Proc. 5. Österr. AI-Tagung, J.Retti,K.Leidlmair (eds.), Springer-Verlag (1989)

Equational Theorem Proving and Rewrite Rule Systems *)

Franz Winkler

Institut für Mathematik and
Research Institute for Symbolic Computation
Johannes Kepler Universität Linz

Abstract

Equational theorem proving is interesting both from a mathematical and a computational point of view. Many mathematical structures like monoids, groups, etc. can be described by equational axioms. So the theory of free monoids, free groups, etc. is the equational theory defined by these axioms. A decision procedure for the equational theory is a solution for the word problem over the associated algebraic structure. From a computational point of view, abstract data types are basically described by equations. Thus, proving properties of an abstract data type amounts to proving theorems in the associated equational theory.

One approach to equational theorem proving consists in associating a direction with the equational axioms, thus transforming them into rewrite rules. Now in order to prove an equation $a = b$, the rewrite rules are applied to both sides, finally yielding reduced versions a' and b' of the left and right hand sides, respectively. If a' and b' agree syntactically, then the equation holds in the equational theory. However, in general this argument cannot be reversed; a' and b' might be different even if $a = b$ is a theorem. The reason for this problem is that the rewrite system might not have the Church–Rosser property. So the goal is to take the original rewrite system and transform it into an equivalent one which has the desired Church–Rosser property.

We show how rewrite systems can be used for proving theorems in equational and inductive theories, and how an equational specification of a problem can be turned into a rewrite program.

1. Introduction

We give an overview of "automated reasoning with equations". Equations, i.e. formulas of the form $s = t$, are very common in mathematics, logic, and computer science. We concentrate on two questions concerning such equations: (1) how can we automatically deduce new equations from given ones, and (2) how can we compute with such equations?

Example 1.1: The abstract data type $QUEUE$ (over some element type EL) contains operations *newq* (constructing the empty queue), *app* (appending two queues), and *add* (adding a new element to a queue).

*) Work reported herein has been supported by the *Fonds zur Förderung der wissenschaftlichen Forschung* under Projekt Nr. P6763.

These operations have to satisfy the equations

(Q1) $app(x, newq) = x$,
(Q2) $app(x, add(y, z)) = add(app(x, y), z)$,
(Q3) $app(app(x, y), z) = app(x, app(y, z))$.

Do the equations (Q1) – (Q3) imply that

(H) $app(x, app(add(y, z), w)) = app(add(app(app(x, newq), y), z), w)$?

In fact they do, because

$$\begin{aligned} &\underline{app(x, app(add(y, z), w))} =_{Q3,\Leftarrow} \\ &app(\underline{app(x, add(y, z))}, w) =_{Q2,\Rightarrow} \\ &app(add(app(\underline{x}, y), z), w) =_{Q1,\Leftarrow} \\ &app(add(app(app(x, newq), y), z), w). \end{aligned}$$

Here by the subscripts we indicate which equation is used on which subterm and in which direction, i.e. whether the left hand side is replaced by the associated right hand side or vice versa.

In general the problem is that we do not know in which sequence the given equations have to be used and in which direction. Furthermore we do not have an upper bound on the lenght of a derivation. ■

Example 1.2: In mathematics a *group* is a set with operations 1 (arity 0), $\cdot$ (arity 2), $^{-1}$ (arity 1), satisfying the group axioms

(G1) $1 \cdot x = x$,
(G2) $x^{-1} \cdot x = 1$,
(G3) $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

Again we are interested in deciding equations of the form $s = t$, where both s and t are terms constructed from variables and the group operations, for instance

$$(x^{-1} \cdot (x \cdot y))^{-1} = (x^{-1} \cdot y)^{-1} \cdot x^{-1} \ ?$$ ■

Example 1.3: We consider an equational definition of the quotient and remainder of non-negative integers. The predicate div is defined as

$$div(x, y, q, r) \iff x = q \cdot y + r \quad \wedge \quad r < y.$$

So an equational definition of the predicate div is

(D1) $div(u + y + 1, y + 1, q + 1, r) = div(u, y + 1, q, r)$,
(D2) $div(x, x + u + 1, 0, x) = true$.

We want to use such an equational definition for actually computing the quotient and remainder of two nonnegative numbers. ■

In the following chapters we define the syntax and semantics of equational theories, show how term rewriting systems can be applied for deciding equations modulo equational

theories, describe the Knuth–Bendix completion procedure for term rewriting systems, discuss the termination of the Knuth–Bendix procedure, and show how the completion procedure can be applied for proving theorems in inductive theories and for programming with rewrite rules. For extensive introductions to equational theories, rewrite rules and the completion procedure we refer to [HO 80], [Wi 84] and [Bu 85]. There the interested reader may also find proofs of the theorems.

2. Equational theories

Let $\mathcal{S}$ be a set of *sorts* (for instance, $\mathcal{S} = \{EL, QUEUE\}$ in Example 1.1). A *signature* over $\mathcal{S}$ is a set of operators Σ together with an (implicit) typing function T, which assigns to every operator $f \in \Sigma$ a type $\mathrm{T}(f)$ of the form $S_1 \times \cdots \times S_n \to S$, where $S_1, \ldots, S_n, S \in \mathcal{S}$. If $n = 0$, then the operator f is called a *constant*.

