7th SA COMPUTER RESEARCH SYMPOSIUM
7th Southern African Computer Research Symposium

Karos Indaba Hotel, Johannesburg
1 July 1982

PROCEEDINGS

Guest Editor: Judy M Bishop

Organised by the SA Institute of Computer Scientists
in association with the Computer Society of SA

Sponsored by Persetel and the FRD
When the first SA Computer Symposium was held at the CSIR in the early eighties, it was unique. There was no other forum at the time for the presentation of research in computer science. In the intervening decade, conferences, symposia and workshops have sprung up in response to demand, and now there are several successful ventures, some into their third or fourth iteration. Each of these addresses a specific topic - for example, hypermedia, expert systems, parallel processing or formal aspects of computing - and attracts a specialised audience, well versed in the subject and eager to learn more. For the main part, the proceedings are informal, and certainly not archival.

SACRS, though, is still unique, in that it deliberately covers a broad spectrum of research in computing, and in addition, seeks to provide a lasting record of the proceedings. To achieve the second aim, we negotiated with the SA Institute of Computer Scientists for the proceedings to form a special issue of the SA Computer Journal, and the copy you have in front of you is the result. The collaboration between the symposium committee and the journal's editorial board placed high standards on the refereeing and final presentation of the papers, to the symposium's benefit, while we were still able to maintain a fresh, audience-oriented approach to the selection of papers.

This is SACJ's first such special issue, and the largest issue (at 145 pages) to date. We hope that it is only the beginning of future such collaborations.

In all 29 papers were received, all were refereed twice, and 19 were chosen for presentation by the programme committee. All the papers were thoroughly revised by the authors on the basis of the referee's comments, and the committee's suggestions aimed at making the material more accessible to a broadly-based audience. Papers had to be new, and not to have been presented elsewhere, a requirement that is still unusual within the SA conference round.

A third goal of SACRS has been to invite keynote speakers, usually from overseas. This year, we are fortunate to present Dr Vinton Cerf, the father of the Internet and a world-renown expert on computer networks. Although his paper is not available for this special issue, it will appear later in SACJ. Through the good offices of Professor Chris Brink of UCT, we also have three other speakers from Germany, Canada and the US adding interest to the event, and two of their papers appear in this issue.

The programme committee originally devised a theme for the symposium - "Computing in the New South Africa". We received several queries as to the meaning of this theme, but unfortunately few papers that addressed it directly. One prospective author went as far as to enquire whether computer research would survive in the new South Africa. Another felt that his work was definitely not in the theme, as it was genuine, old world, basic, theoretical science! Nevertheless, there are two papers that consider one of South Africa's key issues, that of language. Others look at the success we have achieved in applying technology to mining, and the future of low-cost operating systems. In all, the mix of papers represents a balance between the theoretical and the practical, the past and the future, all firmly based in the computing of the present.

Organising the symposium has involved the hard work of several people, and I would like to thank in particular:

• Derrick Kourie, my co-organiser, and the editor of SACJ for his invaluable advice and hard work throughout the planning and implementation stages;
• Riel Smit, the production editor, for attaining such a high standard in such a short time for so many papers;
• Gerrit Prinsloo and the staff at the CSSA for their efficient and quite delightfully unfussy organisation;
• Persetel for their very generous sponsorship of R25000, and Tim Schumann for taking a genuine interest in our events;
• the Foundation for Research Development for sponsoring Vint Cerf's visit;
• and finally the Department of Computer Science of the University of Pretoria for providing the ideal working conditions for undertaking ventures of this kind, and especially Roelf van den Heever for his unfailing encouragement and support.

Judy M Bishop
Organising Chairman, SACRS 1992
Guest Editor, SACJ Special Issue
Referees

The journal draws on a wide range of referees. The following were involved in the refereeing of the papers selected for this special issue. Their role in certifying the papers and their contribution to enhancing the quality of papers is sincerely appreciated.

