{L}$.
7. A set of ground atoms $\mathcal{A}$ *forces*
   - a ground atom a if $a \in \mathcal{A}^*$, and the literal $\neg a$ if $a \notin \mathcal{A}^*$;
   - a ground clause C if it forces some literal $l \in C$;
   - a non-ground clause C if it forces any ground instance of C;
   - a set of clauses $\mathcal{S}$ if it forces all clauses from $\mathcal{S}$.

The next lemma is the only place where the predicate ordering is used directly.

Lemma 11. *Let $\mathcal{L}$ be confluent, $a \in \mathcal{L}$, an atom b, $b \leq a$, has a rewriting proof P in $\mathcal{L}$, but $b \notin (\mathcal{L} \setminus \{a\})^*$. Then a and b have the same maximal term, say s, and the proof P has the form a, P', where a is not used in the proof P'. If $a \neq b$, then the literal derived by P' is smaller than b.*

We are sketching the proof of *refutational completeness* of the inference system $\mathcal{I}$, *i.e.*, that there exists a model (multialgebra) satisfying all the clauses from $\mathcal{S}$ if the empty clause is not derivable from $\mathcal{S}$. We call a set of clauses $\mathcal{S}$ *consistent* if it does not contain the empty clause. The main result is

Theorem 12. *If a set of clauses $\mathcal{S}$ is consistent then it has a model.*

The construction proceeds in two main steps. Given a consistent set $\mathcal{S}$, we select a set of atoms $\mathcal{R}$ and show that $\mathcal{R}$ is a *forcing set* for $\mathcal{S}$. Then $\mathcal{R}$ can be used to construct a multimodel which satisfies $\mathcal{S}$ in the way it was done in [KW94].

Let $\mathcal{G}$ denote the set of ground instances of form $\mathcal{S}$, *i.e.*, of clauses $C\sigma$, where $C \in \mathcal{S}$ and σ is a ground substitution (involving only Φ functional symbols.). The starting point of the model construction is the set of maximal literals of ground instances of $\mathcal{S}$:

$$\mathcal{L}_0 \stackrel{\text{def}}{=} \{\max(C) : C \in \mathcal{G}\}. \tag{19}$$

For an $\mathcal{L}$ and an $l \in \mathcal{L}$, we define the set $\mathcal{L}_l \stackrel{\text{def}}{=} \{a \in \mathcal{L} : a < l\}$

Definition 13. Let $\mathcal{A}$, $\mathcal{G}$ be sets of ground atoms and clauses, respectively, C be a ground clause and l a ground literal.

1. C with $\max(C) = l$ is *redundant* in $\mathcal{A}$ and $\mathcal{G}$ if either
   - C is forced by $\mathcal{A}_l$ or
   - the set $\mathcal{G}$ contains another clause $C' < C$ with $\max(C') = l$, and C' is not forced by $\mathcal{A}_l$.

2. l is *redundant* in $\mathcal{A}$ and $\mathcal{G}$ if either
   - $l = s \oplus s'$, where $\oplus \in \{\not\succ, \not\succ, \not\succ\}$, l overlaps $\bar{l}$ at the top position, and the clause $\mathsf{B}(l, \bar{l}, \Lambda)$ is not forced by $\mathcal{A}$, or
   - $\mathcal{A} \cup \{l\}$ contains an atom r overlapping l at a position p, and such that the critical clause $\mathsf{CC}(r, l, p) < \{l\}$ and is not forced by $\mathcal{A}$, or
   - every clause $C \in \mathcal{G}$ with $\max(C) = l$ is redundant in $\mathcal{A}$.
3. C, l with $\max(C, l) = l$ is *productive* for l in $\mathcal{A}$ if $\mathcal{A}$ does not force C.

Definitions 13.2 and 10.7 of redundancy and forcing are so related, that all negative literals that are not forced are redundant. Since any forced literal makes redundant all clauses containing it, any negative literal appears redundant.

If $\mathcal{L}_i$ is known, and l_i is the minimal redundant literal in $\mathcal{L}_i$, we let:

$$\mathcal{L}_{i+1} \stackrel{\text{def}}{=} \mathcal{L}_i \setminus \{l_i\}, \qquad \mathcal{R} \stackrel{\text{def}}{=} \textstyle\bigcap_{i \in \mathbb{N}} \mathcal{L}_i. \tag{20}$$

For a literal $l \in \mathcal{R}$ we denote by $\mathcal{R}_{l'}$ the set $\mathcal{R}_l \cup \{l\}$, while for $l \notin \mathcal{R} : \mathcal{R}_{l'} = \mathcal{R}_l$. Before sketching the proof that $\mathcal{R}$ is the forcing set for $\mathcal{S}$, we list some of its properties.