Given a signature Σ over a set of sorts $\mathcal{S}$, a Σ–*algebra* is an algebraic structure "implementing" the sorts and operators of Σ. I.e. a Σ–algebra is a pair $(\mathcal{A}, \mathcal{F})$, where $\mathcal{A}$ is an $\mathcal{S}$–indexed family of sets ($\mathcal{A}_S$, for $S \in \mathcal{S}$, is called the *carrier set* of sort S) and $\mathcal{F}$ is a Σ–indexed family of functions (the operations of the algebra) such that

- $\mathcal{F}_f$ is a constant in $\mathcal{A}_S$ if $\mathrm{T}(f) = S$, and
- $\mathcal{F}_f$ is a function from $\mathcal{A}_{S_1} \times \cdots \times \mathcal{A}_{S_n}$ to $\mathcal{A}_S$ if $\mathrm{T}(f) = S_1 \times \cdots \times S_n \to S$.

A special Σ–algebra is the *algebra of ground terms* or *initial algebra* $\mathcal{G}(\Sigma)$. For every sort S the carrier of sort S, $\mathcal{G}(\Sigma)_S$, contains every constant of type S in Σ and if $\mathrm{T}(f) = S_1 \times \cdots \times S_n \to S$ and $t_i \in \mathcal{G}(\Sigma)_{S_i}$ for $1 \leq i \leq n$, then $ft_1 \ldots t_n$ is in $\mathcal{G}(\Sigma)_S$. Nothing else is in the carrier of $\mathcal{G}(\Sigma)$. An element of $\mathcal{G}(\Sigma)_S$ is called a *ground term of sort* S. The operations of this algebra are defined such that $\mathcal{F}_f$ takes ground terms $t_1, \ldots, t_n$ (of the appropriate sorts) and maps them to the ground term $ft_1 \ldots t_n$. For better readability we often write $f(t_1, \ldots, t_n)$ instead of $ft_1 \ldots t_n$.

Admitting additional special ground terms $V_S = \{v_S^1, v_S^2, \ldots\}$ of every sort S, V an $\mathcal{S}$–indexed family of sets, we get the *term algebra* $\mathcal{T}(\Sigma, V)$ over the signature Σ and the variable set V. An element of $\mathcal{T}(\Sigma, V)_S$ is called a *term of sort* S.

Example 2.1: Let $\mathcal{S} = \{NAT, QUEUE\}$ be the set of sorts, Σ the following signature

operator	type
0	NAT
$succ$	$NAT \to NAT$
$newq$	$QUEUE$
app	$QUEUE \times QUEUE \to QUEUE$
add	$QUEUE \times NAT \to QUEUE$

Then 0, $succ(succ(0))$, $succ(succ(succ(succ(0))))$ are ground terms of sort NAT over Σ, and $newq$, $add(app(newq, newq), succ(0))$ are ground terms of sort $QUEUE$ over Σ.

We choose $V_{NAT} = \{x_1, x_2, \ldots\}$ as variables of sort NAT and $V_{QUEUE} = \{y_1, y_2, \ldots\}$

as variables of sort $QUEUE$. Then 0, $succ(succ(x_5))$ are terms of sort NAT, and $newq$, $add(app(y_7, newq), succ(x_3))$ are terms of sort $QUEUE$. ■

The two basic operations on terms are *substitution* and *replacement* of subterms. A substitution is a homomorphism of the term algebra $\mathcal{T}(\Sigma, V)$ into itself, which changes only a finite number of variables. If the substitution σ changes the variable v_1 to the term t_1, ... ,v_k to the term t_k and leaves every other variable unchanged, we write σ as $\{t_1 \rightarrow v_1, \ldots, t_k \rightarrow v_k\}$. A replacement is a partial function of two terms s, t and an *occurrence* (sequence of positive integers) $p = p_1 p_2 \cdots p_l$, which changes the subterm of s at p to the term t. We write $s[p \leftarrow t]$ for the replacement of the subterm of s at p by t. If $l = 0$, i.e. p is the empty sequence Λ, then $s[p \leftarrow t] = t$. If $l > 0$ and $s =$

$$f_1(t^1_1, ..., t^1_{p_1-1}, f_2(t^2_1, ..., t^2_{p_2-1}, \cdots f_l(t^l_1, ..., \boldsymbol{t^l_{p_l}}, ..., t^l_{n_l}) \cdots, t^2_{p_2+1}, ..., t^2_{n_2}), t^1_{p_1+1}, ..., t^1_{n_1})$$

then $s[p \leftarrow t] =$

$$f_1(t^1_1, ..., t^1_{p_1-1}, f_2(t^2_1, ..., t^2_{p_2-1}, \cdots f_l(t^l_1, ..., \boldsymbol{t}, ..., t^l_{n_l}) \cdots, t^2_{p_2+1}, ..., t^2_{n_2}), t^1_{p_1+1}, ..., t^1_{n_1}).$$

Otherwise the replacement is undefined. By s/p we denote the *subterm of s at p*. In the notation above $s/p = s$ if $p = \Lambda$ and otherwise $s/p = t^l_{p_l}$.

