HARDWARE, SOFTWARE AND PEOPLEWARE

UNISA

SAICSIT 2001

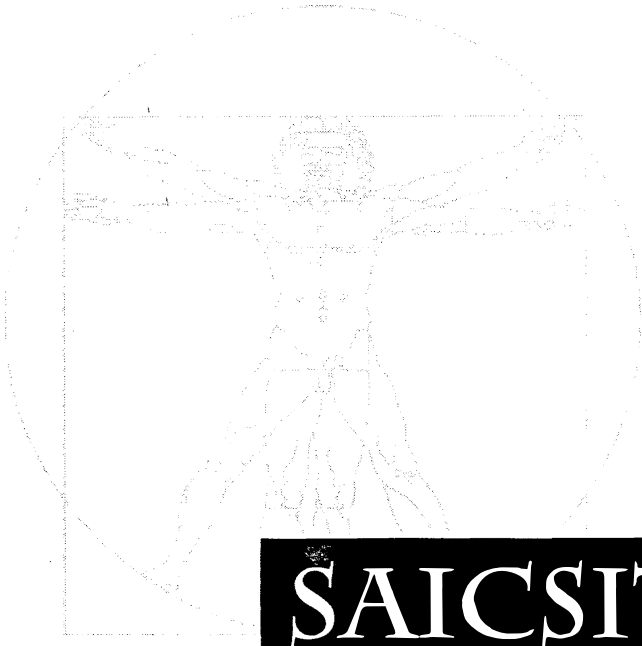
Edited by
Karen Renaud
Paula Kotzé
Andries Barnard

HARDWARE, SOFTWARE AND PEOPLEWARE

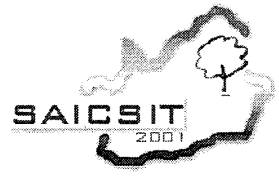
South African Institute of Computer Scientists and Information Technologists Annual Conference

25 – 28 September 2001

Pretoria, South Africa



SAICSIT 2001



Edited by Karen Renaud, Paula Kotzé & Andries Barnard

University of South Africa, Pretoria

Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists

First Edition, First Impression
ISBN: 1-86888-195-4

Published by Unisa Press
University of South Africa
P O Box 392, Pretoria, 0003

Cover Design by Tersia Parsons

Editors: Karen Renaud, Paula Kotzé & Andries Barnard

Electronic Publication by the Editors

Printed by Unisa Press
2001

Table of Contents

Abstracts of Electronic Papers

Message from the SAICSIT President

The South African Institute of Computer Scientists and Information Technologists (SAICSIT) was formed in 1982 and focuses on research and development in all fields of computing and information technology in South Africa. Now in the 20th year of its existence, SAICSIT has come of age, and through its flagship series of annual conferences provides a showcase of not only the best research from the Southern-African region, but also of international research, attracting contributions from far afield. SAICSIT does, however, not exist or operate in isolation.

More than 50 years have passed since the first electronic computer appeared in our society. In the intervening years technological development has been exponential. Over the last 20 years there has been a vast growth and pervasiveness of computing and information technology throughout the world. This has led into the expansion and consolidation of research into a diversity of new technologies and applications in diverse cultural environments. During this period huge strides have also been made in the development of computing devices. The processing speed of computers has increased thousand-fold and memory capacity from megabytes to gigabytes in the last decade alone. The Southern African region did not miss out on these developments.

It is hardly possible for such quantitative expansion not to bring a change in quality. Initially computers had been developed mainly for purposes such as automation for the improvement of processing, labour-reduction in production and automation control of machinery, with artificial intelligence, which made great strides in the 1980s, seen as the ultimate field to which computers could be applied. As we moved into the 1990s it was recognized that such an automation route was not the only direction in the improvement of computers. The expansion of processing power has enabled image data to be incorporated into computer systems, mainly for the purpose of improving human utilisation. For most computer technologies of the 1990s, including the Internet and virtual reality, automation was not the ultimate purpose. Humans were increasingly actively involved in the information-processing loop. This involvement has gradually increased as we move into the 21st century. Development of computer technology based not on automation, but on interaction, is now fully established.

The method of interaction has significantly changed as well. The expansion of computer ability means that the same function can be performed far more cheaply and on smaller computers than ever before. The advent of portable and mobile computers and pervasive computing devices is ample evidence of this. The need for users to be at the same location as a computer in order to reap the benefits of software installed on that computer is becoming an obsolete notion. Time and space are no longer constraints. One of the most discussed impacts of computing and information technology is *communication* and the easy accessibility of information. This changes the emphasis for research and development – issues such as cultural, political, and economic differences must, for example, be accommodated in ways that researchers have not previously considered. Our goal should be to enable users to benefit from technological advances, hence matching the skills, needs, and expectations of users of available technologies to their immense possibilities.

The conference theme for the SAICSIT 2001 Conference – *Hardware, Software and Peopleware: The Reality in the Real Millennium* – aims to reflect technological developments in all aspects related to computerised systems or computing devices, and especially reflect the fact that each influences the others.

Not only has SAICSIT come of age in the 21st century, but so has the research and development community in Southern Africa. The outstanding quality of papers submitted to SAICSIT 2001, of which only a small selection is published in this collection, illustrates both the exciting and developing nature of the field in our region. I hope that you will enjoy SAICSIT 2001 and that it will provide opportunities to cultivate and grow the seeds of discussion on innovative and new developments in computing and information technology.

Paula Kotzé
SAICSIT President

Message from the Chairs

Running this conference has been rewarding, exciting and exhausting. The response to the call for papers we sent out in March was overwhelming. We received 64 paper submissions for our main conference and twelve for the postgraduate symposium. We had a panel of internationally recognized reviewers, both local and international. The response from the reviewers was impressive – accepting a variety of papers and *mostly* returning the reviews long before the due date. We were struck, once again, by the sheer magnanimity of academia – as busy as we all are, we still manage to contribute fully to a conference such as SAICSIT.

After an exhaustive review process, where each paper was reviewed by at least three reviewers, the program committee accepted 26 full research papers and 14 electronic papers. Five papers were referred to the postgraduate symposium, since they represented work in progress – not yet ready for presentation to a full conference but which nevertheless represented sound and relevant research. The papers published in this volume therefore represent research of an internationally high standard and we are proud to publish it. Full electronic papers will be available on the conference web site (http://www.cs.unisa.ac.za/saicsit2001/).

