Copy two of the re-launched FAC-FACTS should be with you before the winter solstice celebrations begin. Whilst we have you in the festive and hopefully generous spirit we are asking you to renew your membership of BCS FACS Special Interest Group, see pages (3-8).

There has been serious concern over the whereabouts of F. X. Reid, our roving reporter Victor Zemanticz has news hot off the press to allay some of our readers concern.

Finally, FACS-FACTS is an "open newsletter" - it welcomes contributions, particularly reports of works by colleagues just starting on research. Why not send us your draft research reports? The newsletter provides an excellent forum to get your early thoughts circulated and "reviewed."

So go on send send send... your departmental technical reports, conference and workshop notices and reports, research papers etc.

Happy winter solstice

Jawed Siddiqi
For the period up to December 91 (or any part thereof) the following fees will be charged for membership of BCS-FACS:

- **BCS member**: £8
- **non-BCS member**: £25

The BCS rate will also be available to various overseas organisations with which BCS has reciprocal arrangements (e.g. ACM, GI, AFCET, ACIA, etc.).

For this fee members will receive our FACS-FACfS newsletter and be eligible for a discount rate at FACS conferences and workshops (the discount will vary from event to event but will usually be substantial).

Additionally members are eligible to subscribe at a 70% discount to our new journal "Formal Aspects of Computing". This rate is only available to FACS members. Members may also apply for membership of the EATCS (the European Association for Theoretical Computer Science) at a special discount rate. This membership will run for 12 months and include 3 issues of the EATCS Bulletin.

To obtain either of these discounted services please complete the relevant forms and return together with the appropriate supplementary fee.

---

**BCS-FACS membership application**

Name ...........................................  ‘BCS' membership No. ............................ .
Address ................................................ .

I enclose a cheque for the following Fees

- £8 FACS (BCS Member)
- £25 FACS (non-BCS Member)
- £18 FAC Journal Vol 1 1989*
- £18 FAC Journal Vol 2 1990*
- £20 FAC Journal Vol 3 1991*
- £6 EATCS**

Total: £______________

If a receipt is required please tick box and enclose a stamped self-addressed envelope □.

Cheques should be made payable to BCS-FACS and must be in pounds sterling.

Send to:  Dr D J Cooke
          Department of Computer Studies
          Loughborough University of Technology
          Loughborough
          Leicestershire
          LE11 3TU

*Complete and return the Journal subscription form with your FACS subscription.

**Complete and return the EATCS form with your FACS subscription.
Application for Subscription to FAC Journal

I am a member of the BCS-FACS specialist group and wish to subscribe to the Formal Aspects of Computing journal (4 issues per year) at the special reduced rate.

Please send me:

☐ FAC Journal Volume 1 1989 (£18)
☐ FAC Journal Volume 2 1990 (£18)
☐ FAC Journal Volume 3 1991 (£20)

Name: ........................................................................................................
Address: ...................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................
........................................................................................................

Date: ........................................................................................................
Signed: .....................................................................................................

For FACS use only:

Certified for BCS-FACS

Date: ........................................................................................................
Signed: .....................................................................................................

Received by FACS: Date: .............. Initials: ..............
Sent to Springer: Date: .............. Initials: ..............
Actioned by Springer: Date: .............. Initials: ..............
Application For Membership of EATCS

I am a member of BCS-FACS and -

*I would like to join EATCS
*I would like to renew my membership of EATCS

(* Please delete as appropriate)

I have paid the appropriate fee to BCS-FACS and I understand that this will be forwarded to EATCS in due course.

Name: .......................................................................................... ..
Address: ..................................................................................... .
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
.....................................................................................................
Date: ..................................... Signature: ............................................................ .

For FACS use only:

Certified for BCS-FACS:

Date: ..................................... Signature: ............................................................ .
Report on the BCS-FACS Meeting
"An Introduction to FOREST"

September 25-26, 1989, Imperial College, London

The FOREST Workshop consisted of several talks over two days. The most important of these have been covered by this report. Each section gives a flavour of what individual speakers covered, even though some of them gave more than one presentation.

Introduction

The FOREST (Formal REquirements Specificalion) project is an attempt to combat the problems of imprecisely stated and continually changing requirements of computing systems. It is predominately concerned with requirements capture and specification. The FOREST approach has been developed by an academic/industrial consortium, involving GEC Research, AEA Harwell Laboratory, Imperial College and GEC/Marconi Avionics. Its application is targeted at industrial scale real-time systems which, typically, have the following characteristics:

(i) interaction with external objects,
(ii) real-time requirements and constraints,
(iii) conformance to very high standards of reliability and safety,
(iv) software systems with a long expected lifetime, exceeding several generations of hardware.

The FOREST approach brings formal methods techniques to bear upon the requirements capture and specification with the following advantages:

(i) freedom from semantic ambiguity,
(ii) the ability to reason early on about the consequences of requirements,
(iii) the potential for using software tools in the analysis of specifications.

'Requirements for Requirements Specs' (Tom Maibaum)

This presentation was divided into two distinct halves, the first tackling the assumptions that the FOREST team made when developing their logic, followed by a general description of logic. The second half described the Modal Action Logic (MAL) that the FOREST team have used as their underlying logic for the FOREST specification language.

Three different types of assumptions were identified, technical, methodological and personal. For the technical assumptions the underlying theme was that the nature of the application influences the nature of the formalism. Hence the following characteristics were considered to be important:

(i) the specification language (or logic) has to be capable of dealing with multi-component, asynchronous systems,
(ii) time is an important factor in the above environment,
(iii) there is a need to deal explicitly with causality and sequencing,
(iv) actions/transitions need to be distinguished from functions/relations,
(v) the environment and the system needs to be treated as a single unit for description,
(vi) context of environment is important, instead of pre/post conditions, because transitions are not instantaneous.

The methodological assumptions need to ascertain how the logic is to be used. The following were a few of the major considerations:

(i) the logic needs to support the concepts that engineers usually use,
(ii) the logic needs to facilitate 'loose specifications',
(iii) the logic should be modular so as to suit different classes of application.

The final set of assumptions, the personal assumptions, recognised that 'taste' is an important factor, as it is in any design activity. The FOREST team had the following personal tastes:

(i) the notion of a state should be an implicit data type,
(ii) durations of time intervals and time dependent properties should also be implicit in the logic.

Having described the assumptions that were made within the FOREST framework for their particular logic, logics in general were described. A logic is a pair consisting of the language of the logic, \( L \), and a derivability relation (a relation that can be used to deduce all of the logical consequences of the language), \( \vdash \). i.e. \( <L,\vdash> \). Logics are generally presented in an axiomatic fashion. In order to move to a characterisation of a particular application (i.e. a specification) we need to introduce extralogical symbols into our logic \( L \). Thus a specification is a theory presentation, which is also a pair \( <L_s,\Delta> \), where \( L_s \) is the extralogical language of the theory presentation and \( \Delta \) is the set of axioms of the theory.

The FOREST team uses a logic called M[AL], Modal Action Logic. MAL is built up in a hierarchy, in the following order: many sorted first order logic, agents & actions, deontic operators (indicating what must occur, may occur, may not occur, etc), intervals and combinators. An important part of MAL is the actions and agents level. These take the form of two new sorts, \( A_C \) (actions) & \( A_g \) (agents). Terms of sort \( A_g \) (\( A_C \)) are generated by using symbols (analogous to functions) whose resulting sort is \( A_g \) (\( A_C \)). For example (within a telephone system):

user : telephone_id \( \rightarrow A_g \)

push: natural \( \rightarrow A_c \) (ie. pushing a natural number onto a stack produces something of type action)

Both of the sorts \( A_g \) and \( A_C \) are logical and can therefore be used in all kinds of specification (eg. a stack is a user-defined sort and therefore cannot be used in all specifications).

