Special Issue: SAICSIT '99
The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

World-Wide Web: http://www.cs.up.ac.za/sacj/

Editor
Prof. Derrick G. Kourie
Department of Computer Science
University of Pretoria, Hatfield 0083
dkourie@cs.up.ac.za

Sub-editor: Information Systems
Prof. Niek du Plooy
Department of Informatics
University of Pretoria, Hatfield 0083
nduplooy@econ.up.ac.za

Production Editors
Andries Engelbrecht
Department of Computer Science
University of Pretoria, Hatfield 0083

Herna Viktor
Department of Informatics
University of Pretoria, Hatfield 0083
sacj_production@cs.up.ac.za

Editorial Board

Prof. Judith M. Bishop
University of Pretoria, South Africa
jbishop@cs.up.ac.za

Prof. Richard J. Boland
Case Western University, U.S.A.
boland@spider.cwry.edu

Prof. Trevor D. Crossman
University of Natal, South Africa
crossman@bis.und.ac.za

Prof. Donald. D. Cowan
University of Waterloo, Canada
dcowan@csg.uwaterloo.ca

Prof. Jürg Gutknecht
ETH, Zürich, Switzerland
gutknecht@inf.eth.ch

Prof. R. Nigel Horspool
University of Victoria, Canada
nigelh@csr.csc.uvic.ca

Prof. Fred H. Lochovsky
University of Science and Technology, Hong Kong
fred@cs.ust.hk

Prof. Kalle Lyytinen
University of Jyväskylä, Finland
kalle@cs.jyu.fi

Dr. Jonathan Miller
University of Cape Town, South Africa
jmiller@geb2.uct.ac.za

Prof. Mary L. Soffa
University of Pittsburgh, U.S.A.
soffa@cs.pitt.edu

Prof. Basie H. von Solms
Rand Afrikaanse Universiteit, South Africa
basie@rkw.rau.ac.za

Subscriptions

Southern Africa
Annual R80.00
Single copy R40.00
Elsewhere
US$40.00
US$20.00

An additional US$15 per year is charged for airmail outside Southern Africa.

+27 (11) 315-1319 Fax: +27 (11) 315-2276
Contents

Preface
  P. Machanick ............................................................... 1

Research Articles
Active Learning: Issues and Challenges for Information Systems and Technology
  RD Quilling, GJ Erwin and O Petkova ........................................ 5

A Generic Modelling Framework for Interactive Authoring Support Environments
  Paula Kotze ............................................................... 15

  O Petkova and JD Roode ....................................................... 26

An Information-Theoretic Semantics for Belief Change
  T Meyer ............................................................... 33

A Complexity Metrics Model for Software Correction
  A Törn, T Andersson and K Enholm ........................................ 40

A Conceptual Design for High-Volume Data Processing of Warehouse Database into Multidimensional Database
  Paisarn Trakulsuk and Vichit Avatchanakorn ........................................ 49

A Pragmatic Approach to Bitemporal Databases: Conceptualization, Representation and Visualisation
  Chiyaba Njovu and WA Gray ....................................................... 58

A Building Recognition System
  SP Levitt and B Dwolatzky ....................................................... 68

Computer Programming and Learning to Write
  John Barrow ............................................................... 77

Co-operating to Learn using JAD Technologies
  TA Thomas ............................................................... 87

Critical Success Factors for the Implementation of DSS at a Selection of Organisations in Kwazulu/Natal
  URF Averweg and GJ Erwin ....................................................... 95