Computer Science and Information Systems academics in South Africa labour under difficult circumstances. *The popularity of IT courses stems from the fact that IT qualifications are in high demand in industry, which leads in turn to a shortage of IT academic staff to teach the courses, even when posts are available. The net result is that fewer people teach more courses to more students. IT departments thus rake in ever-increasing amounts of state subsidy for their universities. These profits, euphemistically labelled "contribution to overhead costs", are deployed in various ways: cross-subsidization of non-profitable departments; maintenance of general facilities; salaries for administrative personnel, etc. Sweeteners of generous physical resources for the IT departments may be provided. We have yet to hear of a University in South Africa where significant concessions have been made in terms of industry-related remuneration. At best, small subventions are provided. As a result, shortages of quality staff remain acute in most IT departments – especially at senior teaching levels. What is even worse is that academics in these departments have to motivate the value of their conference contributions and other IT outputs to selection committees, often dominated by sceptical academic power-brokers from the more traditional departments whose continued survival is underwritten by IT's contribution to overhead costs.*[1]

The papers published in this volume are conclusive evidence of the indefatigability and pertinacity of Computer Science and Information Systems academics and technologists in South Africa. We are proud to be part of such a prestigious and innovative group of people.

In conclusion, we would like to thank the conference chair, Prof Paula Kotzé, for her support. We also specially thank Prof Derrick Kourie for his substantial contribution. Finally, to all of you, contributors, presenters, reviewers and organisers – a big thank you – without you this conference could not be successful.

Enjoy the Conference!
Karen Renaud & Andries Barnard

[1] This taken almost verbatim from Professor Derrick Kourie's SACLA 2001 paper titled: "*The Benefits of Bad Teaching*".

Conference Organisation

General Chair
Paula Kotzé

Programme Chairs
Karen Renaud
Andries Barnard

Organising Committee Chairs
Lucas Venter, Alta van der Merwe

Art and Design
Tersia Parsons

Sponsor Liaison
Paula Kotzé, Chris Bornman

Secretarial & Finances
Christa Prinsloo, Elmarie Havenga

Marketing & Public Relations
Klarissa Engelbrecht, Elmarie van Solms, Adriaan Pottas, Mac van der Merwe

Audio Visual
Tobie van Dyk, Andre van der Poll, Mac van der Merwe

Program Committee

Bob Baber – McMaster Univeristy, Canada
Andries Barnard – University of South Africa
Judy Bishop – University of Pretoria
Andy Bytheway – University of the Western Cape
Andre Calitz – University of Port Elizabeth
Elsabe Cloete – University of South Africa
Carina de Villiers – University of Pretoria
Alan Dix – Lancaster University, United Kingdom
Jan Eloff – Rand Afrikaans University
Andries Engelbrecht – University of Pretoria
Chris Johnson – University of Glasgow, United Kingdom
Paul Licker – University of Cape Town
Paula Kotzé – University of South Africa
Derrick Kourie – University of Pretoria
Philip Machanick – University of the Witwatersrand
Gary Marsden – University of Cape Town
Don Petkov – University of Natal in Pietermaritsburg
Karen Renaud – University of South Africa
Ian Sanders – University of the Witwatersrand
Derrick Smith – University of Cape Town
Harold Thimbleby – Middlesex University, United Kingdom
Theda Thomas – Port Elizabeth Technikon
Herna Viktor – University of Pretoria, South Africa
Bruce Watson – Universities of Pretoria and Eindhoven
Janet Wesson – University of Port Elizabeth

Referees

Molla Alemayehu
Trish Alexander
Adi Attar
Bob Baber
Andries Barnard
John Barrow
Judy Bishop
Gordon Blair
Arina Britz
Andy Bytheway
André Calitz
Charmain Cilliers
Elsabe Cloete
Gordon Cooper
Richard Cooper
Annemieke Craig
Thad Crews
Quintin Cutts
Michael Dales
Carina de Villiers
Alan Dix
Dunlop Mark
Elize Ehlers
Jan Eloff
Andries Engelbrecht
Klarissa Engelbrecht
David Forsyth
John Galletly
Vashti Galpin
Wayne Goddard
Alexandré Hardy
Scott Hazelhurst
Johannes Heidema
Tersia Hörne
Chris Johnson
Bob Jolliffe
Paula Kotzé
Derrick Kourie
Les Labuschagne
Paul Licker
Philip Machanick
Anthony Maeder
David Manlove
Gary Marsden
Thomas Meyer
Elsa Naudé
Martin Olivier
Don Petkov
Pekka Pihlajasaari
Nelisha Pillay
Laurette Pretorius
Karen Renaud
Ingrid Rewitzky
Sheila Rock
Markus Roggenbach
Ian Sanders
Justin Schoeman
Martie Schoeman
Elsje Scott
Derek Smith
Elmé Smith
Adrie Stander
Harold Thimbleby
Theda Thomas
Judy Van Biljon
Alta Van der Merwe
André van der Poll
Tobias Van Dyk
Lynette van Zijl
Lucas Venter
Herna Viktor
Bruce Watson
Janet Wesson

Conference Sponsors

Keynote Abstracts

The specification of a multi-level marketing business

J.A. Van der Poll[a] P. Kotzé[b]

University of South Africa
Department of Computer Science and Information Systems
[a]vdpolja@unisa.ac.za [b]kotzep@unisa.ac.za

Abstract

*A formal specification of a multi-level marketing (MLM) business is presented. Specifying a MLM business boils down to specifying properties of and operations on mathematical forests and trees. The usefulness of the model-based specification language, **Z**, is investigated as a vehicle for a formal specification of these. Proof obligations aimed at corroborating the aptness of the specification are stated and discharged using the resolution-based, first-order theorem prover OTTER. Reasoning about two simple properties of the specification illustrates the utility of two automated reasoning strategies in the literature, namely avoiding equality and applying resonance.*
Keywords: *automated reasoning, formal specification, multi-level marketing, OTTER, resolution, set theory,* Z
Computing Review Categories: *D.2.4, F.3.1, F.4.1, I.2.3*

1 Introduction

A formal specification of a multi-level marketing (MLM) business is presented using **Z** (e.g. Spivey [12]). A MLM business may be modelled by a mathematical *forest* consisting of various *trees*. The study of forests and trees is well established in Computer Science (see e.g. Baase et al. [2]). These structures have been used to describe various entities, for example, the specification of a text editor (Scheurer [11]) in ordinary Zermelo-Fraenkel set theory (Enderton [5]), but as far as we are aware, such structures have never been used to specify a MLM business and its various intrinsic operations in **Z**.