Lemma 14. *$l \in \mathcal{L}_0$ (a clause $C \in \mathcal{G}$ with $\max(C) = l$) is redundant in some $\mathcal{L}_i$ with $l_i > l \Leftrightarrow$ it is redundant in every $\mathcal{L}_j$ with $j > i \Leftrightarrow$ it is redundant in $\mathcal{R}$.*

Lemma 15. *$\mathcal{R}$ is confluent.*

Lemma 16. *For any $l \in \mathcal{R}$, there is a clause $C \in \mathcal{G}$ productive for l in $\mathcal{R}_l$.*

Definition 17. A set $\mathcal{G}$ is *relatively closed* if any application of a rule from $\mathcal{I}$ with premises from $\mathcal{G}$ yields a clause whose each ground instance is in $\mathcal{G}$ or is redundant in $\mathcal{G}$.

The main technical result, giving the completeness theorem is

Theorem 18. *Let $\mathcal{G}$ be consistent and relatively closed set of ground clauses, $\mathcal{L}_0$ and $\mathcal{R}$ be defined by (19) and (20). Each $l \in \mathcal{L}_0$ satisfies the following conditions:*

I1. *if l is not redundant in $\mathcal{R}$, then for any $a \in \mathcal{R}_l$ there exists a clause $C \in \mathcal{G}$ productive for a in $\mathcal{R}_l$,*

I2. *if l is redundant in $\mathcal{R}$, then $\mathcal{R}_l$ forces any clause $C \in \mathcal{G}$ with $\max(C) = l$.*

Proof. By contradiction, assuming that l is a minimal literal in $\mathcal{L}_0$ not satisfying the theorem, i.e., either

B1. l, being included in $\mathcal{R}$, "spoils" productiveness of some clause $C \in \mathcal{G}$ with $a = \max(C) \leq l$, *i.e.*, C is productive for a in $\mathcal{R}_l$, but is not productive in $\mathcal{R}_l \cup \{l\}$, or

B2. l, being not included in $\mathcal{R}$, leaves unforced by $\mathcal{R}_l$ some clause $C \in \mathcal{G}$ with $\max(C) = l$.

The following lemma allows us to obtain contradiction in both cases:

Lemma 19. *The following conditions about a clause D and a literal l cannot be satisfied simultaneously in a relatively closed $\mathcal{G}$:*

1. *D is a ground instance of conclusion of some proof rule from $\mathcal{I}$ with premises from $\mathcal{G}$,*

2. *for any clause* $D' \leq D$, *if* $D' \in \mathcal{G}$, *then* $\mathcal{R}_l$ *forces* D',
3. $\mathcal{R}_l$ *does not force* D,
4. $\max(D) < l$.

Assumption B1 means that $\max(C) = a$, and $C \setminus \{a\}$ is not forced by $\mathcal{R}_l$, but there exists $b \in C$ such that $b \neq a$ and $b \in \mathcal{R}^*_{l'}$. By Lemma 16, there is a clause D, l such that $l = \max(D)$ and D is not forced by $\mathcal{R}_l$. Lemma 11 implies that we can apply the compositionality resolution to C and D, l producing the clause $E = D, b, \neg c$ with $\max(E) < l$ which satisfy the conditions of Lemma 19 thus leading to a contradiction.
Assuming B2, we have a clause D such that $C = (D, l)$, $\max(D) < l$ and C is not forced by $\mathcal{R}_l$. Then l is redundant and we analyse the three alternatives of Definition 13.2. The last one is impossible since C is non-redundant. In the first case, contradiction follows from Lemma 19 after application of reflexivity resolution. In the second case, Lemma 16 and superposition rule give again a pair satisfying the conditions of Lemma 19, thus yielding a contradiction.

Thus $\mathcal{R}$ is the forcing set for $\mathcal{S}$. The construction of a multialgebra from $\mathcal{R}$ is given in [KW94].

6 Conclusions and directions for future work

We presented a theoretical case study of transferring the ideas from ordered paramodulation to a particular non-equational reasoning system. We concentrated on a particular logic, but the obtained results are of general interest. Firstly, because reasoning with set-valued relations is very common and, secondly, because the introduced techniques should be applicable to the more general binary relations. Together with L. Bachmair and H. Ganzinger, [BG94b, BG95], we have shown how term-rewriting techniques for theorem proving and methods of restricting inference rules can be applied to more general binary relations than congruences. Specifically, we demonstrated how term-rewriting can work in a languge with a very restricted substitutivity into variables and in the presence of existential quantification.

We were not concerned here with inference restrictions other than those imposed by the term ordering. Probably, also the ideas from *basic paramodulation* [BGLS95] could be transferred to our logic, because the level of deterministic terms is entirely analogous to the equational case.

Our results may be particularly relevant to the area of relational specifications [BGS94], where composition of relations also hides existential quantification. We intend to investigate the possible applications and extensions in this direction [KMW95]. Another interesting possibility would be to extend our results to typing relations.

References

[Bez90] M. Bezem. Completeness of Resolution Revisited. *Theoretical Computer Science*, 74, pp.27-237, (1990).