Definition 2.2: Let Σ be a signature and V a variable set over the set of sorts $\mathcal{S}$. An *equation* over $\mathcal{T}(\Sigma, V)$ is a pair of terms s, t, written as $s = t$, where s and t are of the same sort.

Now let E be a set of equations and s, t two terms in $\mathcal{T} = \mathcal{T}(\Sigma, V)$. Then s and t are *provably equal modulo E*, or $s = t$ is *provable modulo E* ($E \vdash s = t$) iff $s = t$ can be derived in the following *equational calculus*:

(G1) $\dfrac{}{u_1 = u_2}$ for all $u_1 = u_2 \in E$

(G2) $\dfrac{}{u_1 = u_1}$, $\dfrac{u_1 = u_2}{u_2 = u_1}$, $\dfrac{u_1 = u_2, u_2 = u_3}{u_1 = u_3}$ (reflexivity, symmetry, transitivity)

(G3) $\dfrac{u_1 = u_2}{\sigma(u_1) = \sigma(u_2)}$ (substitution rule)

(G4) $\dfrac{u_1 = u'_1, \ldots, u_n = u'_n}{f(u_1, \ldots, u_n) = f(u'_1, \ldots, u'_n)}$ (replacement rule)

where the u's are terms, σ is a substitution, and f is an operator of appropriate type.

The set $=_E = \{s = t \mid E \vdash s = t\}$ is called the *equational theory generated by E*, and E is called a *basis* of $=_E$.

For convenience we often write $s =_E t$ instead of $E \vdash s = t$. ■

Provability of an equation is a purely syntactical notion. Now let us turn to the semantics of equations, i.e. a definition of validity.

Definition 2.3: Let Σ be a signature and V a variable set over $\mathcal{S}$. Let $(\mathcal{A}, \mathcal{F})$ be a Σ–algebra. An *assignment* in $(\mathcal{A}, \mathcal{F})$ is a homomorphism ν from $\mathcal{T}(\Sigma, V)$ in $(\mathcal{A}, \mathcal{F})$. Such an assignment is uniquely determined by the values of the variables in V and by the functions associated with the operators in Σ.

An equation $s = t$ over $\mathcal{T}(\Sigma, V)$ is *valid* in $(\mathcal{A}, \mathcal{F})$ (or $(\mathcal{A}, \mathcal{F})$ is a *model* of $s = t$) iff for all assignments ν in $(\mathcal{A}, \mathcal{F})$ we have $\nu(s) = \nu(t)$.

The set of Σ–algebras in which all the equations of a set of equations E are valid, is called the *variety* of E, $var(\Sigma, E)$.

s and t are *semantically equal modulo* E, or $s = t$ is *valid modulo* E ($E \models s = t$) iff $s = t$ is valid in every Σ–algebra in $var(\Sigma, E)$. ∎

Example 2.4: Let $\mathcal{S}$, Σ, and V be as in Example 2.1. In Example 1.1 we have shown that $\{(\text{Q1}), (\text{Q2}), (\text{Q3})\} \vdash H$.

Let $\mathcal{S} = \{S\}$, $\Sigma = \{1, \cdot, ^{-1}\}$, $V = \{x, y, z, \ldots\}$ and $G = \{(\text{G1}), (\text{G2}), (\text{G3})\}$ as in Example 1.2. We show that

$$x^{-1} \cdot ((y^{-1} \cdot y) \cdot x) = z^{-1} \cdot z$$

is valid modulo G, i.e. in the equational theory of free groups. Let $(A, \{one, times, inv\})$ be a Σ–algebra in $var(\Sigma, G)$, i.e. for all $a, b, c \in A$

(G'1) $times(one, a) = a$
(G'2) $times(inv(a), a) = one$
(G'3) $times(times(a, b), c) = times(a, times(b, c))$

Then no matter what the assignment ν assigns to the variables x, y, z, we get

$$\begin{aligned}\nu(x^{-1} \cdot ((y^{-1} \cdot y) \cdot x)) = times(inv(\nu(x)), times(times(inv(\nu(y)), \nu(y)), \nu(x))) &=_{(\text{G}'2)} \\ times(inv(\nu(x)), times(one, \nu(x))) &=_{(\text{G}'1)} \\ times(inv(\nu(x)), \nu(x)) =_{(\text{G}'2)}\ one &=_{(\text{G}'2)} \\ times(inv(\nu(z)), \nu(z)) = \nu(z^{-1} \cdot z).&\end{aligned}$$

So $G \models x^{-1} \cdot ((y^{-1} \cdot y) \cdot x) = z^{-1} \cdot z$. ∎

In contrast to the situation in predicate calculus the notions of provability and validity coincide for equational theories.

Theorem 2.5 (Birkhoff [Bi 35]): Let E be the basis of an equational theory over Σ and V, and s, t terms in $\mathcal{T}(\Sigma, V)$. Then

$$E \models s = t \quad \Longleftrightarrow \quad E \vdash s = t.$$ ∎

So for deciding the validity of an equation modulo E is suffices to show that the equation can be syntactically derived from E.