A lucrative feature of a formal specification is that the specifier can reason about the properties of the specification. One can show that the specification has certain characteristics or that certain undesirable consequences are absent. Various reasoning mechanisms are available to a specifier. Some well-known systems are the term-rewriting theorem provers like CaDiZ (Toyn [14]) and Z-Eves (Saaltink [10]). These systems are all interactive reasoning assistants. Alternatively, one can use a fully automatic reasoning assistant like OTTER (McCune [7]) or Gandalf (Tammet [13]). In this paper we use OTTER to discharge proof obligations arising from the specification. OTTER is a first-order, resolution-based theorem prover developed by William C. McCune at the Argonne National Laboratory in Illinois, USA. Two important sections often present in the input to OTTER are the *usable list* and the *set-of-support (sos) list*. One way in which to use OTTER is to place the relevant facts known to be true in the proof attempt in the usable list and the negation of what we want to prove in the sos.

Our MLM specification below gives rise to a rather large state space which in turn poses demanding challenges to a resolution-based automated reasoning assistant when reasoning about the properties of state components. We show that proof attempts of two rather simple properties call for the application of two important reasoning strategies, namely avoiding equality and using resonance. These strategies are only two of many. Other strategies for aiding the operation of a resolution-based reasoning assistant have been developed and presented in [16].

2 Background to MLM systems

Traditional multi-level marketing (MLM) businesses (e.g. GNLD [6]) have been around for a number of years and many conventional businesses are now starting to add some form of MLM to their existing operation. A multi-level marketing business normally markets consumable products[1] and operates as follows:

A new *distributor* (also called a member) pays a registration fee and joins the business either as a direct associate of the company, or under an existing distributor called a *sponsor*. The sponsor does not sponsor the new distributor with money, but rather with knowledge and advice about the business. Both the sponsor and the new distributor then go on to each sponsor more new distributors, and so on. In this way a *network* of distributors of the products of the company is built. The sponsor is also called the *upline* of the new distributor, while the new distributor is generally known as the *downline* of the sponsor. All

[1]One of the most common consumable products in such a business is soap, since it normally has a high turnover. Alternatively, one can trade with luxury items like motor cars or houses, resulting in lower turnover and often lower profits.

distributors have to renew their registration annually to remain in the business. An example of a MLM network is shown in Figure 1.

The distributors $A1$, $A2$, and $A3$ in Figure 1 associated with the company directly are called the roots of the forest (or network in MLM terms).

The 'work' in a MLM business involves a number of steps:

- **Step 1:** Become a distributor of the consumable products available from the company.
- **Step 2:** Sponsor others to become product distributors as well.

Each product has a *point value* (pv for short) as well as a *business value*[2] (bv) associated with it. Both the points and the business values are accumulated per distributor throughout a calendar month. At the end of the month the total business value (called a *turnover* in MLM terms) in the network for each distributor is calculated, and the distributor is paid (in the appropriate currency) a certain percentage of the total business value for his or her group. This is called a *bonus.*

The point value determines, on a sliding scale, the percentage to be used in the calculation of a bonus — a lower percentage for a lower turnover and a higher percentage for a higher turnover. Bonuses are the main source of income for distributors in such a network.[3] A distributor qualifies for a bonus at the end of a month *only* after having accumulated a certain number of points through personal product consumption and a (possibly different) number of points generated by downline distributors.

Distributors may also buy products from the company and sell these to customers at a profit. A customer is somebody who uses the products, but did not join the MLM business to become a distributor. A distributor may furthermore sell her or his business (or part of it) to an existing distributor, or somebody outside the MLM business who then automatically becomes a distributor. Multi-level marketing companies normally have fixed guidelines regarding such a transaction. (See e.g. the section *The marketing plan*, pages 20 — 23 in [6].)

We start with the following natural language requirements definition:

> Specify a system where new distributors may join a MLM business and become consumers of products available from the company. A distributor is allocated a unique identity code upon enrollment. Other information to be maintained for a distributor include the name, address, personal pv, and personal bv. Every product carries a point value as well as a business value. The system must allow for operations like enrolling a new distributor, ordering of products, calculation of bonuses at the end of a month and removing a distributor from the network of distributors.

Our specification is presented in the **Z** specification language (e.g. Spivey [12], Potter et al. [8]). From the above requirements definition we identify the following basic types:

$$[ID, Name, Address, PV, BV, Bonus, Message]$$

The basic type *Message* above represents the set of all possible messages generated by the system. Its sole purpose is to provide feedback to users.

In our model we adopt a global viewpoint taken by the company that creates the environment within which distributors can build their businesses.

One possibility is that the company views the businesses as a group of top-level upline distributors, each with their own network of downline distributors. This viewpoint corresponds to the idea of a *forest*, where the *roots* of the trees in the forest represent the top-level upline distributors, and the network of downline distributors for each top-level upline is represented by a corresponding *tree.* This viewpoint fits the generic model for forests and trees put forward by Scheurer [11].

In this case, the state space is given by:

MLM

$$\begin{array}{l} known : \mathbb{P}\, ID \\ NRoots : \mathbb{P}\, ID \\ NUplines : ID \leftrightarrow ID \\ NDist : ID \nrightarrow \\ \quad Name \times Address \times PV \times BV \times Bonus \\ \hline known = \mathrm{dom}\, NDist \\ \mathrm{dom}\, NUplines \cup \mathrm{ran}\, NUplines \subseteq known \\ NRoots = known \setminus \mathrm{ran}\, NUplines \\ Inj(NUplines) \\ (\forall Netw) \\ \quad (Netw \subseteq known \land NRoots \subseteq Netw \land \\ \quad\quad ClosedBy(Netw, NUplines) \\ \quad\quad\quad \longrightarrow Netw = known) \end{array}$$

Schema *MLM* introduces 4 components and 2 auxiliary predicates:

- The set *known* contains the identity codes of all distributors known to the system.

[2]The business value is an amount which is some indication of the price of the product.

[3]These multi-level marketing businesses are sometimes called pyramids. In a typical pyramid, the sponsor always earns more than any of his or her downline. While this might be true for some MLM businesses, it is not the case in general. Often the amount allocated to a downline is subtracted from the gross income of the sponsor (i.e. the upline). In this way it is quite possible for a downline to earn more per month than the upline. The use of a sliding scale furthermore ensures that the amount subtracted from the gross bonus of an upline is not more than the gross bonus itself.