Enhancing the Predictability of Two Popular Software Reliability Growth Models
  Peter A Keiller and Thomas A Muzzuchi ........................................ 105
<table>
<thead>
<tr>
<th>Title</th>
<th>Authors</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalised Unification of Finite Temporal Logic Formulas</td>
<td>Scott Hazelhurst</td>
<td>110</td>
</tr>
<tr>
<td>Harmonizing Global Internet Tax: A Collaborative Extranet Model</td>
<td>E Lawrence and B Garner</td>
<td>119</td>
</tr>
<tr>
<td>Improving Object Oriented Analysis by Explicit Change Analysis</td>
<td>Lui Yu, Siew Chee Kong, Yi Xun and Miao Yuan</td>
<td>128</td>
</tr>
<tr>
<td>Reconciling the Needs of New Information Systems Graduates and Their Employers in Small, Developed Countries</td>
<td>Rodney Turner and Glenn Lowry</td>
<td>136</td>
</tr>
<tr>
<td>Shortest Delay Scheduling Algorithm for Lossless Quality Transmission of Stored VBR Video under Limited Bandwidth</td>
<td>Fei Li, Yan Liu, Jack Yiu-Bun Lee and Ishfaq Ahmad</td>
<td>146</td>
</tr>
<tr>
<td>Software Croma Keying in an Immersive Virtual Environment</td>
<td>Frans van der Berg and Vali Lalioti</td>
<td>155</td>
</tr>
<tr>
<td>Some Automata-Theoretic Properties of ( \exists )-NFA</td>
<td>Lynette van Zijl and Andries PJ van der Walt</td>
<td>163</td>
</tr>
<tr>
<td>The CILT Multi-Agent Learning System</td>
<td>Hema L Viktor</td>
<td>176</td>
</tr>
<tr>
<td>The Development of a Generic Framework for the Implementation of Cheap, Component-Based Virtual Video-Conferencing System</td>
<td>Soteri Panagou and Shaun Bangay</td>
<td>185</td>
</tr>
<tr>
<td>The Role of Experience in User Perceptions of Information Technology: An Empirical Examination</td>
<td>Melia Handzic and Graham Low</td>
<td>194</td>
</tr>
<tr>
<td>What are Web Sites Used for: Cost Savings, Revenue Generating or Value Creating?</td>
<td>Man-Ying Lee</td>
<td>201</td>
</tr>
<tr>
<td>New Ideas Papers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approaches to Video Transmission over GSM Networks</td>
<td>Bing Du and Anthony Maeder</td>
<td>210</td>
</tr>
<tr>
<td>From Information Security Baselines to Information Security Profiles</td>
<td>Rossouw von Solms and Helen van der Haar</td>
<td>215</td>
</tr>
<tr>
<td>Grounded Theory Methodology in IS Research: Glaser versus Strauss</td>
<td>J Smit</td>
<td>219</td>
</tr>
<tr>
<td>Introducing a Continuum of Abstraction-Led Hierarchical Search Techniques</td>
<td>Robert Zimmer and Robert Holte</td>
<td>223</td>
</tr>
</tbody>
</table>
Multimedia as a Positive Force to Leverage Web Marketing, with Particular Reference to the Commercial Sector

Stan Shear ................................................................. 229

Understanding HCI Methodologies

Peter Warren ............................................................... 234

Electronically Published Papers

Experience Papers
A Java Client/Server System for Accessing Arbitrary CANopen Fieldbus Devices via the Internet
Dieter Bühler, Gerd Nusser, Gerhard Gruhler and Wolfgang Küchlin ............................................ 239

An Object-Oriented Framework for Rapid Client-side Integration of Information Management Systems
Ralf-Dieter Schimkat, Wolfgang Küchlin and Rainer Krautter ......................................................... 244

Distributed Operating Systems: A Study in Applicability
Jürgen Prange and Judith Bishop ...................................................... 249

Formal Verification with Natural Language Specifications: Guidelines, Experiments and Lessons so far
Alexander Holt ............................................................... 253

Introducing Research Methods to Computer Science Honours Students
Vashti Galpin, Scott Hazelhurst, Conrad Mueller and Ian Sanders ................................................. 258

Visualising Eventuality Structure
ST Rock .......................................................... 264

Electronically Published Papers
SAICSIT'99

South African Institute of Computer Scientists and Information Technologists

Annual Research Conference 17-19 November 1999

Prepare for the New Millennium

Is there life after y2k?

Mount Amanzi Lodge, Hartebeespoort, South Africa

Sponsors

http://www.cs.wits.ac.za/

PHDS research/PHDS.html

Think different.
Preface

Philip Machanick, Overall Chair: SAICSIT'99

Running SAICSIT'99, the annual research conference of the South African Institute for Computer Scientists and Information Technologists, has been quite an experience.

SAICSIT represents Computer Science and Information Systems academics and professionals, mainly those with an interest in research. When I took over as SAICSIT president at the end of 1998, the conference had not previously been run as an international event. I decided that South African academics had enough international contacts to put together an international programme committee, and a South African conference would be of interest to the rest of the world.