3. Rewrite rule systems

One approach to deciding equational theories $=_E$ is to use rewrite rule systems. The idea is to associate a direction (left to right or right to left) with every equation of the

basis E. This leads to a "reduction relation" $\rightarrow_E$ on the terms such that $=_E$ is equal to the reflexive, symmetric, transitive closure of $\rightarrow_E$. Then the decision problem for $=_E$ can be solved if $\rightarrow_E$ has some nice properties: (1) $\rightarrow_E$ always terminates after a finite number of applications and (2) $\rightarrow_E$ has the so called Church-Rosser property. We will deal with (1) and (2) in this chapter. But first we have to give some definitions and present basic facts about such rewrite rule systems.

From now on we assume that a set of sorts $\mathcal{S}$, a signature Σ and a variable set V over $\mathcal{S}$ are given. By $\mathcal{T}$ we denote the set of terms over Σ and V.

Definition 3.1: A *rewrite rule* is a pair of terms $s \rightarrow t$ of equal sort such that every variable occurring in t also occurs in s. A *rewrite rule system* (or *rrs* for short) is a set of rewrite rules.

A rewrite rule system R generates a relation $\rightarrow_R$ on the set of terms $\mathcal{T}$ in the following way:
$s \rightarrow_R t \iff$ there exists a rule $l \rightarrow r \in R$, a substitution σ, and an occurrence p in s, such that $s/p = \sigma(l)$ and $t = s[p \leftarrow \sigma(r)]$. ■

Whenever we have a binary relation $\rightarrow$ on a set M, then by $\rightarrow^+$, $\rightarrow^*$, $\leftrightarrow^*$ we denote the transitive, reflexive–transitive, and reflexive–transitive–symmetric closure of $\rightarrow$, respectively. By $\leftarrow$ we denote the inverse of $\rightarrow$. Furthermore we define
$x \uparrow y \iff$ there exists a z such that $x \leftarrow z \rightarrow y$,
$x \uparrow^* y \iff$ there exists a z such that $x \leftarrow^* z \rightarrow^* y$,
$x \downarrow y \iff$ there exists a z such that $x \rightarrow z \leftarrow y$,
$x \downarrow^* y \iff$ there exists a z such that $x \rightarrow^* z \leftarrow^* y$.

Definition 3.2: Starting from a term t and successively reducing it by the rules in some rewrite rule system R might lead (after finitely many steps) to a term t' which cannot be reduced further: $t \rightarrow_R t_1 \rightarrow_R t_2 \rightarrow_R \cdots \rightarrow_R t_k \rightarrow_R t'$. In this case t' is called a *normal form of t modulo R*. t' is said to be *irreducible* or *in normal form modulo R*.

If every reduction chain $t \rightarrow_R t_1 \rightarrow_R t_2 \rightarrow_R \cdots$ terminates with a normal form then R is a *terminating* or *Noetherian* rewrite rule system. ■

If we are able to orient the equations in some set of equations E such that we get a rewrite rule system R, then instead of studying $=_E$ we could just study $\leftrightarrow^*_R$.

Lemma 3.3: Let R be a rewrite rule system, E the set of equations $\{l = r \mid l \rightarrow r \in R\}$. Then $\leftrightarrow^*_R = =_E$.

Example 3.4: Orienting the group axioms (Examples 1.2 and 2.4) leads to the rewrite rule system RG:

$$\begin{array}{ll} (RG1) & 1 \cdot x \rightarrow x, \\ (RG2) & x^{-1} \cdot x \rightarrow 1, \\ (RG3) & (x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z). \end{array}$$

For this rewrite rule system we have $\leftrightarrow^*_{RG} = =_G$. Using RG we can reduce
$x^{-1} \cdot ((y^{-1} \cdot y) \cdot x) \rightarrow_{(RG2)} x^{-1} \cdot (1 \cdot x) \rightarrow_{(RG1)} x^{-1} \cdot x \rightarrow_{(RG2)} 1$ and
$z^{-1} \cdot z \rightarrow_{(RG2)} 1$.

So $x^{-1} \cdot ((y^{-1} \cdot y) \cdot x) \leftrightarrow^*_{RG} z^{-1} \cdot z$ and therefore $x^{-1} \cdot ((y^{-1} \cdot y) \cdot x) = z^{-1} \cdot z$ is in the equational theory generated by G.

On the other hand, also $x \cdot 1 = x$ is in $=_E$, but this fact cannot be proved by reduction modulo RG. Both sides of this equation are irreducible modulo RG. ∎

So in general, for deciding $\leftrightarrow^*_R$ it is not enough to reduce both sides to normal form and check for syntactic equality. What we need is the so called Church–Rosser property, which can be defined for arbitrary binary relations on some set M.