John Barrow
Ronnie Becker
Danie Behr
Sonia Berman
Liesbeth Botha
Theo Bothma
Chris Brink
Peter Clayton
Ian Cloete
Antony Cooper
Elise Ehlers
Quintin Gee
Andy Gravell
Wendy Hall
David Harel
Scott Hazellhurst
Derrick Kourie
Willems Labuschagne
Doug Laing
Dave Levy
Graham McLeod
Hans Messerschmidt
Deon Oosthuizen
Ben Oosthuysen
Neil Pendock
D Petkov
Martin Rennhackkamp
Cees Roon
Jan Roos
John Shochot
Morris Sloman
Riel Smit
Pat Terry
Walter van den Heever
Lynn van der Vegte
Herman Venter
Herna Viktor

UNISA
University of Cape Town
University of Pretoria
University of Cape Town
University of Pretoria
University of Pretoria
University of Cape Town
Rhodes University
Stellenbosch University
CSIR
Rand Afrikaanse University
DALSIG
University of Southampton
University of Southampton
The Weizmann Institute of Science
University of the Witwatersrand
University of Pretoria
UNISA
ISM
University of Natal
University of Cape Town
University of the Orange Free State
University of Pretoria
University of Pretoria
University of the Witwatersrand
University of Natal
Stellenbosch University
University of Pretoria
University of Pretoria
University of the Witwatersrand
Imperial College, London
University of Cape Town
Rhodes University
University of Pretoria
University of Pretoria
University of Fort Hare
Stellenbosch University
Statenets – An Alternative Modelling Mechanism for Performance Analysis

Lynette Lewis
Institute for Applied Computer Science, University of Stellenbosch, 7600 Stellenbosch
lynette@cs.sun.ac.za

Abstract

Statecharts were developed by Harel to specify complex reactive systems. Statecharts proved to be rich in modelling power and convenience [9] and well suited to mechanical implementation. However, the semantic definition of statecharts is complex, so that little work has been done on the mathematical aspects of statecharts. This paper informally introduces a simplified version of statecharts, called statenets, with a stochastic extension. The stochastic statenet is a step towards combining the modelling convenience of the statechart with the analysis capabilities already developed for stochastic Petri nets. The statenet is therefore suitable for both verification and performance analysis purposes, while retaining reasonable modelling convenience. The paper contains an example of designing a system with the statenet, and gives an informal description of analysis methods for the statenet.

Keywords: Performance analysis, reactive systems, statecharts, statenets, stochastic Petri nets.

Received March 1992, Accepted April 1992

1 Introduction

In the design of complex computer hardware or software, it is often expedient to design the system using a formal mathematical model. Such a mathematical model allows the system to be analyzed for both correctness and performance criteria during the design phase, thus avoiding the appearance of costly design errors during implementation.

Until the 1980s, different mathematical models and different tools were used for the design, the correctness analysis and the performance analysis of systems. Designs were usually made with a graphical visualization tool; correctness analysis required correctness specifications to be set up in logical formulae, which were often processed using a theorem prover; performance analysis required the use of a queueing model.

Stochastic Petri nets were then proposed as a unified design and analysis methodology, where the system may be designed as a (stochastic) Petri net, and correctness and performance results computed from the reachability graph of the stochastic Petri net. The major drawback of stochastic Petri nets, however, is that they lead to rather large models which are difficult to visualize for the design phase [13].

In this paper I propose to alleviate the complexity of the design phase with the introduction of the stochastic statenet, which is an extension of a simplified version of Harel’s statecharts. Statecharts were developed by David Harel in the mid 1980s, and are described in a number of papers [9, 10, 11, 12]. Harel developed the statecharts as an answer to the challenge of Green [8] who claimed that there are no adequate graphical specification tools available to model interactive behaviour. Finite state machines and their transition diagrams, for example, are easy to understand, but suffer from an exponential state explosion problem. Statecharts proved to be rich in modelling power and are practical for the design of large systems [9, 14]. Stochastic statenets utilize these characteristics of statecharts, and in addition allow for correctness and performance analysis.

The rest of this paper is organised as follows: Section 2 contains a description of the Distributed Queue Dual Bus (DQDB) protocol for Metropolitan Area Networks (MANs) as a modelling example, and gives the stochastic Petri net model of DQDB. Also in section 2 an informal introduction to statenets is given by means of the DQDB example, contrasting the modelling convenience of the stochastic Petri net with the statenet model for DQDB. Section 3 contains a short informal discussion on the analysis mechanisms needed for stochastic statenets.