I felt that we could make this transition at relatively low cost, given that we could advertise via mailing lists, and encourage electronic submission of papers (to reduce costs of redistributing papers for review).

The first prediction turned out to be correct, and we were able to put together a strong programme committee.

As a result, we had an unprecedented flood of papers: 100 submitted from 21 countries. As papers started to come in, it became apparent that we needed more reviewers. It was then that the value of the combination of old-fashioned networking (people who know people) and new-fashioned networking (the Internet) became apparent. While the Internet made it possible to convert SAICSIT into an international event at relatively low cost, the unexpected number of papers made it essential to find many additional reviewers on short notice. Without the speed of e-mail to track people down and to distribute papers for review, the review process would have taken weeks longer, and it would have been much more difficult to track down as many new reviewers in so little time.

Even so, the number of referees who were willing to help on short notice was a pleasant surprise.

The accepted papers cover an interesting range of subjects, from management-interest Information Systems, to theoretical Computer Science, with subjects including database, Java, temporal logic and implications of e-commerce for tax.

In addition, we were very fortunate in being able invite the president of the ACM, Barbara Simons as a keynote speaker. Consequently, the programme for SAICSIT'99 should be very interesting to a wide range of participants.

We were only able to find place in the proceedings for 36 papers out of the 100 submitted, of which only 24 are full research papers. While this number of papers is in line with our expectation of how many papers would be accepted in each category, we did not have a hard cut-off on the number of papers, but accepted all papers which were good enough, based on the reviews. Final selection was made by myself as Programme Chair, and Derrick Kourie, as editor of the South African Computer Journal. Additional papers are published via the conference web site.

We believe that we have put together a quality programme, and hope you will agree.

Acknowledgments

I would like to thank the South African Computer Journal production team, Andries Engelbrecht and Herna Viktor, respectively from the Department of Computer Science and Informatics, University of Pretoria, for their work on producing the proceedings.

The reviewers listed overleaf did an excellent job: many wrote very detailed reports, sometimes after being called in on very short notice. Inevitably, there were some glitches resulting from the unexpected workload, but the buck stops with the programme chair: I promise to do better next time.

I would also like to thank my own department for putting up with the extra work and expense that running a conference entails. I tried not to burden them with too much extra work, but our secretaries, Zalm Gowar and Leanne Reddy, inevitably had to take on some extra work. John Ostrowick provided valuable assistance with design of our web pages and call for papers poster. Carol Kernick, who handles our finances and membership records, did a fine job of keeping up with the demands of the conference.

Finally, I would like to thank our sponsors, whose contribution made this conference been possible:

- PricewaterhouseCoopers – sponsored generous prizes and the conference banquet
- National Research Foundation (NRF) – provided financial support
- University of the Witwatersrand – provided financial support
- Programme for Highly Dependable Systems, University of the Witwatersrand – provided financial support
- Standard Bank – provided financial support

SART/SACJ, No 24, 1999
Editorial

- Apple Computer – provided equipment for the conference
- Qualica – provided technical support including helping with the conference web site

Web Site
For more information about SAICSIT, including a pointer to the conference site, see <http://www.saicsit.org.za>.

Referees
- Department of Computer Science, University of Pretoria
  – Derrick Kourie
  – Bruce Watson
  – Vali Lalioti
  – Andries Engelbrecht
  – Ivan Mphahlele
- German National Research Center for Information Technology - GMD
  – Gernot Goebbels
- School of Math Stat and IT, Natal University Pietermaritzburg
  – Peter Warren
- Graduate School of Business, University of Cape Town
  – Kurt April
- CSIR
  – James Jardine
- Department of Computer Science, Rhodes University
  – Shaun Bangay
  – Peter Clayton
  – John Ebden
  – Richard Foss
  – George Wells
  – Peter Wentworth
- Information Technology Division, Rhodes University
  – Caro Watkins
- Department of Information Systems, Rhodes University
  – Brenda Mallinson
- Department of Informatics, University of Pretoria
  – Carina de Villiers
  – Herna Viktor
  – Niek du Plooy
  – Elsje van Rooyen
  – Machdel Matthee
  – Alan Abrahams
  – Jackie Phahlamohlaka
- School of Information Technology & Engineering, University of Ottawa
  – Stan Szpakowicz
  – Dwight Makaroff
- Department of Information Systems, Victoria University
  – Glenn Lowry
  – Peter Shackleton
  – Tas Adam
  – Alastair Wallace
  – Stephen Burgess
  – Dave Burgess
  – Julie Fisher
  – Jerzy Lepa
  – Geoff Sandy
  – Rod Turner
  – Alastair Wallace
  – Andrew Wenn
- Peninsula School of Computing and Information Technology, Monash University
  – Ainslie Ellis
- School of Information Management and Systems, Monash University
  – Angela Carbone
- School of Computer Science & Software Engineering, Monash University
  – Dianne Hagan
  – Judy Sheard
- School of Management Information Systems, Deakin University
  – John Lamp
  – David Mackay
  – Philip Joyce
- Marie van der Klooster
- Bill Hewett
- Jennie Carroll
- Rodney Carr

