Point-free substitution

Eindhoven University of Technology

Department of Mathematics and Computing Science

Point-free substitution

by

A. Bijlsma and C.S. Scholten

94/38

ISSN 0926-4515

Computing Science Report 94/38
Eindhoven, September 1994

Point-free substitution

A. Bijlsma and C.S. Scholten

June 14, 1994

0 Introduction

In modern treatments of predicate calculus [2], no mention is made of states or variables up until the point where substitution is introduced. There, suddenly, the abstraction that reigned before is cast to the winds, and a substitution is defined as the result of a textual replacement of variable names by expressions. It is the purpose of this note to remedy this breach of style by proposing a new characterization of substitution, one that is meaningful also in point-free models, and is equivalent to the classical definition [3] wherever the latter is applicable. More precisely, we prove that

> *a predicate transformer is a substitution according to the classical definition iff it is both universally conjunctive and universally disjunctive.*

To this purpose, we introduce a number of postulates limiting the set of valid models for the predicate calculus until we finally arrive at a context where variables are available.

Our first postulate concerns the existence of a covering set of point predicates. In Section 1 we deduce from this that universal junctivity of f is equivalent to the existence of a point predicate transformer h such that, for every point predicate p and every predicate x,

$$[p \Rightarrow f.x] \;\equiv\; [h.p \Rightarrow x] \quad .$$

In Section 2 we postulate the existence of a state space and prove that universal junctivity of f is equivalent to the existence of a state transformer g such that, for every predicate x,

$$[f.x \equiv x \circ g] \quad .$$

Our third and final postulate introduces variables and enables us to prove, in Section 3, equivalence to a classical definition of substitution.

Throughout, we assume familiarity with predicate calculus as developed in [2].

1 Point predicates

Definition 1 The set P of *point predicates* is defined by

$$p \in P \;\equiv\; (\forall x :: [p \Rightarrow x] \not\equiv [p \Rightarrow \neg x]) \quad , \tag{1}$$

where p and x range over the predicates. Functions mapping P into itself will be called *point predicate transformers.*

□

The definition of P makes sense in every model for the predicate calculus, but there exist models where P is the empty set [4, page 29]. In order to exclude such surprises, we introduce our first postulate.

Postulate 2 $[(\exists p : p \in P : p)]$.
□

The following lemma shows that now every predicate may be written as a disjunction of point predicates.

Lemma 3 *For every predicate* x,

$$[x \equiv (\exists p : p \in P \ \wedge \ [p \Rightarrow x] : p)] \ .$$

Proof For any x, with the range $p \in P$ omitted,

$$\begin{array}{ll}
 & x \\
\equiv & \quad \{\text{Postulate 2}\} \\
 & x \ \wedge \ (\exists p :: p) \\
\equiv & \quad \{\wedge \text{ over } \exists\} \\
 & (\exists p :: x \wedge p) \\
\equiv & \quad \{\text{splitting the range}\} \\
 & (\exists p : [p \Rightarrow x] : x \wedge p) \ \vee \ (\exists p : \neg[p \Rightarrow x] : x \wedge p) \\
\equiv & \quad \{\text{eliminating implications, using (1)}\} \\
 & (\exists p : [p \equiv x \wedge p] : x \wedge p) \ \vee \ (\exists p : [x \wedge p \equiv \mathit{false}] : x \wedge p) \\
\equiv & \quad \{\text{Leibniz}\} \\
 & (\exists p : [p \equiv x \wedge p] : p) \ \vee \ (\exists p : [x \wedge p \equiv \mathit{false}] : \mathit{false}) \\
\equiv & \quad \{\text{reintroducing implication; term } \mathit{false}\} \\
 & (\exists p : [p \Rightarrow x] : p) \ .
\end{array}$$

□

Lemma 3 enables us to formulate our first alternative characterization of universal junctivity.