2 Example – the DQDB MAN protocol

In the early 1980s, a hybrid between local area networks (LANs) and wide area networks came to the fore: the Metropolitan Area Network (MAN). MANs typically cover a geographical area of tens of kilometres, and are used to interconnect existing LANs. In addition, MANs are assumed to have broadband (B-ISDN) capabilities, enabling them to handle voice, data and video traffic at high speeds [5]. Experimental MANs already exist at this stage, and some companies are starting to market MANs commercially.

Distributed Queue Dual Bus (DQDB) is the IEEE chosen standard medium access protocol for Metropolitan Area Networks. Let us look at the DQDB protocol in more detail.

DQDB – How does the protocol work?
This description of the DQDB protocol is based on the discussion in Abeysundara and Kamal [1]. The description
DQDB is based on a dual-unidirectional bus architecture. It is a slotted system, with a network controller station at the upstream end of a bus generating frames at fixed intervals. A frame consists of several slots, including a busy bit and request bits in the header. The DQDB protocol controls the access of each node to these slots.

Access to the two buses are symmetrical, so that I discuss transmission on only one bus, without any loss of generality. Call the forwarding bus A and the backwards bus B, and consider node i. Node i can be in one of four states: Idle, active, countdown or transmission. In the idle state, the node monitors the frames on buses A and B. If a request frame passes on bus B, it increases the node i request (REQ) counter. If an empty frame passes on bus A, it decreases the node i REQ counter. The node thus keeps track of requests from all its upstream (higher priority) nodes.

When node i has data to transmit, it enters the active state. In the active state, it copies the contents of its REQ counter to its countdown (CD) counter, and zeroes its REQ counter. It immediately enters the countdown state.

In the countdown state, node i monitors buses A and B. For each request on bus B, it increases its REQ counter. For each empty frame passing on a bus, it decreases its CD counter. When the CD counter reaches zero, it knows that none of its upstream nodes have outstanding requests, and it can start transmission of its data. When transmission is complete, it enters the idle state again.

The next two subsections provide a comparison of the relative modelling convenience of stochastic Petri nets (SPNs) and statenets. I use the simplified behaviour of node i in the DQDB protocol as the modelling example.

DQDB – An SPN model
The idle state is given by the subpart marked (a) of the SPN in figure 1. eA represents an empty frame on bus A, while rB represents a request frame on bus B.

From the idle state, a self-request for data transmission prompts a state change to the active state – see figure 1(b). The copying of the REQ counter to the CD counter takes place with the repeated firing of transition Copy.

Once the copying of the counter is complete and the REQ counter is zero, the countdown phase is entered. See figure 1(c) for the countdown phase. When the CD counter is zero, the transmission of the data can start.

Figure 1(d) gives the complete SPN model of this part of the behaviour of node i in the DQDB protocol. Given the complete SPN, one can now use a computerized tool, such as GreatSPN [3], to construct the reachability graph and compute performance measures for the problem. These tools usually follow an approach whereby the reachability graph is used to construct the Continuous Time Markov Chain (CTMC) of the problem and then use standard analysis techniques on the CTMC.

Let us now construct the statenet model of the same problem.

DQDB – A Statenet Model
A statenet consists graphically of a number of rounded rectangles, called states, which may be nested. There are two types of states: XOR (sequential) states and AND (concurrent) states. AND states are indicated graphically by a dashed line separating the substates which may execute in parallel – see figure 2(a). States can be connected by arrows, representing events that cause state changes. The event arrows may cross state boundaries, and are then known as hierarchical arrows (see event h in figure 2(b)). Default arrows in the statenet indicate the default child state to enter on entry to a parent state.

The two major differences between the statechart and the statenet is the absence of the advanced arrows in the statenet, such as the history and selective arrows; and the fact that no output is allowed on events in the statenet. The statenet thus still contains the two major characteristics of the statechart – the hierarchical composition of states, and the orthogonal composition of states. I will now use the statenet to design the DQDB node i behaviour.

The node i can be in any of four states, as shown in figure 3(a). The idle state is given in figure 3(b), the active state in figure 3(c), the countdown state in figure 3(d) and the transmission state in figure 3(e). Here the state AssP represents the assembling of packets, while ChB represents a check on the buffer for more packets to be sent.

The statenet model above is clearly simpler and easier to understand than the stochastic Petri net model in the previous section. The ease with which the statenet model can be constructed results from the orthogonal composition and the hierarchical nesting of states.