• Computer and Automation Research Institute, Hungarian Academy of Sciences
  - Ferenc Vajda

• Department of Computer Science, University of the Witwatersrand
  - Ian Sanders
  - David Lubinsky
  - Conrad Mueller
  - Yinong Chen
  - Bob Baber
  - Vashti Galpin
  - András Salamon
  - Scott Hazelhurst
  - Philip Machanick
  - Zoltán Fazekas

• Department of Electrical Engineering, University of the Witwatersrand
  - Barry Dwolatzky
  - Farzin Aghdasi

• Department of Mathematics, University of the Witwatersrand
  - Jonathan Burgess

• School of Law, University of the Witwatersrand
  - Victoria Bronstein

• Edward Nathan & Friedland Attorneys, Johannesburg
  - Justine White

• Department of Information Systems & Technology, University of Durban-Westville
  - Geoff Erwin

• Department of Computer Science, Clemson University, Clemson, SC
  - Karen Hay

• Logical SA
  - Philip Green

• Department of Computer Science, University of Bristol
  - Alan Chalmers

• Department of Systems and Computer Science, Howard University, Washington, D.C
  - Todd Shurn
  - John Trimble

• Department of Computer Science, University of Cape Town
  - Sonia Berman

• King’s College, London
  - Peter Wood

• Department of Information Systems, University of Cape Town
  - Paul Licker
  - Steve Erlank
  - Alemayehu Molla

• Department of Department of Mathematics and Applied Mathematics, University of Cape Town
  - Ingrid Rewitzky
  - Peter Jipsen
  - Renkuan Guo

• Department of Information Systems, University of the Western Cape
  - Andy Bytheway

• Department of Electrical Engineering and Computer Science, The George Washington University, Washington, DC
  - Rachelle Heller

• School of Biophysical Sciences and Electrical Engineering, Swinburne University of Technology
  - Ian Macdonald

• Electrical and Electronic Systems Engineering, Queensland University of Technology
  - Anthony Maeder

• Computer Science Department, Åbo Akademi
  - Aimo Törn
  - Kaisa Sere
  - Mats Aspnas

• Division of Informatics, University of Edinburgh
  - Graeme Ritchie
  - Arturo Espinosa-Romero
  - Jon Oberlander
Editorial

- Chris Brew
- Alexander Holt

Department of Computer Science and Engineering, Helsinki University of Technology
- Tommi Junnila
- Nisse Husberg

Technical University of Budapest, Department of Measurement and Information Systems
- Andras Pataricza

NASA-Goddard Space Flight Center
- Nigel Ziyad

Department of Computer Science & Information Systems, University of Port Elizabeth
- Janet Wesson
- Leon Nicholls

Computer Technology Department, Indiana University Purdue University Indianapolis
- Tim Price

Department of Electrical Engineering and Computer Science, The University of Michigan
- Trevor Mudge

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
- David Forsyth

School of Computer Science, University of Birmingham
- Mark Ryan

Faculty of Mathematics, Computer Science, Physics & Astronomy, University of Amsterdam
- Carlos Areces

Department of Computer Science, Universidade do Vale do Rio dos Sinos - UNISINOS Rio Grande do Sul
- Marcelo Walter

School of Information Technology & Mathematical Sciences, University of Ballarat
- Binh Pham
Formal verification with natural language specifications: guidelines, experiments and lessons so far

Alexander Holt

Abstract

The industrial take-up of formal verification techniques remains limited. Allowing specifications to be expressed in natural language (perhaps augmented with diagrams) offers the prospect of increasing the usability of verification tools. We suggest guidelines for the development of such systems, and describe a prototype which provides an English interface to the SMV model checker by translating specification sentences to formulas of temporal logic. Limitations are discussed, and prospects for future development considered.

