FACS FACTS

The Newsletter of the BCS Formal Aspects of Computing Science SIG

Series 2 Vol.1 No. 2 November '90.

Editor Jawed Siddiqi

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

FACS Officers

Chairman
Dr D J Cooke
John
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU
Telephone: 0509 222676
Email: john@uk.ac.lut.cs-vaxa

Secretary
B T Denvir
Tim
37 Orpington Road
Winchmore Hill
London N21 3PD
Telephone: 01 882 5853(home), (0225) 444700
Email: btd@uk.co.praxis

Treasurer
Dr R G Stone
Roger
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU
Telephone: 0509 222686
Email: roger@uk.ac.lut.cs-vaxa

Newsletter Editor
Dr J I A Siddiqi
Jawed
Dept. of Computer Studies
Sheffield City Polytechnic
Pond Street
Sheffield S1 1WB
Telephone: 0742 720911

Publicity
Dr B Q Monahan
Brian
Dept. of Computer Science
The University
Manchester M13 9PL
Tel: home 0223 314019 / 061 275 6137
Email: bqm@uk.ac.man.cs

Dr P N Scharbach
Peter
Information Technology Research Unit
BP Research Centre
Chertsey Road
Sunbury-On-Thames
Middlesex TW16 7LN
Telephone: (0932) 762570

Specialist Groups Management Committee Representative

Dr T H Axford
Tom
Computer Science Dept.,
University of Birmingham
PO BOX 363,
Birmingham
B15 2TT.
Telephone: 021 414 4779
Email: AxfordTH@uk.ac.bham

Steve Webster
Department of Computer Science
Bournemouth Polytechnic
Wallisdown Road
Poole
Dorset BH12 5BB

David Till
Comp Sci
City U
Northampton Sq, EC1V 0HB

John Boarder
"Cotteff"
New Inn Road
Beckley Oxon OX3 9SS

Dr T H Axford
Computer Science Dept.,
University of Birmingham
PO BOX 363,
Birmingham B15 2TT.

D Blyth
Incord Ltd
15, Sherwood Avenue
Ferndown
Wimborne, Dorset BH22 8JS

Dr D J Cooke
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU

B T Denvir
37 Orpington Road
Winchmore Hill
London N21 3PD

Dr A J J Dick
Racal Research Ltd
Worton Drive
Reading RG2 0SB

Prof. S J Goldsack
Dept of Computing,
Imperial College
180 Queen's Gate,
London SW7 2BZ

Dr R J Mitchell
32 Lincoln Avenue
Peacehaven
E Sussex BN10 7JT

Dr B Monahan
Dept of Computer Science
The University
Manchester
M13 9PL

Dr A Norcliffe
Dept of Mathematical Sciences
Sheffield City Polytechnic
Sheffield S1 1WB

Dr P N Scharbach
Information Technology Research Unit
BP Research Centre
Chertsey Road
Sunbury-On-Thames
Middlesex TW16 7LN

R C Shaw
Praxis Systems PLC
20 Manvers Street
Bath BA1 1PX

Dr J I A Siddiqi
Dept. of Computer Studies
Sheffield City Polytechnic
Pond Street
Sheffield S1 1WB

Prof D Simpson
Dept of Computing
Brighton Polytechnic,
Moulsecoomb,
Brighton BN2 4AT

Dr R G Stone
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU

BCS-FACS FEES 1991

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 (eg. ACM, GI, AFCET, ACIA, etc.).

For this fee members will receive our FACS-FACTS 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 SpecificaTion) 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 conditons, because transitions are not instantaneous.

The methodological asumptions 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 withing 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 derivabiltiy relation (a relation that can be used to deduce all of the logical consequences of the language), $\models$. ie. $<L,\models>$. Logics are generally presented in an axiomatic fashion. In order to move to a characterisation of a particular application (ie. 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 Δ is the set of axioms of the theory.

The FOREST team uses a logic called M[A]L, 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 → Ag

push: natural → Ac (ie. pushing a natural number onto a stack produces something of type action)

Both of the sorts Ag and Ac 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 Ag and Ac can be used in modal formulae. For example if $A \in Ag$, $a \in Ac$ and α 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) α holds. An example of this is:

[Tom,Close(Account_id)]not(has_access(Tom,account_id))

A more common use of this type of formula is:

$$Pre \rightarrow [A,a]Post$$

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

$[A,a](\varphi \rightarrow \psi) \rightarrow ([A,a]\varphi \rightarrow [A,a]\psi)$ Distributivity

$[A,a]not(\alpha) \rightarrow not([A,a]\alpha)$ 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 P(ermitted) and O(bliged). These can be used in the following manner:

P(Agent,Action),
O(Agent,Action),
Pref(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.

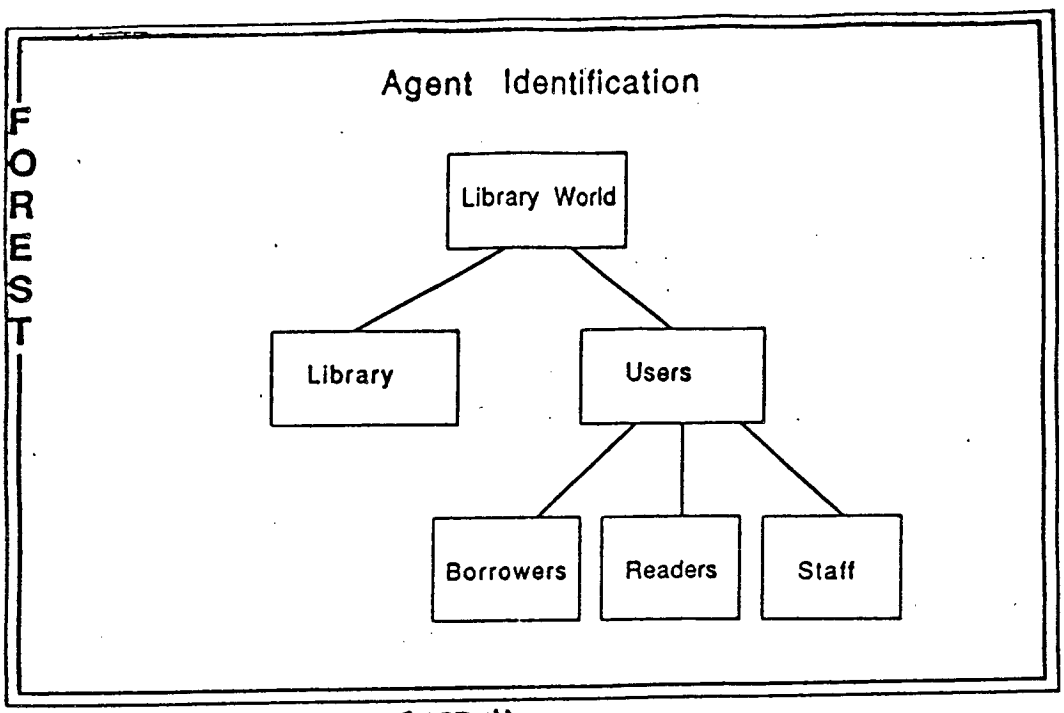
Formal methods bring us freedom from ambiguity and a basis upon which we can reason. Much of this reasoning can be done automatically by software tools, eg. detecting missing parts of specifications, verification, animation. However, formal methods do not particularly help with controlling complexity, 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 diagramatically 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 targetted 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) identificaition 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.

