Relational algebra and equational proofs

Eindhoven University of Technology

Department of Mathematics and Computing Science

Relational Algebra and Equational Proofs

by

Eric de Kogel

93/23

Computing Science Note 93/23
Eindhoven, July 1993

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

Relational Algebra and Equational Proofs

Eric de Kogel *

June 30, 1993

Abstract

We show that two concepts involving equational provability can be elegantly formalized in terms of a relational algebra, equipped with two special-purpose mappings. We derive some calculus in order to prove that these concepts are equivalent, and that they are sound and complete.

We illustrate the use of the relational framework by a few examples. We show how decidability of provability of an equation from a finite set of variable-free equations, where all equations are variable-free. Then we discuss a method by Reeves ([3]) to deal with equations in semantic tableaux, that we can now prove to be complete in a very simple way.

Finally we discuss an equational proof format that is naturally induced by the relational formulation, and serves as a guideline in finding proofs. The relation between Reeves' rules and the construction of such proofs is made explicit.

1 Introduction

We present characterizations of equational provability in terms of a relational algebra. These characterizations are very compact, yet intuitively clear, and they are subject to formal manipulation. We can in fact easily establish equivalence of different characterizations after deriving some simple relational calculus.

We shall restrict ourselves to equations between terms of first-order logic. In order to keep the presentation clear we do not discuss the role of variables. As a starting point of the discussion we take the well-known proof rule of 'replacing equals by equals' (section 3). This can be formalized in terms of a relational algebra quite easily, yielding a characterization of all equations that can be proved with this rule. We then go on, transforming this characterization into a less redundant one.

Our description permits us to show that the problem of proving an equation from a finite set of equations is decidable, again assuming that all equations are variable-free.

Then we discuss a method by Reeves ([3]) to handle equations in semantic tableaux. Using the relational calculus we can easily prove it to be sound and complete.

*Co-operation Centre Tilburg and Eindhoven Universities (SOBU)
c/o Dept. of Philosophy, Tilburg University, Tilburg, the Netherlands
e-mail kogel@kub.nl

It is shown how the relational charactarization of equational provability naturally induces a very compact format for actual equational proofs, and also sheds some light on the heuristics of finding such a proof. It is shown that application of Reeves' rules can be interpreted as a method of constructing such proofs. Moreover, Reeves' rules in their original formulation tell what proof steps we are allowed to make, the heuristic guidance provided by a description in terms of relational algebra also shows what proof steps are worth trying.

2 Terms, Equations, Interpretations

We define terms, equations and interpretations as usual in first-order logic. We consider terms without variables only.

2.1 Terms and Equations

We assume that for every natural number n an enumerable set $Func_n$ exists, containing n-ary function symbols. From these function symbols terms can be built.

Definition 2.1 The set $Term$ of terms is defined to be the smallest set satisfying:

- If $c \in Func_0$, then $c\langle\rangle \in Term$. We shall abbreviate $c\langle\rangle$ to c, and call c a constant.
- If $f \in Func_n$ and $t_1, ..., t_n \in Term$, then $f\langle t_1, ..., t_n\rangle \in Term$.

We shall consider the former requirement to be a special case of the latter. Here we specified them both, just to give a clear inductive definition.

f is the *outermost function symbol* of $f\langle t_1, ..., t_n\rangle$. $t_1, ..., t_n$ are the *arguments* of $f\langle t_1, ..., t_n\rangle$. □

Definition 2.2 A term s is a *subterm* of t iff it follows from the following rules.

- Every term is a subterm of itself.
- A subterm of an argument of t is also a subterm of t.

□

Definition 2.3 An *equation* is an expression of the form $s \doteq t$, where s and t are terms. □

Remark Note the distinction between the angled brackets $\langle\rangle$ used in terms, and the parentheses (). This must remind us that the angled brackets and terms are syntactic objects. Parentheses serve to denote function applications and to avoid ambiguity in expressions, as usual. For the same reason we distinguish between $\doteq$ and $=$. □

2.2 Interpretations

Terms are supposed to refer to objects in some 'universe of discourse'. This universe can be any non-empty set U. What object a term refers to is determined by an interpretation.

Definition 2.4 Let U be a non-empty set. An *interpretation* I with universe U maps each n-ary function symbol f, to a function $f^I : U^n \to U$. I is naturally extended, to map terms in U:

$$I(f(t_1, ..., t_n)) = f^I(I(t_1), ..., I(t_n)).$$

□

Definition 2.5 The symbol $\models$ can have three different meanings.

- An equation can be either true or false in an interpretation. $s \doteq t$ is true in I iff $I(s) = I(t)$. In this case we write $I \models s \doteq t$. If $s \doteq t$ is false in I, i.e., $I(s) \neq I(t)$, then we write $I \not\models s \doteq t$.
- If E is a set of equations, then we write $I \models E$ iff every equation in E is true in I.
- If $s \doteq t$ is true in all interpretations I such that $I \models E$, then we write $E \models s \doteq t$. We say that $s \doteq t$ logically follows from E.

□

3 Equational Provability

The intuitive meaning of equality in this context is that we may 'replace equals by equals' in a term, without changing the interpretation of that term. That is, the original term and the resulting term have the same interpretation. A proof rule based on this intuition is the following: Start with a term s. If a subterm of s occurs on one hand of an equation in E, then we may replace it by the other hand of the equation. Then we can proceed with the resulting term, and so on, until we obtain t. If and only if we can obtain t this way, we say that we can prove $s \doteq t$ from E, and we write $E \vdash s \doteq t$.

We shall formalize this in terms of a relational algebra, and then show in theorem 3.2 that $E \vdash s \doteq t$ iff $E \models s \doteq t$. The relations in our algebra are subsets of $Term \times Term$, and if r is a relation, we shall write $s\ [r]\ t$ rather than $(s,t) \in r$. Further we sometimes write $t_0\ [r_1]\ t_1\ [r_2]\ t_2$ instead of $t_0\ [r_1]\ t_1 \wedge t_1\ [r_2]\ t_2$.

The first useful relation is associated with the set E. The relation itself is called *eqns*, and is defined by

$$s\ [eqns]\ t \Leftrightarrow s \doteq t \in E.$$

Please note that *eqns* can be any arbitrary relation, it need not be reflexive, transitive or symmetric.

Further we define the relation *subst*. $s\ [subst]\ t$ is true iff a subterm of s occurs as one hand of an equation in E and t is the result of replacing this subterm by the other hand of the equation. Following the structure of definition 2.2 we can define *subst*.

Definition 3.1 s $[subst]$ t is true iff it follows from the following rules:

- s $[subst]$ t if s $[eqns]$ t or t $[eqns]$ s;
- s $[subst]$ t if s and t have the same outermost function symbol, say f, $f \in Func_n$, i.e., we can write $s = f\langle s_1, ..., s_n\rangle$ and $t = f\langle t_1, ..., t_n\rangle$, and
  - for some index i, $1 \leq i \leq n$, s_i $[subst]$ t_i, and
  - for all indices j with $j \neq i$, $1 \leq j \leq n$, $s_j = t_j$.

Note that *subst* depends on *eqns*. In order to keep the notation compact we shall not express this dependency explicitly. □

Let $subst^*$ be the reflexive and transitive closure of *subst*. Hence, s $[subst^*]$ t is true iff, for some natural number n, there exist terms $t_0, ..., t_n$ such that:

$$s = t_0\ [subst]\ t_1\ [subst]\ t_2 ... t_{n-1}\ [subst]\ t_n = t.$$

That is, s $[subst^*]$ t is true iff $E \vdash s \doteq t$. Note that *subst* is symmetric, so $subst^*$ is an equivalence relation. The proof rule of 'replacing equals by equals' is sound and complete: every equation that logically follows from E, can be proved this way.

Theorem 3.2 For all terms s and t, s $[subst^*]$ $t \Leftrightarrow (E \models s \doteq t)$.

Proof ($\Rightarrow$) 'Replacing equals by equals' is obviously a sound rule. Hence, s $[subst^*]$ $t \Rightarrow (E \models s \doteq t)$.