[BG94a] L. Bachmair, H. Ganzinger. Rewrite-Based Equational Theorem Proving with Selection and Simplification. *Journal of Logic and Computation*, 4(3), pp. 217–247 (1994).

[BG94b] L. Bachmair, H. Ganzinger. Rewrite Techniques for Transitive Relations. *Proc. 9th IEEE Symposium on Logic in Computer Science*, pp. 384–393. IEE Computer Society Press (1994).

[BG95] L. Bachmair, H. Ganzinger. *Ordered Chaining Calculi for First-Order Theories of Binary Relations.* Technical Report MPI-I-95-2-009, Max-Planck-Institut f. Informatik, Saarbrücken, (1995).

[BGLS95] L. Bachmair, H. Ganzinger, C. Lynch, W. Snyder. Basic paramodulation. *Information and Computation*, 121(2), pp. 172–192 (1995).

[BGS94] R. Berghammer, T. Gritzner, G. Schmidt. Prototyping relational specifications using higher-order objects. In *Proc. Int. Workshop on Higher Order Algebra, Logic and Term Rewriting (HOA'93)*, Amsterdam, The Netherlands, Sept. 1993, LNCS, vol. 816, pp. 56–75 (1994).

[BK95] M. Białasik, B. Konikowska, *Reasoning with Nondeterministic Specifications*, ICS PAS Report no. 793, Institute of Computer Science, Polish Academy of Sciences, (1995).

[DJ90] N. Dershowitz, J.-P. Jouannaud. Rewrite systems. In: J. van Leeuwen (ed.) *Handbook of theoretical computer science*, vol. B, chap. 6, pp.243-320. Amsterdam: Elsevier, (1990).

[DM79] N. Dershowitz, Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22:8,pp.465-476, (1979).

[DO92] A. Dovier, E. Omodeo, E. Pontelli, G.-F. Rossi. Embedding finite sets in a logic programming language. LNAI, vol. 660, pp.150-167, (1993).

[Hes88] W.H. Hesselink. A Mathematical Approach to Nondeterminism in Data Types. *ACM Trans. on Programming Languages and Systems*, 10, pp.87-117, (1988).

[Hus92] H. Hussmann. Nondeterministic algebraic specifications and nonconfluent term rewriting. *Journal of Logic Programming*, 12, pp.237-235, (1992).

[Hus93] H. Hussmann. *Nondeterminism in Algebraic Specifications and Algebraic Programs.* Birkhäuser Boston, (1993).

[Jay92] B. Jayaraman. Implementation of Subset-Equational Programs. *Journal of Logic Programming*, 12:4, pp.299-324, (1992).

[Kap88] S. Kaplan. Rewriting with a Nondeterministic Choice Operator. *Theoretical Computer Science*, 56:1, pp.37-57, (1988).

[KMW95] V. Kriaučiukas, S. Meldal, M. Walicki. Nondeterministic Algebraic Specifications in Relational Syntax, in *Proc. of 7th Nordic Workshop on Programming Theory*, Report No. 86, Programming Methodology Group, Göteborg University, (1995).

[KW94] V. Kriaučiukas, M. Walicki. Reasoning and Rewriting with Set-Relations I: Ground-Completeness. In *Proceedings of CSL'94*, LNCS, vol. 933, pp. 264–278, (1995).

[LA93] J. Levy, J. Agustí. Bi-rewriting, a term rewriting technique for monotonic order relations. In *RTA'93*, LNCS, vol. 690, pp.17-31, (1993).

[Mos89] P.D. Mosses. Unified algebras and institutions. In *LICS'89, Proc. 4th Ann. Symp. on Logic in Computer Science*, pp. 304–312, IEEE, (1989).

[SL91] W. Snyder, C. Lynch. Goal directed strategies for paramodulation. In *Proc. 4th Int. Conf, on Rewriting Techniques and Applications*, LNCS, vol. 488, pp. 150–161, Berlin, (1991).

[PP91] J. Pais, G.E. Peterson. Using Forcing to Prove Completeness of Resolution and Paramodulation. *Journal of Symbolic Computation*, 11:(1/2), pp.3-19, (1991).

[SD86] J. Schwartz, R. Dewar, E. Schonberg, E. Dubinsky. *Programming with sets, an introduction to SETL.* Springer Verlag, New York, (1986).

[Wal93] M. Walicki. *Algebraic Specifications of Nondeterminism.* Ph.D. thesis, Institute of Informatics, University of Bergen, (1993).

[WM95a] M. Walicki, S. Meldal. Multialgebras, Power algebras and Complete Calculi of Identities and Inclusions. In *Recent Trends in Data Type Specification*, LNCS, vol. 906, pp. 453–468 (1995).

[WM95b] M. Walicki, S. Meldal. A Complete Calculus for Multialgebraic and Functional Semantics of Nondeterminism. *ACM Trans. on Programming Languages and Systems*, vol. 17, No. 2, pp. 366–393 (1995).