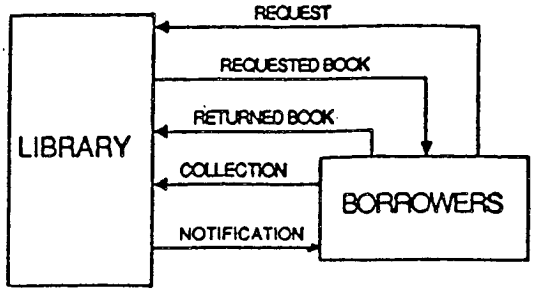


STEP (i)

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.

FOREST

Library Example



STEP (ii)

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)

S	I	A	O	D
Source Agent	→ Input Dataflow	→ Action	→ Output Dataflow	→ Destination Agent

Library Action Table

S	I	A	O	D
Borrower	→ request	→ issue	- - → book	- - → borrower
			→ transaction record	→ library
library	→ transaction record	→ adjust records		
Borrower	→ collection	→ issue	→ book	→ borrower
			→ transaction record	→ library
Borrower	→ returned book	→ check-in	- - → note	- - → borrower
			→ transaction record	→ library

FOREST

Borrower Action Table:

S	I	A	O	D
		request →	request →	library
library →	book →	return →	returned book →	library
library →	notification →	collect →	collection →	library

FOREST

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

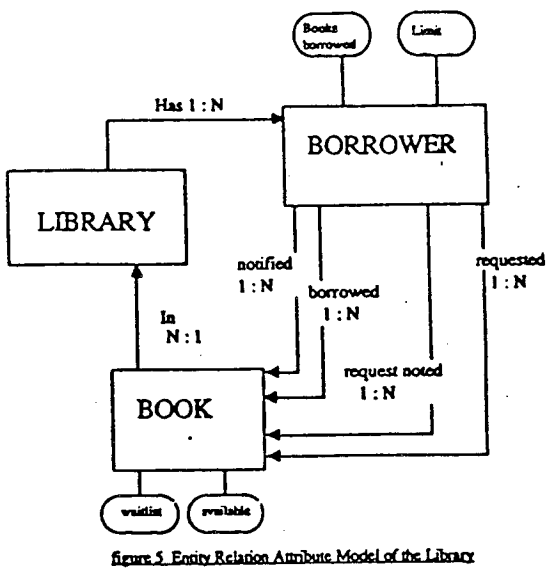


figure 5 Entity Relation Attribute Model of the Library

FOREST

STEP (iv)

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

where u is a user
b is a book
F on the table

then the user is not permitted to put in a request for b but the library is obliged to issue the book.

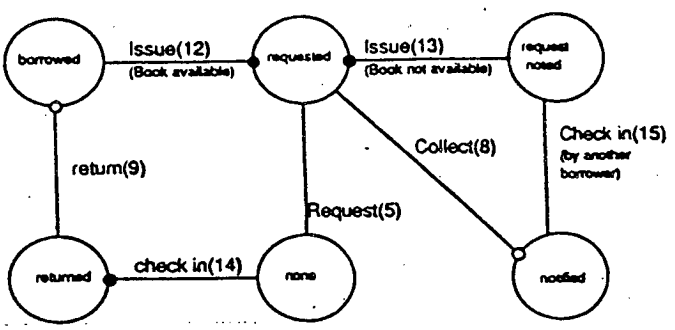
$books\_borrowed(u) < limit(u)$	T				F			
$available(b)$	T		F		T		F	
$status(u,b) = none$	T	F	T	F	T	F	T	F
$status(u,b) = requested$	F	T	F	T	F	T	F	T
$[b, request]$	per	¬per	per	¬ per	¬ per	*	¬ per	*
$[library, issue(b)]$	¬per	obl	¬ per	¬ per	¬ per	*	¬ per	*

(The asterisk identifies unattainable situations)

STEP (v)

Step (vi) (permission and obligation analysis) has been ommitted 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

per (u,request(b)) ↔ (books borrowed(u) < limit(u))
∧ status(u,b) = none

[u,request(b)] (status(u,b) = requested ∧ obl (library, issue(u,b)))

Protect against multiple obligations:

∃ S.OBLS(library, S) → [u,request(b)]
(status(u, b) = requested ∧
∃S'.OBLS(library, S') ∧ S' = S | obl (library, issue(u,b)))

FOREST

Some Library Axioms

The effects of issuing a book are defined by the axiom:

available(b) ∧ books borrowed(u) = N →
[library, issue(u,b)]
status(u,b) =borrowed ∧ books borrowed(u) = N + 1

today = d ∧ available(b) →
[library, issue(u, b)]
obl(u, return(b), today < d + 14)

FOREST

'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 facilites. 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(socarates) and
for all X: human(X) implies mortal(X)

Prove: mortal(socrates)

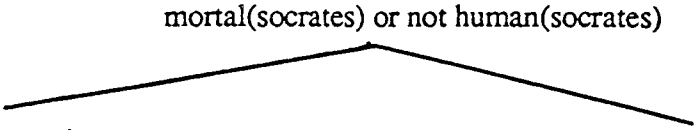
Negate the proposition: not mortal(socrates)

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) — not human(socrates)

contradicts negation of proposition given — 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 demonstate 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 dagram (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, ie 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 propostional tableau rules:

Rule A:

$\neg\neg a$	$a \wedge b$	$\neg(a \wedge b)$	$\neg(a \Rightarrow b)$
a	a	$\neg a$	a
	b	$\neg b$	$\neg b$

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

Rule B:

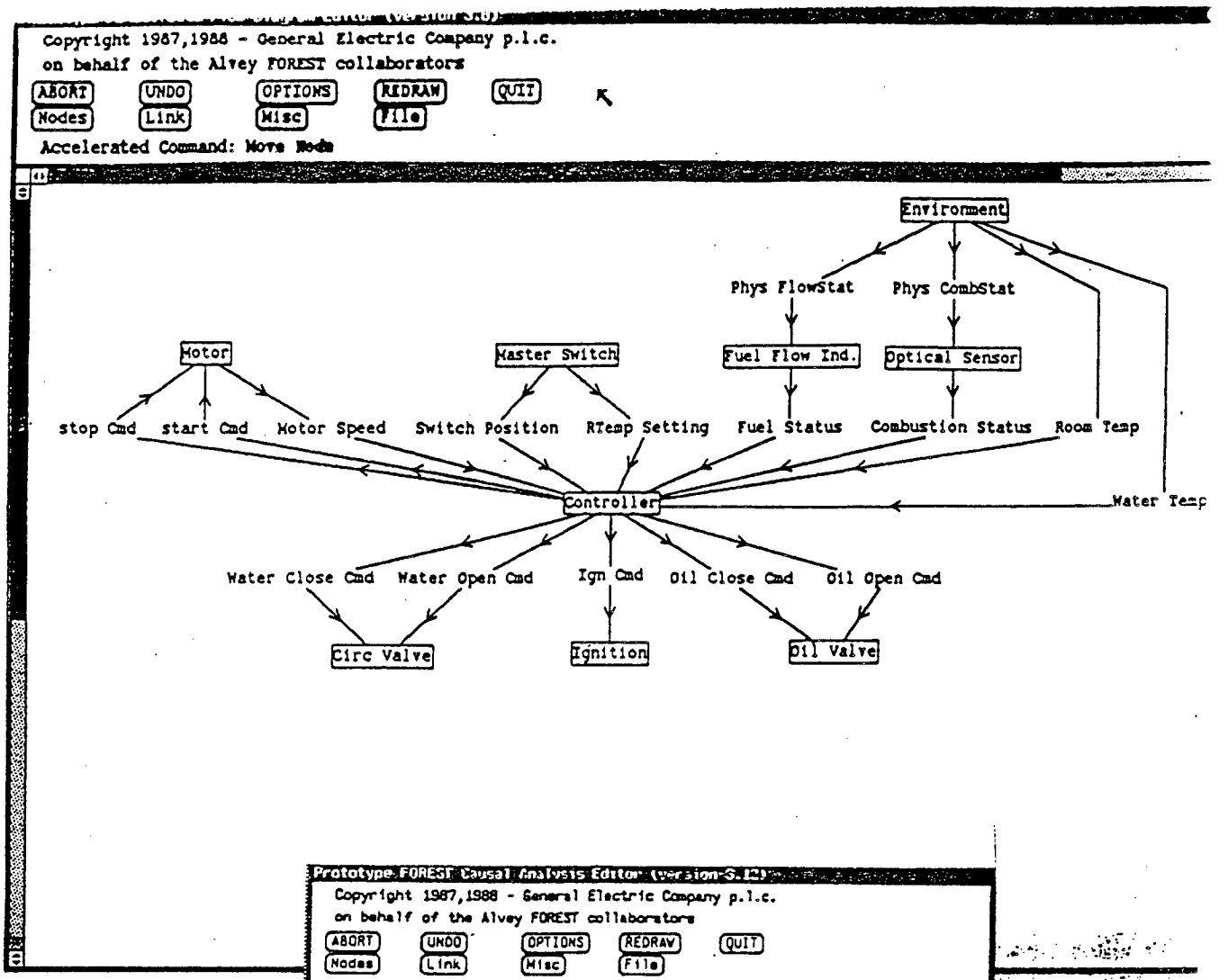
$a \vee b$	$\neg(a \wedge b)$	$a \Rightarrow b$
a | b	$\neg a \mid \neg b$	$\neg a \mid b$

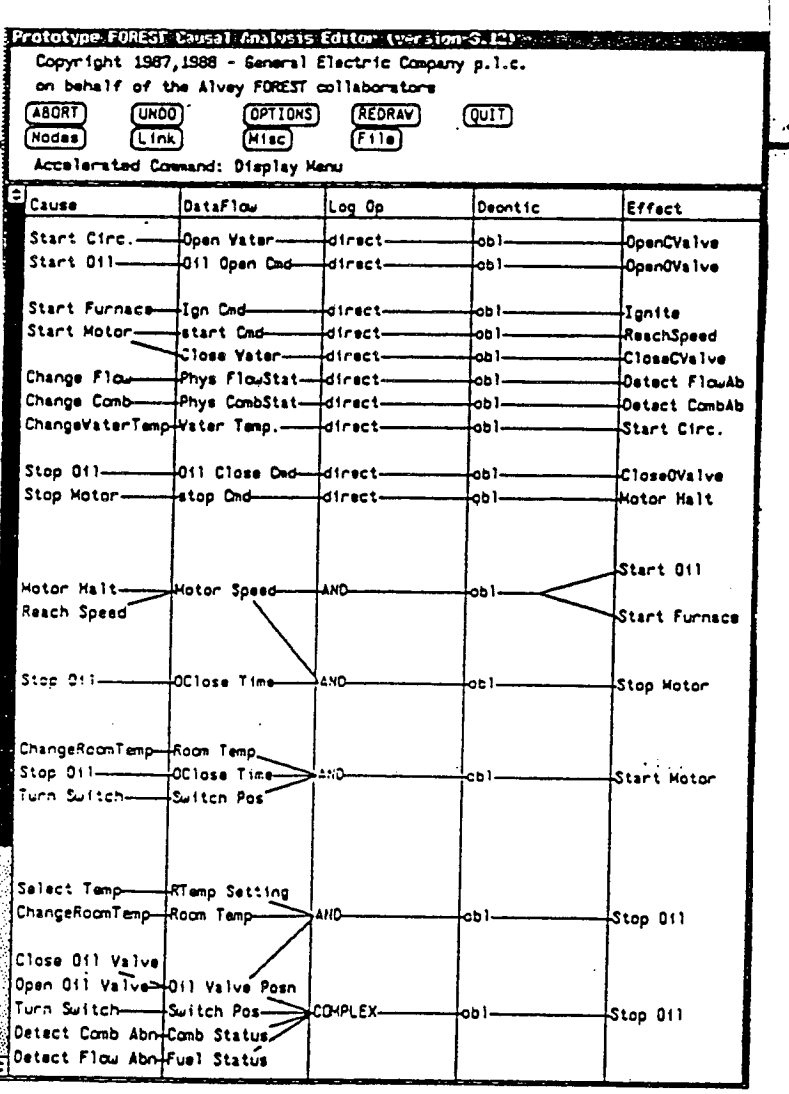
The vertical bar represents the opening of two parallel branches of the tableau.