($\Leftarrow$) Suppose that $\neg(s\ [subst^*]\ t)$. Then we must prove that there is an interpretation I such that $I \models E$ and $I \not\models s \doteq t$. $subst^*$ is an equivalence relation, and if we denote the equivalence class of a term t by $[t]$, then I, defined by $I(t) = [t]$, is such an interpretation, with the set of equivalence classes as its universe. It follows from the definitions of *subst* and $subst^*$ that each equation in E is true in I, while $\neg s$ $[subst^*]$ t is equivalent to $I \not\models s \doteq t$. We have not yet shown that I is indeed an interpretation. To do this, we must show that, for all function symbols f, the interpretation f^I is well-defined by $f^I([t_1], ..., [t_n]) = [f\langle t_1, ..., t_n\rangle]$. Therefore we must show that

$$[s_1] = [t_1] \wedge ... \wedge [s_n] = [t_n] \Rightarrow [f\langle s_1, ..., s_n\rangle] = [f\langle t_1, ..., t_n\rangle]$$

which is equivalent to

$$s_1\ [subst^*]\ t_1 \wedge ... \wedge s_n\ [subst^*]\ t_n \Rightarrow f\langle s_1, ..., s_n\rangle\ [subst^*]\ f\langle t_1, ..., t_n\rangle.$$

In this formulation it will be proved in theorem 5.10. □

4 Relational Algebra

Subjects of study are:

- the set *Rel* of relations on $Term \times Term$,
- the set of mappings $Rel \rightarrow Rel$.

Unless stated otherwise, the word *relation* refers to an element of *Rel*, and the word *mapping* refers to a mapping $Rel \rightarrow Rel$.

4.1 Elementary Operations

As basic operators on relations we use set union $\cup$, and 'matrix multiplication' $\circ$. If a and b are relations, then the relation $a \circ b$ is defined by

$$s\ [a \circ b]\ t \quad \Leftrightarrow \quad (\exists m \in Term\ s\ [a]\ m\ [b]\ t)$$

$\circ$ binds stronger than $\cup$. So $a \cup b \circ c$ must be read $a \cup (b \circ c)$.

Theorems 4.1, 4.2 and 4.3 list a few properties of $\circ$. Their proofs are straightforward and therefore they are omitted.

Theorem 4.1 $\cup$ is commutative, and $\circ$ distributes over $\cup$. That is, for all relations a, b and c

$$\begin{aligned} a \circ (b \circ c) &= (a \circ b) \circ c \\ (a \cup b) \circ c &= a \circ c \cup b \circ c \\ a \circ (b \cup c) &= a \circ b \cup a \circ c \end{aligned}$$

□

Theorem 4.2 If, for some index set Ind, $\{a_i \mid i \in Ind\}$ is a set of relations, then

$$\begin{aligned} (\cup i \in Ind\ a_i) \circ b &= \cup i \in Ind\ (a_i \circ b), \\ b \circ (\cup i \in Ind\ a_i) &= \cup i \in Ind\ (b \circ a_i). \end{aligned}$$

□

Theorem 4.3 $\circ$ is monotonic, that is,

$$a_1 \subseteq a_2 \wedge b_1 \subseteq b_2 \quad \Rightarrow \quad a_1 \circ b_1 \subseteq a_2 \circ b_2.$$

□

Definition 4.4 $\perp$ is the empty set. $\perp$ is the identity element of $\cup$, and the zero element of $\circ$. □

Definition 4.5 $id = \{(t,t) \mid t \in Term\}$. That is, id is the identity relation on $Term \times Term$. id is the identity element of $\circ$. □

Definition 4.6 Define natural powers of a relation a by

$$a^0 := id \qquad a^{n+1} := a \circ a^n$$

□

Definition 4.7 As a convenient abbreviation we introduce, for every relation a

$$a^* := id \cup a \cup a^2 \cup a^3 \cup \ldots = \cup n \in \mathbb{N}\ a^n.$$

Hence, a^* is the reflexive, transitive closure of a. □

Theorem 4.8 For all relations a and b, $(a \cup b)^* = (b^* \circ a)^* \circ b^*$.

Proof Suppose that $s\ [(a \cup b)^*]\ t$ is true, so there is an $n \in \mathbb{N}$ such that $s\ [(a \cup b)^n]\ t$. Find relations $r_1, ..., r_n \in \{a, b\}$ such that $s\ [r_1 \circ ... \circ r_n]\ t$. Writing out the sequence $r_1, ..., r_n$ symbolically we get a string of n characters, each character being an 'a' or a 'b'. Suppose the string contains m 'a's, then obviously

$$r_1 \circ ... \circ r_n \subseteq \underbrace{b^* \circ a \circ ... \circ b^* \circ a}_{m \text{ times}} \circ b^* = (b^* \circ a)^m \circ b^*.$$

This proves $(a \cup b)^* \subseteq (b^* \circ a)^* \circ b^*$. The converse inclusion can easily be shown with a similar argument. □

4.2 Least Fixpoints

We often define a relation r by an inductive definition like: '$s\ [r]\ t$ is true iff this follows from the following rules ...'. It will turn out that the rules that follow can often be summarized as $T(r) \subseteq r$, where T is a mapping. Now there may be many relations r' fulfilling $T(r') \subseteq r'$. Let us collect these solutions in the set $\mathcal{S} = \{r' \mid T(r') \subseteq r'\}$. The definition of r then involves two requirements:

- $r \in \mathcal{S}$, and
- $s\ [r]\ t$ is true iff for all $r' \in \mathcal{S}$ $s\ [r']\ t$ is true.

It is not hard to see that this means that r is the 'smallest' element of $\mathcal{S}$, where 'smallest' refers to the partial ordering $\subseteq$. Under certain conditions, that are always fulfilled in those cases we are interested in, there will be such a smallest element, hence r is well-defined. Moreover, r will be the smallest relation satisfying $r = T(r)$. Such an r is called the least fixpoint of T. Hence, some theory about least fixpoints is interesting for us and we discuss it here. All theorems and definitions in this subsection can also be found in in [2], though we adapted them to our purposes.

Definition 4.9 A relation r is a *fixpoint* of a mapping T iff $r = T(r)$. □

Definition 4.10 A relation r is the *least fixpoint* of a mapping T iff r is a fixpoint of T, and if r' is a fixpoint of T, then $r \subseteq r'$. The least fixpoint of T is denoted μT. □

Definition 4.11 A mapping T is *monotonic* iff for all relations r_1 and r_2, $r_1 \subseteq r_2 \Rightarrow T(r_1) \subseteq T(r_2)$. □

Theorem 4.12 Every monotonic mapping T has a least fixpoint.

Proof Define $\mathcal{S} = \{r \mid T(r) \subseteq r\}$. Note that $\mathcal{S}$ is not empty: If $\top = Term \times Term$, then $T(\top) \subseteq \top$. Let $l = \cap r \in \mathcal{S}\ r$.

For all $r \in \mathcal{S}$ we have $T(r) \subseteq r$ and $l \subseteq r$. Monotonicity of T yields $T(l) \subseteq T(r) \subseteq r$. So we find $(\forall r \in \mathcal{S}\ T(l) \subseteq r)$, which equivales $T(l) \subseteq l$. Monotonicity yields $T(T(l)) \subseteq T(l)$, hence $T(l) \in \mathcal{S}$ and therefore $l \subseteq T(l)$.

All this implies $l = T(l)$. As S contains all fixpoints of T, and l is smaller than every element of S, $l = \mu T$. □

From the proof of theorem 4.12 we can immediately deduce:

Theorem 4.13 If T is a monotonic mapping, then μT is the least solution r of $T(r) \subseteq r$. In other words, for all relations r

$$T(r) \subseteq r \Rightarrow \mu T \subseteq r.$$

□

Theorem 4.14 If S and T are monotonic mappings and $S(r) \subseteq T(r)$ for all relations r, then $\mu S \subseteq \mu T$.

Proof We find that $S(\mu T) \subseteq T(\mu T) = \mu T$. Since μS is the *smallest* relation r such that $S(r) \subseteq r$, we find $\mu S \subseteq \mu T$. □

Definition 4.15 A set R of relations is *directed* iff for every finite subset $\{r_1, ..., r_n\}$ of R there is an $r \in R$ such that $r_1 \subseteq r$ and ... and $r_n \subseteq r$. Note that R must be non-empty. □

Theorem 4.16 Let $R = \{r_1, ..., r_n\}$ be a finite, nonempty set of relations such that $r_1 \subseteq r_2 \subseteq ... \subseteq r_n$ or let $R = \{r_0, r_1, r_2, ...\}$ be an enumerable set of relations such that $r_0 \subseteq r_1 \subseteq r_2 \subseteq ...$. Then R is a directed set of relations.

Proof Let $\{r_{i_1}, ..., r_{i_m}\}$ be a finite subset of R. If N is the largest number in $\{i_1, ..., i_m\}$, then $r_{i_1} \subseteq r_N$ and ... and $r_{i_m} \subseteq r_N$. □

Definition 4.17 A mapping T is *continuous* iff for every directed set of relations R, $T(\cup r \in R\ r) = \cup r \in R\ T(r)$. □

Theorem 4.18 A continuous mapping is also *monotonic*.