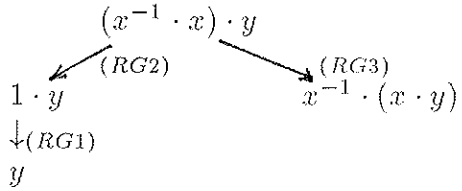
Definition 3.5: Let $\rightarrow$ be a binary relation on a set M.
$\rightarrow$ is *Church–Rosser* or *complete* iff $x \leftrightarrow^* y \implies x \downarrow^* y$ for all $x, y \in M$.
$\rightarrow$ is *confluent* iff $x \uparrow^* y \implies x \downarrow^* y$ for all $x, y \in M$.
$\rightarrow$ is *locally confluent* iff $x \uparrow y \implies x \downarrow^* y$ for all $x, y \in M$. ∎

For Noetherian rewrite rule systems the test for the Church–Rosser property can be reduced to the test for local confluence.

Theorem 3.6: Let $\rightarrow$ be a binary relation on a set M.
(a) $\rightarrow$ is Church–Rosser if and only if $\rightarrow$ is confluent.
(b) If $\rightarrow$ is Noetherian, then $\rightarrow$ is confluent if and only if it is locally confluent. ∎

Observe that modulo a complete rrs R every term t has a uniquely defined normal form. This can be seen by the following argument: if t' and t'' are two normal forms of t, then $t' \leftrightarrow^*_R t''$, so there must be a term u such that $t' \rightarrow^*_R u \leftarrow^*_R t''$. But both t' and t'' are irreducible, so $t' = u = t''$.

Example 3.7: The rewrite rule system (RG) is not locally confluent. This can be seen by considering the two terms $1 \cdot y$ and $x^{-1} \cdot (x \cdot y)$. They have a common predecessor, but $1 \cdot y$ can only be reduced to y and the second term is irreducible.



∎

If M is a term algebra and $\rightarrow$ is the reduction relation modulo a rewrite rule system R, then it suffices to carry out the test for local confluence for finitely many so called "critical" situations $x \leftarrow z \rightarrow y$.

Definition 3.8: Let R be a rewrite rule system, $l_1 \to r_1$, $l_2 \to r_2$ two rules in R, p an occurrence in l_1 such that l_1/p is not a variable. Furthermore assume that l_1/p and l_2 can be unified by a most general unifier σ such that $\sigma(l_1/p)$ and l_1 have no variables in common. Then $(\sigma(l_1[p \leftarrow r_2]), \sigma(r_1))$ is a *critical pair* in R. ■

Observe that the critical pairs are the results of reducing "smallest" terms which can be reduced in two different ways. $\sigma(l_1[p \leftarrow r_2]) \leftarrow_R \sigma(l_1) \to \sigma(r_1)$.

Theorem 3.9 (Knuth, Bendix [KB 67]): Let R be a rewrite rule system such that $\to_R$ is Noetherian. Then $\to_R$ is Church–Rosser if and only if $c_1 \downarrow^* c_2$ for all critical pairs (c_1, c_2) in R. ■

We can use Theorem 3.9 to test any given rewrite rule system R whether it has the Church–Rosser property. Now suppose that in the course of this test we find a critical pair (c_1, c_2) in R such that $c_1 \to_R^* c_1'$, $c_2 \to_R^* c_2'$, c_1, c_2 are irreducible modulo R and $c_1 \neq c_2$. If $c_1' \to c_2'$ (or $c_2' \to c_1'$) is a rewrite rule which does not disturb the termination property, then we can add it to the rule system R and thereby force a common successor of c_1' and c_2'. This enlargement of the rule system is correct, because $c_1' \leftrightarrow_R^* c_2'$ and therefore $\leftrightarrow_R^* = \leftrightarrow_{R \cup \{c_1' \to c_2'\}}^*$. If this process stops, then we get an rrs which is Church–Rosser and terminating. Such a rrs is called *canonical.*

Completion procedure ([KB 67]) 3.10:
Input: R, a finite Noetherian rrs
Output: R', a finite Noetherian rrs, such that $\leftrightarrow_R^* = \leftrightarrow_{R'}^*$ and $\to_{R'}$ is Church–Rosser or "failure"

$R' := R$; $C :=$ set of critical pairs in R;
while $C \neq \emptyset$ **do**
    $(c_1, c_2) :=$ an element of C;
    $C := C \setminus \{(c_1, c_2)\}$;
    $(d_1, d_2) :=$ normal forms of (c_1, c_2) modulo $\to_{R'}$;
    if $d_1 \neq d_2$ and $R'' = R' \cup \{d_1 \to d_2\}$ (or $R'' = R' \cup \{d_2 \to d_1\}$) is a Noetherian rrs
    then $R' := R''$
    else exit with "failure"
    endif;
    $C := C \cup$ set of critical pairs formed with the new rule
endwhile ■

This completion procedure does not always terminate with a complete rrs. Firstly, a pair of normal forms (d_1, d_2) might not be orientable, i.e. it is not possible to transform it into a rewrite rule which preserves the termination property of the rewrite rule system. In this case the procedure terminates with "failure". Secondly, the process of adding new rewrite rules might not stop. Nevertheless, in many interesting examples the completion procedure does produce a complete rewrite rule system.