The sorts \( A_g \) and \( A_C \) can be used in modal formulae. For example if \( A \in A_g \), \( a \in A_C \) and \( \alpha \) is a formula then

\( [A,a]\alpha \)

is a formula which can be read as: if \( A \) does \( a \), then (in the following resulting state) \( \alpha \) holds. An example of this is:

\( [\text{Tom},\text{Close(Account_id)}]\text{not(has_access(Tom,account_id))} \)
A more common use of this type of formula is:

\[ \text{Pre} \rightarrow [A,a]\text{Post} \]

However, we may also have hypothetical situations such as:

\[ [A,a]\alpha \rightarrow \beta \]

To complement these agent/action formulae there exists a set of agent/action axioms, such as

\[ [A,a](\varphi \rightarrow \psi) \rightarrow ([A,a]\varphi \rightarrow [A,a]\psi) \quad \text{Distributivity} \]

\[ [A,a]\neg(\alpha) \rightarrow \neg([A,a]\alpha) \quad \text{But not necessarily the converse.} \]

Built upon the idea of actions and agents is the notion of normative behaviour. Basically if doing an action in a 'good' state should lead to a 'good' state then this action should be permitted, and vice versa. To help characterise this behaviour we introduce two new operators \( \mathcal{P} \text{(ermitted)} \) and \( \mathcal{O} \text{(bliged)} \). These can be used in the following manner:

\[ \mathcal{P}(\text{Agent,Action}), \quad \mathcal{O}(\text{Agent,Action}), \quad \text{Pref}(\text{Agent,Action}). \]

The last of the three uses permits an agent to refrain from an action.

'Requirements Method - Background to SCS' (Stephen Goldsack)

This presentation can basically be divided into two sections. The first was a general discussion of why we need formal methods and what is required of them. The second was a look at the steps involved in SCS (Structured Common Sense), ie. the FOREST approach to requirements capture and specification.

Formal methods bring us freedom from ambiguity and a basis upon which we can reason. Much of this reasoning can be done automatically by software tools, eg. detecting missing parts of specifications, verification, animation. However, formal methods do not particularly help with controlling complexity, with readability or with creating structure. We must keep these problems in mind when developing formal methods as logic on an industrial scale describes a large amount of inter-related data and typically has complexity well beyond one "headfull." Therefore, there is a need for a methodology so that the analyst always knows what his next step is. He is then able to find all of the necessary information and can express it under a formal framework. A method imposes a structure upon the analyst, and gives him the languages to describe his thoughts, whether diagrammatically or textually. These languages should be supported by software tools so that thoughts can be quickly created and modified. Some methods already exist, eg. SSADM, CORE, MASCOT, and on the whole they work well for developing (normal) programs. A method needs to be targeted at the types of system that are to be developed. They need to consider the type of information that is to be handled, whether the system needs to be looked at from many viewpoints, whether there is granularity in the system (ie. is there more than one level) and how we wish to express the system, eg. in a programming language, logically, algebraically.
The FOREST method, SCS, is aimed at real-time systems and consists of the following steps:

(i) identification of the system components (agents),
(ii) identification of the interactions between the agents (dataflows),
(iii) identification of the actions that each dataflow performs for the system, what inputs are required and what outputs are produced and where they come from and go to (action tabulation),
(iv) data analysis - determination of the structure and logical properties of data flows. This can be achieved in a number of ways, eg. structure analysis (Jackson), ERA modelling,
(v) permission and obligation analysis - the determination of what post conditions imply other permissible or obliged actions. This is basically why something happens,
(vi) temporal analysis - when does a particular action occur?
(vii) generation of the specification. Having completed all of the preceding steps the specification of the system should be an easier process.

One may wish to have other views of the system. These can also be incorporated into the SCS steps. For example, one may wish to use decision tables, state machines or Petri Nets.

Having outlined the SCS steps a library system was used as an example to illustrate them. As the copies of the slides are fairly self-explanatory I will not discuss them in great detail, if at all.

Consider the diagram labelled step (ii). Here we find that the agents have been narrowed down from the previous diagram to just the library and the borrowers. This was done to simplify the problem so that each step could be covered quickly. As can be seen, there are already five dataflows between the two agents. Any more agents would have just complicated the example.
In the next three diagrams (step (iii)) we have the list of actions that can be performed in the system. Where there is more than one possible output dataflow for a given action, it is represented by an arrow for each possibility. The dashed arrows indicate side effects or supplementary actions.

**Action Tabulation**  
*STEP (iii)*

<table>
<thead>
<tr>
<th>S</th>
<th>I</th>
<th>A</th>
<th>O</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source → Input → Action → Output → Destination</td>
<td>Agent Dataflow</td>
<td>Dataflow</td>
<td>Agent</td>
<td></td>
</tr>
</tbody>
</table>

**Library Action Table**

<table>
<thead>
<tr>
<th>S</th>
<th>I</th>
<th>A</th>
<th>O</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Borrower → request → issue → book → borrower</td>
<td>transaction → library record</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>library → transaction → adjust records</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Borrower → collection → issue → book → borrower</td>
<td>transaction → library record</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Borrower → returned → check-in → note → borrower</td>
<td>transaction → library record</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Borrower Action Table:

<table>
<thead>
<tr>
<th>S</th>
<th>I</th>
<th>A</th>
<th>O</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>request</td>
<td>request</td>
<td>library</td>
<td></td>
</tr>
<tr>
<td></td>
<td>library</td>
<td>book</td>
<td>returned book</td>
<td></td>
</tr>
<tr>
<td></td>
<td>library</td>
<td>notification</td>
<td>collection</td>
<td>library</td>
</tr>
</tbody>
</table>

FOREST

Step (iv) shows the entity relationship model of the library system.

![Entity Relationship Model](image)

Figure 1. Entity Relationship Attribute Model of the Library

Now consider the diagram marked step (v). There are two actions at the bottom of this table. The boxes in the corresponding rows indicate whether an action is permissible, not permissible or obliged, when certain conditions hold. For example, when

books_borrowed(u) < limit(u)
and available(b)
and not(status(u, b) = none)
and status(u, b) = requested
then the user is not permitted to put in a request for b but the library is obliged to issue the book.

where u is a user
b is a book
F on the table
**STEP (u)**

Step (vi) (permission and obligation analysis) has been omitted but a state diagram has been included to illustrate how an alternative diagram can illustrate the various actions (and what the sequence of actions is).

State Diagram for User/book Pair

The next diagrams give some examples of library axioms expressed in the FOREST specification language. The first axiom states that it is permitted for u to request book b if u has borrowed less than his limit and u has not had any previous requests or has not already been issued with b (status(u,b) = none) and vice versa. The second axiom states that if the agent u (ie person u) requests book b then the resulting state is that the u has requested book b and the library is obliged to issue the book to u.

**Some Library Axioms**

\[
\text{per } (u,\text{request}(b)) \leftrightarrow (\text{books borrowed}(u) < \text{limit}(u)) \\
\land \text{status}(u,b) = \text{none}
\]

\[
[u,\text{request}(b)] (\text{status}(u,b) = \text{requested} \land \text{obl (library, issue}(u,b)))
\]

Protect against multiple obligations:

\[
\exists S.\text{OBL}(\text{library}, S) \rightarrow [u,\text{request}(b)] \\
(\text{status}(u, b) = \text{requested} \land \exists S'.\text{OBL}(\text{library}, S') \land S' = S \land \text{obl (library, issue}(u,b))]
\]

FOREST
Some Library Axioms

The effects of issuing a book are defined by the axiom:
\[
\text{available}(b) \land \text{books borrowed}(u) = N \rightarrow \\
\text{library}, \text{issue}(u,b) \\
\text{status}(u,b) = \text{borrowed} \land \text{books borrowed}(u) = N + 1
\]
\[
today = d \land \text{available}(b) \rightarrow \\
\text{library}, \text{issue}(u,b) \\
\text{obl}(u, \text{return}(b), today < d + 14)
\]

'Tools: Explanation and Demonstration' (Bill Quirk)

This presentation described the tools that the FOREST team have designed to support the FOREST approach to system specification. Firstly it posed the question of why we need tools at all. Tools are useful because they can remove the drudgery of having to rewrite/draw particular system characterisations when parts are changed, they can do automatic checking of syntax, data typing, etc and they can maintain format consistency. Tools must also be able to make formal methods palatable to the engineer. The FOREST set of tools operate on Sun Workstations. The tool set contains the following diagramatic editors to aid the SCS steps: Agent Identification (AGH), Dataflow Diagram (DFD), Action Tabulation (SIAOD), ERA Analysis (ERA) and Causal Analysis (CANE). All of these editors provide some automated support, ie there is still need for some user involvement. They all access the same database and are therefore able to automatically generate parts of new SCS steps automatically from information that already exists in the database as a result of using other SCS editors.

The tools also have validation facilities. These come in the form of an animator, a theorem prover and a MAL translator. The theorem prover uses the tableau method of theorem proving. An example of the tableau method is as follows:

From: human(socrates) and
for all X: human(X) implies mortal(X)

Prove: mortal(socrates)

Negate the proposition: not mortal(socrates)

Rewrite human(X) implies mortal(X) as
mortal(X) or not human(X)

combine with mortal(socrates)

mortal(socrates) or not human(socrates)

mortal(socrates)

contradicts negation of proposition
given
not human(socrates)

contradicts human(socrates) -

Both of these arms derive a contradiction and thus the theorem is proved.
As well as being able to prove propositions about a particular system, safety properties can often be proven, and in particular a subset of these properties can be proven automatically.