The standard statenet as given in figure 3 can only be used to design a system. However, if it were possible to extract state space and reachability information from the statenet, it will be possible to analyse its correctness. Moreover, if one can then allow for probabilistic information to be kept, it will be possible to analyse the performance of the model – similar to the SPN model analysis.

In the next section I suggest an analysis procedure for the statenet.

3 Statenets - the Analysis Method
Let us start with a definition of the syntax and graphical representation of a statenet, before we explore its execution policy and hence its analysis.

Syntax and Graphical Representation
Definition 1 A statenet SN is a 7-tuple

\[ SN = (S, E, C, P, C, T, I) \]

where S is a set of states, E is a set of arcs, C is the level function, P is the parent function, C is the condition function on the arcs, T is a type function on the states and I is a partitioning function.

The set of states S contains all the states in the system. States may be nested, but there is only one toplevel state.
Figure 1. DQDB Node i

Figure 2. States in a Statenet
Figure 3. Statenet of DQDB Node i
One set of states may be indicated as the start states, and one set as the final states. As states can be nested, the level function $L$ indicates the system-wide nesting depth of a given state. Nested states are known as children of their directly surrounding state, and $P$ returns the parent of a given child.

As in statecharts, the states in a statenet are of two types: XOR and AND states. The type function $T$ associates with each state its type. In AND states, the different substates belong to different categories, in the sense that the states in each AND component form a sequential entity, but the different sequential entities can be executed in parallel. Hence the partitioning function $T$.

The arcs $E$ are directed binary arcs, connecting states in $S$. The arcs of the statenet are much simpler than the arcs in a statechart, where various types of arcs are allowed. The arcs play a major role in the definition of the execution policy of statenets, as we will see below.

Let us define the components of a statenet more formally.

**Definition 2** Let $S$ be the set of states in the statenet $SN$. Then $\epsilon$ is an element of $S$, where $\epsilon$ denotes the empty (or non-existent) state.

**Definition 3** In the statenet $SN$, the level function $L$ takes the set of states to the natural numbers,

$$L : S \rightarrow \mathbb{N},$$

satisfying the toplevel state condition:

$$\exists x \in S : x \neq \epsilon \land L(x) = 0 \land$$

$$\forall y \in S : (x \neq y \land \epsilon \neq y) \Rightarrow L(y) > 0$$

$L$ is then recursively defined as

$$\forall x \in S : L(x) = L(P(x)) + 1$$

Also, $L(\epsilon) = 0$.

**Definition 4** In the statenet $SN$, the parent function $P$, 

$$P : S \rightarrow S,$$

has $P(\epsilon) = \epsilon$, and $P(x) = \epsilon$ iff $L(x) = 0$.

Note that $P$ is not transitive, so that only the immediate parent of a given state is returned by $P$. The preference of the parent function over the original statechart's subblob function ($\sigma : S \rightarrow 2^{S \times S}$, which gives all the children of a state) will become clear when the statenet execution policy is defined below.

**Definition 5** In the statenet $SN$, the type function $T$ is defined as

$$T : S \rightarrow \{\text{AND, XOR}\}$$

**Definition 6** In the statenet $SN$, the partitioning function $T$ is defined as

$$T : S \rightarrow 2^S,$$

where it holds that, if

$$T(x) = \{\{s_0, \ldots, s_{0, m_0}\}, \ldots, \{s_n, \ldots, s_{n, m_n}\}\}$$

then

$$\forall i : 0 \leq i \leq n : L(s_{i, 0}) = L(s_{i, 1}) = \ldots = L(s_{i, m_i})$$

It follows that $L(x) + 1 = L(y)$ for any state $y$ which is a child of $x$. Call $L(x) + 1$ the partition level of state $x$.

In other words, all the states in one partition must be on the same level and all the states in the different partitions of state $x$ must have the same level. Note that this restriction does not occur in the original statechart's partitioning function – again it simplifies the definition of the execution policy of the statenet. Clearly, the restriction does not influence the modelling convenience of the statenet, as this situation occurs by default in modelling.

**Definition 7** The arcs $E$ of a statenet $SN$ is a set of binary directed arcs, with

$$E \subseteq S \times S$$

Arcs are labeled with event names, and may also have a condition attached to the event.