After a canonical rrs R' has been computed by the completion procedure, some of the

rules in R' can be eliminated. Suppose that the left hand side of the rule (1) $l_1 \to r_1$ can be reduced by the rule (2) $l_2 \to r_2$. Then in computing normal forms for terms the rule (1) is superfluous, because whenever a term is reducible by (1) it is also reducible by (2). Finally, however, a unique normal form is reached, because R' is complete. So the rule (1) can be deleted from R' without changing the relation $\leftrightarrow^*_{R'}$. The resulting rrs is still canonical.

Example 3.11: The axioms (Q1) – (Q3) of Example 1.1 can be oriented such that the left hand side is reduced to the right hand side. This gives a terminating rrs RQ. Running the completion procedure on RQ finally leads to the canonical system RQ':

$$
\begin{aligned}
&app(x, newq) \to x\\
&app(x, add(y,z)) \to add(app(x,y),z)\\
&app(app(x,y),z) \to app(x, app(y,z))\\
&app(x, app(newq,y)) \to app(x,y)\\
&app(x, app(add(y,z),w)) \to app(add(app(x,y),z),w)
\end{aligned}
$$

Using the rrs RQ', both sides of (H) (in Example 1.1) can be reduced to $app(add(app(x,y),z),w)$, so the equation is proved. ∎

Example 3.12: We consider the rrs RG of Example 3.4. As we have seen in Example 3.7, RG is not Church–Rosser. So we might try to complete it, i.e. add additional rules which do not change the associated equivalence relation $\leftrightarrow^*_{RG}$ and also do not destroy the termination property. Choosing $(1 \cdot y, x^{-1} \cdot (x \cdot y))$ as the critical pair (c_1, c_2) in the completion procedure, we detect that $(d_1, d_2) = (y, x^{-1} \cdot (x \cdot y))$ and $RG \cup \{d_2 \to d_1\}$ is again a Noetherian rrs. So we add $d_2 \to d_1$ as a new rule. This, of course, leads to new critical pairs.

Finally the procedure stops, because all the critical pairs can be reduced to identical normal forms. At this stage the rrs contains 20 rules, 10 of which can be eliminated. A complete rrs RG' for the equational theory of free groups is

(1)	$1 \cdot x \to x$
(2)	$x^{-1} \cdot x \to 1$
(3)	$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$
(4)	$x^{-1} \cdot (x \cdot y) \to y$
(5)	$x \cdot 1 \to x$
(6)	$1^{-1} \to 1$
(7)	$(x^{-1})^{-1} \to x$
(8)	$x \cdot x^{-1} \to 1$
(9)	$x \cdot (x^{-1} \cdot y) \to y$
(10)	$(x \cdot y)^{-1} \to y^{-1} \cdot x^{-1}$

Every theorem in the equational theory of free groups, i.e. every term equation which can be derived from the three group axioms, can be proved by reducing the two sides of the equation modulo this rewrite rule system and checking for syntactical equality. ∎

The completion procedure has been refined in various ways. Huet [Hu 81] has developed a version which gives a semidecision procedure for the associated equational theory. An approach to speeding up the procedure is described in [WB 83].

The two important properties a rrs might have are the Church–Rosser property and the termination property. We have already dealt with the Church–Rosser property, so let us now turn to termination. Termination of rewrite rule systems is needed twice in the process of deciding an equational theory: we need it for computing normal forms in the completion procedure and we need it for being able to infer that a locally confluent rrs (which is computed by the completion procedure) is actually confluent.

Unfortunately, termination of rewrite rule systems is an undecidable property [De 85]. However, various sufficient conditions for termination have been developed. For instance, termination can be proved if one has a simplification ordering compatible with the rrs. A *simplification ordering* is a partial ordering $\succ$ on a set of terms $\mathcal{T}$ such that for all terms $t, u \in \mathcal{T}$ and for all operators f

$$t \succ u \Longrightarrow f(\cdots t \cdots) \succ f(\cdots u \cdots) \text{ and}$$
$$f(\cdots t \cdots) \succ t.$$

Theorem 3.13: If R is a rrs over a set of terms $\mathcal{T}$ and $\succ$ is a simplification ordering on $\mathcal{T}$ such that $t \to_R u \Longrightarrow t \succ u$ for all terms $t, u \in \mathcal{T}$, then R is terminating. ∎

Example 3.14: Let $\mathcal{T}$ be the set of terms for the group signature of Example 2.4. Let τ be the following map from $\mathcal{T}$ to $\mathbb{N}$: $\tau(1) = 2$, $\tau(x) = 2$ for every variable x, and $\tau(u \cdot v) = \tau(u) \cdot \tau(v)$, $\tau(u^{-1}) = 2 \cdot \tau(u)$ for all terms $u, v \in \mathcal{T}$. We get a simplification ordering $\succ$ on $\mathcal{T}$ by setting

$$u \succ v \Longleftrightarrow \tau(u) > \tau(v).$$

With respect to this simplification ordering the left hand side of every rule in RG' is greater than the right hand side, so by Theorem 3.13 RG' is terminating. ∎

For an overview of sufficient conditions for termination we refer to [De 85].