Theorem 4 *For predicate transformer* f, *the following are equivalent:*

(i) there exists a point predicate transformer h *such that for every point predicate* p *and every predicate* x,

$$[p \Rightarrow f.x] \ \equiv \ [h.p \Rightarrow x] \ , \tag{2}$$

(ii) f *is universally conjunctive and universally disjunctive.*

Proof of (i)⇒(ii) Choose h to satisfy (2). We begin by proving that f is universally conjunctive. For any any set V of predicates we have, the ranges $x \in V$ and $p \in P$ being understood,

$$\begin{array}{ll}
 & [f.(\forall x :: x) \ \equiv \ (\forall x :: f.x)] \\
\equiv & \quad \{\text{Lemma 3, twice}\} \\
 & [(\exists p : [p \Rightarrow f.(\forall x :: x)] : p) \ \equiv \ (\exists p : [p \ \Rightarrow \ (\forall x :: f.x)] : p)] \\
\Leftarrow & \quad \{\text{termwise equality}\} \\
 & (\forall p :: [p \ \Rightarrow \ f.(\forall x :: x)] \ \equiv \ [p \ \Rightarrow \ (\forall x :: f.x)]) \ .
\end{array}$$

Now for any p,

$$
\begin{array}{ll}
 & [p \;\Rightarrow\; f.(\forall x :: x)] \\
\equiv & \qquad \{(2) \text{ with } x := (\forall x :: x)\} \\
 & [h.p \;\Rightarrow\; (\forall x :: x)] \\
\equiv & \qquad \{\text{distribution}\} \\
 & (\forall x :: [h.p \Rightarrow x]) \\
\equiv & \qquad \{(2)\} \\
 & (\forall x :: [p \Rightarrow f.x]) \\
\equiv & \qquad \{\text{distribution}\} \\
 & [p \;\Rightarrow\; (\forall x :: f.x)] \;\;,
\end{array}
$$

which proves the universal conjunctivity.

In order to prove f's universal disjunctivity, it suffices to prove

$$f = f^* \quad , \tag{3}$$

since—loosely speaking—f's conjunctivity is f^*'s disjunctivity (see Theorem 6.9 of [2]). Indeed, for any p in P and any predicate x,

$$
\begin{array}{ll}
 & [p \;\Rightarrow\; f^*.x] \\
\not\equiv & \qquad \{(1) \text{ with } x := f^*.x\} \\
 & [p \;\Rightarrow\; \neg f^*.x] \\
\equiv & \qquad \{\text{conjugate}\} \\
 & [p \;\Rightarrow\; f.(\neg x)] \\
\equiv & \qquad \{(2) \text{ with } x := \neg x\} \\
 & [h.p \;\Rightarrow\; \neg x] \\
\not\equiv & \qquad \{(1) \text{ with } p := h.p, \text{ using } h.p \in P\} \\
 & [h.p \;\Rightarrow\; x] \\
\equiv & \qquad \{(2)\} \\
 & [p \;\Rightarrow\; f.x] \;\;,
\end{array}
$$

from which, again with the help of Lemma 3, (3) follows.

Proof of (ii)$\Rightarrow$(i) From f's universal conjunctivity it follows that there exists a predicate transformer g with

$$[g.x \;\Rightarrow\; y] \;\equiv\; [x \;\Rightarrow\; f.y] \tag{4}$$

for all predicates x, y (see Theorem 11.1 of [2]). A correspondence of this kind is sometimes called a *Galois connection* [1]. We wish to take h as the restriction of g to P; this yields the proof obligation

$$(\forall p : p \in P : g.p \in P) \quad ,$$

which is discharged as follows: for $p \in P$ and any predicate y,

$$
\begin{array}{ll}
 & [g.p \;\Rightarrow\; y] \\
\equiv & \qquad \{(4)\} \\
 & [p \;\Rightarrow\; f.y]
\end{array}
$$

$$
\begin{array}{ll}
\equiv & \quad \{ [f.y \equiv \neg f.(\neg y)] \text{, see below} \} \\
 & [p \Rightarrow \neg f.(\neg y)] \\
\not\equiv & \quad \{ (1) \} \\
 & [p \Rightarrow f.(\neg y)] \\
\equiv & \quad \{ (4) \} \\
 & [g.p \Rightarrow \neg y] \quad ,
\end{array}
$$