Proof Consider relations r_1 and r_2 such that $r_1 \subseteq r_2$, i.e., $r_2 = r_1 \cup r_2$. $\{r_1, r_2\}$ is directed (theorem 4.16). Continuity of T yields $T(r_2) = T(r_1 \cup r_2) = T(r_1) \cup T(r_2)$ and hence, $T(r_1) \subseteq T(r_2)$. □

Definition 4.19 We define natural powers of a mapping F by:

$$F^0(r) := r \qquad F^{n+1}(r) := F(F^n(r))$$

Theorem 4.20 Let T be a continuous mapping, then

- $\bot \subseteq T(\bot) \subseteq T^2(\bot) \subseteq T^3(\bot) \subseteq ... \subseteq \mu T$;
- $\mu T = \cup n \in \mathbb{N}\ T^n(\bot)$.

Proof $\bot \subseteq T(\bot) \subseteq T^2(\bot) \subseteq T^3(\bot) \subseteq ... \subseteq \mu T$ follows easily from the monotonicity of T and the obvious facts $\bot \subseteq T(\bot)$ and $\bot \subseteq \mu T$.

$\{T^n(\bot) \mid n \in \mathbb{N}\}$ is a directed set of relations (theorem 4.16); the continuity of T implies

$$\begin{aligned} & T(\cup n \in \mathbb{N}\ T^n(\bot)) \\ = & \\ & \cup n \in \mathbb{N}\ T(T^n(\bot)) \\ = & \\ & \bot \cup (\cup n \in \mathbb{N}\ T^{n+1}(\bot)) \\ = & \qquad \{\text{since } \bot = T^0(\bot)\} \\ & \cup n \in \mathbb{N}\ T^n(\bot). \end{aligned}$$

So $\cup n \in \mathbb{N}\ T^n(\bot)$ is a fixpoint of T. It is also the least fixpoint: We already saw that $(\forall n \in \mathbb{N}\ T^n(\bot) \subseteq \mu T)$ and hence, $\cup n \in \mathbb{N}\ T^n(\bot) \subseteq \mu T$. □

Theorem 4.21 If T is a continuous mapping and a is a relation such that $a \subseteq \mu T$, then $\mu T = \cup n \in \mathbb{N}\ T^n(a)$.

Proof For all $n \in \mathbb{N}$ we can prove $T^n(\bot) \subseteq T^n(a) \subseteq \mu T$, using the monotonicity of T and $\bot \subseteq a \subseteq \mu T$. Using theorem 4.20 we find

$$\mu T \ = \ \cup n \in \mathbb{N}\ T^n(\bot) \ \subseteq \ \cup n \in \mathbb{N}\ T^n(a) \ \subseteq \ \mu T,$$

and we conclude that $\mu T = \cup n \in \mathbb{N}\ T^n(a)$. □

4.3 Some Useful Fixpoints

The previous section taught us that an inductive definition of a relation r of the form 's $[r]$ t is true iff this follows from the following rules ...' is often equivalent to the definition $r = \mu T$ for an appropriately chosen monotonic mapping T. We shall often write this definition as

$$r := T(r)$$

which we call a defining equation for r. In the appendix it is shown that all mappings that we shall actually use are continuous. Hence, we can use the 'fixpoint construction' of theorems 4.20 and 4.21. A few important fixpoint definitions, and the relations they actually define, will now be discussed.

Notation We use the symbol $\mapsto$ to avoid introducing new names for mappings. For instance, instead of 'the mapping H defined by $H(r) = F(r) \cup G(r)$' we shall simply write 'the mapping $r \mapsto F(r) \cup G(r)$'. □

Theorem 4.22 Let a and b be arbitrary, but fixed, relations.

1. $\mu(r \mapsto a \cup b \circ r) = b^* \circ a$.

2. $\mu(r \mapsto a \cup r \circ b) = a \circ b^*$.

3. $\mu(r \mapsto a \cup r \circ b \circ r) = (a \circ b)^* \circ a$.

Proof We only discuss 1. One can prove 2 and 3 analogously.

According to the appendix, $T = r \mapsto a \cup b \circ r$ is a continuous mapping; hence, we may construct μT as in theorem 4.20: $\mu T = \cup n \in \mathbb{N}\ T^n(\bot)$. With induction one can prove $T^{n+1}(\bot) = a \cup b \circ a \cup ... \cup b^n \circ a = \cup i \in \{0,...,n\}\ (b^i \circ a)$, and hence, $\mu T = \cup n \in \mathbb{N}\ (b^n \circ a) = (\cup n \in \mathbb{N}\ b^n) \circ a = b^* \circ a$. Theorem 4.2 justifies the second equality. □

Example 4.23 We reconsider definition 2.2 of the notion subterm. If we define the relations *subterm* and *argument* by

- s [*subterm*] t iff s is a subterm of t,
- s [*argument*] t iff s is an argument of t,

then definition 2.2 happens to define *subterm* in terms of *argument*: A term s is a *subterm* of t iff it follows from the following rules:

- Every term is a subterm of itself, hence, $id \subseteq$ *subterm*.
- A subterm of an argument of t is also a subterm of t. Hence, if there is a term t' such that s [*subterm*] t' [*argument*] t, then s [*subterm*] t. For short: *subterm* $\circ$ *argument* $\subseteq$ *subterm*.

So *subterm* is the smallest relation including id and *subterm* $\circ$ *argument*, i.e., *subterm* $= \mu(r \mapsto id \cup r \circ argument)$. According to theorem 4.22 (part 2) *subterm* $= id \circ argument^* = argument^*$, clearly showing that subterms of t are t, arguments of t, arguments of arguments of t, etc.. Note that the same relation *subterm* would have been defined by *subterm* $= \mu(r \mapsto id \cup argument \circ r)$, corresponding to a definition in which the second rule reads 'an argument of a subterm of t is also a subterm of t'. □

Unfortunately we shall encounter more complex fixpoint definitions than the ones in theorem 4.22, basically because the relations a and b mentioned there are replaced by relations depending on the relation that is being defined. We discuss such definitions in the next theorems.

Theorem 4.24 Let $T : Rel \times Rel \to Rel$ be such that for every relation a, the mappings $r \mapsto T(a,r)$ and $r \mapsto T(r,a)$ are monotonic. Then $\mu(r \mapsto T(r,r)) = \mu(r_1 \mapsto \mu(r_2 \mapsto T(r_1,r_2)))$.

Proof Define the relation $a = \mu(r \mapsto T(r,r))$ and the mapping $F = (r_1 \mapsto \mu(r_2 \mapsto T(r_1,r_2)))$. Finally let $b = \mu F$. So we must prove $b = a$.

Firstly, $b = F(b) = \mu(r_2 \mapsto T(b,r_2)) = T(b,b)$. So b is a fixpoint of $r \mapsto T(r,r)$, and, since a is the *least* fixpoint of that mapping, $a \subseteq b$.

Secondly, $a = T(a,a)$, so a is a fixpoint of $r_2 \mapsto T(a,r_2)$. Now, $F(a)$ is by definition the *least* fixpoint of that mapping, so $F(a) \subseteq a$. According to theorem 4.13, this implies $\mu F = b \subseteq a$. □

Theorem 4.25 Let A and B be monotonic mappings. Then

1. $\mu(r \mapsto A(r) \cup B(r) \circ r) = \mu(r_1 \mapsto B(r_1)^* \circ A(r_1))$,
2. $\mu(r \mapsto A(r) \cup r \circ B(r)) = \mu(r_1 \mapsto A(r_1) \circ B(r_1)^*)$,
3. $\mu(r \mapsto A(r) \cup r \circ B(r) \circ r) = \mu(r_1 \mapsto (A(r_1) \circ B(r_1))^* \circ A(r_1))$.

Proof 1 follows easily from theorem 4.24, defining T by

$$T(r_1, r_2) \quad = \quad A(r_1) \cup B(r_1) \circ r_2.$$

Then $\mu(r_2 \mapsto T(r_1, r_2)) = B(r_1)^* \circ A(r_1)$, according to theorem 4.22. One can prove 2 and 3 analogously. □

5 Back to Equational Proofs

Now we are ready to analyze the topics introduced in section 3 by means of the concepts of relational algebra.

First we want to formalize definition 3.1, in which the relation *subst* is defined, in terms of the relational algebra. Unfortunately, we have no means yet to say anything about the arguments of terms, as is required in the second rule of definition 3.1. Hence, we define a special-purpose mapping *Pick*.