**Definition 8** In a statenet $SN$ the condition function $C$ is

$$C : E \rightarrow 2^S$$

So, when an arc is labeled event/condition, it means that the event can execute if the global system is in any of the states given in the condition. As a shorthand notation, allow the condition to be written as

$$\text{in}(s_1, \ldots, s_j) \land \ldots \land \sim \text{in}(s_k, \ldots, s_l),$$

The condition function $C$ thus controls the execution of a statenet, similar to the markings for transition firing in a Petri net.

We are now ready to consider an execution policy for statenets.

**An Execution Policy for Statenets**

The definition of the semantics of statecharts is quite complex [6, 12]. In (stochastic) Petri nets, on the other hand, the execution is driven by tokens which indicate the currently "enabled" states. Such a token-oriented execution rule is easily definable since there is no state (place or transition) nesting in a Petri net. Nested states in statecharts prohibit such a simplistic firing rule – for example, if a state is enabled, does it mean that its parent is also enabled? Also, arcs are allowed to cross levels, so that it becomes necessary to consider the global system state at all times, with time-step executions.

My approach in the definition of the statenet execution policy will be to make a number of simplifying assumptions first, explain the policy under those assumptions, and then relax the assumptions one by one to reach the full operational description for the statenet.

**Assumption 1** Assume that arcs do not cross levels, that is if $e = (x_1, x_2)$, then $L(x_1) = L(x_2)$.

**Assumption 2** Assume that all start states are on level 1.

**Assumption 3** Assume that there are no conditions on events.
In this simplified statenet the strategy is to put down an initial marking on the toplevel (level 1) states — that is, all the start states are active and all the others are inactive. This marking can be indicated by a vector \( M = [s_0, \ldots, s_n] \), where \( s_i = 0 \) indicates that state \( s_i \) is inactive and \( s_i = 1 \) indicates that state \( s_i \) is active. Indicate this initial marking by \( M_0 \). Now, consider all the arcs connecting level 1 states. Define a function \( \delta \) (called the next function) which, given a marking \( M_i \) and a set of arcs on a given level, produces a marking \( M_{i+1} \) as follows:

**Definition 9** Consider marking

\[
M_i = [s_1, \ldots, s_j, \ldots, s_n]
\]

If \( s_j = 1 \) and there is an arc \( e = (s_j, s_k) \) in \( E \), then \( \delta \) produces the marking \( M_{i+1} \) with the elements \( s_k = 1 \) and \( s_j = 0 \). The other elements of the marking do not change.

These marking changes on one level can be used to construct a reachability graph for that level, similar to the reachability graph of a Petri net. However, the level of a state in a statenet does not carry any significance or meaning as far as the actual problem being modeled is concerned — the number of children in a state simply indicates the complexity (or, the amount of activity) that takes place in that subpart of the statenet. So, it must be possible to combine the reachability graphs for each level of the statenet in order to arrive at a single meaningful state space problem being modeled. A similar problem arose with the definition of synchronous statecharts [7].

Under the three assumptions given above, such a combination is surprisingly simple. The trick is to identify all the lowest-level children states in all the states (i.e. those that do not have any children themselves — call these the atomic states). Find the maximum depth of the atomic states, and call this the depth of the statenet. Now, add an auxiliary component to the marking vector for the depth of the statenet for each atomic state that is on a higher level than the depth of the statenet. The reachability graph for the depth of the statenet will then represent the global state of the system.

If there are conditions on events (assumption 3 above relaxed), the rule for the next function changes slightly:

**Definition 10** Consider marking

\[
M_i = [s_1, \ldots, s_j, \ldots, s_n]
\]

If \( s_j = 1 \) and there is an arc \( e = (s_j, s_k) \) in \( E \), then \( \delta \) produces the marking \( M_{i+1} \) with the elements \( s_k = 1 \) and \( s_j = 0 \). The other elements of the marking do not change.

If there is a condition \( \text{in}_{s_k} \) on arc \( e \), then the marking change will only take place if \( s_k = 1 \) holds in \( M_i \). If there is a condition \( \text{~in}_{s_k} \) on arc \( e \), then the marking change will only take place if \( s_k = 0 \) holds in \( M_i \).

The same analysis method applies to the marking rule above. Assumption 2 above is not restrictive in modelling convenience, and will not be relaxed.