from which $g.p \in P$ follows by (1). The second step in the above derivation is justified since for any y

$$
\begin{array}{ll}
 & f.y \equiv \neg f.(\neg y) \\
\equiv & \quad \{ \text{eliminating the equivalence} \} \\
 & (f.y \vee f.(\neg y)) \ \wedge \ \neg(f.y \wedge f.(\neg y)) \\
\equiv & \quad \{ f \text{ is finitely disjunctive and finitely conjunctive} \} \\
 & f.(y \vee \neg y) \ \wedge \ \neg f.(y \wedge \neg y) \\
\equiv & \quad \{ \text{Excluded Middle} \} \\
 & f.true \ \wedge \ \neg f.false \\
\equiv & \quad \{ f \text{ is conjunctive and disjunctive over the empty set} \} \\
 & true \quad .
\end{array}
$$

□

Remark Inspection of the proofs shows that we have not explicitly used the universal disjunctivity of f, only disjunctivity over finite (possibly empty) sets. This observation, however, does not strengthen the theorem, because every universally conjunctive predicate transformer that is disjunctive over finite sets is also universally disjunctive. This was proved by Scholten [5], as a generalization of a theorem of Van der Woude [2, Theorem 6.25].
□

The reader who is already convinced that (i) of Theorem 4 captures the notion of substitution may quit here. Others may wish to read on.

End of scope of Postulate 2.

2 State transformers

In this section, we restrict ourselves to predicates as functions on a state space. We shall see that this implies the existence of a covering set of point predicates as claimed in Postulate 2, which reappears below as Lemma 10.

Postulate 5 *The predicates are boolean functions on some set S, such that*

$$(\forall x : x \in V : x).s \ \equiv \ (\forall x : x \in V : x.s) \quad , \tag{5}$$

$$(\neg y).s \ \equiv \ \neg(y.s) \quad , \tag{6}$$

$$[y] \ \equiv \ (\forall t : t \in S : y.t) \tag{7}$$

for $s \in S$, predicate y and set V of predicates.
□

We shall call S the *state space* and its elements *states*; a function mapping S into itself is called a *state transformer*. No doubt some readers would prefer denoting the operators and quantifiers on the right hand side, which take boolean constants as their operands, differently from those on the left hand side, which operate on boolean functions. We have not found such a distinction to be useful.

Notice that (5) and (6) guarantee that $(.s)$ distributes over all boolean operators; in particular, it follows that

$$(x \Rightarrow y).s \;\equiv\; x.s \;\Rightarrow\; y.s \quad , \tag{8}$$

$$(x \equiv y).s \;\equiv\; x.s \;\equiv\; y.s \quad . \tag{9}$$

In order to prepare for Lemma 10, we define predicates C_t as follows:

Definition 6 For state t, predicate C_t is defined by

$$(\forall s : s \in S : C_t.s \;\equiv\; t = s) \quad .$$

□

The predicates C_t allow us to express application of a predicate to a state differently, as is shown in the next lemma.

Lemma 7 *For state t and predicate x,*

$$x.t \;\equiv\; [C_t \Rightarrow x] \quad .$$

Proof

$$\begin{array}{ll} & [C_t \Rightarrow x] \\ \equiv & \qquad \{(7)\} \\ & (\forall s : s \in S : (C_t \Rightarrow x).s) \\ \equiv & \qquad \{(8)\} \\ & (\forall s : s \in S : C_t.s \Rightarrow x.s) \\ \equiv & \qquad \{\text{Definition 6}\} \\ & (\forall s : s \in S : t = s \;\Rightarrow\; x.s) \\ \equiv & \qquad \{\text{trading; one-point rule}\} \\ & x.t \quad . \end{array}$$

□

Now we are ready to show that set of the C_t equals the set of point predicates.