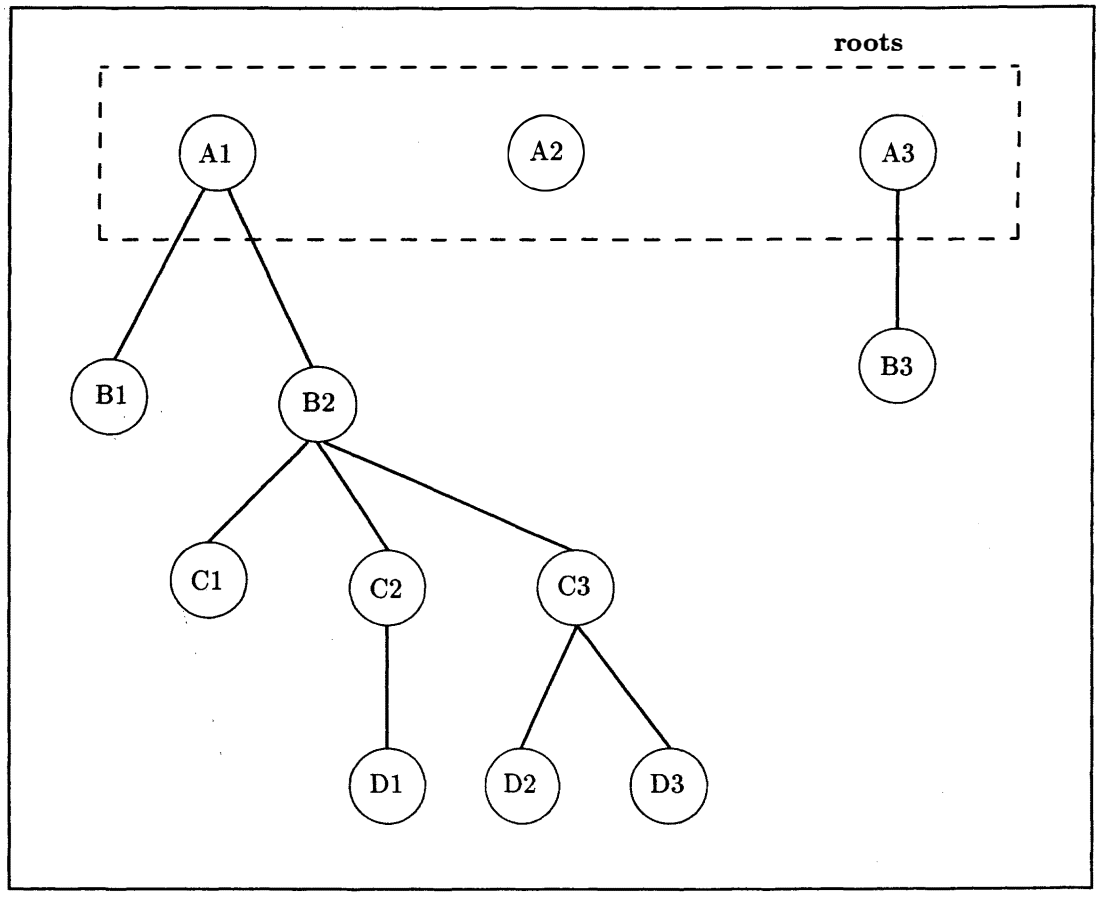


Figure 1: An example network

- *NRoots* represents the distributors who do not have an upline in the business but joined the company directly.
- The network of distributors is represented by relation *NUplines*.
- The function *NDist* represents a mapping from a unique identity code to the particulars for that distributor. Note that *NDist* is partial, since for a given value of *NDist* it need not be the case that dom *NDist* = *ID*; neither is it injective since two (or more) distributors may have the same particulars (i.e. name, address, etc.).
- Relation *NUplines* is injective since every distributor in the business has at most one upline. This property is captured by the predicate *Inj*(*NUplines*). The definition of *Inj* is:

$$(\forall R)(Inj(R) \leftrightarrow (\forall i)(\forall j)(\forall k)(((i,k) \in R \land (j,k) \in R) \longrightarrow (i = j))) \quad (1)$$

Note that the first coordinate of a pair in *NUplines* represents the *upline* of a *downline* which is the second coordinate of the pair.

- The purpose of the last predicate in *MLM* is to ensure that *known*, which represents the set of all known identity codes, is the smallest inductive set of identity codes generated by relation *NUplines*, starting from the root nodes, *NRoots*. Predicate *ClosedBy* is given by:

$$(\forall Netw)(\forall R)(ClosedBy(Netw,R) \leftrightarrow (\forall i)(\forall j)(i \in Netw \land (i,j) \in R \longrightarrow j \in Netw)) \quad (2)$$

3 System operations

Our first operation defined on the state is to register a new distributor. A new distributor (say p) may register either directly with the company (i.e. becomes a root node) or under an existing distributor (say q) in the network. As is customary in **Z**, an unused identity code for the new distributor is generated by the system.

The following schema registers a new distributor under an existing one:

Register_with_upline

ΔMLM
$p!, q? : ID$
$name? : Name;\ addr? : Address$
$mes! : Message$

$p! \notin known \land q? \in known$
$known' = known \cup \{p!\}$
$NUplines' = NUplines \cup \{q? \mapsto p!\}$
$NDist' = NDist \cup$
$\{p! \mapsto (name?, addr?, 0, 0.0, 0.0)\}$
$mes! = New\_distributor\_added$

Initial product information pertaining to the new distributor is reflected in specifying the personal point value to be 0 and both the business value and potential bonus equal to the real value 0.0.

Note that the schema makes no mention of the relationship between the before and after state values of the state component *NRoots*. The reason is that *NRoots* is unaffected by the above operation and this can be proven from the other predicates in the schema. Section 4 presents guidelines for constructing a proof of this property.

Alternatively, a new distributor joins the company directly. Therefore, the new distributor becomes a root element and a dummy, unknown identity code is supplied for the upline:[4]

Register_no_upline

ΔMLM
$p!, q? : ID$
$name? : Name;\ addr? : Address$
$mes! : Message$

$p! \notin known \land q? \notin known$
$known' = known \cup \{p!\}$
$NRoots' = NRoots \cup \{p!\}$
$NUplines' = NUplines$
$NDist' = NDist \cup$
$\{p! \mapsto (name?, addr?, 0, 0.0, 0.0)\}$
$mes! = New\_distributor\_added$

An error condition is generated by *No_more_codes* when attempting to add a new distributor but there are no free identity codes.

No_more_codes

ΞMLM
$mes! : Message$

$known = ID$
$mes! = No\_more\_identity\_codes$

[4]The use of a dummy identity code is simply to make the structure of the input syntactically similar to that of operation *Register_with_upline*, a decision that facilitates the process of combining of the 2 operations into a single operation, *Register* below, using schema disjunction.