A user may wish to animate his/her specification. This gives the user a further checking mechanism which can demonstrate that the specification exhibits the behaviour that is expected by the specifier.

Specifications are animated by first translating them into Prolog and then running test scenarios with user-graphic interfaces. The graphic interfaces are problem dependent and thus the tools do not produce automatic representations of them. This tool is one of the most interesting and perhaps would be one of the most useful for the engineer as he/she would be able to see his/her specification at work.

Consider the next four diagrams. These are illustrations of the FOREST tools. The first diagram shows SCS step (ii), a dataflow diagram (DFD) for a heating system. The second shows the corresponding Action Tabulation. The third diagram shows the animation of a lift specification and finally the fourth shows the animation of a railway specification. Notice that both of the last two have different graphical interfaces which reflect the nature of the application.

'Verification and Validation' (Jim Cunningham)

This presentation explored the tableau method of theorem proving in more detail than the last presentation. More specifically the FOREST system uses a MAL Tableau, i.e. a tableau method tailored to MAL. One of the basic observations about a tableau method is that although the 'state' space may be large (and infinite in the first order case), an unsatisfiable theory has no model, so its tableau will close (finitely).

The following is a set of propositional tableau rules:

**Rule A:**

\[
\begin{align*}
\neg \neg a & \quad a \land b \quad \neg (a \land b) \quad \neg (a \Rightarrow b) \\
\hline
a & a & \neg a & a \\
b & \neg b & \neg b
\end{align*}
\]

Where there is more than one result these indicate both are on the same branch of the tableau.

**Rule B:**

\[
\begin{align*}
a \lor b & \quad \neg (a \land b) \quad a \Rightarrow b \\
\hline
\neg a \lor \neg b & \neg a \lor \neg b \neg a \lor b
\end{align*}
\]

The vertical bar represents the opening of two parallel branches of the tableau.

**Rule C:**

\[
\begin{align*}
\forall x a & \quad \neg \exists a \\
a (x/c) & \quad \neg a (x/c) \\
\forall x a & \quad \neg \exists a
\end{align*}
\]

where \( c \) is any constant.
Top : down

Second: up

First : down

Ground: up

Satisfied axiom(s) : 6 7 8 9 10 11 12 13
Meta rules checked.
lift is obliged to do action move_from(ground)

FINISH [move_from(ground), lift1]

Satisfied axiom(s) : clock 25 27 28 29
Satisfied axiom(s) : 6 7 9 10 11 12 13
Meta rules checked.
lift is obliged to do action move_from(floor1)

Obliged Sequence : (set_signal(s10,stop), interlock)

FINISH [set_signal(s10,stop), interlock]

Obliged Sequence : (set_signal(s10,stop), interlock)

Obliged Sequence : (set_route(s10, stop), interlock)
Rule D:

\[ \exists a \quad \neg \forall x \ a \]

\[ a (x/d) \quad \neg a (x/d) \]

where d is a constant new to the branch.

Added to these first order rules we have some MAL rules.

Modality introduces two new rules. The basic idea is that for actions (or non-actions) we open a new tableau, and if the new tableau is closed then so is the original one.

Rule E:

\[ [Ag,Ac] \alpha \]

Rule F:

\[ \neg [Ag,Ac] \alpha \]

\[ \tau (\delta, \alpha) \quad \tau (\delta', \neg \alpha) \]

where \( \tau (\delta, \alpha) \) asserts that a new (or previously generated) tableau \( \delta \) has \( \alpha \) as a root formula. Similarly, \( \tau (\delta', \neg \alpha) \) asserts that a new tableau \( \delta' \) has \( \neg \alpha \) as a root formula.

The six rules are the basic tableau rules but to make them implementable the following changes are necessary.

1. Unification is introduced by using a variable in rule C instead of a constant. This variable can then be unified with a pre-existing constant in order to close a branch.

2. Rule D is avoided by skolemisation.

3. A linear, depth-first strategy is introduced.

Many other less significant changes also have to be made to the rules so that they are efficiently implemented.

The meeting ended with a short presentation about the commercial aspirations of the team. Basically a spin-off company will market the FOREST tools.

References

Costa & Cunningham Logical Animation, Department of Computing, Imperial College, London SW7 2BZ.

Martin Start Loughborough University
Dame Rumour has been up to her tricks again. The unusually (for him) taciturn Dr. F. X. Reid, has been 'sighted' all over the world, as though he were some kind of flying saucer. Indeed, in some quarters (never mind which) he is held responsible for the Wiltshire corn circles. One correspondent (R. R. Raskolnikov, GCHQ) claims that Reid is working on a Thatcherite deontic logic, in which Economics is combined with Truth. Another (A. Einstein, Brighton Poly) has told me that Reid is applying Quantum Mechanics to resolve the Country's inflation problems. The idea is that if a coin is small enough then the Uncertainty Principle applies, so that if you try to spend the coin, you lose it. This reduces the money supply. My correspondent suggests that the new 5 pence piece is a prototype for this 'quantum money'. (Nice try, Albert).

Mrs. E. W. D. Schlock (Anabaptist College, Monrovia) is convinced that Reid ('A hairy, malodorous introvert, whose presence in the establishment is contrary to the Laws of God and the Statutes of the College') is applying Chaos Theory or Fractal Theory ('whichever is in fashion at the time') to Programming Metrics. Thank you for the pamphlets, Mrs. Schlock, but your 1* is in the Proslogion and your 2* is in the Monadology and Leibnitz's argument is even sillier than St. Anselm's.

Dr. F. X. Lurk (Open U., Beijingo) tells me that Reid is working for the International Marxist Conspiracy, but we've known this for years.

Professor Lampost (somewhere in Patagonia) claims to have photographed Reid in a post office in Ulan Bator, but this claim, it emerged, was merely a pretext for showing me his holiday slides. ('I know it's in here someplace - say, there's the front of the British embassy - kind of run-down, I guess'). P. McCartney (Mull of Kintyre) believes that Reid has really been dead for the past ten years. (No such luck, Paul). Prof. Dr. F. X. de Roeever (whom God forbid) of Utrecht has not expressed an opinion. A. B. See (Marin Co. Retreat for the Uncool) sees it all in terms of 'energy centres, right?' and insists that Reid is an avatar of the Sun God1 and that he is working for the US DoD.2 Mr. D. Shtoompf (Heidelberg, Pa.) claims that Reid is his external examiner and that in many a drunken conversation Reid has violently asserted that his mission in life is (a) to rid Theoretical Computer Science of Category Theory once and for all (b) to get those b*st*rd*s at *mp*r**l (c) to play the Mozart A minor piano sonata right through without any mistakes. Mr. Shtoompf has sent me a 230 page document that I may read sometime. Dr. M. Z. Badhandat-Sxcrabble (Gdansk) says that Dr. Reid has been 'very fair' to her and is a nice old gentleman, if a little eccentric. But - 'hide the Port'! Professor C. A. R. Trollope (Hamburg) asserts that Reid's existence is irrelevant to the functionality of parallel programs and is therefore not the

1 Whatever that means.
2 That's probably what it means.
concern of Concurrency Theory. 'Why needlessly multiply hypotheses?' (Keep it up, Prof). Ms. M. H. Roberts writes, 'Dr. Reid is one of my lodgers. Please, please tell me how I can get rid of him!' The Dowager Countess of Lucan pleads, 'Give yourself up, son!!' The Soviet Miners Union is asking for their money back. Field Marshal S. Hobson-Jobson (Baghdad) tells us that Reid is busy designing large-scale digital range-finders for oil pipelines. (my italics). Mr. N. Debbitt (Dransylvania) says that everything will go swimmingly once Reid is evicted from his council house.

Surprisingly enough, none of my correspondents have accused Reid of working for the EEC Commission (that well-known video game). Those who know the old boy will deduce from this that he is in Brussels getting into some serious data-base hacking.