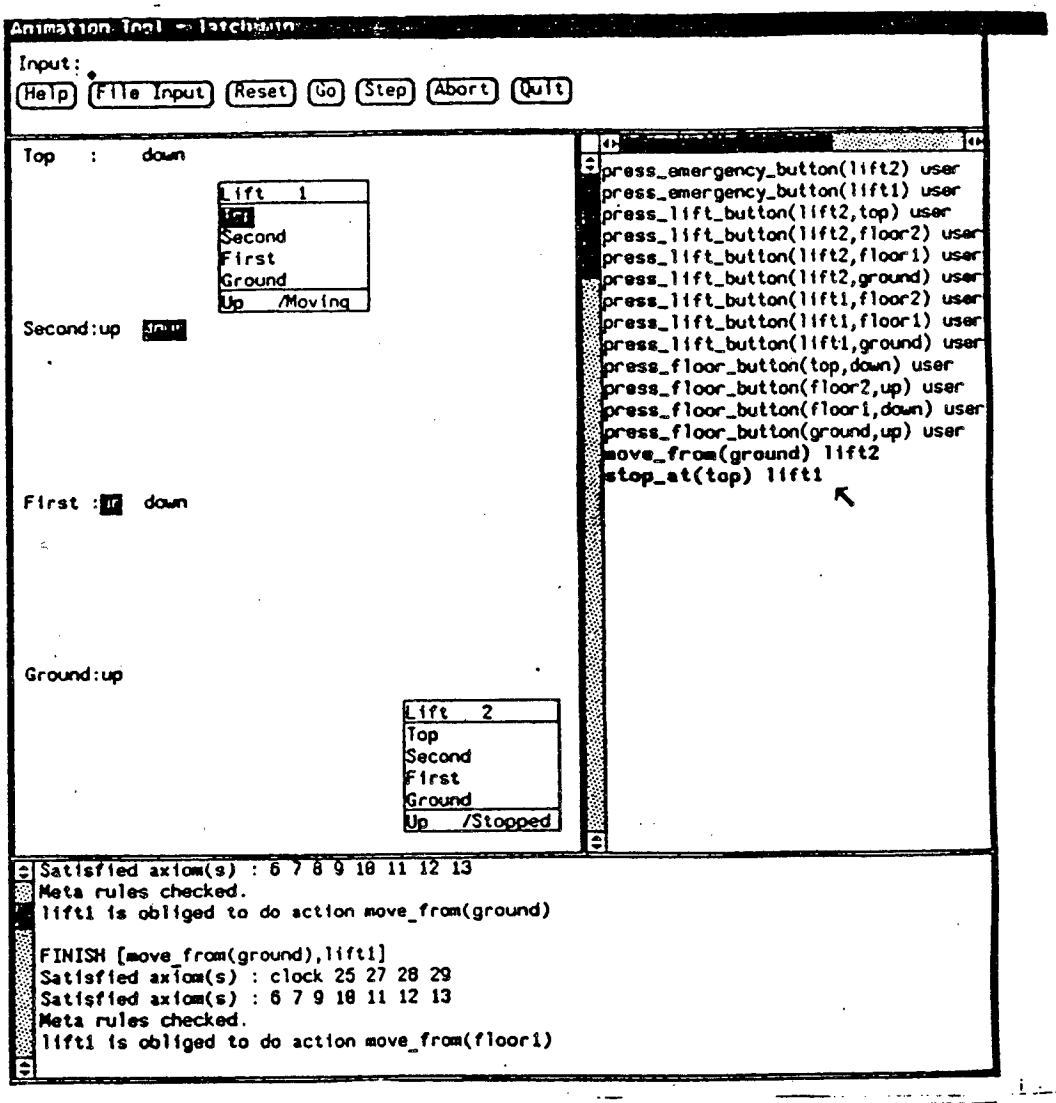
Rule C:

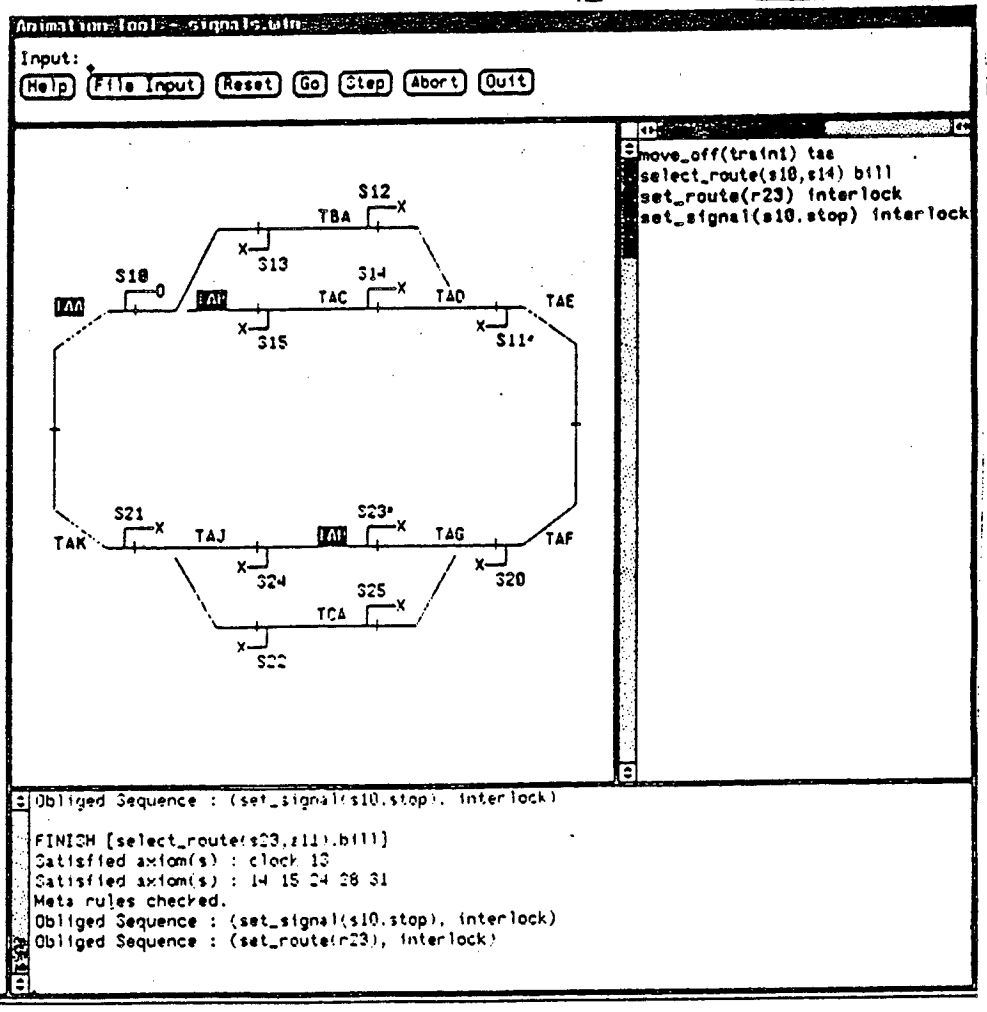
$\forall x a$	$\neg \exists a$
a (x/c)	$\neg$a (x/c)
$\forall$ x a	$\neg \exists$a

where c is any constant.