Finally, a robust operation to register a distributor is given by:

$Register \mathrel{\widehat{=}}$
$Register\_with\_upline \lor Register\_no\_upline \lor$
No_more_codes

Next, we specify an operation to capture a product order placed by a distributor.

Order

ΔMLM
$id? : ID$
$pv? : PV;\ bv? : BV$
$mes! : message$

$id? \in known$
$(\exists pv : PV;\ bv : BV \bullet$
$pv = third(NDist(id?)) + pv? \land$
$bv = fourth(NDist(id?)) + bv? \land$
$NDist' =$
$NDist \oplus$
$\{id? \mapsto$
$(first(NDist(id?)), second(NDist(id?)),$
$pv, bv, fifth(NDist(id?)))\})$
$mes! = Order\_captured$

Schema *Order* introduces some functions:

- The functions *first*, *second*, and so forth project out an element at the appropriate position in the tuple.
- The symbol $\oplus$ is the relational overriding operator and its effect is to replace a tuple in a relation. Therefore, $NDist'$ is obtained from *NDist* by replacing the tuple with first coordinate $id?$ as specified above.

Supplying an unknown identity code results in:

Unknown_distributor

ΞMLM
$id? : ID$
$mes! : Message$

$id? \notin known$
$mes! = Unknown\_distributor$

A robust operation for placing an order is:

$Order\_product \mathrel{\widehat{=}} Order \lor Unknown\_distributor$

An important operation is to calculate the nett bonus for a distributor. The first step is to specify an operation to calculate the gross bonus to be paid to a distributor. For any distributor, say $id?$, the gross bonus is kept as the fifth coordinate of $NDist(id?)$.

Gross_bonus

ΔMLM
$id? : ID$
$mes! : Message$

$id? \in known \land$
$(\exists\, distpv, totpv : PV;\ distbv, totbv : BV \bullet$
$distpv = third(NDist(id?)) \land$
$distbv = fourth(NDist(id?)) \land$
$totpv = \sum_{i \in NDescs(id?)} third(NDist(i)) \land$
$totbv = \sum_{i \in NDescs(id?)} fourth(NDist(i)) \land$
$fifth(NDist'(id?)) =$
if $distpv \geq 100$ **then**
if $(totpv < 500)$ **then** 0.0
else if $(500 \leq totpv < 1000)$ **then**
$0.05 * totbv$
else if $(1000 \leq totpv < 2000)$ **then**
$0.1 * totbv$
else if $(2000 \leq totpv < 3000)$ **then**
$0.15 * totbv$
else if $(3000 \leq totpv < 4000)$ **then**
$0.2 * totbv$
else $0.25 * totbv$ [5]
else 0.0)

The function *NDescs* recursively calculates all descendents of a distributor (say p), including p:

$$NDescs(p) = \{p\} \cup \bigcup \{NDescs(j) \cup \{j\} \mid (p, j) \in NUplines\}$$

for *NUplines* a component of the *MLM* state space.

Schema *Gross_bonus* specifies a gross bonus for a distributor *id*? along the following lines: If the personal pv of *id*? is less than 100 or *id*?'s group pv is less than 500 then the bonus is 0.0. If the personal pv is greater than 99 and the group pv is 500 or more then the group pv determines the percentage to be used in the calculation of the gross bonus for *id*?.

Next, we specify a nett bonus calculation:

Nett_bonus

ΞMLM
$id? : ID$
$bonus! : Bonus$
$mes! : Message$

$id? \in known$
$bonus! =$
$fifth(NDist(id?)) -$
$\sum_{(id?, i) \in NUplines} fifth(NDist(i))$
$mes! = Bonus\_calculated$

A nett bonus for distributor *id*? is specified as the difference between the gross bonus for *id*? and the sum of the gross bonuses of all the *immediate* downlines of *id*?.

A robust bonus calculation is specified as either a gross bonus calculation followed by the calculation of a nett bonus, or an appropriate error condition:

$$Bonus \mathrel{\widehat=} (Gross\_Bonus \fatsemi Nett\_Bonus) \lor Unknown\_distributor$$

Our last operation deals with the scenario where a distributor decides to quit the business. Suppose a *root* distributor, say $A1$ in Figure 1, decides to drop out of the network. This has the following effect:

(1) Remove $A1$ as a root element, i.e.

$$NRoots' = NRoots \setminus \{A1\} \qquad (3)$$

(2) Remove the pairs $A1 \mapsto B1$ and $A1 \mapsto B2$ from the network, i.e.

$$NUplines' = \{A1\} \lhd NUplines \qquad (4)$$

The symbol $\lhd$ is known as the domain anti-restriction operator. Its effect is to remove all tuples with the first coordinate being an element of the set on the left of the operator from a relation.

(3) Make $B1$ and $B2$ root elements, i.e.

$$NRoots'' = NRoots' \cup \{B1, B2\} \qquad (5)$$

As a schema operation we have:

DelRoot

ΔMLM
$id? : ID$
$mes! : Message$

$id? \in NRoots \land$
$known' = known \setminus \{id?\} \land$
$NDist' = \{id?\} \lhd NDist \land$
$NUplines' = \{id?\} \lhd NUplines \land$
$(\exists\, FRoots : \mathbb{P}\, ID \bullet$
$FRoots = \{r \mid (id?, r) \in NUplines\} \land$
$NRoots' = (NRoots \setminus \{id?\}) \cup FRoots) \land$
$mes! = Distributor\_deleted$

Note that appropriate after state values are also specified for *known* and *NDist*.

Alternatively, if a *non-root* distributor, say $B2$, decides to quit the business then the following actions are applicable:

(1) Node $B2$ is deleted from the forest. Deleting $B2$ from the forest amounts to removing the pairs $B2 \mapsto C1$, $B2 \mapsto C2$ and $B2 \mapsto C3$ from the network, i.e.

$$NUplines' = NUplines \rhd \{C1, C2, C3\} \qquad (6)$$

[5] Normally, there would be different bonus structures for point values 'much larger' than the 4000 mark, but consideration of these is beyond the scope of this introductory specification.

as well as removing the pair $A1 \mapsto B2$ from the network, i.e.

$$NUplines'' = NUplines' \rhd \{B2\} \quad (7)$$

The symbol $\rhd$ is known as the range anti-restriction operator. Its effect is to remove all tuples with the second coordinate being an element of the set on the right of the operator from a relation. Note that equation (6) can also be written as $\{B2\} \lhd NUplines$. The reason for using the format in (6) is to simplify the definition of the after state value of *NUplines* in schema *Del_Not_Root* below.