Definition 5.1 For every relation r and terms s and t, s $[Pick(r)]$ t is true iff

- s and t have the same outermost function symbol, say $f \in Func_n$, i.e. we can write $s = f\langle s_1, ..., s_n \rangle$ and $t = f\langle t_1, ..., t_n \rangle$. Further,
- for exactly one index i, $1 \leq i \leq n$, s_i $[r]$ t_i is true, and finally
- for all indices j with $j \neq i$, $1 \leq j \leq n$, $s_j = t_j$.

□

We shall define the relation $\overline{eqns}$ as the symmetric closure of *eqns*, i.e., s $[\overline{eqns}]$ t iff s $[eqns]$ t or t $[eqns]$ s. Then we can restate definition 3.1 as: s $[subst]$ t is true iff it follows from the rules 's $[subst]$ t if s $[\overline{eqns}]$ t' and 's $[subst]$ t if s $[Pick(subst)]$ t'. This yields the following defining equation for *subst*:

$$subst := \overline{eqns} \cup Pick(subst).$$

Remark One can prove $subst = \cup n \in \mathbb{N}\ Pick^n(\overline{eqns})$. We shall not need this characterization however.

Now we have a simple and compact characterization of those equations that logically follow from E. Yet it has a disadvantage, as an example will show.

Example 5.2 Show that $a_1 \doteq b_1, a_2 \doteq b_2 \models f\langle a_1, a_2\rangle \doteq f\langle b_1, b_2\rangle$. In this case $E = \{(a_1, b_1), (a_2, b_2)\}$, and we must prove $f\langle a_1, a_2\rangle\ [subst^*]\ f\langle b_1, b_2\rangle$. This is not hard to do:

$$f\langle a_1, a_2\rangle\ [subst]\ f\langle b_1, a_2\rangle\ [subst]\ f\langle b_1, b_2\rangle.$$

But we might as well have written

$$f\langle a_1, a_2\rangle\ [subst]\ f\langle a_1, b_2\rangle\ [subst]\ f\langle b_1, b_2\rangle.$$

These two 'proofs' are essentially the same of course, but the characterization by $subst^*$ does not bring this to light so clearly. □

In the example it is obvious that we can obtain $f\langle b_1, b_2\rangle$ from $f\langle a_1, a_2\rangle$ by replacing both arguments by a new term and the order in which we do so doesn't matter.

In general, since $subst = \overline{eqns} \cup Pick(subst)$, we can use theorem 4.8 to obtain $subst^* = (\overline{eqns} \cup Pick(subst))^* = (Pick(subst)^* \circ \overline{eqns})^* \circ Pick(subst)^*$. Now $s\ [Pick(subst)]\ t$ is true, if t can be obtained by replacing a subterm of an argument of s by an appropriate term. Hence $Pick(subst)^*$ refers to an arbitrary number of such substitutions. The example suggests that the order of two such substitutions is irrelevant, if they apply to arguments on different positions. The following theorem makes this more explicit.

Theorem 5.3 For all relations r, $f\langle s_1, ..., s_n\rangle\ [Pick(r)^*]\ f\langle t_1, ..., t_n\rangle$ is true iff $s_1\ [r^*]\ t_1$ and ... and $s_n\ [r^*]\ t_n$.

Proof Suppose that $f\langle s_1, ..., s_n\rangle\ [Pick(r)^N]\ f\langle t_1, ..., t_n\rangle$ is true for some natural number N. This means that $f\langle s_1, ..., s_n\rangle$ can be transformed into $f\langle t_1, ..., t_n\rangle$ by N times replacing an argument by another term, such that whenever we replace a term s' by t', $s'\ [r]\ t'$ holds. Now let the total number of replacements of an argument on the i-th position $(1 \leq i \leq n)$ be N_i. Hence, $N_1 + ... + N_n = N$ and $s_1\ [r^{N_1}]\ t_1 \wedge ... \wedge s_n\ [r^{N_n}]\ t_n$.

We can similarly show that $s_1\ [r^{N_1}]\ t_1 \wedge ... \wedge s_n\ [r^{N_n}]\ t_n$ implies that $f\langle s_1, ..., s_n\rangle\ [Pick(r)^N]\ f\langle t_1, ..., t_n\rangle$, for $N = N_1 + ... + N_n$. □

This can be expressed more easily if we define another special purpose mapping.

Definition 5.4 Define the mapping $Args$. For every relation r, terms s and t, $s\ [Args(r)]\ t$ is true iff

- s and t have the same outermost function symbol, say $f \in Func_n$, i.e., we can write $s = f\langle s_1, ..., s_n\rangle$ and $t = f\langle t_1, ..., t_n\rangle$. Further,
- $s_1\ [r]\ t_1 \wedge ... \wedge s_n\ [r]\ t_n$

Note that $c\ [Args(r)]\ c$ is true, if $c \in Func_0$. □

Theorem 5.3 can now be restated as:

Theorem 5.5 For all relations r, $Pick(r)^* = Args(r^*)$.

Proof This follows from theorem 5.3, if we note that $s\ [Pick(r)^*]\ t$ can only be true if s and t have the same outermost function symbol:

- $Pick(r)^0 = id$, and if $s\ [id]\ t$, s and t must of course have the same outermost function symbol.
- For $n \geq 1$, it easily follows from the definition of $Pick$ that $s\ [Pick(r)^n]\ t$ can only be true if s and t have the same outermost function symbol.

□

Our characterization of $subst^*$ can now be further rewritten:

$$\begin{aligned} subst^* &= (Pick(subst)^* \circ \overline{eqns})^* \circ Pick(subst)^* \\ &= (Args(subst^*) \circ \overline{eqns})^* \circ Args(subst^*). \end{aligned}$$

This yields a nice characterization of $subst^*$, since in theorem 5.9 we shall prove that $subst^*$ is not just a fixpoint of $r \mapsto (Args(r) \circ \overline{eqns})^* \circ Args(r)$, but even the *least* fixpoint of that mapping. We shall name this fixpoint pr, a mnemonic for *p*rovable equation.

Definition 5.6 $pr := (Args(pr) \circ \overline{eqns})^* \circ Args(pr)$. □

Finally we show that pr is indeed a good characterization of provable equations. First we prove two convenient lemmas.

Lemma 5.7 id is the only fixpoint of $Args$.

Proof Define the depth $|t|$ of a term t by

$$|f\langle t_1, ..., t_n\rangle| := 1 + \max(|t_1|, ..., |t_n|).$$

In particular, we define $|c| := 1$ iff $c \in Func_0$.

Let r be a fixpoint of $Args$. Then, $s\ [r]\ t$ equivales $s\ [Args(r)]\ t$, and hence can only be true if s and t have the same outermost function symbol. If s and t have different outermost function symbols, then neither $s\ [r]\ t$ nor $s\ [id]\ t$.

By induction on N we prove that if $|s|, |t| \leq N$, then $s\ [r]\ t$ iff $s = t$.

- If $|s| = |t| = 1$, then s and t are constants, and, as they must have the same outermost function symbol, we find $s = t$.
- Induction hypothesis: if $|s| \leq N$ and $|t| \leq N$, then $s\ [r]\ t$ iff $s = t$.

Suppose $|f\langle s_1, ..., s_n\rangle| \leq N + 1$ and $|f\langle t_1, ..., t_n\rangle| \leq N + 1$. Hence, $|s_1|, ..., |s_n|, |t_1|, ..., |t_n| \leq N$. Then

$$\begin{aligned} & f\langle s_1, ..., s_n\rangle\ [r]\ f\langle t_1, ..., t_n\rangle \\ \Leftrightarrow \quad & \{r = Args(r)\} \\ & f\langle s_1, ..., s_n\rangle\ [Args(r)]\ f\langle t_1, ..., t_n\rangle \\ \Leftrightarrow \quad & \end{aligned}$$

$$
\begin{array}{ll}
 & s_1\ [r]\ t_1 \wedge ... \wedge s_n\ [r]\ t_n \\
\Leftrightarrow & \quad \{\text{Induction hypothesis}\} \\
 & s_1 = t_1 \wedge ... \wedge s_n = t_n \\
\Leftrightarrow & \\
 & f\langle s_1, ..., s_n\rangle = f\langle t_1, ..., t_n\rangle
\end{array}
$$

We conclude that $s\ [r]\ t$ iff $s = t$ for all terms s and t, so $r = id$. Note that $s_1 = t_1 \wedge ... \wedge s_n = t_n \Leftrightarrow f\langle s_1, ..., s_n\rangle = f\langle t_1, ..., t_n\rangle$ shows that id is indeed a fixpoint of $Args$. □

Lemma 5.8 $id \subseteq pr$.