Keywords: formal verification, natural language specification, model checking

1 Formal verification

The contribution that formal verification of software and hardware can make to system design is now widely recognized. Much work remains to be done, however, to make formal verification techniques computationally feasible for large-scale problems. Further obstacles to the widespread adoption of these techniques are the need for expert users, and the mismatch between currently available verification tools and established design processes.

This paper addresses part of the expert user issue: it investigates the use of natural language as the primary interface in the formulation of system specifications. We do not attempt a survey of the field; instead, guidelines for development are suggested, and specific experiences discussed. In particular, we describe an experimental system which integrates a model checker for temporal logic with a natural language processing application. The core of this system is a convertor that translates (a formal representation of the meaning of) specification sentences written in English to formulas of temporal logic, suitable for input to the model checker. We discuss some of the problems involved, and relate our experiences to the guidelines. Finally, we consider ways in which our system could be further developed and improved.

Techniques for formal verification can be categorized in many ways. From the perspective of industrial application, a relevant distinction is between those approaches that require some knowledge of theorem proving techniques, and those which rely only on the correct expression of specifications in an appropriate formal language. In both cases, we are assumed to possess a formal model of the hardware or software system being verified. The latter category of approaches features model checking as its most well known technique, and model checking forms the basis of the prototype system discussed below.

2 Justification for linguistic and diagrammatic interfaces

Most formal verification approaches require the precise expression of a system's intended properties (one might consider equivalence checkers, for example, to be exceptions). Whether the verification technique involves theorem proving or model checking, a prerequisite is to couch a specification of behaviour in the appropriate formal language—typically a logic. But many target users of verification technology are not logicians, although they may well have clear and precise intuitions about the properties they wish to verify. In the hardware realm, for example, timing diagrams are commonly used to make explicit and objective claims about the temporal behaviour of circuits or components. But timing diagrams are not logic. (At least not as they stand; formalizations are possible, however [2].)

So there is strong motivation to build interfaces that permit hardware and software engineers who are not experts in logic to use verification tools. Since most tools based on theorem proving require the user to acquire some logical expertise for the purpose of driving the prover—not merely expressing the specifications—it is model checking approaches that appear to offer the greatest immediate usability gain from such interfaces. Here we consider the use of interfaces based on natural language, and peripherally, diagrams.

3 Guidelines

A number of ideas have guided the work described here, some based on theoretical considerations and the literature, others on practical experience. What started as internal comments on the development of our own system has evolved into a set of more general guidelines for the deve-
opment of natural language interfaces to verification tools, which is the spirit in which they are presented here. The role of diagrams is also considered to some extent, though much work remains to be done in this area.

1. **Be interactive.** Give appropriate feedback to the user whenever possible, whether about linguistic processing, semantic interpretation or verification results. For example, if there are two plausible readings of a natural language specification, we need to ask the user which is intended. Presenting two logical formulas is not appropriate—instead, one could explain the ambiguity, or back-generate two disambiguated natural language specifications, or perhaps use some diagrammatic device to convey the differing senses.

2. **Use a controlled language.** Although contemporary systems can process largely unrestricted English (if shallowly), it doesn’t make sense to try and do so for the purposes of a verification interface. It’s better that the user acquire a sense of what language the system understands (through interactivity), and that he or she start to learn canonical ways to express specific formal notions with appropriate fragments of natural language. For diagrams, of course, defining a controlled language is the only feasible way to proceed.

3. **Derive linguistic information from the application.** Once a formal model of the system to be verified is available, that can be used to make predications about the words and phrases a user is likely to employ. When verifying a circuit, for example, we can anticipate that the labels used for signals in the circuit will appear as nouns in natural language specifications. (This example is relatively straightforward; in general, satisfying this guideline may be much harder.) As the object of verification changes, derived linguistic data should alter dynamically in correspondence.