(2) Nodes $C1$, $C2$ and $C3$ become immediate downlines of $A1$:

$$NUplines''' = NUplines'' \cup \{A1 \mapsto C1, A1 \mapsto C2, A1 \mapsto C3\}$$

As a schema we have ($^\sim$ denotes an inverse):

Del_Not_Root

ΔMLM
$id? : ID$
$mes! : Message$

$id? \in \text{ran } NUplines \land$
$known' = known \setminus \{id?\} \land$
$NDist' = \{id?\} \lhd NDist \land$
$(\exists q : ID;\ FRoots : \mathbb{P}\, ID \bullet$
$\quad q = NUplines^\sim(id?) \land$
$\quad FRoots = \{r \mid (id?, r) \in NUplines\} \land$
$\quad NUplines' =$
$\quad\quad (NUplines \rhd (FRoots \cup \{id?\})) \cup$
$\quad\quad\quad \{(q, r) \mid r \in FRoots\})$

A robust operation for deleting a distributor is given by:

$$DelDist \mathrel{\widehat=} Del\_Root \lor Del\_Not\_Root \lor Unknown\_distributor$$

Summarising the MLM operations in this section gives:

$$Register \mathrel{\widehat=} Register\_with\_upline \lor Register\_no\_upline \lor No\_more\_codes$$

$$Order\_Product \mathrel{\widehat=} Order \lor Unknown\_distributor$$

$$Bonus \mathrel{\widehat=} (Gross\_Bonus \fatsemi Nett\_Bonus) \lor Unknown\_distributor$$

$$Del\_Dist \mathrel{\widehat=} Del\_Root \lor Del\_Not\_Root \lor Unknown\_distributor$$

4 Reasoning about the specification

In **Z**, system operations are implicitly assumed to preserve the state invariant (see e.g. Diller et al. [4]), hence proving that the state invariant is preserved by an operation is not considered a necessary proof obligation in **Z**. Nevertheless, it is a good idea to check that an operation does not incur unpredictable results as far as the invariant is concerned. In systems like the **B** method (Abrial [1], Waeselynck [17]) ensuring that an operation preserves the invariant is considered a proof obligation.

For example, suppose a specifier wants to verify that the property

$$NRoots' = known' \setminus \text{ran } NUplines' \quad (8)$$

which is a conjunct of the invariant, is preserved by operation *Register*. Suppose a new distributor p is registered below an existing distributor, q. Consider two restrictions specified in *Register_with_upline*:

$$known' = known \cup \{p\} \quad (9)$$
$$NUplines' = NUplines \cup \{q \mapsto p\} \quad (10)$$

Suppose we define:

$$NewRoots = known' \setminus \text{ran } NUplines' \quad (11)$$

and pose the negation of the following equality in the OTTER set-of-support list

$$(NRoots = NewRoots) \quad (12)$$

then the theorem prover fails to find any proof for (12) in 20 minutes. It is well known that equality reasoning poses demanding challenges to automated reasoning assistants (e.g. Boyer et al. [3], Quaife [9]). Therefore, if we apply the first axiom of Zermelo-Fraenkel set theory (see e.g. Enderton [5]), namely Extensionality, and simply replace (12) with (13) below

$$(\forall x)(x \in NRoots \leftrightarrow x \in NewRoots) \quad (13)$$

then OTTER finds a short proof in *0.49 seconds.*[6] Note that this result also implies that the set of root nodes (i.e. *NRoots*) is not affected by operation *Register_with_upline*.

Another example of a proof is to show that the cardinality of a set is preserved, e.g.

$$\#NDist' = \#NDist \quad (14)$$

after operation *Order_product*. Suppose the precondition $id? \in known$ is satisfied and the after state value

[6] Proof performed on Linux Red Hat 6.2 on a Pentium III running at 600 MHz.

of *NDist* is as reflected in operation *Order*:

$$
\begin{aligned}
&NDist' = \\
&\quad NDist \oplus \\
&\quad\quad \{id? \mapsto \\
&\quad\quad\quad (first(NDist(id?)), \\
&\quad\quad\quad\quad second(NDist(id?)), pv, bv, bonus)\} \qquad (15)
\end{aligned}
$$

Possible definitions for reasoning about the cardinality of a set are (e.g. [15]):

$$(\forall A)(\#A = 0 \leftrightarrow A = \varnothing) \qquad (16)$$

$$
\begin{aligned}
&(\forall A)(\forall n)(\#A = n + 1 \leftrightarrow \\
&\quad (\exists x)(x \in A \wedge \#(A \setminus \{x\}) = n)) \qquad (17)
\end{aligned}
$$

In a more procedural fashion relational overriding, $\oplus$, for any two relations R and S is defined as:

$$R \oplus S = (\mathrm{dom}\, S \lhd R) \cup S \qquad (18)$$

Hence, definition (15) is unfolded into the two formulae (19) and (20):

$$
\begin{aligned}
&(\forall u)(\forall v)(\forall w)(\forall x)(\forall y)(\forall z) \\
&\quad (6TUP(u, v, w, x, y, z) \in NDist'' \leftrightarrow \\
&\quad\quad 6TUP(u, v, w, x, y, z) \in NDist \wedge u \neq id?) \qquad (19)
\end{aligned}
$$

The set *NDist''* represents the entity $(\mathrm{dom}\, S \lhd R)$ in definition (18) above.

$$
\begin{aligned}
&(\forall u)(\forall v)(\forall w)(\forall x)(\forall y)(\forall z) \\
&\quad (6TUP(u, v, w, x, y, z) \in NDist' \leftrightarrow \\
&\quad\quad (6TUP(u, v, w, x, y, z) \in NDist'' \vee \\
&\quad\quad\quad 6TUP(u, v, w, x, y, z) = \\
&\quad\quad\quad\quad 6TUP(id?, name, addr, pv, bv, bonus))) \qquad (20)
\end{aligned}
$$

where *name*, *addr*, *pv*, *bv* and *bonus* represent the last 5 coordinates of a tuple from *NDist*.

Proof obligation (14) above boils down to showing that if we start by claiming that $\#NDist = k + 1$ (say), then $\#NDist' = k+1$. OTTER fails to find any proof of this property using the standard definitions (16) and (17). One of the reasons for failure is that these definitions are not suitable for a proof of (14), especially not when we are dealing with sets of tuples instead of simple sets.

We start our attack on the problem by employing a reasoning strategy called *resonance* described in Wos [18]. The aim of the resonance strategy is to give preference (for directing a program's reasoning) to formulae that have the same syntactic shape (ignoring variables) as one or more of the patterns supplied by the specifier. Such formulae are called *resonators*.