Lemma 8 *For all predicates p,*

$$p \in P \;\equiv\; (\exists t : t \in S : [p \equiv C_t]) \quad .$$

Proof of LHS $\Rightarrow$ RHS For any predicate p we have, with t ranging over the states,

$$
\begin{array}{ll}
 & (\exists t :: [p \equiv C_t]) \\
\equiv & \quad \{\text{mutual implication}\} \\
 & (\exists t :: [p \Rightarrow C_t] \ \wedge \ [C_t \Rightarrow p]) \\
\Leftarrow & \quad \{\text{predicate calculus, guided by the form of (1)}\} \\
 & (\forall t :: [p \Rightarrow C_t] \ \equiv \ [C_t \Rightarrow p]) \ \wedge \ (\exists t :: [C_t \Rightarrow p]) \\
\equiv & \quad \{\text{Lemma 7 with } x := p\text{, twice}\} \\
 & (\forall t :: [p \Rightarrow C_t] \ \equiv \ p.t) \ \wedge \ (\exists t :: p.t) \\
\equiv & \quad \{\text{Lemma 7 with } x := \neg p\text{; de Morgan}\} \\
 & (\forall t :: [p \Rightarrow C_t] \ \equiv \ \neg[C_t \Rightarrow \neg p]) \ \wedge \ \neg(\forall t :: \neg p.t) \\
\equiv & \quad \{\text{contraposition; (7)}\} \\
 & (\forall t :: [p \Rightarrow C_t] \ \equiv \ \neg[p \Rightarrow \neg C_t]) \ \wedge \ \neg[p \Rightarrow \mathit{false}] \\
\Leftarrow & \quad \{\text{(1) with } x := C_t\} \\
 & p \in P \ \wedge \ \neg[p \Rightarrow \mathit{false}] \\
\equiv & \quad \{\text{(1) with } x := \mathit{false}\} \\
 & p \in P \ \wedge \ [p \Rightarrow \mathit{true}] \\
\equiv & \quad \{\text{second conjunct is } \mathit{true}\} \\
 & p \in P \quad .
\end{array}
$$

Proof of LHS $\Leftarrow$ RHS With p and x ranging over the predicates and t over the states,

$$
\begin{array}{ll}
 & (\forall p :: p \in P \ \Leftarrow \ (\exists t :: [p \equiv C_t])) \\
\equiv & \quad \{\Leftarrow \text{ over } \exists\} \\
 & (\forall p, t :: p \in P \ \Leftarrow \ [p \equiv C_t]) \\
\equiv & \quad \{\text{trading; one-point rule}\} \\
 & (\forall t :: C_t \in P) \\
\equiv & \quad \{(1)\} \\
 & (\forall t, x :: [C_t \Rightarrow x] \not\equiv [C_t \Rightarrow \neg x]) \\
\equiv & \quad \{\text{Lemma 7}\} \\
 & (\forall t, x :: x.t \not\equiv \neg x.t) \\
\equiv & \quad \{\text{term is } \mathit{true}\} \\
 & \mathit{true} \quad .
\end{array}
$$

□

The following 'dummy transformation rule' is an immediate consequence of Lemma 8.

Lemma 9 *For every predicate transformer f,*

$$[(\mathsf{Q}\, t : t \in S : f.C_t) \ \equiv \ (\mathsf{Q}\, p : p \in P : f.p)] \quad ,$$

where $\mathsf{Q} = \forall$ or $\mathsf{Q} = \exists$.

Proof

$$
\begin{array}{ll}
 & (\mathsf{Q}\, t : t \in S : f.C_t) \\
\equiv & \quad \{\text{one-point rule}\} \\
 & (\mathsf{Q}\, t : t \in S : (\mathsf{Q}\, p : [p \equiv C_t] : f.p)) \\
\equiv & \quad \{\text{generalized range split}\} \\
 & (\mathsf{Q}\, p : (\exists t : t \in S : [p \equiv C_t]) : f.p) \\
\equiv & \quad \{\text{Lemma 8}\} \\
 & (\mathsf{Q}\, p : p \in P : f.p) \quad .
\end{array}
$$

□

As announced above, we are now able to prove Postulate 2.