Copyright 1987,1988 - General Electric Company p.l.c.
on behalf of the Alvey FOREST collaborators
ABORT
UNDO
OPTIONS
REDRAW
QUIT
Nodes
Link
Misc
File
Accelerated Command: Move Node
Environment
Phys FlowStat
Phys CombStat
Motor
Master Switch
Fuel Flow Ind.
Optical Sensor
stop Cmd
start Cmd
Motor Speed
Switch Position
RTemp Setting
Fuel Status
Combustion Status
Room Temp
Water Temp
Controller
Water Close Cmd
Water Open Cmd
Ign Cmd
Oil Close Cmd
Oil Open Cmd
Circ Valve
Ignition
Oil Valve


Prototype FOREST Causal Analysis Editor (version 6.1a)
Copyright 1987,1988 - General Electric Company p.l.c.
on behalf of the Alvey FOREST collaborators
ABORT
UNDO
OPTIONS
REDRAW
QUIT
Nodes
Link
Misc
File
Accelerated Command: Display Menu
Cause
DataFlow
Log Op
Deontic
Effect
Start Circ.
Open Vater
direct
obl
OpenCValve
Start Oil
Oil Open Cmd
direct
obl
OpenOValve
Start Furnace
Ign Cmd
direct
obl
Ignite
Start Motor
start Cmd
direct
obl
ReachSpeed
Close Vater
direct
obl
CloseCValve
Change Flow
Phys FlowStat
direct
obl
Detect FlowAb
Change Comb
Phys CombStat
direct
obl
Detect CombAb
ChangeVaterTemp
Vater Temp.
direct
obl
Start Circ.
Stop Oil
Oil Close Cmd
direct
obl
CloseOValve
Stop Motor
stop Cmd
direct
obl
Motor Halt
Motor Halt
Reach Speed
Motor Speed
AND
obl
Start Oil
Start Furnace
Stop Oil
OClose Time
AND
obl
Stop Motor
ChangeRoomTemp
Stop Oil
Turn Switch
Room Temp
OClose Time
Switch Pos
AND
obl
Start Motor
Select Temp
ChangeRoomTemp
RTemp Setting
Room Temp
AND
obl
Stop Oil
Close Oil Valve
Open Oil Valve
Turn Switch
Detect Comb Abn
Detect Flow Abn
Oil Valve Posn
Switch Pos
Comb Status
Fuel Status
COMPLEX
obl
Stop Oil


Animation Tool - latchlift
Input:
Help
File Input
Reset
Go
Step
Abort
Quit
Top : down
Lift 1
Top
Second
First
Ground
Up /Moving
Second:up
First : up down
Ground:up
Lift 2
Top
Second
First
Ground
Up /Stopped
press_emergency_button(lift2) user
press_emergency_button(lift1) user
press_lift_button(lift2,top) user
press_lift_button(lift2,floor2) user
press_lift_button(lift2,floor1) user
press_lift_button(lift2,ground) user
press_lift_button(lift1,floor2) user
press_lift_button(lift1,floor1) user
press_lift_button(lift1,ground) user
press_floor_button(top,down) user
press_floor_button(floor2,up) user
press_floor_button(floor1,down) user
press_floor_button(ground,up) user
move_from(ground) lift2
stop_at(top) lift1
Satisfied axiom(s) : 6 7 8 9 10 11 12 13
Meta rules checked.
lift1 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.
lift1 is obliged to do action move_from(floor1)


Animation Tool - signals.bin
Input:
Help
File Input
Reset
Go
Step
Abort
Quit
S12
TBA
S13
S10
TAA
TAB
S14
TAC
TAD
TAE
S15
S11
S21
TAK
TAJ
S23
TAG
TAF
S24
S20
S25
TCA
S22
move_off(train1) tae
select_route(s10,s14) bill
set_route(r23) interlock
set_signal(s10.stop) interlock
Obliged Sequence : (set_signal(s10.stop), interlock)
FINISH [select_route(s23,s11),bill]
Satisfied axiom(s) : clock 13
Satisfied axiom(s) : 14 15 24 28 31
Meta rules checked.
Obliged Sequence : (set_signal(s10.stop), interlock)
Obliged Sequence : (set_route(r23), interlock)

Rule D:

$\exists a$	$\neg \forall x\, a$
a (x/d)	$\neg$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:	Rule F:
$[Ag,Ac]\alpha$	$\neg[Ag,Ac]\alpha$
$\tau(\sigma,\alpha)$	$\tau(\sigma',\neg\alpha)$

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

F. X. Reid Not Sighted in Belgium

The FACS FACTS roving correspondent
Victor Zemanticz

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 Roever (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 God[1] and that he is working for the US DoD.[2] Mr. D. Shtroompf (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*rds at *mp*r**l (c) to play the Mozart A minor piano sonata right through without any mistakes. Mr. Shtroompf 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 desparate 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.

FORMAL ASPECTS OF COMPUTING

CONTENTS OF VOLUME 2 (1990)

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

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

Contents

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 [ISO8807] 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 as 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 [GG89] 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 it 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 [SNGM87] 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 [TJ89] 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 $\mathcal{S}$ such that terms with sort $\mathcal{S}$ 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 $f{:}s_1 \times \ldots \times s_n \rightarrow s$, $f{:}t_1 \times \ldots \times t_n \rightarrow t$, if for all $i=1,\ldots,n$, $s_i \leq t_i$, then $s \leq t$.

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) $f(g(f(x))) \rightarrow g(f(x))$

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:

R1)	$f(g(f(x)))$	$\rightarrow$	$g(f(x))$
R2)	$f(g(g(f(x))))$	$\rightarrow$	$g(g(f(x)))$
R3)	$f(g(g(g(f(x)))))$	$\rightarrow$	$g(g(g(f(x))))$

etc.

We use R^{∞} to denote this infinite sequence.