Proof Let T be the mapping $r \mapsto (Args(r) \circ \overline{eqns})^* \circ Args(r)$, so $pr = \mu T$. Now we find that for all relations r

$$
\begin{array}{ll}
 & Args(r) \\
= & \\
 & (Args(r) \circ \overline{eqns})^0 \circ Args(r) \\
\subseteq & \\
 & (Args(r) \circ \overline{eqns})^* \circ Args(r) \\
= & \\
 & T(r).
\end{array}
$$

So, theorems 4.14 and 5.7 yield $id = \mu Args \subseteq \mu T = pr$. □

Theorem 5.9 $pr = subst^*$.

Proof First consider a fixpoint construction of $subst$. Let S be the mapping $r \mapsto \overline{eqns} \cup Pick(r)$, so $subst = \mu S$. S is continuous and if $subt_n = S^n(\perp)$, then, by theorem 4.20, $subst = \mu S = \cup n \in \mathbb{N} S^n(\perp) = \cup n \in \mathbb{N} subst_n$.

Because $r \mapsto r^*$ is continuous, and $\{subst_0, subst_1, subst_2, ...\}$ is a directed set of relations it follows that $subst^* = (\cup n \in \mathbb{N} subst_n)^* = \cup n \in \mathbb{N} subst_n^*$.

Now, consider a fixpoint construction of pr. Let P be the mapping $r \rightarrow (Args(r) \circ \overline{eqns})^* Args(r)$, so $pr = \mu P$. Then T is continuous, and by lemma 5.8, $id \subseteq \mu P = pr$. Define $pr_n = P^n(id)$. Then, by theorem 4.21 $pr = \mu P = \cup n \in \mathbb{N} P^n(id) = \cup n \in \mathbb{N} pr_n$.

So we have $subst^* = \cup n \in \mathbb{N}\ subst_n^*$ and $pr = \cup n \in \mathbb{N}\ pr_n$. If we prove that $subst_n^* = pr_n$ for all $n \in \mathbb{N}$, then we are done. We do so by induction:

- $pr_0 = id = \perp^* = subst_0^*$
- Induction hypothesis: $pr_n = subst_n^*$.
- Induction step. $pr_{n+1} = P(pr_n) = S(subst_n)^* = subst_{n+1}^*$, since

$$
\begin{array}{ll}
 & P(pr_n) \\
= &
\end{array}
$$

$$
\begin{array}{ll}
 & (Args(pr_n) \circ \overline{eqns})^* \circ Args(pr_n) \\
= & \quad \{\text{induction hypothesis}\} \\
 & (Args(subst_n^*) \circ \overline{eqns})^* \circ Args(subst_n^*) \\
= & \quad \{\text{theorem 5.5}\} \\
 & (Pick(subst_n)^* \circ \overline{eqns})^* \circ Pick(subst_n)^* \\
= & \quad \{\text{theorem 4.8}\} \\
 & (\overline{eqns} \cup Pick(subst_n))^* \\
= & \\
 & S(subst_n)^*
\end{array}
$$

We remark for later use that $subst_0 \subseteq subst_1 \subseteq subst_2 \subseteq ... \subseteq subst$ implies $subst_0^* \subseteq subst_1^* \subseteq subst_2^* \subseteq ... \subseteq subst^*$ and hence, $pr_0 \subseteq pr_1 \subseteq pr_2 \subseteq ... \subseteq pr$. □

Reconsidering example 5.2, we find that $f\langle a_1, a_2\rangle\ [pr]\ f\langle b_1, b_2\rangle$ is true since $f\langle a_1, a_2\rangle\ [Args(pr)]\ f\langle b_1, b_2\rangle$. There is no need to specify whether $a_1\ [pr]\ b_1$ is proved earlier or later than $a_2\ [pr]\ b_2$.

As a by-product, we can now complete the proof of theorem 3.2 in a relatively easy way:

Theorem 5.10 For all $f \in Func_n$, and terms $s_1, ..., s_n, t_1, ..., t_n$:

$$s_1\ [subst^*]\ t_1 \wedge ... \wedge s_n\ [subst^*]\ t_n \Rightarrow f\langle s_1, ..., s_n\rangle\ [subst^*]\ f\langle t_1, ..., t_n\rangle.$$

Proof The theorem is equivalent to $Args(subst^*) \subseteq subst^*$. Using $subst^* = pr$ this can be proved immediately:

$$Args(pr) \ \subseteq\ (Args(pr) \circ \overline{eqns})^* \circ Args(pr) \ =\ pr.$$

□

6 An Application: Decidability

It is well-known that one can decide whether or not

$$s_1 \doteq t_1, ..., s_n \doteq t_n \models s_0 \doteq t_0.$$

A very simple decision procedure is discussed in [5]. Let $\mathcal{T}$ be the set of terms occurring in this problem, namely $s_0, t_0, ..., s_n, t_n$ and all of their subterms. Let $id_{\mathcal{T}} = \{(t,t) \mid t \in \mathcal{T}\}$.

Then, for every relation r, $s\ [id_{\mathcal{T}} \circ r \circ id_{\mathcal{T}}]\ t$ equivales $s\ [r]\ t \wedge s, t \in \mathcal{T}$. So, $id_{\mathcal{T}} \circ r \circ id_{\mathcal{T}}$ is the restriction of r to $\mathcal{T} \times \mathcal{T}$. As an abbreviation we write $[r]_{\mathcal{T}}$ instead of $id_{\mathcal{T}} \circ r \circ id_{\mathcal{T}}$.

For instance, $\overline{eqns} = [\overline{eqns}]_{\mathcal{T}}$, since $s\ [\overline{eqns}]\ t$ already implies that $s, t \in \mathcal{T}$. Further, $[Args(r)]_{\mathcal{T}} = [Args([r]_{\mathcal{T}})]_{\mathcal{T}}$, since $s\ [[Args(r)]_{\mathcal{T}}]\ t$ equivales $s\ [Args(r)]\ t \wedge s, t \in \mathcal{T}$, and if s and t are in $\mathcal{T}$, then so are all of their arguments.

A more complicated property is stated in the next lemma.

Lemma 6.1 For all relations a, b, c, $[(a \circ [b]_{\mathcal{T}})^* \circ c]_{\mathcal{T}} = ([a]_{\mathcal{T}} \circ b)^* \circ [c]_{\mathcal{T}}$.

Proof

$$
\begin{array}{ll}
 & [(a \circ [b]_{\mathcal{T}})^* \circ c]_{\mathcal{T}} \\
= & \\
 & id_{\mathcal{T}} \circ (a \circ id_{\mathcal{T}} \circ b \circ id_{\mathcal{T}})^* \circ c \circ id_{\mathcal{T}} \\
= & \quad \{\text{see remark below}\} \\
 & (id_{\mathcal{T}} \circ a \circ id_{\mathcal{T}} \circ b)^* \circ id_{\mathcal{T}} \circ c \circ id_{\mathcal{T}} \\
= & \\
 & ([a]_{\mathcal{T}} \circ b)^* \circ [c]_{\mathcal{T}}.
\end{array}
$$

Remark: We use an equality of the form $id_{\mathcal{T}} \circ (r \circ id_{\mathcal{T}})^* = (id_{\mathcal{T}} \circ r)^* \circ id_{\mathcal{T}}$. The reader easily verifies that for all $n \in \mathbb{N}$, $id_{\mathcal{T}} \circ (r \circ id_{\mathcal{T}})^n = (id_{\mathcal{T}} \circ r)^n \circ id_{\mathcal{T}}$. □

Now we consider the fixpoint construction according to theorem 4.21, also used in the proof of theorem 5.9, $pr = \cup n \in \mathbb{N}\ pr_n$, where

$$pr_0 := id \quad \text{and} \quad pr_{n+1} := (Args(pr_n) \circ \overline{eqns})^* \circ Args(pr_n).$$

The restriction to $\mathcal{T} \times \mathcal{T}$ of the relations involved in this construction is discussed in the next theorem.