Lemma 10 $[(\exists p : p \in P : p)]$.

Proof

$$
\begin{array}{ll}
 & [(\exists p : p \in P : p)] \\
\equiv & \qquad \{\text{Lemma } 9\} \\
 & [(\exists t : t \in S : C_t)] \\
\equiv & \qquad \{(7)\} \\
 & (\forall s : s \in S : (\exists t : t \in S : C_t.s)) \\
\equiv & \qquad \{\text{Definition } 6\} \\
 & (\forall s : s \in S : (\exists t : t \in S : t = s)) \\
\equiv & \qquad \{\text{instantiation } t := s\} \\
 & \mathit{true} \quad .
\end{array}
$$

□

On account of Lemma 10 we are allowed to import every result from Section 1, in particular Theorem 4. We are now in a position to present another property equivalent to universal junctivity.

Theorem 11 *For predicate transformer f the following are equivalent:*
(i) there exists a state transformer g such that, for every predicate x,

$$[f.x \equiv x \circ g] \quad ,$$

(ii) f is universally conjunctive and universally disjunctive.

Proof In the proof, we let dummy x range over the predicates, p over the point predicates, t over the states, g over the state transformers, and h over the point predicate transformers.
We start by transforming both (i) and (ii) into comparable shapes.

$$
\begin{array}{ll}
 & \text{(i)} \\
\equiv & \qquad \{\text{by definition}\} \\
 & (\exists g :: (\forall x :: [f.x \equiv x \circ g])) \\
\equiv & \qquad \{(7)\} \\
 & (\exists g :: (\forall x, t :: (f.x \equiv x \circ g).t)) \\
\equiv & \qquad \{(9)\} \\
 & (\exists g :: (\forall x, t :: f.x.t \ \equiv \ x.(g.t))) \\
\equiv & \qquad \{\text{Lemma 7 with } t := g.t\} \\
 & (\exists g :: (\forall x, t :: f.x.t \ \equiv \ [C_{g.t} \Rightarrow x])) \quad ,
\end{array}
$$

and

$$
\begin{array}{ll}
 & \text{(ii)} \\
\equiv & \qquad \{\text{Theorem } 4\} \\
 & (\exists h :: (\forall p, x :: [p \Rightarrow f.x] \ \equiv \ [h.p \Rightarrow x]))
\end{array}
$$

$$\begin{array}{ll} \equiv & \{\text{Lemma 9}\} \\ & (\exists h :: (\forall x,t :: [C_t \Rightarrow f.x] \;\equiv\; [h.C_t \Rightarrow x])) \\ \equiv & \{\text{Lemma 7 with } x := f.x\} \\ & (\exists h :: (\forall x,t :: f.x.t \;\equiv\; [h.C_t \Rightarrow x])) \;\; . \end{array}$$

The equivalence of

$$(\exists g :: (\forall x,t :: f.x.t \;\equiv\; [C_{g.t} \Rightarrow x])) \tag{10}$$

and

$$(\exists h :: (\forall x,t :: f.x.t \;\equiv\; [h.C_t \Rightarrow x])) \tag{11}$$

is proved by mutual implication. First, let a state transformer g be given that is a witness for (10). Then, for every state t, $g.t$ is also a state, and hence, by Lemma 8, $C_{g.t}$ is a point predicate. Again by Lemma 8, every C_t is a point predicate; from Definition 6 it follows immediately that distinct states t correspond to distinct point predicates C_t. Consequently, a point predicate transformer h can be defined by

$$(\forall t :: [h.C_t \equiv C_{g.t}]) \;\; . \tag{12}$$

This h is a witness for (11).

Conversely, let a point predicate transformer h be given that is a witness for (11). Let t be any state. Then C_t is a point predicate by Lemma 8 and so, therefore, is $h.C_t$. Again by Lemma 8, the point predicate $h.C_t$ equals C_s for some state s. Now define $g.t$ to be s. This defines a state transformer g satisfying (12); it is a witness for (10).
□

Again, the reader who is convinced that (i) of Theorem 11 captures the notion of substitution may quit here.