It can easily be seen that the rules in R^{∞} fall into a clear pattern:

$f(g^n(f(x))) \rightarrow g^n(f(x))$ for any $n > 0$.

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

$t = g^n(f(x))$ 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^{∞} 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^{∞}. 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 Σ, brackets, or variables in Σ, 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:

$N \rightarrow f(x_1,...,x_n)$
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 Σ, let R^∞ be the infinite sequence of rules generated by the Knuth-Bendix completion algorithm. We partition R^∞ into Q and Q^∞, where Q^∞ is the infinite sequence we wish to generalise; i.e. we aim to replace Q^∞ by a finite set of rules. Note, in general, R^∞ 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^∞ 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^∞ can be generalised. We do not concern ourselves here with the generation of the grammar from the finite subset of the infinite sequence Q^∞ 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^∞,
- there is a sort X in Σ such that every term in $L(G)$ has sort X and X is minimal among such sorts.

Let $Z = (\{X,\mathcal{S}\}, \{(\mathcal{S},X)\},\{\})$ 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 Σ as follows:

Step 1 (add sorts)

For every non-terminal N in V, add the sort $\mathcal{N}$ 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 Σ, then add the pair (t,U) to < in Z, i.e. order $t < U$.

Define the partial function sort: $V \cup T \rightarrow$ Sorts of $Z \cup$ VarSorts, where VarSorts is the set of sorts of the variable terminals in G, by:

sort(t) = T if t is a variable-terminal of sort T,
sort (t) = t if t was defined in Step 1,
sort (N) = $\mathcal{N}$ if N is a nonterminal.

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: $\mathcal{N}$
to Z.

For every production of the form $N \rightarrow f(x_1,...x_n)$, $n > 0$, add the operator
f: sort(x_1) x ... x sort(x_n)$\rightarrow \mathcal{N}$
to Z.

For every production of the form $N \rightarrow N'$ where N' is a nonterminal, add the pair
($\mathcal{N}'$,$\mathcal{N}$)
to the relation < in Z, i.e. order $\mathcal{N}' < \mathcal{N}$.

Step 3 (combine Z and Σ)

Let Σ' be the union of Z and Σ.

Step 4 (ensure regularity)

For each n-ary operator f, $n \geq 0$, in Σ', for each pair of arities
f: s_1 x ... x $s_n \rightarrow t$
f: s'_1 x ... x $s'_n \rightarrow t'$
with $\sim(t' \leq t \vee t \leq t')$

for each sequence of sorts $<u_1,...,u_n>$ such that
for all i=1,...,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 GLB(t,t') to Σ',
add the pairs (GLB(t,t'), t) and (GLB(t,t'), t') to the relation < in Σ',
if for some r we have r < t and r < t',
then add (r,GLB (t,t')) to the relation < in Σ',
add a new arity f: u_1 x ... x $u_n \rightarrow$ GLB(t,t') to Σ'.

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

Step 5 (ensure monotonicity)

For each n-ary operator f, $n \geq 0$, in Σ', for each ordered pair of arities
f: s_1 x ... x $s_n \rightarrow t$
f: s'_1 x ... x $s'_n \rightarrow t'$
if for all i=1,...,n $s_i \geq s'_i$, then:

if t'>t then delete the arity f: $s'_1 \times \ldots \times s'_n \to t'$ from Σ'.

Step 6 (remove redundant sorts)

For every sort s in Σ', if s does not occur in an operator arity, then delete s from Σ'.

5. An Example

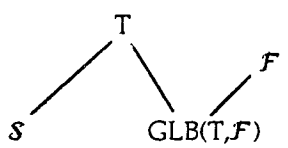
Consider the language of the varying parts from the example presented in §2: the set of terms {g(f(x)), g(g(f(x))),...}. A grammar for this language is

$S \to g(S) \mid g(F)$
$F \to f(x)$

The result of the algorithm is:

	<	Sorts	F
Step 1			
	$\mathcal{S} < T$	$T, \mathcal{S}, \mathcal{F}$	
Step 2			
			g: $\mathcal{S} \to \mathcal{S}$
			g: $\mathcal{F} \to \mathcal{S}$
			f: $T \to \mathcal{F}$
Step 3			
			g: $T \to T$
			f: $T \to T$ *
			c: T
Step 4			
	$GLB(T,\mathcal{F}) < T$	$GLB(T,\mathcal{F})$	f: $T \to GLB(T,\mathcal{F})$
	$GLB(T,\mathcal{F}) < \mathcal{F}$		
Step 5		(delete *)	

Result:



$(T_{\Sigma'})_{\mathcal{S}}$ = {g(f(x)), g(g(f(x))),...}
= language of start symbol S

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

i) a complete, confluent rewrite set;
ii) a conservative extension of R^{∞}.

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 Σ 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 Σ.

References

[AS83] D. Angluin, C.H. Smith, A Survey of inductive inference: theory and methods, Computing Surveys 15, 3, pp.237-269, 1983.

[Co89] H. Comon, Inductive Proofs by Specification Transformation, in Proc. Rewriting Techniques and Applications, 3rd. Intl. Conference, Lecture Notes in Computer Science 355, Springer-Verlag, 1989.

[CW90] D. Cohen, P. Watson, An efficient representation of arithmetic for term rewriting, submitted for publication, 1990.

[GG89] S.J. Garland, J.V. Guttag, An Overview of LP, The Larch Prover, Proc. Rewriting Techniques and Applications, 3rd. Intl. Conference, Lecture Notes in Computer Science 355, Springer-Verlag, 1989.

[He88] M. Hermann, Vademecum of Divergent Term Rewriting Systems, CRIN 88-R-082, Centre de Recherche en Informatique de Nancy, 1988.

[Ho 85] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[HU79] J. Hopcroft, D. Ullman, Introduction to automata theory, languages, and computation, Addison-Wesley, 1979.

[KB70] D. Knuth, P. Bendix, Simple word problems in universal algebra, in J. Leech, ed., Computational Problems in Abstract Algebra, Pergamon Press, 1970.

[Ki87] H. Kirchner, Schematization of infinite sets of rewrite rules. Application to the divergence of completion processes, in Proc. Rewriting Techniques and Applications, P. Lescanne (ed.), Lecture Notes in Computer Science 256, Springer-Verlag, pp.180-191, 1987.

[KN90] C. Kirkwood, K. Norrie, Some Experiments using Term Rewriting Techniques for Concurrency, submitted for publication, 1990.

[KS84] D. Kapur, G. Sivakumar, Architecture of and experiments with RRL, a rewrite rule laboratory, in Proceedings of an NSF workshop on the Rewrite Rule Laboratory, 1983, GERDC Report 84GEN008.