4. **Adopt logically-based semantic representations.** Computational linguistics can now offer a number of logically sound representations for the semantic content of a discourse. Since the primary goal of an NL verification interface is to deliver logic specifications, it is wise to use an intermediate representation that already has a logical character.

5. **Justify all semantic conversions.** If the interface is to be trusted, it should be possible to justify all of the conversions it performs internally on objects representing the meaning of specifications, from initial semantic representations to the target logic. Theory-based justifications of its machinery serve to legitimise the interface within a formal verification methodology; they may also support the generation of explanations for the user (a common requirement for expert systems).

6. **Integrate ambiguity resolution with verification engine.** Ambiguity is a key problem in nearly all natural language understanding systems. In this case, it’s crucial to exploit the state and output of the theorem prover or model checker as much as possible, as well as the overall specification context. Faced with multiple interpretations of the input, passing them to the verification engine may reveal that some are more likely than others. This information can be used to generate a plausibility ordering, allowing a minimally disruptive confirmation dialogue with the user.

7. **Take account of design paradigms.** Users won’t want to use a system that doesn’t interoperate with their usual work flows.

### Experiment: English specifications for SMV

#### 4.1 Overview

We have built a prototype system which integrates a natural language understanding component with the freely available SMV model checker program [6]. The result is a tool which allows the formal verification of digital circuits using specifications expressed in English. SMV requires specifications to be written in the temporal logic CTL (computation tree logic). A parser for English, returning general-purpose semantic representations, is allied with a convertor from these representations to CTL. The convertor is connected to the SMV model checker, so that inferential information may be used during semantic interpretation.

For more details of this system, see [5]. Some earlier work with similar goals is reported in [7].

#### 4.2 Computation Tree Logic

The SMV program implements a model checking algorithm where circuit properties are expressed in the temporal logic CTL [1]. In models of CTL, the temporal order < defines a tree which branches towards the future.

**CTL formulas that start with A express necessity:**

- $\text{AG} f$ is true at a time $t$ just in case $f$ is true along all paths that branch forward from the tree at $t$ (true globally).
- $\text{AF} f$ holds when, on all paths, $f$ is true at some time in the future.
- $\text{AX} f$ is true at $t$ when $f$ is true at the next time point, along all paths.

Finally, $\text{A} [f \text{ U } g]$ holds if, for each path, $g$ is true at some time, and from now until that point $f$ is true.

Figure 1 [1, p. 247] illustrates a CTL model structure, with the relation < represented by arrows between circles (states), and atomic propositions being the letters contained in a circle. A CTL structure gives rise to an infinite computation tree, and the figure shows the initial part...
Experience Article

Figure 1: A CTL structure and corresponding computation tree

<table>
<thead>
<tr>
<th>formula</th>
<th>sense</th>
<th>at s₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXc</td>
<td>for all paths, at the next state c is true</td>
<td>true</td>
</tr>
<tr>
<td>AGb</td>
<td>for all paths, globally b is true</td>
<td>false</td>
</tr>
<tr>
<td>AF(AX(a ∧ b))</td>
<td>for all paths, eventually there is a state from which, for all paths, at the following state a and b are true</td>
<td>true</td>
</tr>
</tbody>
</table>

Table 1: Interpretation of CTL formulas

of such a tree when s₀ is the initial state. States correspond to points in the course of a computation, and branches represent non-determinism. Formulas of CTL are either true or false with respect to any given model; see Table 1 for three examples interpreted at s₀.

4.3 System structure

The system consists of four components: (i) a parser, (ii) a converter from semantic representations to CTL, (iii) the SMV model checker, and (iv) a module that mediates interaction between the three others, as well as handling input and output with the user (Figure 2). The system currently runs under Solaris and GNU/Linux, and has a web interface, accessible from http://www.ltg.ed.ac.uk/prosper/.

We used the Alvey Natural Language Tools Grammar [3] to implement a parser for a restricted subset of English. The definition of an appropriate subset of English for this task raised interesting issues of its own [4].

The next component converts the semantic representations produced by the Alvey system into CTL formulas. This part of the system was initially based on work by Danny Tidhar [9].