Theorem 6.2 $[pr_{n+1}]_{\mathcal{T}}$ is a function of $[pr_n]_{\mathcal{T}}$.
Proof

$$
\begin{array}{ll}
 & [pr_{n+1}]_{\mathcal{T}} \\
= & \\
 & [(Args(pr_n) \circ \overline{eqns})^* \circ Args(pr_n)]_{\mathcal{T}} \\
= & \quad \{\overline{eqns} = [\overline{eqns}]_{\mathcal{T}}\} \\
 & [(Args(pr_n) \circ [\overline{eqns}]_{\mathcal{T}})^* \circ Args(pr_n)]_{\mathcal{T}} \\
= & \quad \{\text{lemma 6.1}\} \\
 & ([Args(pr_n)]_{\mathcal{T}} \circ \overline{eqns})^* \circ [Args(pr_n)]_{\mathcal{T}} \\
= & \quad \{[Args(pr_n)]_{\mathcal{T}} = [Args([pr_n]_{\mathcal{T}})]_{\mathcal{T}}\} \\
 & ([Args([pr_n]_{\mathcal{T}}) \circ \overline{eqns})^* \circ [Args([pr_n]_{\mathcal{T}})]_{\mathcal{T}}.
\end{array}
$$

□

Now we have $pr_0 \subseteq pr_1 \subseteq pr_2 \subseteq \ldots \subseteq pr$ (see the remark at the end of the proof of theorem 5.9) and hence,

$$[pr_0]_{\mathcal{T}} \subseteq [pr_1]_{\mathcal{T}} \subseteq [pr_2]_{\mathcal{T}} \subseteq \ldots \subseteq [pr]_{\mathcal{T}}.$$

Since there are only finitely many relations on $\mathcal{T} \times \mathcal{T}$, and $[pr_{n+1}]_{\mathcal{T}}$ depends on $[pr_n]_{\mathcal{T}}$ only, we find that there must be an N such that

$$[pr_0]_{\mathcal{T}} \subset [pr_1]_{\mathcal{T}} \subset \ldots \subset [pr_{N-1}]_{\mathcal{T}} \subset [pr_N]_{\mathcal{T}} = [pr_{N+1}]_{\mathcal{T}} = \ldots = [pr]_{\mathcal{T}}.$$

Here $r_1 \subset r_2$ means that $r_1 \subseteq r_2$ and $r_1 \neq r_2$. Assume that there are M terms in $\mathcal{T}$. For all equivalence relations r, let $|r|$ be the number of equivalence classes on $\mathcal{T}$ with respect to r. Then we find $|pr_0| = M$, as $pr_0 = id$, and for all n we find $|pr_n| \geq |pr_{n+1}|$. In fact, if $[pr_n]_{\mathcal{T}} \subset [pr_{n+1}]_{\mathcal{T}}$, then $|pr_n| > |pr_{n+1}|$. Of course there will always be at least one equivalence class, so $|pr_N| \geq 1$, and we find

$$M = |pr_0| > |pr_1| > \ldots > |pr_{N-1}| > |pr_N| = |pr| \geq 1.$$

Obviously the maximum value for $|pr_n|$ is $M - n$, and, as $|pr_N|$ must be greater than, or equal to, 1, we find that the maximum value for N is $M - 1$.

If E contains only finitely many equations, one can of course construct the equivalence classes on $\mathcal{T}$ with respect to pr_n explicitly, and thus decide whether or not $E \models s_0 \doteq t_0$. The method described in [5] can be proved to do just that.

Example 6.3 Let $E = \{a \doteq f\langle a\rangle\}$ and prove that $E \models a \doteq f\langle f\langle a\rangle\rangle$. So, $\mathcal{T} = \{a, f\langle a\rangle, f\langle f\langle a\rangle\rangle\}$ and $\mathcal{T}$ contains $M = 3$ elements. The reader easily verifies that the equivalence classes on $\mathcal{T}$ are

- $\{a\}$, $\{f\langle a\rangle\}$, $\{f\langle f\langle a\rangle\rangle\}$ with respect to pr_0,
- $\{a, f\langle a\rangle\}$, $\{f\langle f\langle a\rangle\rangle\}$ with respect to pr_1,
- $\{a, f\langle a\rangle, f\langle f\langle a\rangle\rangle\}$ with respect to pr_2, and $[pr_2]_{\mathcal{T}} = [pr]_{\mathcal{T}}$.

In this case we find the maximum value for N which is $M - 1 = 2$. □

7 An Application: Equality in Semantic Tableaux

In his paper [3], Reeves proposes to extend the theorem proving method based on semantic tableaux, tableaux method for short, with rules to deal with equations. We shall discuss this method only as far as equality is concerned, and, in theorem 7.3, recognize our characterization of equational provability in it. For detailed discussions of the tableaux method see [1] or [4].

Definition 7.1 A *sequent* S is a finite set of signed equations. A signed equation is an equation labeled with a $+$ or a $-$. Hence, $+s \doteq t$ and $-s \doteq t$ are signed equations. An equation labeled with a $+$ is a positive equation, an equation labeled with a $-$ is a negative equation. A sequent is *satisfiable* iff there is an interpretation in which every positive equation is true, and every negative equation is false. □

The tableaux method is a refutation method, hence we hope to show that a sequent is not satisfiable. To do so, so-called tableau rules are given. If we can conclude that a sequent is not satisfiable by means of these rules, we say that the sequent closes. Reeves defines the following rules concerning equality:

rule 1: S closes if S contains a negative equation of the form $-t \doteq t$, where t may be any term.

rule 2: S closes if S contains a negative equation of the form $-f\langle s_1, ..., s_n\rangle \doteq f\langle t_1, ..., t_n\rangle$, and the sequents $S \cup \{-s_1 \doteq t_1\}$ and ... and $S \cup \{-s_n \doteq t_n\}$ all close.

rule 3: S closes if S contains a negative equation $-s \doteq t$.and a positive equation $+s' \doteq t'$ (or a positive equation $+t' \doteq s'$) and the sequents $S \cup \{-s \doteq s'\}$, $S \cup \{-t' \doteq t\}$ both close.

All rules happen to have the form 'S closes if the sequents $S \cup \{-s_1 \doteq t_1\}$ and ... and $S \cup \{-s_n \doteq t_n\}$ all close, under certain conditions'. (In rule 1 $n = 0$, in rule 2 n is the arity of a function symbol, and in rule 3 $n = 2$.) We call S the input sequent, the $S \cup \{-s_i \doteq t_i\}$ are the output sequents.

Every rule involves exactly one negative equation in the input sequent, and every output sequent contains the same positive equations as the input sequent. Then it is easily seen that, loosely speaking, a sequent closes iff at least one negative equation is individually responsible for this. More formally, if S_T is a set of positive equations then the sequent $S_T \cup \{-s_1 \doteq t_1, ..., -s_n \doteq t_n\}$ closes iff at least one of the sequents $S_T \cup \{-s_1 \doteq t_1\}$, ..., $S_T \cup \{-s_n \doteq t_n\}$ closes. This is a good starting point for a formalization in terms of the relational algebra.

Choose a fixed arbitrary set S_T of positive equations. S_T is represented by the relation $eqns$: s $[eqns]$ t is true iff $+s \doteq t \in S_T$. As before $\overline{eqns}$ denotes the symmetric closure of $eqns$. Further we define the relation pr': s $[pr']$ t is true iff the sequent $S_T \cup \{-s \doteq t\}$ closes. As the reader might expect, pr' will turn out to be equal to pr.

We translate the tableau rules to a definition of pr'. Consider rule 3:

rule 3: $S_T \cup \{-s \doteq t\}$ closes if S_T contains a positive equation $+s' \doteq t'$ (or a positive equation $+t' \doteq s'$) and the sequents $S_T \cup \{-s \doteq s', -s \doteq t\}$, $S_T \cup \{-t' \doteq t, -s \doteq t\}$ both close.

Translating this literally, using that $S_T \cup \{-s_1 \doteq t_1, -s_2 \doteq t_2\}$ closes iff s_1 $[pr']$ $t_1 \vee$ s_2 $[pr']$ t_2 we obtain

rule 3: $s[pr']$ t if there are terms s' and t' such that s' $[\overline{eqns}]$ t' and (s $[pr']$ $t \vee s$ $[pr']$ s') and (s $[pr']$ $t \vee t'$ $[pr']$ t).

If we leave out the tautological conditions we get:

rule 3: s $[pr']$ t if there are terms s' and t' such that s $[pr']$ s' $[\overline{eqns}]$ t' $[pr']$ t. That is, $pr' \circ \overline{eqns} \circ pr' \subseteq pr'$

Translating all rules this way we get the following definition of pr':

Definition 7.2 s $[pr']$ t is true iff it follows from the following rules:

rule 1: $id \subseteq pr'$.

rule 2: $Args(pr') \subseteq pr'$.

rule 3: $pr' \circ \overline{eqns} \circ pr' \subseteq pr'$.

Hence, pr' is defined by $pr' := id \cup Args(pr') \cup pr' \circ \overline{eqns} \circ pr'$. □

Now we can prove $pr' = pr$, and then the completeness of Reeves' rules immediately follows.

Theorem 7.3 $pr' = pr$.