Let us now relax assumption 1. First, consider all the possible ways in which arcs may cross levels:

1. An arc may connect two atomic states on different levels.
2. An arc may connect an atomic state and a non-atomic state on different levels.
3. An arc may connect two non-atomic states on different levels.

Suppose an arc connects two atomic states on different levels. The addition of auxiliary components to the marking vector for the depth of the statenet ensures that this situation is handled by the reachability graph generation algorithm given before.

Now, suppose an arc connects an atomic and a non-atomic state on different levels, originating at the atomic state (the same argument holds if the arc has the reverse direction). The intended meaning of such an arc is that the event causes a state change from the atomic state to the non-atomic state. However, remember that the non-atomic state must have one or more default children states. Therefore, the meaning of the arc from the atomic to the non-atomic state is the same as one or more arcs from the atomic state to the default states of the non-atomic state. Replace the arc with these arcs. Repeat the argument, until the original arc has been replaced by a number of arcs from atomic states to atomic states. The reachability graph generation then proceeds as in the first case.

Suppose an arc connects two non-atomic states on different levels. Replace the arc as for the second case. Then, repeat the procedure on each of the resultant arcs, but with the origin of the arc instead of the destination of the arc. Again, the complete statenet will contain arcs from atomic states to atomic states on the depth of the statenet. The reachability graph generation then proceeds again as before.

The behaviour of the statenet has now been described in terms of state changes caused by events, and these are reflected in marking vectors from which a reachability graph can be constructed. Moreover, the reachability graph is a “flat” representation, similar to the state change reachability graph of a Petri net. To analyse the statenet, one can therefore use the standard Petri net analysis techniques [4].

As an example, let us analyse the DQDB statenet.

**Analysis of the DQDB example**

Refer again to figure 3 of the statenet for the DQDB node \( i \) behaviour. Then the statenet DQDB is given by the 7-tuple

\[
DQDB = (S, E, \mathcal{L}, \mathcal{P}, \mathcal{C}, T, I)
\]

with

\[
S = \{ \epsilon, \text{DQDB}, \text{Idle}, \text{Active}, \text{Countdown}, \text{Transmit}, \text{Idle-MonB}, \text{Idle-MonA}, \text{Idle-REQ+}, \text{Idle-REQ-}, \text{Active-CopyB}, \text{Countdown-MonB}, \text{Countdown-REQ+}, \text{Countdown-MonA}, \text{Countdown-TestCD}, \text{Transmit}, \text{Transmit-Send}, \text{Transmit-Send-AsmP}, \text{Transmit-Send-ChB} \}
\]

and

\[
E = \{ (\text{Idle,Active}), (\text{Active,Countdown}) \}
\]
Then, the level of state DQDB is 0, the level of states Idle, Active, Countdown and Transmit is 1, the level of states Transmit-Send-AsmP and Transmit-Send-ChB is 3, and the level of all the other states is 2. State Idle is the start state, and there is no final state. All the level 2 states are atomic, except for state Transmit-Send, and the depth of the statenets is 3. To simplify the construction of the reachability graph, number the states of statenets DQDB from 1 to 19, in the order in which they are given in the definition above. Then states 7 to 15 and states 18 and 19 are in the reachability graph of the statenet, with states 7 to 15 as auxiliary components in the 11-component marking vector. Number the marking vectors as follows:

1. 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
2. 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
3. 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4. 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
5. 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
6. 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
7. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
8. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
9. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

The reachability graph of the statenet DQDB is given in figure 4.

The statenet can at this stage not be used for performance analysis, since no probabilistic information is available. What would the addition of probabilistic information to the statenet entail?

**Stochastic Statenets**

Statenets can be extended to stochastic statenets in much the same way that Petri nets are extended to stochastic Petri nets [17].

Consider the statenet that results from the arc replacement step in the final reachability graph generation algorithm described in the previous section. With each event connecting atomic states, associate a random variable which represents the “firing” delay of the event.

Since the reachability graph of a stochastic statenet corresponds closely to the reachability graph of a safe and bounded stochastic Petri net, it is possible to apply the analysis techniques for stochastic Petri nets to stochastic statenets. This analysis is even simpler, since Petri net
structural specific problems (such as conflict) cannot occur in a statenet.