As we have seen above, the termination property is essential for the application of the completion procedure. There are certain types of axioms which make termination impossible. A typical example is the commutativity axiom $f(x, y) = f(y, x)$. This problem can be overcome by treating such an axiom not as a rewrite rule but keeping it as an equation and carrying out the completion procedure modulo this equation, or in general modulo a set of equations E. Basically this means computing the most general unifier (or a generating set for the unifiers) modulo E. Peterson and Stickel [PS 81] have described a completion procedure modulo a set of equations E which contains associativity and commutativity axioms.

4. Inductive theories

Definition 4.1: Let Σ be a signature, E a set of equations over Σ. The *inductive theory* of E is the set of equations $s = t$ which are valid in the *initial algebra* of E, $\mathcal{I}(E)$. $\mathcal{I}(E)$ is the quotient of the algebra of ground terms $\mathcal{G}(\Sigma)$ modulo the congruence $=_E$. ■

The inductive theory of a set of equations E contains the equational theory of E. For proving theorems in inductive theories usually some sort of inductive argument is used, e.g. induction on the structure of the terms. We follow [De 83] in describing how the completion procedure can be used for proving theorems in inductive theories.

Example 4.2: The following rrs L is a complete system for reversing list structures with basic elements a and b:

(1) $a^r \to a$
(2) $b^r \to b$
(3) $(x \cdot y)^r \to (y^r \cdot x^r)$

The equation $x^{r^r} = x$ holds for all ground lists. This can be proved for instance by induction on the depth of the list. So if we denote by LE the set of equations corresponding to L, then $x^{r^r} = x$ is in the inductive theory of LE, but it is not in the equational theory of LE. ■

Theorem 4.3: Let E be the basis of an equational theory, R a complete rrs for $=_E$. Let $R(\mathcal{G})$ be the set of irreducible ground terms modulo R. Let $s = t$ be an equation which can be oriented into a rewrite rule r such that $R \cup \{r\}$ is Noetherian. Then $s = t$ is not in the inductive theory of E if and only if running the completion procedure on $R \cup \{r\}$ results in a rule with a left–hand side that has an instance in $R(\mathcal{G})$. This, provided that the completion procedure does not stop with "failure". ■

Example 4.4: We apply Theorem 4.3 for proving that the equation $x^{r^r} = x$ is in the inductive theory of LE (see Example 4.2). Adding the rule

$$x^{r^r} \to x$$

to the rrs L and running the completion procedure generates no new rule. So the equation indeed belongs to the inductive theory of LE.

On the other hand, adding

$$(x \cdot x)^r \to x^r$$

to the rule system and running the completion procedure generates (among others) the new rules

$$x^r \cdot x^r \to x^r,$$
$$a \cdot a \to a.$$

Since $a \cdot a$ is an irreducible ground term modulo L, this proves that $(x \cdot x)^r = x^r$ is not in the inductive theory of LE. ■

Example 4.5: As another example consider a system AM of axioms for addition and multiplication

$$\begin{gathered} m + 0 = m, \\ m \times 0 = 0, \\ m \times (n+1) = m \times n + m, \\ \textstyle\sum_{i=1}^{0} i = 0, \\ \textstyle\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + n + 1. \end{gathered}$$

Summation is a function of the upper bound, so we write $sum(n)$ instead of $\sum_{i=1}^{n} i$. We use prefix notation for all the operators. Transforming these equations into rewrite rules and running the completion algorithm (modulo a set of equations E containing associativity and commutativity axioms for $+$ and $\times$) yields the complete rewrite system RAM

$$\begin{gathered} +(m, 0) \to m, \\ \times(m, 0) \to 0, \\ \times(m, +(n, 1)) \to +(\times(m, n), m), \\ sum(0) \to 0, \\ sum(+(n, 1)) \to +(sum(n), +(n, 1)), \\ \times(m, 1) \to m, \\ sum(1) \to 1. \end{gathered}$$

The irreducible ground terms are $0,\ 1,\ +(1,1),\ +(1,+(1,1)), \ldots$. Now we want to prove that

$$\text{(H)} \quad \sum_{i=1}^{n} i = \frac{n \times (n+1)}{2}$$

is in the inductive theory of AM. So we add the rule

$$\times(+(1,1), sum(n)) \to \times(n, +(n,1))$$

to the rrs RAM and run the completion algorithm. The only new rule generated is

$$+(sum(n), sum(n)) \to +(\times(n,n), n).$$

No irreducibel ground term becomes reducible by this new rule, so (H) is in the inductive theory of AM. ∎

5. Programming with rewrite rules

Definition 5.1: If P is an n–ary predicate over the ground terms of some term algebra $\mathcal{T}$, and R is a canonical rrs for deciding P, then R is called a *rewrite program* for P. 'Deciding P' means that $P(t_1, \ldots, t_n) \to_R^* true$ if and only if the ground terms $t_1, \ldots, t_n$ satisfy the predicate P. ∎

Example 5.2: We consider the predicate *q and r are the quotient and remainder of x divided by y.*

$$div(x, y, q, r) \iff x = q \cdot y + r \ \wedge \ r < y.$$

A rewrite program for div is RD:

basic rules:

(1) $div(u+y+1, y+1, q+1, r) \to div(u, y+1, q, r)$

(2) $div(x, x+u+1, 0, x) \to true$

specializations of (1) and (2), where some of the variables are set to 0:

(3) $div(u+y+1, y+1, 1, r) \to div(u, y+1, 0, r)$

(4) $div(y+1, y+1, q+1, r) \to div(0, y+1, q, r)$

(5) $div(y+1, y+1, 1, r) \to div(0, y+1, 0, r)$

(6) $div(u+1, 1, q+1, r) \to div(u, 1, q, r)$

(7) $div(u+1, 1, 1, r) \to div(u, 1, 0, r)$

(8) $div(1, 1, q+1, r) \to div(0, 1, q, r)$

(9) $div(1, 1, 1, r) \to div(0, 1, 0, r)$

(10) $div(x, x+1, 0, x) \to true$

(11) $div(0, 1, 0, 0) \to true$

Now we want to use this rewrite program for computing the quotient and remainder of 7 and 3. We add the rule

$$div(7, 3, q, r) \to false(q, r)$$

to RD and complete the system. The completion procedure generates the additional rules

$$div(4, 3, q, r) \to false(q+1, r)$$
$$div(1, 3, q, r) \to false(q+2, r)$$
$$false(2, 1) \to true$$

So the assumption that $div(7, 3, q, r)$ is false leads to a contradiction and the counterexample is $q = 2,\ r = 1$. Hence 2 and 1 are the quotient and remainder, respectively, of 7 and 3. ■

Theorem 5.3 ([De 83]): Let R be a rewrite program for the predicate $P(\bar{x}, \bar{z})$. Running the completion procedure with $R \cup \{P(\bar{s}, \bar{z}) \to false(\bar{z})\}$ as input, where $\bar{s}$ are ground terms and $\bar{z}$ are variables, and with a simplification ordering under which the constant $true$ is less than any other term, will generate an answer rule with a left–hand side $false(\bar{t})$ if and only if ground terms $\bar{r}$ exist such that $P(\bar{s}, \bar{r})$ is true. This, provided that the completion procedure does not stop with "failure". ■

Actually one gets a solution of the problem by substituting ground terms for the variables in the term $\bar{t}$.

Example 5.4: We can also use the rewrite program RD for solving the reverse problem: find x and y such that $div(x, y, 2, 1)$ holds. For solving this reverse problem we add the rule

$$div(x, y, 2, 1) \to false(x, y)$$

to RD and complete the system. The completion procedure generates the additional rules

$$div(u, y+1, 1, 1) \to false(u+y+1, y+1)$$
$$div(u, y+1, 0, 1) \to false(u+y+y+1+1, y+1)$$
$$false(y+y+1+1+1, y+1) \to true$$

So whenever we substitute ground terms into $2y + 3$ and $y + 1$ we get numbers whose quotient and remainder are 2 and 1, respectively. ∎

References

[Bi 35] G. Birkhoff: "On the structure of abstract algebras", *Proc. Cambridge Phil. Soc.* **31**, 433–454 (1935)

[Bu 85] B. Buchberger: "Basic features and development of the critical-pair/completion procedure", *Proc. Rewriting Techniques and Applications*, J.-P. Jouannaud (ed.), 1–45, Springer–Verlag, LNCS 202 (1985)

[De 83] N. Dershowitz: "Applications of the Knuth–Bendix Completion Procedure", Techn. Rep. ATR-83(8478)-2, Laboratory Operations, THE AEROSPACE CORP., El Segundo, Calif. 90245 (1983)

[De 85] N. Dershowitz: "Termination", *Proc. Rewriting Techniques and Applications*, J.-P. Jouannaud (ed.), 180–224, Springer–Verlag, LNCS 202 (1985)

[Hu 81] G.P. Huet: "A Complete Proof of Correctness of the Knuth–Bendix Completion Algorithm", *J. Comp. and Syst. Sci.* **23**, 11–21 (1981)

[HO 80] G.P. Huet, D. Oppen: "Equations and rewrite rules: A survey", in: *Formal Languages: Perspectives and Open Problems*, R. Book (ed.), Academic Press (1980)

[KB 67] D.E. Knuth, P.B. Bendix: "Simple word problems in universal algebras", in: *Computational Problems in Abstract Algebra*, J. Leech (ed.), Pergamon Press (1970)

[PS 81] G.E. Peterson, M.E. Stickel: "Complete Sets of Reductions for Some Equational Theories", *J.ACM* **28/2**, 233–264 (1981)

[Wi 84] F. Winkler: *The Church–Rosser property in computer algebra and special theorem proving: An investigation of critical-pair/completion algorithms*, Dissertation, Inst. f. Math., Univ. Linz (1984)

[WB 83] F. Winkler, B. Buchberger: "A criterion for eliminating unnecessary reductions in the Knuth–Bendix algorithm", *Colloquia Mathematica Societatis János Bolyai – 42. Algebra, Combinatorics and Logic in Computer Science*, North–Holland (1986)