3 Substitution

We retain Postulate 5, but add another postulate in order to introduce coordinates into the state space.

Postulate 12 *All states are functions defined on the same finite set.*
□

The elements of this finite set will be called *variables*. With the aid of Postulate 12, we can give a conventional definition of substitution; see, for instance, [3]. As usual, a *structure* is a function on the state space.

Definition 13 A state transformer g is called an *update* iff there exists a list v of distinct variables and an equally long list φ of structures, such that

$$g.s.w = \begin{cases} s.w & \text{if } w \notin v \;\; , \\ \varphi_k.s & \text{if } w = v_k \end{cases} \tag{13}$$

for state s and variable w. A *substitution* is a predicate transformer f such that, for all predicates x,

$$[f.x \equiv x \circ g] \quad ,$$

where g is an update.
□

Example 14 Consider the state space spanned by the integer variables a, b and c. An example of a structure on this state space is the function mapping each state s to $s.a + s.b$ An example of an update on this space is g defined by

$$g.s.a = s.b \quad , \quad g.s.b = s.a + s.b \quad , \quad g.s.c = s.c$$

for every state s. An example of a substitution on this space is f defined by

$$f.x.s = x.(g.s)$$

for every state s. The substitution f would traditionally be denoted by $(a, b := b, a + b)$.
□

Observe that lists v and φ as occurring in (13) are not uniquely determined by the state transformer g. Indeed, if $w \notin v$, lists v and φ may be extended with w and ψ respectively, where ψ is defined by $\psi.s = s.w$ for all states s. Hence, without loss of generality, we may assume v to be a list of *all* variables. In that case the first alternative in (13) does not occur.

Lemma 15 *Every state transformer is an update.*

Proof Let g be a state transformer. Let v be a list of all variables; define list φ by

$$\varphi_k.s = g.s.v_k$$

for every index k and all s. Then g satisfies (13).
□

Combination of Lemma 15 and Theorem 11 finally yields the result promised in the Introduction:

Theorem 16 *For predicate transformer f the following are equivalent:*
(i) f is a substitution,
(ii) f is universally conjunctive and universally disjunctive.
□

End of scope of postulates 5 and 12.

References

[1] R.C. Backhouse and J.C.S.P. van der Woude, 'Demonic operators and monotype factors'. *Math. Struct. in Comp. Sci.* **3** (1993), 417–433.

[2] E.W. Dijkstra and C.S. Scholten, *Predicate calculus and program semantics*. Springer-Verlag, New York, 1990.

[3] W.H. Hesselink, *Programs, recursion and unbounded choice: predicate-transformation semantics and transformation rules*. Cambridge Tracts in Theoretical Computer Science; 27. Cambridge University Press, 1992.

[4] J.D. Monk and R. Bonnet, *Handbook of boolean algebras*, vol. I. North-Holland, Amsterdam, 1989.

[5] C.S. Scholten, *A generalization of Van der Woude's theorem*. Memorandum CSS144. Beekbergen, 1988.

Computing Science Reports **Department of Mathematics and Computing Science**
Eindhoven University of Technology

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.
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.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.
93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.

93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Komatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.
94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doornbos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.

94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.
94/30	J.C.M. Baeten J.A. Bergstra Gh. Ştefanescu	Process Algebra with Feedback, p. 22.
94/31	B.W. Watson R.E. Watson	A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.
94/32	J.J. Vereijken	Fischer's Protocol in Timed Process Algebra, p. 38.
94/33	T. Laan	A formalization of the Ramified Type Theory, p.40.
94/34	R. Bloo F. Kamareddine R. Nederpelt	The Barendregt Cube with Definitions and Generalised Reduction, p. 37.
94/35	J.C.M. Baeten S. Mauw	Delayed choice: an operator for joining Message Sequence Charts, p. 15.
94/36	F. Kamareddine R. Nederpelt	Canonical typing and Π-conversion in the Barendregt Cube, p. 19.

94/37	T. Basten R. Bol M. Voorhoeve	Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.