Proof In definition 5.6 pr is defined by $pr = \mu(r \mapsto (Args(r) \circ \overline{eqns})^* \circ Args(r))$. According to theorem 4.25 this is equivalent to $pr = \mu(r \mapsto Args(r) \cup r \circ \overline{eqns} \circ r)$. Lemma 5.8 states that $id \subseteq pr$, so, pr might as well have been defined by $pr = \mu(r \mapsto id \cup Args(r) \cup r \circ \overline{eqns} \circ r)$. Since pr' is defined by the same equation, we conclude $pr' = pr$. □

Theorem 7.4 Reeves' rules are sound and complete. That is, a sequent closes iff it is not satisfiable.

Proof If the sequent $S_T \cup S_F$, $S_F = \{-s_1 \doteq t_1, ..., -s_n \doteq t_n\}$, does not close, then

$$\neg s_1\ [pr]\ t_1 \wedge ... \wedge \neg s_n\ [pr]\ t_n.$$

Since $pr = subst^*$, and there is an interpretation I such that $I \models s \doteq t$ equivales $s\ [subst^*]\ t$ (see the proof of theorem 3.2), we find that all equations in S_T are true in I, while those in S_F are false.

On the other hand, if $S_T \cup S_F$ does close, then at least one of the negative equations follows from the positive equations. Hence, it is impossible that all positive equations are true, while all negative equations are false.

Thus, $S_T \cup S_F$ is satisfiable iff it does not close. □

8 Actual Proofs

Until now we have only discussed equational provability, while it is important of course to give actual proofs. Fortunately, the definitions of pr almost immediately induce a very compact proof format, which is also a guideline in finding such proofs.

As an example the definition of pr according to Reeves' rules is discussed here: $pr := id \cup Args(pr) \cup pr \circ \overline{eqns} \circ pr$. Assume that $s\ [pr]\ t$ is true, because $s\ [Args(pr)]\ t$ is. In that case for some n and some n-ary function symbol f we can write $s = f\langle s_1, ..., s_n\rangle$ and $t = f\langle t_1, ..., t_n\rangle$. Assume there are proofs $p_1, ..., p_n$ for $s_1\ [pr]\ t_1$,...,$s_n\ [pr]\ t_n$, respectively. Now we propose to consider $f\langle p_1, ..., p_n\rangle$ as a proof for $s\ [Args(pr)]\ t$. Note that this implies that each term t is a proof for $t\ [id]\ t$, if id is considered to be defined as a fixpoint of $Args$.

Another possibility is that $s\ [pr]\ t$ is true because $s\ [pr \circ \overline{eqns} \circ pr]\ t$ is, in which case there are terms s' and t' such that $s\ [pr]\ s'\ [\overline{eqns}]\ t'\ [pr]\ t$. There will be a proof p for $s\ [pr]\ s'$ and a proof q for $t'\ [pr]\ t$, and p and q are 'glued' together by an equation. We define $p \doteq q$ to be a proof for $s\ [pr]\ t$ indicating this.

We shall always identify proofs $(p_1 \doteq p_2) \doteq p_3$ and $p_1 \doteq (p_2 \doteq p_3)$, and write $p_1 \doteq p_2 \doteq p_3$ in both cases. All proofs for $s\ [pr]\ t$ yielded by the definition will then be of the form

$$p_0 \doteq \ldots \doteq p_i$$

where there are terms $s{=}s_0$, t_0, ..., s_i, $t_i{=}t$ such that each p_j is a proof for $s_j\ [Args(pr)]\ t_j$, and p_j is linked to p_{j+1} by an equation in E, that is, $s_j\ [\overline{eqns}]\ t_{j+1}$.

It is noteworthy that this description directly reflects the definition of pr by $pr := (Args(pr) \circ \overline{eqns})^* \circ Args(pr)$.

Example 8.1 Consider the problem of finding a proof of $f\langle e\rangle \doteq g\langle c\rangle$ from $E = \{a \doteq b, c \doteq d, f\langle a\rangle \doteq g\langle c\rangle, b \doteq f\langle b\rangle, e \doteq g\langle d\rangle\}$. (The example is also discussed in [3]. It is originally due to [5].) We start with a framework of this proof, that we denote

$$f\langle e\rangle \cdots g\langle c\rangle.$$

We can obviously not have $f\langle e\rangle\ [Args(pr)]\ g\langle c\rangle$ because $f \neq g$, so we have to find terms s and t, such that $s\ [\overline{eqns}]\ t$. If proofs p and q can be found for $f\langle e\rangle\ [pr]\ s$ and $t\ [pr]\ g\langle c\rangle$, respectively, then $p \doteq q$ is a proof for $f\langle e\rangle\ [pr]\ g\langle c\rangle$. As a framework for p we write $f\langle e\rangle \cdots s$, and as a framework for q we write $t \cdots g\langle c\rangle$. So, the framework for the entire proof becomes $f\langle e\rangle \cdots s \doteq t \cdots g\langle c\rangle$.

To direct the search, we shall assume that p is a proof for $f\langle e\rangle\ [Args(pr)]\ s$. In that case $p \doteq q$ will be a proof for $f\langle e\rangle\ [Args(pr) \circ \overline{eqns} \circ pr]\ g\langle c\rangle$. This corresponds to a definition of pr by $pr := id \cup Args(pr) \cup Args(pr) \circ \overline{eqns} \circ pr$, which is equivalent to the other definitions of pr we have discussed, as can be proved using lemma 5.8 and theorem 4.25.

This leaves two possibilities for the choice of s and t. Either $s = f\langle b\rangle \doteq b$ and $t = b$ or $s = f\langle a\rangle$ and $t = g\langle c\rangle$. The first possibility does not lead to a proof, so we develop the second one. It yields the framework

$$f\langle e\rangle \cdots f\langle a\rangle \doteq g\langle c\rangle \cdots g\langle c\rangle.$$

So the framework for p is $f\langle e\rangle \cdots f\langle a\rangle$. We wanted p to be a proof for $f\langle e\rangle\ [Args(pr)]\ f\langle a\rangle$, so if there is a proof p' for $e\ [pr]\ a$, then $p = f\langle p'\rangle$ for which $f\langle e \cdots a\rangle$ is the framework. The framework $g\langle c\rangle \cdots g\langle c\rangle$ can immediately be completed to the proof $g\langle c\rangle$. This yields the framework

$$f\langle e \cdots a\rangle \doteq g\langle c\rangle.$$

If we proceed the same way we can find a proof with the following steps:

$$f\langle e \cdots e \doteq g\langle d\rangle \cdots a\rangle \doteq g\langle c\rangle$$
$$f\langle e \doteq g\langle d\rangle \cdots g\langle c\rangle \doteq f\langle a\rangle \cdots a\rangle \doteq g\langle c\rangle$$
$$f\langle e \doteq g\langle d \cdots c\rangle \doteq f\langle a\rangle \cdots f\langle b\rangle \doteq b \cdots a\rangle \doteq g\langle c\rangle$$
$$f\langle e \doteq g\langle d \cdots d \doteq c \cdots c\rangle \doteq f\langle a \cdots b\rangle \doteq b \cdots b \doteq a \cdots a\rangle \doteq g\langle c\rangle$$
$$f\langle e \doteq g\langle d \doteq c\rangle \doteq f\langle a \cdots a \doteq b \cdots b\rangle \doteq b \doteq a\rangle \doteq g\langle c\rangle$$
$$f\langle e \doteq g\langle d \doteq c\rangle \doteq f\langle a \doteq b\rangle \doteq b \doteq a\rangle \doteq g\langle c\rangle.$$

The proof that is obtained is $f\langle e \doteq g\langle d \doteq c\rangle \doteq f\langle a \doteq b\rangle \doteq b \doteq a\rangle \doteq g\langle c\rangle$.

Note how we can still recognize in this proof the principle of 'replacing equals by equals'. Reading it from left to right it simply tells us to start with $f\langle e\rangle$, replace e by $g\langle d\rangle$, replace d by c, etc. in order to finally obtain $g\langle c\rangle$.

It is not so hard to implement a proof search procedure (in Prolog, for instance), that finds proofs along these lines. We shall address this topic again in section 9. □

Example 8.2 The way a proof was obtained in example 8.1 can immediately be translated into a way to show that the sequent $S_T \cup \{-f\langle e\rangle \doteq g\langle c\rangle\}$, where $S_T = \{+a \doteq b, +c \doteq d, +f\langle a\rangle \doteq g\langle c\rangle, +b \doteq f\langle b\rangle, +e \doteq g\langle d\rangle\}$, can be closed by means of Reeves' rules.