The truth is simpler and more depressing.3

On the run from the police of several continents, Reid was obliged to apply for a lectureship in an obscure University situated in one of those parts of southern England where the hoi poloi all talk like dishonest taxi drivers. Unaware of the identity and international standing of their new employee, the department in question decided that he was incapable of anything other than administration.4 As a result, the renowned author of the Redundant Sock Theorem has spent the whole of this past year acting as an Examinations Officer for every course that the department could legitimately impose on him, as well as several that they could not. The results were predictable. Obliged to interleave the work of at least five people, Reid suffered a serious stack overflow and the University is now employing an expensive firm of Chartered Accountants to recalculate the results, which serves it right. Unfortunately, the whole business has come to obsess him. When I rang him up the other day, his first words were 'I suppose you're after the diagrams for paper FA 666'. It took me at least five minutes to convince him that all I wanted was some information on his unpublished theorems about quasi-compact ω-closed hyper-upper-lower semi-topoi (the so-called 'Reid objects'). Reid mumbled something pejorative about Per Martin-Lof and rang off.

Reid is desperate to resign, but unfortunately the University regulations state that (a) An Examinations Officer may only be relieved of his task if he shows signs of insanity and (b) No one with all his marbles would want to continue as an Examinations Officer. At this rate he will be flying missions for some time.

In recognition of Reid's extraordinary contribution to Theoretical Computer Science, FACS is setting up a fund and organising an Escape Committee. Send money (no five pence pieces, please) and any digging equipment, forged papers, vaulting horses, civilian clothes etc. to:-

THE F. X. REID APPEAL
FACS FACTS.

Thank you.

3 Depending on your point of view, of course.
4 Later, they discovered that he was no good at that either.
A Reification Calculus for Model-Oriented Software Specification
J. N. Oliveira

Branching versus Linear Yet Again
José Carmo and Amilcar Semadas

Command Algebras, Recursion and Program Transformation
Wim H. Hesselink

A Fast Pattern Matching Algorithm Derived by Transformational and Assertional Reasoning
H. A. Partsch and F. A. Stomp

Partial Order Behaviour and Structure of Petri Nets
Eike Best and Jörg Desel

Axioms and Models of Linear Logic
Wim H. Hesselink

Equational Reasoning About Nondeterministic Processes
Jayadev Misra

Chain Properties of Rule Closures
Miki Hermann

Integrating Predicate Transition Nets with First Order Temporal Logic in the Specification and Verification of Concurrent Systems
Xudong He and John A. N. Lee

Refinement Concepts Formalised in Higher Order Logic
R. J. R. Back and J. Wright

The Projection of Systolic Programs
C. Lengauer and J. W. Sanders

Defining, Analysing and Implementing Communication Protocols using Attribute Grammars
N P Chapman

On the Interconnection of Hopfield Nets
N Soparker and A Silberschatz

Proof Obligations for Blocks and Procedures
J A Ah-Kee

Modelling Multiple Inheritance with Colimits
H Lin and Pong, M. C

A Brief Note on the Halting Problem
Mike Stannett
M. Z. Kwiatkowska, M. W. Shields, R. M. Thomas

Semantics for Concurrency

Proceedings of the International
BCS-FACS Workshop,
Sponsored by Logic for IT (S.E.R.C.),
23–25 July 1990, University of Leicester, UK

Published in collaboration with the
British Computer Society

Springer-Verlag
London Berlin Heidelberg New York
Paris Tokyo Hong Kong
Contents

A General Tableau Technique for Verifying Temporal Properties of Concurrent Programs (Extended Abstract) C. Stirling and D. Walker ........................................... 1
Traps, Free Choice and Home States (Extended Abstract) E. Best, L. Cherkasova, J. Desel and J. Esparza ...................... 16
A Denotational Semantics for Synchronous and Asynchronous Behavior with Multiform Time (Extended Abstract) M. Roncken and R. Gerth ........................................................... 21
From Failure to Success: Comparing a Denotational and a Declarative Semantics for Horn Clause Logic F. S. de Boer, J. N. Kok, C. Palamidessi and J. J. M. M. Rutten ................................................................. 38
Negations of Transactions and Their Use in the Specification of Dynamic and Deontic Integrity Constraints F. P. M. Dignum and J.-J. Ch. Meyer .................................. 61
Experimenting with Process Equivalence B. Bloom and A. R. Meyer ............................................................... 81
Iteration Theories of Synchronization Trees S. L. Bloom, Z. Esik and D. Taubner .............................................. 96
Towards a Theory of Parallel Algorithms on Concrete Data Structures S. D. Brookes and S. Geva ........................................... 116
Causal Automata I: Confluence = (AND, OR) Causality J. Gunawardena ................................................................. 137
A Simple Generalization of Kahn's Principle to Indeterminate Dataflow Networks (Extended Abstract) E. W. Stark ................................................................. 157
Defining Fair Merge as a Colimit: Towards a Fixed-Point Theory for Indeterminate Dataflow D. B. Benson, P. Panangaden and J. R. Russell ...................... 175
High-Level Nets for Dynamic Dining Philosophers Systems G. Findlow and J. Billington ............................................. 185
C. Rattray (Ed.)

Specification and Verification of Concurrent Systems

Published in collaboration with the British Computer Society

Springer-Verlag
London Berlin Heidelberg New York
Paris Tokyo Hong Kong
POSTER SESSIONS

1. Modelling of Distributed Problem Solving using Logic
   Modified Petri Nets
   A. Di Stefano, F. Gibilisco and O. Mirabella .................. 557

2. An Animator for CSP Implemented in HOPE
   L. D. Natanson and W. B. Samson ................................. 575

3. A Concurrent Approach to the Towers of Hanoi
   W. D. Crowe and P. E. D. Strain-Clark ......................... 595

Author Index .................................................. 611

Contents

1. The Interplay of Theory and Practice in a Parallel Object-Oriented Language
   P. America ......................................................... 1

2. Object-Oriented Process Specification
   S. A. Schuman, D. H. Pitt and P. J. Byers ..................... 21

3. Formal Object Oriented Specification of Distributed Systems
   E. Cusack ......................................................... 71

4. The Design and Development of Ada Real-Time Embedded Systems
   R. G. Clark ....................................................... 84

5. Protocol Analysis and Implementation using NPNs and SDL
   K. R. Parker, R. A. Berger and K. E. Cheng ................... 100

6. A Tool for the Performance Analysis of Concurrent Systems
   V. Caragli, A. Faro, M. Malgeri ................................ 121

7. Winston: A Tool for Hierarchical Design and Simulation of Concurrent Systems
   J. Malhotra, R. Shapiro, S. A. Smolka and A. Giacalone ... 140

   D. Roantree and M. Clint ........................................ 153

9. Dynamic Communication Links
   C. M. Holt ....................................................... 184

10. Formal Environment and Tools Description for the Analysis of Real Time Concurrent Systems
    V. S. Alagar and G. Ramanathan ................................ 196

11. An Equivalence Decision Problem in Systolic Array Verification
    P. Abdul and S. Amborg ......................................... 236

12. Should Concurrency be Specified?
    R. L. Ibrahim, J. A. Ogden and S. A. Williams ............... 246

    (extended abstract only)
    M. Joseph and A. Goswami ..................................... 272
The ERIL Project at Glasgow

Muffy Thomas
Department of Computing Science
Glasgow University

The ERIL project at Glasgow is a SERC (IED) funded research project, in collaboration with Royal Holloway and Bedford New College (RHBNC), London University, and Rutherford Appleton Laboratory (RAL). British Telecom/Research and Technology is an industrial "uncle" to the project. The project has been going for nearly one year and will run for another two years.

The overall aim of the project is to investigate the verification requirements of LOTOS specifications and to determine the applicability of equational reasoning and term rewriting to discharging these requirements. In this overview of the project, we explain the background and history of the project and give a brief description of the current research work at Glasgow. We conclude with an extended abstract of a paper on solving divergence in Knuth-Bendix completion.

History

The project is the result of earlier collaboration between Ursula Martin at RHBNC, Jeremy Dick at RAL, and Muffy Thomas at Stirling University (now at Glasgow). Ursula Martin was already leading a SERC project in the area of equational reasoning and term rewriting at RHBNC, Jeremy Dick was working on the ERIL order-sorted equational reasoning system at RAL, and Muffy Thomas was working on the implementation of algebraic specifications and was using ERIL for prototyping and theorem proving. Unfortunately, Jeremy left RAL, but the ERIL work continued under the guidance of David Duce. In October 1988 the "Verification Techniques for LOTOS Specifications" project was proposed with Professor Martin overall project leader; the project began in October 1989. Jeremy still keeps in touch with the group and we informally keep the "ERIL" name.

People

Carron Kirkwood began working as a Research Assistant on the project at Glasgow in October '89, after graduating from Glasgow University. Her interests include specifying and verifying concurrent systems and she has been experimenting with various rewrite systems such as LP [GC89] and RRL [KS84].