The SMV model checker is a self-contained program. As well as reading an input file and accepting command line arguments, it has a line-oriented interactive mode, allowing each SMV invocation to check an arbitrary number of CTL formulas against a given circuit description. The system uses UNIX pipes to communicate with SMV, exploiting the per-circuit interactive mode. The result of an SMV verification check is passed back to the coordination module where it can influence further semantic interpretation.

4.4 Example

In (1), (2) and (3) we show respectively a short sentence from our corpus, its Alvey semantics, and the CTL formula to which we convert it:

(1) After sig₁ is active, sig₂ is active for three cycles.

(2) (DECL (AFTER (uqe (some (e₁) (e₁)))) (BE (uqe (some (e₂) (PRES e₂)))) (ACTIVE (name (the (x₁) (and (sg x₁) (named x₁ sig₁))) (degree unknown))) (and (BE #₁=(uqe (some (e₃) (PRES e₃)))) (ACTIVE (name (the (x₂) (and (sg x₂) (named x₂ sig₁)))) (degree unknown))) (FOR #₁ (uq ((NN \3) (x₃) (and (pl x₃) (CYCLE x₃)))) (timespan unknown)))

(3) AG(sig₁ → AX(sig₂ ∧ AXsig₂ ∧ AXAXsig₂))

5 Observations

Our prototype system fails to do justice to all the points listed in §3. It has only limited interactivity, and makes little attempt to fit with common work patterns in hardware engineering. It also requires much extensive testing against English specification data, and a more sophisticated treatment of ambiguity. From a theoretical standpoint, it needs logical justification of its key conversion component. These latter two points are enlarged on below.

So it's not yet ready for exposure to real users. However it's actively under development, and we intend to perform real user testing during 2000.
5.1 Ambiguity

Sentence (4), taken from our corpus of specification discourses, is ambiguous. The two CTL readings which our system assigns are shown in (5) and (5').

(4) After sig1 becomes active sig2 should not become active until sig3 becomes active.

(5) $A[(\text{sig}1 \rightarrow AX(\neg\text{sig}2)) \cup \text{sig}3]$
(5') $AG(\text{sig}1 \rightarrow AXA[\neg\text{sig}2 \cup \text{sig}3])$

We can use SMV to test the truth of these readings for the circuit being specified, and rank false readings according to the length of the computation tree that SMV requires in order to generate a counter-example. However, it would be desirable to find a metric that is more likely to match the user's actual intentions. We are under no illusions about the difficulty of obtaining—and justifying—such a metric.

5.2 Semantic conversions

The process of converting the semantic representations produced by the Alvey system to CTL formulas has some interesting semantic aspects. For example, temporal information must be converted from first-order relations over Davidsonian event variables—the e1, e2 and e3 of example (2)—to modal operators. It is also necessary to deal with the unscoped nature of the Alvey semantics. At present we do not have a fully worked-out logical account of this conversion.

6 Prospects

We are considering the adoption of a higher-order logic formalism for our semantic representations. This would permit new kinds of inference prior to model checking, and would also allow a more natural conversion from the output of the parser, since the most direct logical reading of the Alvey semantics requires second-order predicates.

The SMV program can only provide our semantic interpretation process with limited information about the success or failure of a model checking attempt. It would be interesting to experiment with model checkers that make a richer perspective available—for example, game theoretic approaches [8].

It should be possible to mix language and diagrams appropriately. This is what people do naturally, after all.

References


Acknowledgements

This work has been carried out within PROSPER (Proof and Specification Assisted Design Environments), ESPRIT Framework IV LTR 26241, http://www.dcs.gla.ac.uk/prosper/. Ewan Klein and Claire Grover are major contributors to the prototype system described above, and I’m grateful to them for many fruitful discussions of the ideas presented here. The anonymous referees made a number of constructive suggestions.
Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines:

- Use wide margins and 1½ or double spacing.
- The first page should include:
  - the title (as brief as possible)
  - the author’s initials and surname
  - the author’s affiliation and address
  - an abstract of less than 200 words
  - an appropriate keyword list
  - a list of relevant Computing Review Categories
  - Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author’s surname, and should be cited in the text according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page, http://www.cs.up.ac.za/sacj, which gives a number of examples.

SACJ is produced using the \LaTeX\ document preparation system, in particular \LaTeX\ 2e. Previous versions were produced using a style file for a much older version of \LaTeX, which is no longer supported.

Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

Proofs

Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book Reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.