[La89] St. Lange, Towards a Set of Inference Rules for Solving Divergence in Knuth-Bendix Completion, Proc. Analogical and Inductive Inference '89, GDR, Lecture Notes in Computer Science 367, Springer-Verlag, 1989.

[MR90] B. Matthews, S. Robinson, Parsing Misfix Operators, in preparation, 1990.

[Mi 80] R. Milner, A Calculus of Communication Systems, Lecture Notes in Computer Science 92, Springer-Verlag, 1980.

[Pa85] P. Padawitz, Parameter Passing Data Type Specifications, in Tapsoft '85, Berlin, Lecture Notes in Computer Science 185, Springer-Verlag, 1985.

[Re90] A. Reid, Designing Data Structures, in Functional Programming, Proc. of 1989 Glasgow Workshop, Springer Workshops in Computing, Springer-Verlag, 1990.

[Sa73] A. Salomaa, Formal Languages, Academic Press, 1973.

[SNGM87] G. Smolka, W. Nutt, J.A. Goguen, J. Meseguer, Order-Sorted Equational Computation, SEKI Report SR-87-14, Universität Kaiserslautern, FRG, 1987.

[Th87] M. Thomas, The Imperative Implementation of Algebraic Data Types, Ph.D. thesis, St. Andrews University, 1987.

[Th89] M. Thomas, From 1 Notation to Another One: An ACT-ONE Semantics for ASN.1, Formal Description Techniques II, S.T. Vuong (ed.), North Holland 1990.

[TJ89] M. Thomas, K.P. Jantke, Inductive Inference for Solving Divergence in Knuth-Bendix Completion, Proc. Analogical and Inductive Inference '89, GDR, Lecture Notes in Computer Science 367, Springer-Verlag,1989.

[TW90] M. Thomas, P.Watson, Solving Divergence in Knuth-Bendix Completion by Enriching Signatures, submitted for publication.

[WD89] P. Watson, A.J.J. Dick, Least Sorts in Order-Sorted Term Rewriting, Technical Report TR-CSD-606, Royal Holloway and Bedford New College, University of London, 1989.

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 (TGI), Universität Hamburg, Rothenbaumchaussee 67/69, D-2000 Hamburg 13.
Sponsors: Special Interest Group for Theoretical Computer Science of the GESELLSCHAFT für INFORMATIK (GI), and the Special Interest Group for Applied Mathematics of AFCET.

Date: **March 3- 6**
Title: **Fifth International Workshop on High-Level Synthesis**
Location: Beulerhoehe, 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

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@dit.upm.es

Date: **November 26 - 29**
Title: **Conference on Software Maintenance**
Location: San Diego, Calif., USA.
Sponsors: IEEE Computer Society, TC on Software Engineering, and the Institute of Electrical and Electronics Engineers, Inc. in cooperation with ACM SIGSOFT, NIT and the Software Maintenance Association.
Contact: Vacoav 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 Adminstration, 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.
Local Organiser: Ms Rosalind Barden, Logica Cambridge Limited, Betjemin House, Cambridge CB2 1LQ, England. Tel. +44 -(0) 223 66343.
Email: rosalind@uk.co.logcam.

Date: **March 4 - 8**
Title: **International Conference & Exhibition CNIT (Tools '91)**
Location: Paris, France.
Acronym: TOOLS '91
Contact: Program Chairman: Jean Bézivin, Conference Chairman: Bertrand Meyer. TOOLS '91, SOL, 14 rue Jean Rey, 75015 Paris, France. Tel. (+33)-1-40 56 03 58, Fax. (+33)-1-40 56 05 81.
Paper Submission Details:
Attn: Jean Bézivin, Laboratoire d'Informatique, Faculté des Sciences et Techniques, Université de Nantes, 2 rue de la Houssinière, 44072 Nantes Cedex, France.
Email: uunet!geocub.greco-prog.fr!bezivin (from the US) or geocub.greco-prog.fr!bezivin (from Europe).

Date: **March 26 - 28**
Title: **7th British Colloquium for Theoretical Computer Science**
Location: Liverpool, England.
Acronym: BCTCS 7
Contact: Paul E Dunne (BCTCS 7), Department of Computer Science, University of Liverpool, Liverpool L69 3BX, Great Britain.
Email: ped@uk.ac.liv.cs.and

Date: **April 8 - 12**
Title: **TAPSOFT'91 Fourth International Joint Conference on the Theory and Practice of Software Development**
Location: Brighton, UK
Acronym: TAPSOFT'91
Contact: TAPSOFT'91, PPL Conference Services, 2 Savoy Hill, London WC2R 0BL. Tel. 071 240 1871, Fax. 071 497 3633.

Date: **April 10 - 12**
Title: **Fourth International Conference on Rewriting Techniques and Applications**
Acronym: RTA 91
Location: Como, Italy.
Sponsors: University of Milan, EATCS, IEEE, TC on MFCS, ACM, SIGACT, SIGART.
Contact: Professor G Degli Antoni, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Moretto da Brescia, 9, I-20133 Milano, Italy. Telephone: Tel. (+39)-02-7575-201/209.
Email: gdantoni@imisiam.bitnet.

Date: **April 21 - 24**
Title: **ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming**
Acronym: PPoPP 91.
Location: Hilton Hotel, Williamsburg, Virginia, USA.
Sponsor: SIGPLAN.
Contact: Dennis Gannon, Dept. of Computer Science, Indiana Univ., 101 Lindley Hall, Bloomington, IN, 47401. Tel. (812) 855-5184.
Email: gannon@iuvax.cs.indiana.edu.

Date: **May 5 - 10**
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).

Date: **May 6 - 8**
Title: **ACM Symposium on Theory of Computing**
Acronym: STOC '91
Location: New Orleans, Louisiana, USA.
Contact: Cris Koutsougeras, Computer Science Dept., Tulane University, New Orleans, LA 70118.
Email: cris@rex.cs.tulane.edu.
Or Contact: Jeff Vitter, 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.

Date: **May 7 - 10**
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: brglez@mcnc.org.

Date: **May 13 - 16**
Title: **13th International Conference on Software Engineering**
Location: Austin, Texas, USA.
Contact: ICSE13, Laszlo A. Belady, MCC, PO BOX 200195, Austin, Texas 78720 USA. Tel. (512) 338-3356.
Email: belady@mcc.com.
Or: ICSE 13, IEEE Computer Society, 1730 Massachusetts Ave., NW, Washington, DC, 20036 USA. Tel. (202) 371-1013.

Date: **May 17 - 18**
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@dcsnet.uwf.edu.