Phil Watson came to Glasgow as a Research Assistant in January '90. After completing a Ph.D. in recursion theory at Leeds, he worked with Ursula at RHBNC on various aspects of term rewriting and equational reasoning before being enticed to the north. His current interests include order-sorted rewriting, representations of place arithmetic for term rewriting, and solving divergence in Knuth-Bendix completion.

Muffy Thomas is a lecturer at Glasgow University and became interested in term rewriting for proving theorems about algebraic specifications in her Ph.D. work [Th87]. Her current interests include the development of LOTOS and specification languages, the translation of ASN.1 (an ISO data type language) into ACT ONE, inductive inference, and solving divergence in Knuth-Bendix completion.
Brian Matthews, who completed an M.Sc. under Jeremy Dick's supervision and has been working on a new implementation of ERIL at RAL [MR90], will move to Glasgow in November '90 to begin studies for a Ph.D. He will be investigating the implementation of LOTOS specifications and the verification issues involved.

Alastair Reid is a Ph.D. student who has been working at Glasgow for two years on the topic of implementing algebraic specifications [Re90].

Background: Equational Reasoning and Term Rewriting

Equational reasoning is the process of deriving the consequences of a given system of equations. The simplest way to produce an equational proof that two terms, or expressions are equal is to keep rewriting subexpressions of one, using the equations, until it is transformed into the other. The process is more efficient if the equations are considered as rewrite rules (rules which represent directed equality). This paradigm is very similar to functional programming; however, in general, rewriting is nondeterministic in the sense that no restrictions are placed on the selection of rules to be applied or on the selection of the subexpression to be rewritten. Moreover, there is no restriction on overlapping rules.

This generality makes rewriting a very powerful computational paradigm. Our motivation for studying equational reasoning and term rewriting techniques is that if a program or hardware device is described by an equational or algebraic specification, then automated theorem proving systems based on term rewriting can help in showing that it meets its specification. More generally, current uses of rewriting include

- automatic theorem proving of equational and inductive theorems
- solving word problems in universal algebra
- generation of solutions to equations (narrowing)
- prototyping of equational specifications
- synthesis of rewrite rules (which may be regarded as programs, or implementations of more abstract specifications)
- proving properties of specifications such as completeness and consistency

Confluence, termination, and typing are three crucial issues in term rewriting. The confluence property ensures that the order of application of rewrite rules is irrelevant, whereas the termination property ensures that all sequences of rewrites are well-founded (there are no infinite sequences). When a set of rewrite rules is confluent and terminating, then each expression, or term, has a unique normal form: an expression which cannot be rewritten. A set of rewrite rules which is confluent and terminating is called complete, and makes equality between expressions decidable since repeated application of the rules reduces any expression to a unique normal form; two terms are equal if and only if they have the same normal forms. The Knuth-Bendix completion algorithm [KB70], given a termination ordering, tests for the confluence property. The algorithm not only tests for confluence, but it also generates a confluent set of rules. It is called a "completion" algorithm because if it terminates (it is actually a semi-algorithm), it generates a complete set of rules which can then be used as a decision procedure for equality.

Background: LOTOS and Program Specification

LOTOS (Language of Temporal Ordering Specification) is based on the concept of specifying a system in terms of observable behaviour. In LOTOS, events are used to denote
the occurrence of something which the specification makes assertions about; one can 
express ordering constraints on events and through their structure, the communication and 
change of information within a system. LOTOS was developed within ISO (International 
Standards Organisation) and has been used for developing formal descriptions of OSI 
(Open Systems Interconnection) standards.

There are two separate components of LOTOS; the first is concerned with processes and 
the second with data.

The process part of the language (basic LOTOS) is based on the process algebra, and 
associated methods, first introduced by Robin Milner in CCS [Mi 80]. The semantics of the 
LOTOS process calculus is based on CCS and comprise a set of inference rules on the 
labelled transitions of processes. The operators for process composition are, however, 
derived from Tony Hoare's work on CSP [Ho 85]. Verification work in the process calculus 
has been concerned largely with proving various classes of equivalence between labelled 
transition systems; for example, observational, bisimulation, and testing equivalence.

The data part of the language is based on the language ACT ONE, a data type 
specification language which uses equational specification with initial algebra 
semantics. Thus, ACT ONE can be given an operational semantics using term rewriting 
techniques.

Since both the process equivalences and data type specifications can be expressed 
equationally, there is much scope for using rewriting techniques for verifying LOTOS 
specifications. It is important to note though that whilst the process part of the language 
is fixed, every LOTOS specification may contain an arbitrary set of data types.

Current Research Areas

Listed below are descriptions of some research areas, recent results and current work.

Order-Sorted Rewriting

Order-sorted algebra and order-sorted rewriting [SNMG87] extends the expressive power 
of equational specifications, particularly for handling partial functions and errors. This 
approach too has its restrictions: the typing system is too syntactic and static. Moreover, 
rewrite rules must be sort preserving or decreasing. In [WD89], Jeremy Dick and Phil 
Watson show how this restriction is overcome by introducing the notion of dynamic, or 
semantic sorts.

An Application of Rewriting Systems: Efficient Representation of Arithmetic

Most specifications encountered in practice include the natural numbers (or integers) and 
the arithmetic operations. The usual style of specifying numbers by a generator 0, a 
successor and predecessor function leads to unwieldy and unnatural expressions of great 
length. In [CW90], Phil Watson and Dave Cohen (at RHBNC) investigate a set of rewrite 
rules for arithmetic using the more convenient place valued representation, i.e. the 
number 31 is represented by the term "3.1", where _ _ is a binary infix operator, instead 
of thirty one applications of the successor operation to 0.

Solving Divergence of Knuth-Bendix Completion

The Knuth-Bendix completion algorithm is not guaranteed to terminate, even when the 
word problem defined by the given system of equations is decidable. When the algorithm 
diverges, an infinite sequence of rules is generated. In [TJ89], Muffy Thomas and Klaus 
Jantke (from Leipzig Technical University) use inductive inference techniques to 
synthesise a finite generalisation of the infinite sequence which is equivalent in some
sense. However, for a given signature, generalisations cannot always be found and the signature must be enriched in order to generalise. In [TW90], Muffy Thomas and Phil Watson give an algorithm which synthesises a finite generalisation of the infinite sequence of rules by enriching the given signature with new sorts, sort relations, and operator arities. An extended abstract of this paper follows.

Concurrency and Rewriting

In [KN90], Kathy Norrie (at RHBNC) and Carron Kirkwood conduct some experiments using the LP and RRL rewriting systems for proving properties of CSP and (basic) LOTOS specifications. Carron found some very interesting examples of infinite sequences of confluent rewrite rules when she attempted to complete various sets of rules for LOTOS weak bisimulation congruence and these will be investigated further.

The project has been considering a case study of three communicating processes at OSI Network level as a source of typical specification and verification problems. Carron Kirkwood has written a LOTOS specification and is currently investigating the problems of representing and proving the verification requirements.

ACT ONE and Persistency

Two of the ACT ONE verification requirements of a new specification with respect to an existing specification, completeness (there are no new objects), and consistency (there are no fewer objects), have been formulated as requirements on the associated term rewriting systems [Pa 85]. Together, these requirements are called persistency. However, they need to be reformulated within the context of ACT ONE and made more accessible through the development of proof procedures and tools. Phil Watson has carried out some preliminary work in this area and there are plans to study, reformulate and implement these procedures in the new ERIL system.

There are some reservations about the use of the data type language ACT ONE within LOTOS. In [Th89], Muffy Thomas has defined a translation from ASN.1 data types to ACT ONE data types and there are plans to continue to study the alternatives and enhancements to the specification of data types within LOTOS.

Acknowledgements

The members, past and present, of the ERIL project are thanked for their contributions.
SOLVING DIVERGENCE IN KNUTH-BENDIX COMPLETION BY

ENRICHING SIGNATURES

Muffy Thomas and Phil Watson
Dept. of Computing Science
University of Glasgow

Extended Abstract

1. Introduction

The Knuth-Bendix completion algorithm [KB70] is not guaranteed to terminate, even when the word problem defined by the given system of equations is decidable. When the completion (semi) algorithm diverges and results in an infinite sequence of confluent rewrite rules, then we only have a semi-decision procedure for the word problem.