We, therefore, rewrite the cardinality definitions in terms of a tuple being removed or added to a set.

First, we give definition for removing a tuple from a set:

$$
\begin{aligned}
&(\forall A)(\forall n)(\forall u)(\forall v)(\forall w)(\forall x)(\forall y)(\forall z)(\forall B) \\
&\quad ((Fun(A) \wedge \#A = n + 1 \wedge \\
&\quad\quad 6TUP(u, v, w, x, y, z) \in A \wedge \\
&\quad\quad\quad (\forall u1)(\forall v1)(\forall w1)(\forall x1)(\forall y1)(\forall z1) \\
&\quad\quad\quad\quad (6TUP(u1, v1, w1, x1, y1, z1) \in B \leftrightarrow \\
&\quad\quad\quad\quad\quad 6TUP(u1, v1, w1, x1, y1, z1) \in A \wedge u1 \neq u)) \\
&\quad\quad\quad\quad \longrightarrow \#B = n) \qquad (21)
\end{aligned}
$$

Definition (21) could be explained as follows. For any given 6-tuple relations A and B, if A is a function and $\#A = n + 1$ then $\#B = n$, provided that B is obtained from A by performing a domain subtraction of a single tuple from A.

A resonance definition for adding a new tuple to a set is:

$$
\begin{aligned}
&(\forall A)(\forall n)(\forall x)(\forall B) \\
&\quad ((\#A = n \wedge x \notin \mathrm{dom}\, A \wedge \\
&\quad\quad ((\exists name)(\exists addr)(\exists pv)(\exists bv)(\exists bonus) \\
&\quad\quad\quad (\forall u1)(\forall v1)(\forall w1)(\forall x1)(\forall y1)(\forall z1) \\
&\quad\quad\quad\quad (6TUP(u1, v1, w1, x1, y1, z1) \in B \leftrightarrow \\
&\quad\quad\quad\quad\quad 6TUP(u1, v1, w1, x1, y1, z1) \in A \vee \\
&\quad\quad\quad\quad\quad\quad 6TUP(u1, v1, w1, x1, y1, z1) = \\
&\quad\quad\quad\quad\quad\quad\quad 6TUP(x, name, addr, pv, bv, bonus)))) \\
&\quad \longrightarrow \#B = n + 1) \qquad (22)
\end{aligned}
$$

Definition (22) states that for any sets A and B and any x where $x \notin \mathrm{dom}\, A$, if $\#A = n$ and B is obtained from A by adding a tuple with x as the first coordinate, then $\#B = n + 1$.

We also need a resonance definition of a domain:

$$
\begin{aligned}
&(\forall R)(\forall u) \\
&\quad (u \in \mathrm{dom}\, R \leftrightarrow \\
&\quad\quad (\exists v)(\exists w)(\exists x)(\exists y)(\exists z) \\
&\quad\quad\quad (6TUP(u, v, w, x, y, z) \in R)) \qquad (23)
\end{aligned}
$$

These changes allow OTTER to find a short proof for (14) in just *1.78 seconds*. The input to the theorem prover appears in Appendix A.

5 Enhancements to the specification

A small enhancement can be made to the above specification.

Distributors sometimes pose queries which need not go through their uplines, but which may be addressed to the company directly. The company appoints some 'distributor relations coordinators' for this purpose and these coordinators are assigned to the top-level

uplines. If the group of distributors belonging to a top-level upline is not too large (according to the judgement of the company), then only one coordinator is assigned to such a group of distributors, but larger groups normally have more than one coordinator assigned to them.

One way of incorporating coordinators in our specification is to define a new function

$$NCoord : ID \pfun \mathbb{P}(Coord)$$

where *Coord* represents the set of coordinators employed by the company. For any distributor identity code d, if $d \notin \mathrm{dom}(NCoord)$ then d is not a top-level upline and therefore d's top-level upline has to be determined first, whereafter the coordinator set for d is obtained. The set could be a singleton for a small network as mentioned above. Note that the variable *NCoord* would become a fifth component of the state, possibly further complicating proof attempts when reasoning about properties of the state.

6 Summary and future research

This paper presented a **Z** specification of part of a multi-level marketing business. In essence, specifying such a business boils down to specifying properties of trees and forests and operations on these. **Z** appears to be a suitable vehicle for this task. We illustrated how some proof obligations arising from the specification may be stated and discharged using an appropriate reasoning assistant. In the process we demonstrated the utility of two reasoning strategies from the literature, namely avoiding equality and using resonance.

The bonus calculation defined in schema *Gross_bonus* is incomplete. Normally in a MLM business there are additional, more advanced bonus structures. For example, one such structure involves the idea of a '4000 point unit': A distributor is paid a bonus of 5% of the business volume[7] of a 4000 point unit on his or her first level, 2% on the business volume of such a unit on the second level, and 1% on the business volume of a similar unit on the third level. These percentages advance on a sliding scale with the top values fixed at 7%, 5%, and 2% respectively. (See e.g. the section *The Marketing Plan*, pages 5 — 6 in [6].)

Additional proof obligations arising from the specification can be identified and the suitability of using OTTER to discharge these can be investigated. For example, the following proof obligation can be formulated:

> The sum of the gross bonuses allocated to *first-level* downline distributors is subtracted from the gross bonus allocated to their parent distributor in the network. Show, therefore, that the sum of the gross bonuses allocated to first-level downline distributors is *less than or equal to* the gross bonus allocated to their immediate upline.

Formally, consider a distributor, say q, with a number of first-level downlines, represented by a set, say D, such that

$$(\forall\, d)(d \in D \leftrightarrow (q, d) \in NUplines)$$

for *NUplines* a component of the MLM state space. Suppose the gross bonus allocated to a distributor, say i, is indicated by $Bonus_i$. The proof obligation is to show that

$$Bonus_q - \sum_{(q,d) \in NUplines} Bonus_d \geq 0$$

OTTER fails to find any proof of the above property. Therefore, an attempt at discharging this proof obligation with OTTER calls for the development of a set of heuristics for reasoning about arithmetic expressions, reminiscent of the heuristics developed in [16] for reasoning about expressions in set theory. Of course, one can also investigate the feasibility of using some of the well-known interactive term-rewriting reasoners like CaDiZ (see [14]) or Z-Eves, [10] to discharge the above proof obligation.

A rather simple way to solve this problem in one swell swoop in **Z** is to include additional conditions in the state invariant in schema *MLM* requiring that the value of every distributor's bonus be non-negative and greater than or equal to the sum of the bonuses of that distributor's immediate downlines. Nevertheless, the sensible specifier will always check (using some proof mechanism) that all relevant operations uphold this property.