Date: **May 21 - 23**
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.

Date: May 22 - 24
Title: **Second International Conference on Algebraic Methodology and Software Technology, AMAST**
Location: Iowa City, Iowa, USA.
Contact: Professor Eugene Madison, University of Iowa, Iowa City, IA, USA.

Date: May 28 - 29
Title: **The 5th Israel Conference on Computer Systems and Software Engineering**
Location: Herzelia, Israel.
Sponsor: IEEE CS.
Contact: Conference Secretariat, c/o OFTRA Ltd., P.O. Box 50432, Tel Aviv, 61500, Israel. Tel. 972-3-664825. Fax. 972-3-660952.

Date: June 10 - 13
Title: **Conference on Parallel Architectures and Languages Europe**
Location: Congress and Meeting Centre "De Koningshof", PO BOX 140, 5500 AC Veldhoven, The Netherlands.
Acronym: PARLE '91.
Contact: Mr F Stoots, Philips Research Laboratries, PO BOX 80.000, 5600 JA Eindhoven, The Netherlands. Fax. +31 40 744748.
Email: stoots@dooma.prl.philips.nl

Date: June 12 - 14
Title: **3rd International Software Configuration Management Workshop**
Location: Norwegian Inst. of Tech., Trondheim, Norway.
Sponsor: SIGSOFT and IEEE Computer Society.
Contact: Reidar Conradi, Division of Computer Systems and Telematics (DCST) Norwegian Inst. of Tech., N-7034. Tel. 47 7 593444.
Email: conradi@idt.unit.no.

Date: June 24 - 28
Title: **SIGPLAN Conference on Programming Languages Design and Implementation**
Acronym: SIGPLAN '91
Location: Madison, Wisconsin, USA.
Sponsor: ACM SIGPLAN.
Contact: Jorgen Knudsen, Aarhus Univ., Computer Science Dept., Nu Munkegade 116, DK-800 Aarhus, Denmark.
Email: jlknudsen@daimi.dk.

Date: June 25 - 27
Title: **The Twenty-First Annual International Symposium on Fault-Tolerant Computing**
Acronym: FTCS 21
Location: Québec, Canada.
Sponsor: IEEE Computer Society.
Contact: (+ Submissions) Prof. Eduard Cerny, Université de Montréal, Dép. d'Informatique et de recherche opérationnelle, P.O. Box 6128, St. A Montréal, Québec, Canada H3C 3J7. (Envelope to be marked "FTCS-21 submission").

Date: June 25 - 28
Title: **Eighth International Conference on Logic Programming**
Acronym: ICLP'91
Location: Paris, France.
Contact: Koichi Furukawa, ICOT, Mita Kokusai Building, 21 F4 - 28 Mita 1, Chome, Minato-ku, Japan 108, Japan.
Email: furukawa@icot.or.jp.
Submission details:
6 copies of the full contribution which must not exceed 5000 words (approximately 20 pages double space). The front page should include name(s) of the author(s), affiliation, complete address, telephone, telex, fax, e-mail.

Date: June 26 - 28
Title: **Twelfth International Conference on Application and Theory of Petri Nets**
Location: Aarhus, Denmark.
Contact: (Programme Committee Chairman) Manuel Silva, Departamento Ingeniería, Eléctrica e Informática, C/María de Luna, 3 (Actur) E-50015 Zaragoza, Spain. Tel. +34 76 51 72 74.
(Organizing Committee Chairman) Kurt Jensen, Computer Science Department, Aarhus University, Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark. Tel. +45 86 12 71 88.
Email: icpn@daimi.aau.dk.

Date: June 27 - 29
Title: **3rd International Conference on Software Engineering and Knowledge Engineering**
Location: Skokie, Ill., USA.
Sponsor: Knowledge System Inst., Inst. for Information Industry, SWIFT, and Univ. of Pittsburgh.
Contact: W.D. Hurley, Dept. of Comp. Sci., Alumni Hall, Univ. of Pittsburgh, Pittsburgh, PA 15260; (412) 624-8843.
Email: hurley@cs.pitt.edu.

Date: July 1 - 3
Title: **International Conference on Computing 1991**
Acronym: COMP 1991
Location: Imperial College of Science and Technology, London, United Kingdom.
Sponsors: IEEE in coop. with SIGSOFT, Charles Babbage Institute (Minneapolis), IFAC, British Computer Society, IEEE-CS, British Association for the Advancement of Science, British Society for the History of Science. The Royal Society, Institution of Electronic and Radio Engineers.
Contact: General Chair, E. Ash CBE FRS, Imperial College of London, Exhibition Road, London SW7 2AZ, United Kingdom. Tel. +44 1 589 5111.

Date: July 8 - 12
Title: **18th International Colloquium on Automata, Languages, and Programming**
Acronym: ICALP '91
Location: Departamento de Informática y Automática, Facultad de Matemáticas, Universidad Complutense, Av. Complutense s/n, 28040 Madrid, Spain.
Contact: Prof. Mario Rodríguez Artalejo.
Email: W450@EMDUCM11.BITNET.

Date: **October 21 - 24**
Title: **Third European Software Engineering Conference**
Contact: Alfonso Fugetta, CEFRIEI, c/o AICA-ESEC '91, P.le Rodolfo Morandi 2, 1.2021 Milan, Italy.
Email: alfonso@imicefr.bitnet.
Acronym: ESEC 91
Location: Milano, Italy.
Sponsors: AFCET et al.
Paper Submission Details:
Submit six copies of full paper and abstract by **Jan. 15, 1991**, to Alex van Lamsweerde, Unite, 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 1, 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.
Contact: Carlo Ghezzi, Dip. di Elettronica Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milano, Italia.
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@trl.oz.au.

1992

Date: **January 20 - 24**
Title: **19th ACM Symposium on Principles of Programming Languages**
Acronym: POPL '92
Location: Albequerque, 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: meyer@lcs.mit.ecu.

Date: **September 3 - 6**
Title: **Category Theory and Computer Science**
Location: Ecole Normale Supérieure, Paris, France.
Contact: David Pitt, Department of Mathematics, University of Surrey, Guildford, Surrey GU2 XH, 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. Chrzastwoski-Wachtel and A. Tarlecki, MFCS'91, Institute of Computer Science, Polish Academy of Sciences, PKiN, 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.
Organisers: 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 Senter, Phoenix, Ariz., USA.
Sponsor: SIGPLAN.
Contact: John Richards, IBM TJ Watson Research Ctr., PO BOX 704, Yorktown Heights, NY 10598; (914) 784-7731.
Email: jtr@ibm.com.