We aim to replace such an infinite sequence of rules with a finite sequence, or set, which is equivalent in the following sense. First, the finite set should preserve the equational theory defined by the given equations, i.e. the finite set should at least be a conservative extension of the infinite sequence. Note, however, that the finite set may be based on a larger signature than the infinite sequence; the rules in the former may use some sorts which do not occur in the latter. Second, the finite set should be canonical, i.e. confluent and terminating. Our approach is based on finding exact generalisations [T]89 of the varying parts of the infinite sequence of rules. Often, exact generalisations cannot be found with respect to the given signature, but they may exist if we enrich the signature. In [La89], the signature is enriched with new operators; here we enrich the signature with new sorts, sort relations and operator arities: the result is an order-sorted signature. The new sorts allow us to capture exactly the varying parts of the rules, in some cases.

In this extended abstract we present an algorithm for synthesising the required new sorts; a more complete approach to finding generalisations of infinite sequences of rules is contained in [TW90]. In §2 we present an example of the kind of problem to be solved. §3 contains some background material and definitions concerning languages and grammars. §4 contains our algorithm which takes as input a signature and a context-free grammar G (with start symbol S) describing the language of the varying parts of the infinite sequence of canonical rewrite rules. It produces an order-sorted signature with a distinguished sort S such that terms with sort S are exactly those words in the language of G. In §5 we apply the algorithm to the example presented in §2.

We assume the usual definitions of order-sorted matching and term rewriting as contained in [SNGM87], for example. We briefly review the concepts of monotonicity and regularity from [SNGM87] below.

An order sorted signature Σ is monotonic iff for every pair of operator functionalities f:s₁ x ... x sₙ → s, f:t₁ x ... x tₙ → t, if for all i=1,...,n, sᵢ ≤ tᵢ, then s ≤ t.

An order sorted signature Σ is regular iff every term has a least sort. We use the notation LS(t) to denote the least sort of a term t, when it exists.
2. Motivation

Consider the set of rules generated by application of the Knuth-Bendix completion algorithm to the rule:

\[ R \) \quad f(g(f(x))) \rightarrow g(f(x)) \]

where we assume a single-sorted signature with no operators apart from \( g: T \rightarrow T \), \( f: T \rightarrow T \) and a constant symbol \( c: T \).

Then the complete set of rules generated is the infinite sequence:

\[
\begin{align*}
R1) \quad & f(g(f(x))) \rightarrow g(f(x)) \\
R2) \quad & f(g(g(f(x)))) \rightarrow g(g(f(x))) \\
R3) \quad & f(g(g(g(f(x))))) \rightarrow g(g(g(f(x)))) \\
& \text{etc.}
\end{align*}
\]

We use \( R^\infty \) to denote this infinite sequence.

It can easily be seen that the rules in \( R^\infty \) fall into a clear pattern:

\[ f(g^n(f(x))) \rightarrow g^n(f(x)) \quad \text{for any } n > 0. \]

In fact, we might observe that all terms of the form

\[ t = g^n(f(x)) \quad \text{for any } n > 0. \]

are qualitatively different from all others; these are exactly the terms for which

\[ f(t) \rightarrow t. \]

Note that we cannot generalise \( R^\infty \) by the rule

\[ f(y) \rightarrow y \]

where \( y \) is a variable of sort \( S \). Such a rule is too powerful - it equates terms which have different normal forms under \( R^\infty \). If we add such a rule, then the new rule set is not a conservative extension of the original.

If we were able to define a variable \( y \) which could only be instantiated by terms of the form \( g^n(f(x)) \), \( n > 0 \), then we would be able to replace the infinite sequence by the single rule \( f(y) \rightarrow y \). The aim of our algorithm is to define a new sort which contains exactly those terms of the form \( g^n(f(x)) \), \( n > 0 \), and to modify the arities of the operators appropriately.

3. Languages and Grammars

In §1 we informally introduced the idea of regarding an infinite set of rewrite rules as a language; we give the relevant language concepts below.

Definition [HU79]

A context-free language is a set of words (finite strings of symbols from our alphabet) each of which is derived from a context-free grammar \( G = (V,C,S,P) \) where:

- \( V \) is a finite set of variables or non-terminals
- \( C \) is a finite set of constants or terminals (our alphabet)
- \( S \) is a special non-terminal called the start symbol
- \( P \) is a set of productions each of which is of the form:
  \[ A \rightarrow s \]

where \( s \) is a string of symbols from \( (V \cup C)^* \) and \( A \) is in \( V \).
Since we are concerned with sets of terms, we assume that the symbols "(" and ")" are terminals in every grammar. Moreover, since terminals will be either operator symbols from \( \Sigma \), brackets, or variables in \( \Sigma \), we refer to the constant operator symbols as constant-terminals and the variables as variable-terminals. We are interested in a special case of context-free grammar as follows.

Definition
A grammar \( G \) is weakly simple iff every production rule in \( G \) has one of the forms:

\[
N \rightarrow f(x_1, \ldots, x_n)
\]

or

\[
N \rightarrow f
\]

or

\[
N \rightarrow N'
\]

where each \( x_i \) is a terminal or non-terminal, \( f \) is a terminal, \( N \) and \( N' \) are non-terminals.

Lemma
Any context-free grammar can be transformed into an equivalent weakly simple grammar.

4. The Algorithm
We will now define our algorithm as follows.

From a given rewriting system \( R \) over signature \( \Sigma \), let \( R^\infty \) be the infinite sequence of rules generated by the Knuth-Bendix completion algorithm. We partition \( R^\infty \) into \( Q \) and \( Q^\infty \), where \( Q^\infty \) is the infinite sequence we wish to generalise; i.e. we aim to replace \( Q^\infty \) by a finite set of rules. Note, in general, \( R^\infty \) may contain more than one sequence to be generalised, in which case the algorithm may be applied in turn to each sequence.

Provided the language of the varying parts of \( Q^\infty \) can be described by a context-free grammar, then we apply the following algorithm which enriches the signature in the appropriate way, so that \( Q^\infty \) can be generalised. We do not concern ourselves here with the generation of the grammar from the finite subset of the infinite sequence \( Q^\infty \) which we can see up to any one time, but we proceed by inspection.

Algorithm
Let \( G \) be a weakly simple context-free grammar with terminals \( C \), non-terminals \( V \) and start symbol \( S \) such that

- \( G \) generates the language of the varying parts of \( Q^\infty \),
- there is a sort \( X \) in \( \Sigma \) such that every term in \( L(G) \) has sort \( X \) and \( X \) is minimal among such sorts.

Let \( Z = (\{X, S\}, \{(S,X),()\}) \) be a triple consisting of sorts, a relation \( < \) on sorts and operator arities. By an abuse of notation, we identify \( < \) with its transitive closure. Note, \( Z \) may only be a fragment of a signature. We now proceed to enrich \( Z \) and combine it with \( \Sigma \) as follows:

Step 1 (add sorts)
For every non-terminal \( N \) in \( V \), add the sort \( N \) to \( Z \) (non-terminals are sorts).

For every constant-terminal \( t \) in \( T \), if \( t \) occurs as an operand in the right hand side of a rule then define a new sort, \( t \), say. Add sort \( t \) and operator \( t : t \) to \( Z \). If \( t \) is a term of sort \( U \) in \( \Sigma \), then add the pair \( (t, U) \) to \( < \) in \( Z \), i.e. order \( t < U \).
Define the partial function sort: \( V \cup T \rightarrow \text{Sorts of } Z \cup \text{VarSorts} \), where VarSorts is the set of sorts of the variable terminals in \( G \), by:

\[
\begin{align*}
\text{sort}(t) &= T & \text{if } t \text{ is a variable-terminal of sort } T, \\
\text{sort}(t) &= t & \text{if } t \text{ was defined in Step 1,} \\
\text{sort}(N) &= N & \text{if } N \text{ is a nonterminal.}
\end{align*}
\]

**Step 2 (add operator arities and sort orderings)**

For every production of the form \( N \rightarrow f \), where \( f \) is a constant-terminal, add the operator 

\( f: N \)

to \( Z \).

For every production of the form \( N \rightarrow f(x_1, \ldots, x_n), n > 0, \) add the operator 

\( f: \text{sort}(x_1) \times \ldots \times \text{sort}(x_n) \rightarrow N \)

to \( Z \).

For every production of the form \( N \rightarrow N' \) where \( N' \) is a nonterminal, add the pair 

\( (N', N) \)

to the relation < in \( Z \), i.e. order \( N' < N \).

**Step 3 (combine \( Z \) and \( \Sigma \))**

Let \( \Sigma' \) be the union of \( Z \) and \( \Sigma \).