7 Acknowledgement

The authors would like to thank the referees for making valuable suggestions regarding this article.

[7] In the MLM world a business value is often called a *business volume*.

Appendix A

The complete input to OTTER for a proof of

$$\#NDist' = \#NDist$$

after operation *Order* is given below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Prove #NDist' = #NDist after %%
%% overriding an element.       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
set(neg_hyper_res).
set(factor).
set(ur_res).
%%
%% Paramodulation.
%% ---------------
set(para_from).
set(para_into).
%%
set(order_eq).
%%
%% Restriction strategies:
%% -----------------------
set(process_input).
clear(para_from_right).
clear(para_into_right).
set(dynamic_demod_all).
set(back_demod).
%%
weight_list(pick_and_purge).

weight(x,5).

end_of_list.
%%
clear(print_given).
clear(print_kept).
clear(print_back_sub).

formula_list(usable).

%% Reflexivity.
%% ------------
(all x (x = x)).

%% Resonance domain definition.
%% ------------------------ ---
(all R u
 ( El(u,dom(R)) <->
  (exists v w x y z
    El(6TUP(u,v,w,x,y,z),R)) )).

%% ##############################
%% Resonance cardinality follows.
%% ##############################

%% Removing element - Domain subtraction:
%% --------------------------------------
(all A n u v w x y z B
 ( ( Card(A,$SUM(n,1)) &
     El(6TUP(u,v,w,x,y,z),A) &
     Fun(A) &
     (all u1 v1 w1 x1 y1 z1
      ( El(6TUP(u1,v1,w1,x1,y1,z1),B) <->
         (El(6TUP(u1,v1,w1,x1,y1,z1),A) &
          -(u1 = u)) )) )
   -> Card(B,n) )).

%% Adding a new element:
%% ---------------------
(all A n x B
 ( ( Card(A,n) &
     -El(x,dom(A)) &
     (exists name addr pv bv bonus
      (all u1 v1 w1 x1 y1 z1
       ( El(6TUP(u1,v1,w1,x1,y1,z1),B) <->
          (El(6TUP(u1,v1,w1,x1,y1,z1),A) |
          (6TUP(u1,v1,w1,x1,y1,z1) =
           6TUP(x,name,addr,pv,bv,bonus))))
       )) )
    -> Card(B,$SUM(n,1)) )).

%% NDist1 = DOMDIFF(NDist,{id}).
%% -----------------------------
(all u v w x y z
 ( El(6TUP(u,v,w,x,y,z),NDist1) <->
    (El(6TUP(u,v,w,x,y,z),NDist) &
     -(u = id)) )).

%% NDist2 =
%%  NDist1 u {(id,name,addr,pv,bv,bonus)}.
%% --------------------------------------
(all u v w x y z
 ( El(6TUP(u,v,w,x,y,z),NDist2) <->
    (El(6TUP(u,v,w,x,y,z),NDist1) |
     (6TUP(u,v,w,x,y,z) =
      6TUP(id,name,addr,pv,bv,bonus))) )).

%% NDist is functional.
%% --------------------
Fun(NDist).

%% El(id,NDist)).
%% --------------
El(6TUP(id,name,addr,pv,bv,bonus),NDist).

%% #NDist = k + 1.
%% ---------------
Card(NDist,$SUM(k,1)).

%% Fact about NDist1.
%% ----------- ------
-El(id,dom(NDist1)).
```

```
end_of_list.

formula_list(sos).

%% #NDist2 = k + 1.
%% ----------------
-Card(NDist2,$SUM(k,1)).

end_of_list.
```

References

[1] J.-R. Abrial. *The B Book: Assigning Programs to Meanings.* Cambridge University Press, August 1996.

[2] S. Baase and A. Van Gelder. *Computer Algorithms: Introduction to Design & Analysis.* Addison-Wesley, 2000.

[3] R. Boyer, E. Lusk, W. McCune, R. Overbeek, M. Stickel, and L. Wos. Set Theory in First-Order Logic: Clauses for Gödel's Axioms. *Journal of Automated Reasoning*, 2(3):287 – 327, September 1986.

[4] A. Diller and R. Docherty. **Z** and Abstract Machine Notation: A Comparison. In J. Bowen and J. Hall, editors, *Z User Workshop: Proceedings of the Eighth Z User Meeting, Cambridge, U.K., 29 - 30 June 1994*, Workshops in Computing, pages 250 – 263. Springer-Verlag, 1994.

[5] H. Enderton. *Elements of Set Theory.* Academic Press, Inc., 1977.

[6] Golden Neo-Life Diamite International. *Distributor Business Guide: The Business Plan*, July 1997.

[7] W. W. McCune. *OTTER 3.0 Reference Manual and Guide.* Argonne National Laboratory, Argonne, Illinois, August 1995. ANL-94/6.

[8] B. Potter, J. Sinclair, and D. Till. *An Introduction to Formal Specification and Z.* Prentice-Hall, 2nd edition, 1996.

[9] A. Quaife. *Automated Development of Fundamental Mathematical Theories.* Automated Reasoning Series. Kluwer Academic Publishers, 1992.

[10] M. Saaltink. **Z** and Eves. In J. Nicholls, editor, *Z User Workshop: Proceedings of the Sixth Annual Z User Meeting, York, U.K., 16 - 17 December 1991*, pages 223 – 242. British Computer Society, Springer-Verlag, 1992.

[11] T. Scheurer. *Foundations of Computing : System Development with Set Theory and Logic.* International Computer Science Series. Addison-Wesley, 1994.

[12] J. M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, London, 2nd edition, 1992.

[13] T. Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199 – 204, 1997.

[14] I. Toyn. CaDiZ Type Checker and Theorem Prover Home Page. http://www.cs.york.ac.uk/~ian-/cadiz/home.html.

[15] J. Van der Poll. *Automated Support for Set-Theoretic Specifications.* PhD thesis, Department of Computer Science & IS, University of South Africa, June 2000.

[16] J. Van der Poll and W. Labuschagne. Heuristics for Resolution-Based Set-Theoretic Proofs. *South African Computer Journal*, Issue 23:3 – 17, July 1999.

[17] H. Waeselynck. Developing Safe Software: Software development using the **B**-method. LAAS-CNRS, France. Short course: University of the Witwatersrand, February 1999.

[18] L. Wos. The Resonance Strategy. *Computers and Mathematics with Applications*, 29(2):133 – 178, February 1995. (Special issue on Automated Reasoning).