First we apply rule 3 with respect to $+f\langle a\rangle \doteq g\langle c\rangle$ and $-f\langle e\rangle \doteq g\langle c\rangle$, yielding the two sequents

$$S_T \cup \{-f\langle e\rangle \doteq g\langle c\rangle, -f\langle e\rangle \doteq f\langle a\rangle\}$$
$$S_T \cup \{-f\langle e\rangle \doteq g\langle c\rangle, -g\langle c\rangle \doteq g\langle c\rangle\}$$

Note that the negative equations that are added to the sequent correspond to the empty spaces in the framework $f\langle e\rangle \cdots f\langle a\rangle \doteq g\langle c\rangle \cdots g\langle c\rangle$.

The second sequent is immediately closed by rule 1, so we consider the first sequent, and apply rule 2 to $-f\langle e\rangle \doteq f\langle a\rangle$. This corresponds to the replacement of the framework $f\langle e\rangle \cdots f\langle a\rangle$ by $f\langle e \cdots a\rangle$, and yields the following sequent:

$$S_T \cup \{-f\langle e\rangle \doteq g\langle c\rangle, -f\langle e\rangle \doteq f\langle a\rangle, -e \doteq a\}$$

In this way the entire derivation of a proof in example 8.1 can be translated in terms of Reeves' rules. □

9 Discussion

We are interested in equational proofs, since we wish to extend an existing tableaux based theorem prover, implemented in Prolog (see [4]), with rules to deal with equality. In fact we have already implemented a simple prototype of such an extended theorem prover. In this prototype we find the relation pr implemented in a straightforward manner. Only slight adaptations were necessary to cope with the use of logic variables, and to prevent the theorem prover from getting into infinite loops. Having the proof search guided by the definition of pr, more or less as it happens in example 8.1, has the advantage of being goal directed, as the theorem prover attempts to complete the framework of a proof, whereas this proof can be output when it is completed.

Future research will aim at further improving the extended theorem prover, such that it is provided with better heuristics. An interesting issue in this respect is that a simple Knuth-Bendix-like completion procedure, completing a finite set of equations, can also be described in terms of the relational algebra. We hope to be able to generalize this description to cope with the use of logic variables in the tableaux method. We hope to report on this subject in a forthcoming paper.

A Appendix: Continuity

Theorem A.1 All mappings that we can construct using the operators we defined, are continuous and hence monotonic. We specify this:

- For every relation a, the constant mapping $r \mapsto a$ is continuous.
- The mapping $r \mapsto r$ is continuous.
- If F and G are continuous mappings, then so are

$$\begin{aligned} r &\mapsto F(r) \cup G(r), \\ r &\mapsto F(r) \circ G(r), \\ r &\mapsto F(G(r)). \end{aligned}$$

- If $\mathcal{F}$ is a set of continuous mappings, then $r \mapsto \cup F \in \mathcal{F}\ F(r)$ is also continuous.
- *Pick* and *Args* are continuous.

Proof We shall discuss a few examples. Let R be a directed set of relations.

- Let F and G be continuous mappings, then $r \mapsto F(r) \circ G(r)$ is continuous.

$$\begin{aligned} & F(\cup r \in R\ r) \circ G(\cup r \in R\ r) \\ = & \quad \{F \text{ and } G \text{ are continuous}\} \\ & (\cup r \in R\ F(r)) \circ (\cup r \in R\ G(r)) \\ = & \quad \{\text{theorem 4.2, rename dummies}\} \\ & \cup r_1, r_2 \in R\ F(r_1) \circ G(r_2) \\ = & \quad \{\text{see remark below}\} \\ & \cup r \in R\ F(r) \circ G(r) \end{aligned}$$

  Remark: R is directed, so for the subset $\{r_1, r_2\}$ of R there is an $r \in R$ such that $r_1 \subseteq r$ and $r_2 \subseteq r$. Monotonicity of F and G implies $F(r_1) \subseteq F(r)$ and $G(r_2) \subseteq G(r)$. Theorem 4.3 implies $F(r_1) \circ G(r_2) \subseteq F(r) \circ G(r)$.

- Let F and G be continuous mappings, then $r \mapsto F(G(r))$ is also continuous. $G(R) = \{G(r) \mid r \in R\}$ is also a directed set of relations, since all its finite subsets can be written as $\{G(r_1), ..., G(r_n)\}$, where $\{r_1, ..., r_n\} \subseteq R$. Since R is directed, there is an $r \in R$ such that $r_1 \subseteq r \wedge ... \wedge r_n \subseteq r$. Since G is monotonic this implies $G(r_1) \subseteq G(r) \wedge ... \wedge G(r_n) \subseteq G(r)$.

$$\begin{aligned} & F(G(\cup r \in R\ r)) \\ = & \quad \{G \text{ is continuous}\} \\ & F(\cup r \in R\ G(r)) \\ = & \end{aligned}$$

$$
\begin{array}{ll}
 & F(\cup g \in G(R)\ g) \\
= & \quad \{F \text{ is continuous, } G(R) \text{ is directed}\} \\
 & \cup g \in G(R)\ F(g) \\
= & \\
 & \cup r \in R\ F(G(r))
\end{array}
$$

- If $\mathcal{F}$ is a set of continuous mappings, then $r \mapsto \cup F \in \mathcal{F}\ F(r)$ is also continuous.

$$
\begin{array}{ll}
 & \cup F \in \mathcal{F}\ F(\cup r \in R\ r) \\
= & \quad \{\text{each } F \text{ is continuous}\} \\
 & \cup F \in \mathcal{F}\ (\cup r \in R\ F(r)) \\
= & \\
 & \cup r \in R\ (\cup F \in \mathcal{F}\ F(r))
\end{array}
$$

- *Pick* is continuous. For simplicity we shall prove this using terms with a binary outermost function symbol f. Generalization is straightforward.

$$
\begin{array}{ll}
 & f\langle s_1, s_2\rangle\ [Pick(\cup r \in R\ r)]\ f\langle t_1, t_2\rangle \\
\Leftrightarrow & \quad \{\text{definition of } Pick\} \\
 & (s_1\ [\cup r \in R\ r]\ t_1 \wedge s_2 = t_2) \vee \\
 & \vee\ (s_1 = t_1 \wedge s_2\ [\cup r \in R\ r]\ t_2) \\
\Leftrightarrow & \\
 & ((\exists r\ r \in R\ s_1\ [r]\ t_1) \wedge s_2 = t_2) \vee \\
 & \qquad \vee\ (s_1 = t_1 \wedge (\exists r\ r \in R\ s_2\ [r]\ t_2)) \\
\Leftrightarrow & \quad \{\text{predicate calculus}\} \\
 & (\exists r\ r \in R\ (s_1\ [r]\ t_1 \wedge s_2 = t_2) \vee \\
 & \qquad \vee\ (s_1 = t_1 \wedge s_2\ [r]\ t_2)) \\
\Leftrightarrow & \quad \{\text{definition of } Pick\} \\
 & f\langle s_1, s_2\rangle\ [\cup r \in R\ Pick(r)]\ f\langle t_1, t_2\rangle
\end{array}
$$

Note that we did not use the fact that R is directed.

□

Consequences For all $n \in \mathbb{N}$, the mapping $r \mapsto r^n$ is continuous. The mapping $r \mapsto r^*$ is continuous.

Proof According to theorem A.1, $r \mapsto id = r^0$ and $r \mapsto r = r^1$ are continuous. Given that $r \mapsto r$ and $r \mapsto r^k$ are continuous we find that $r \mapsto r \circ r^k = r^{k+1}$ is also continuous. Finally, given that for all $n \in \mathbb{N}$, $r \mapsto r^n$ is continuous, we find that $r \mapsto \cup n \in \mathbb{N}\ r^n = r^*$ is continuous. □

References

[1] Fitting, M.: 'First-Order Logic and Automated Theorem Proving, Springer Verlag, (1990).

[2] Lloyd J.W.: 'Foundations of Logic Programming', Springer Verlag, (1984).
id., Second, Extended Edition, (1987).

[3] Reeves S.V.: 'Adding Equality to Semantic Tableaux', Journal of Automated Reasoning **3**, pp. 225-246, (1987).

[4] Swart, H.C.M. de and Ophelders W.M.J.: 'Tableaux versus Resolution; A Comparison', Fundamenta Informaticae **18**, pp. 109-127, (1993).

[5] Shostak, R.E.: 'An Algorithm for Reasoning about Equality', C.A.C.M. **21**, pp. 583-585, (1978).

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.
91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.

91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.
91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.

92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.
92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.

92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.

93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness?, p. 24.
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.