**Step 4 (ensure regularity)**

For each n-ary operator \( f, n \geq 0, \) in \( \Sigma' \), for each pair of arities 

\( f: s_1 \times \ldots \times s_n \rightarrow t \)

\( f: s'_1 \times \ldots \times s'_n \rightarrow t' \)

with \( t \leq t' \),

for each sequence of sorts \( <u_1, \ldots, u_n> \) such that 

for all \( i = 1, \ldots, n, u_i \leq s_i, \) and \( u_i \leq s'_i, \) and \( u_i \) is maximal among such sorts, 

do:

add the new sort \( \text{GLB}(t, t') \) to \( \Sigma' \), 

add the pairs \( (\text{GLB}(t, t'), t) \) and \( (\text{GLB}(t, t'), t') \) to the relation < in \( \Sigma' \), 

if for some \( r \) we have \( r < t \) and \( r < t' \),

then add \( (r, \text{GLB}(t, t')) \) to the relation < in \( \Sigma' \),

add a new arity \( f: u_1 \times \ldots \times u_n \rightarrow \text{GLB}(t, t') \) to \( \Sigma' \).

(Note: any of these substeps must be omitted if done already. The new sorts are intended to be greatest lower bounds, thus the sorts \( \text{GLB}(x, y) = \text{GLB}(y, x), \) 
\( \text{GLB}(x, \text{GLB}(y, z)) = \text{GLB}(x, y, z), \) etc.)

**Step 5 (ensure monotonicity)**

For each n-ary operator \( f, n \geq 0, \) in \( \Sigma' \), for each ordered pair of arities 

\( f: s_1 \times \ldots \times s_n \rightarrow t \)

\( f: s'_1 \times \ldots \times s'_n \rightarrow t' \)

if for all \( i = 1, \ldots, n, s_i \geq s'_i, \) then:
if \( t > t \) then delete the arity \( f : s'_1 \times \ldots \times s'_n \rightarrow t' \) from \( \Sigma' \).

Step 6 (remove redundant sorts)

For every sort \( s \) in \( \Sigma' \), if \( s \) does not occur in an operator arity, then delete \( s \) from \( \Sigma \).

5. An Example

Consider the language of the varying parts from the example presented in §2: the set of terms \( \{g(f(x)), g(g(f(x))), \ldots \} \). A grammar for this language is

\[
S \rightarrow g(S) \mid g(F) \\
F \rightarrow f(x)
\]

The result of the algorithm is:

\[
\begin{array}{c|ccc}
\text{Step 1} & < & \text{Sorts} & \uparrow \\
S < T & T, S, F & & \\
\text{Step 2} & & g : S \rightarrow S \\
& & g : F \rightarrow S \\
& & f : T \rightarrow F \\
\text{Step 3} & & g : T \rightarrow T \\
& & f : T \rightarrow T \\
& & c : T \\
\text{Step 4} & \text{GLB}(T, F) < T & \text{GLB}(T, F) & f : T \rightarrow \text{GLB}(T, F) \\
& \text{GLB}(T, F) < F & & \\
\text{Step 5} & (\text{delete } *) & & \\
\end{array}
\]

Result:

\[
T \\
\downarrow \\
S \quad \text{GLB}(T, F) \\
\uparrow \\
F
\]

\[
(T \Sigma)^S = \{g(f(x)), g(g(f(x))), \ldots \} \\
= \text{language of start symbol } S
\]

Now the single rule \( f(y) \rightarrow y \) with variable \( y \) of sort \( V \) is

i) a complete, confluent rewrite set;
ii) a conservative extension of \( R^\infty \).

Moreover, the order-sorted signature is monotonic and regular.
7. Conclusions

The algorithm which we have given is only a part of the full process of transforming an infinite set of rewrite rules $R$ (or more accurately a divergent case of Knuth-Bendix completion) into a finite complete set of rules. We have shown that if we enrich the original signature $\Sigma$ in an appropriate way then at least in some cases we arrive at a signature in which at least there exists a complete set of rules which forms a conservative extension of the original set, which may not be true in $\Sigma$.

References


Date: January 9 - 11
Title: International Workshop on Formal Methods in VLSI Design
Location: San Juan, Puerto Rico.
Sponsor: SIGDA.
Contact: P.A. Subrahmanyan, AT&T Bell Labs, Rm. 4E-530, Homdel, NJ 07733. Tel. (201) 949-5812.
Email: subra@vax135.att.com.

Date: January 21 - 23
Title: The Eighteenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages
Sponsor: SIGACT, SIGPLAN.
Acronym: POPL 91
Location: Orlando, Florida, USA.
Contact: David Wise, Indiana Univ., Computer Science Dept., 101 Lindley Hall, Bloomington, IN 47405; Tel. (812) 855-4866.
Email: dswise@iuvax.cs.indiana.edu.

Date: February 12 - 15
Title: Second International Workshop on Parsing Technologies
Location: Cancun, Quintana Roo, Mexico.
Contact: Joan Maddamma, IWPT-91 Workshop Sec., (412) 268-9656; fax: (412) 621-5473.
Email: jfm@cs.cmu.

Date: February 13 - 15
Title: 5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation
Location: Torino, Italy.
Contact: Maria Carla Calzarossa, Dipartimento di Informatica e Sistemistica, Universita’ di Pavia Via Abbiategrasso, 209, 27100 Pavia, Italy. Tel. +39 (382) 391 350.
Email: mcc@ipvpel.infin.it.

Date: February 14 - 16
Title: 8th Symposium on Theoretical Aspects of Computer Science
Acronym: STACS 91
Location: Hamburg, West Germany.
Contact: Professor Dr Matthias Jantzen, Fachbereich Informatik (TG1), Universität Hamburg, Rothenbaumchaussee 67/69, D-2000 Hamburg 13.
Sponsors: Special Interest Group for Theoretical Computer Science of the GESELLSCHAFT füriNFORMATIK (GI), and the Special Interest Group for Applied Mathematics of AFZET.

Date: March 3- 6
Title: Fifth International Workshop on High-Level Synthesis
Location: Beuelerhoeche, West Germany.
Sponsors: SIGDA, IEEE and DATC.
Contact: Raul Camposano, IBM TJ Watson Research Center, P.O. Box 218, Yorktown Heights, NY, 10598. Tel. (914) 945-3871.
Email: raulc at ibm.com.

Date: March 4 - 7
Title: ACM 19th Computer Science Conference
Location: San Antonio, Tex., USA.
Sponsor: ACM.
Contact: C. Jinshong Hwang, Dept. of Computer Science, Southwest Texas State University, San Marcos, TX 78666. Tel. (512) 245-3434.
Email: CSHWANG@SWTEXAS.

FORTHCOMING EVENTS - 22 October 1990

Date: November 5 - 8
Title: Third International Conference on Formal Description Techniques
Acronym: FORTE'90
Location: Madrid, Spain.
Contact: Juan Quemada, ETSI Telecomunicacion, Ciudad Universitaria s/n, E-28040, Madrid, Spain. Tel: +34-1-5495700.
Email: jquemada@dii.upm.es

Date: November 26 - 29
Title: Conference on Software Maintenance
Location: San Diego, Calif., USA.
Contact: Vacos Rajlich, Wayne State University, Department of Computer Science, Detroit, Michigan 48202. Tel. (313)-577-2477
Email: rajlich@cs.wayne.edu.

Date: December 3 - 5
Title: ACM SIGSOFT '90: 4th Symposium on Software Development Environments
Acronym: SDE4
Location: Irvine, Calif., USA.
Sponsor: ACM SIGSOFT, SIGPLAN.
Contact: Dewayne E. Perry, AT&T Bell Laboratories, 4 SDE, 600 Mountain Avenue, Murray Hill, NJ 07974. Tel. (201) 582 2529.
Email: dep@allegra.att.com.

Date: December 3 - 8
Title: Fourth International Workshop on CASE
Acronym: CASE 90
Location: Irvine, Calif., USA.
Contact: Ronald J. Norman, San Diego State Univ., College of Business Administration, San Diego, CA 92182; (619) 594-3734.

Date: December 17 - 19
Title: 10th Conference on Foundations of Software Technology and Theoretical Computer Science
Location: Bangalore, India.
Contact: Y.N. Srikant, Indian Inst. of Science, Bangalore 560 012, India, phone (812) 334-411.

1991

Date: January 9 - 11
Title: BCS-FACS: 4th UK Refinement Workshop
Location: Cambridge, UK.
Co-Sponsor: Logica Cambridge Limited.
Email: rosalind@uk.co.logcam.
May 5 - 10
Date:
Title: Mathematical Fundamentals of Database and Knowledge Base Systems
Acronym: MFDBS·91
Location: Goehren, Germany
Contact: Bernhard Thalheim, Rostock Univ., Computer Science Dept., 2500 Rostock, GDR.
Tel. (37) (81) 45430.

Paper Submission Details:
Submit 5 copies of detailed abstract (more than four pages) or full draft paper (15 pages by 25 October 1990).

May 6 - 8
Date:
Title: ACM Symposium on Theory of Computing
Acronym: STOC '91
Location: New Orleans, Louisiana, USA.
Contact: Chris Koutsougeras, Computer Science Dept., Tulane University, New Orleans, LA 70118.
Email: criS@rex.cs.tulane.edu.
Or Contact: Jeff Viner, Department of Computer Science, Brown University, Providence, Rhode Island, 02912-1910.
Email: jsv@cs.brown.edu.

Paper Submission Details:
Authors are requested to send sixteen copies of a detailed abstract (not a full paper) by Nov. 7, 1990 to: Joseph Halpern, STOC '91 Program Chair, IBM Almaden Research Center, Dept. K53/802, 650 Harry Rd., San Jose, CA 95120-6099.

May 7 - 10
Date:
Title: International Workshop on Logic Synthesis
Location: Research Triangle Park, N.C., USA.
Sponsor: SIGDA and MCNC.
Contact: Franc Brglez, BNR/MCNC, P.O. Box 12889, Research Triangle Park, NC; (919) 248-1925.
Email: brglee@mcnc.org.

May 13 - 16
Date:
Title: 13th International Conference on Software Engineering
Location: Austin, Texas, USA.
Contact: ICSE13, Lazzlo A. Belady, MCC, PO BOX 200195, Austin, Texas 78720 USA.
Tel. (512) 338-3356.
Email: belady@mcc.com.

May 17 - 18
Date:
Title: International Symposium on Software Reliability Engineering
Location: Austin, Texas, USA.
Sponsors: IEEE CS Technical Committee on Software Engineering and National Aeronautics and Space Administration.
Contact: John C. Munson, Division of Comp. Sci., Univ. of West Florida, Pensacola, FL 32514; (904) 474-2989.
Email: jmunson@dcsmnet.uwf.edu.

May 21 - 23
Date:
Title: 4th Software Engineering Standards Application Workshop
Location: San Diego, Calif., USA.
Sponsor: IEEE CS.
Contact: Vera Edelstein, NYNEX Corp., 500 Westchester Ave., White Plains, NY 10604; Tel. (914) 683-2888.
June 25 - 28
Eighth International Conference on Logic Programming
ICLP'91
Paris, France.
Koichi Furukawa, ICOT, Mita Kokusai Building, 21 F4 - 28 Mita 1, Chome, Minato-ku, Japan 108, Japan.
furukawa@icot.or.jp.

June 26 - 28
Twelfth International Conference on Application and Theory of Petri Nets
Aarhus, Denmark.
Manuel Silva, Departamento Ingenieria, Electrica e Informatica, C/Marfa de Luna, 3 (Actur) E-50015 Zaragoza, Spain. Tel. +34 76 51 72 74.
Kurt Jensen, Computer Science Department, Aarhus University, Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark. Tel. +45 86 12 71 88.
icps@daimi.aau.dk.

June 26 - 28
International Conference on Computing 1991
COMP 91
Imperial College of Science and Technology, London, United Kingdom.
IEEE in coop. with Sigsic, Easics, British Association for the Advancement of Science, British Society for the History of Science. The Royal Society, Institution of Electronic and Radio Engineers.
General Chair, E. Ash CBE FRS, Imperial College of London, Exhibition Road, London SW7 2AZ, United Kingdom. Tel. +44 1 589 5111.
W. D. Hurley, Dept. of Comp. Sci., Alumni Hall, Univ. of Pittsburgh, Pittsburgh, PA 15260; (412) 624-8843.
hurley@cs.pitt.edu.

July 1 - 3
International Conference on Computing 1991
The 21st Annual International Symposium on Fault-Tolerant Computing
FTCS 21
Quebec, Canada.
Prof. Mario Rodriguez Arteaga.
W450@EMDCOM11.BITNET.
Date: October 21 - 24
Title: Third European Software Engineering Conference
Contact: Alfonso Fuggetta, CEFRIEI, c/o AICA-ESEC '91, P.le Rodolfo Morandi 2, 1.2021 Milan, Italy.
Email: alfonso@imicefr.bitnet.
Acronym: ESEC 91
Location: Milan, Italy.
Sponsors: AFCET et al.
Paper Submission Details:
Submit six copies of full paper and abstract by Jan. 15, 1991, to Alex van Lamwenande, Unie, d'Informatique, Univ. Catholique de Louvain, Place Sainte Barbe, 2 B-1348 Louvain-La-Neuve, Belgium.
Email: esec@info.ucl.ac.be.

Date: October 25 - 26
Title: Sixth Int'l Workshop on Software Specification and Design
Location: Como, Italy.
Sponsor: SIGSOFT and IEEE-CS.
Contact: Jean-Pierre Finance, CRIN Univ. de Nancy I, Campus Scientifique, BP 239, 54506 Vandoeuvre les, 33 83 91 21.
Email: finance@loria.crin.fr.
Paper Submission Details:
Submit five copies of regular or position paper by January 21 1991 to Carlo Ghezzi.
Email: relett@24imipoli.bitnet.

Date: December 3 - 5
Title: The Fourth International Workshop on Petri Nets and Performance Models
Acronym: PNPM 91.
Location: Melbourne, Australia.
Sponsor: Telecom Australia.
Contact: Jonathan Billington, Telecom Australia Research Laboratories, PO BOX 249, Clayton, Vic., 3168, Australia. Tel. +61-3-5416416.
Email: j.billington@telstra.com.

1992

Date: January 20 - 24
Title: 19th ACM Symposium on Principles of Programming Languages
Acronym: POPL '92
Location: Albuquerque, N. Mex., USA.
Sponsor: SIGACT, SIGPLAN.
Contact: Stuart Feldman, Bell Commun. Research, 445 South St., Rm. 2E-386, Morristown, NJ 07960-1910. Tel. (201) 829-4305.
Email: sif@bellcore.com.

Date: July 14 - 19
Title: 6th Annual IEEE Logic in Computer Science Symposium
Location: Amsterdam, Netherlands.
Sponsors: IEEE, EATCS, ASL in cooperation with ACM SIGACT.
Contact: Albert Meyer, MIT Lab for Computer Science, ME43-315, 545 Technology Square, Cambridge, MA 02139. Tel. (617) 253-6024.
Email: meyera@cs.mit.edu.

Date: September 3 - 6
Title: Category Theory and Computer Science
Location: Ecole Normale Superieure, Paris, France.
Contact: David Pitt, Department of Mathematics, University of Surrey, Guildford, Surrey GU2 7H, UK.
Email: dhp@cs.surrey.ac.uk.
Local arrangements: Pierre-Louis Curien, LIENS, 45 rue d'Ulm, 75230 Paris Cedex 05, France.
Email: curien@dmi.ens.fr.

Date: September 9 - 13
Title: 16th International Symposium on Mathematical Foundations of Computer Science
Acronym: MFCS'91
Location: Warsaw, Poland.
Sponsors: Institute of Computer Science of the Polish Academy of Sciences and the Institute of Informatics of Warsaw University.
Contact: P. Chrzastowski-Wachtel and A. Tarlecki, MFCS'91, Institute of Computer Science, Polish Academy of Sciences, PKN, P.O. Box 22, 00-901 Warsaw, Poland.
Paper Submission Details:
Authors are invited to submit 5 copies of a draft paper to A. Tarlecki by 15 JANUARY 1991.

Date: September 16 - 18
Title: Software Engineering for Real Time Systems
Location: Royal Agricultural College, Cirencester, UK.
Sponsor: IEE.
Organizers: Institution of Electrical Engineers.
Contact: Conference Services, The Institution of Electrical Engineers, Savoy Place, London WC2R 0BL, UK. Tel. 071 240 1871 Ext. 222. Telex. 261176 IEE LDN G. Fax. 071 240 7735.

Date: October 6 - 11
Title: OOPSLA '91
Location: Phoenix Convention Center, Phoenix, Ariz., USA.
Sponsor: SIGPLAN.
Contact: John Richards, IBM TJ Watson Research Ctr., PO BOX 704, Yorktown Heights, NY 10598; (914) 784-7731.
Email: jnr@ibm.com.