I am currently investigating such analysis methods, and summarize some of these issues in the conclusion.

4 Conclusion

In this paper I introduced the stochastic statenet as a unified approach to design, correctness analysis and performance analysis. The design suitability of the statenet was illustrated with an example, and some mathematical definitions were given for the stochastic statenet.

Several problems still exist before the stochastic statenet concept can be implemented as one unified design and analysis system. These include:

- The extension of the statenet with more modelling convenience, closer to the statechart.
- The comparison of the statenet with other modelling techniques and languages, such as SPNs, Esterel [2] and Argonaute [15].
- Further investigation into the mathematical properties of statenets, in order to develop a theory of statenets.
- The implementation of a unified design and analysis environment for statenets.
- The application of the statenet methodology to large practical problems.

References

8. T R G Green, 'Pictures of programs and other processes, or how to do things with lines', Behavior Inform. Tech. 1:3–36.

Acknowledgements: The author wishes to thank the Department of Computer Science, University of Stellenbosch, for their financial support.
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 papers. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or 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 1 1/2 or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - 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. Figures should be submitted as original line drawings/printouts, and not photocopies.
- 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 in square brackets (1, 2, 3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):
1. As (a) \( \TeX \) file(s), either on a diskette, or via e-mail/tftp – a \( \TeX \) style file is available from the production editor;
2. In camera-ready format – a detailed page specification is available from the production editor;
3. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.
Further instructions on how to reduce page charges can be obtained from the production editor.
4. In a typed form, suitable for scanning.

Charges
Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for \( \TeX \) or camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

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

Proofs
Proofs of accepted papers in categories 3 and 4 above will 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.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications
Letters to the editor are welcomed. They should be signed, and should be limited to less than 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 specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References
## Contents

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward</td>
<td>i</td>
</tr>
<tr>
<td>Referees</td>
<td>ii</td>
</tr>
<tr>
<td>Machine Translation from African Languages to English</td>
<td>1</td>
</tr>
<tr>
<td>DG Kourie, WJ vd Heever and GD Oosthuizen</td>
<td></td>
</tr>
<tr>
<td>Automatically Linking Words and Concepts in an Afrikaans Dictionary</td>
<td>9</td>
</tr>
<tr>
<td>PZ Theron and I Cloete</td>
<td></td>
</tr>
<tr>
<td>A Lattice-Theoretic Model for Relational Database Security</td>
<td>15</td>
</tr>
<tr>
<td>A Melton and S Shenoi</td>
<td></td>
</tr>
<tr>
<td>Network Partitions in Distributed Databases</td>
<td>22</td>
</tr>
<tr>
<td>HL Viktor and MH Rennhackkamp</td>
<td></td>
</tr>
<tr>
<td>A Model for Object-Oriented Databases</td>
<td>27</td>
</tr>
<tr>
<td>MM Brand and PT Wood</td>
<td></td>
</tr>
<tr>
<td>Quantifier Elimination in Second Order Predicate Logic</td>
<td>35</td>
</tr>
<tr>
<td>D Gabbay and HJ Ohlbach</td>
<td></td>
</tr>
<tr>
<td>Animating Neural Network Training</td>
<td>44</td>
</tr>
<tr>
<td>E van der Poel and I Cloete</td>
<td></td>
</tr>
<tr>
<td>HiLOG - a Higher Order Logic Programming Language</td>
<td>53</td>
</tr>
<tr>
<td>RA Paterson-Jones and PT Wood</td>
<td></td>
</tr>
<tr>
<td>ESML - A Validation Language for Concurrent Systems</td>
<td>59</td>
</tr>
<tr>
<td>PJA de Villiers and WC Visser</td>
<td></td>
</tr>
<tr>
<td>Semantic Constructs for a Persistent Programming Language</td>
<td>65</td>
</tr>
<tr>
<td>SB Sparg and S Berman</td>
<td></td>
</tr>
<tr>
<td>The Multiserver Station with Dynamic Concurrency Constraints</td>
<td>75</td>
</tr>
<tr>
<td>CF Kriel and AE Krzesinski</td>
<td></td>
</tr>
<tr>
<td>Mechanizing Execution Sequence Semantics in HOL</td>
<td>81</td>
</tr>
<tr>
<td>G Tredoux</td>
<td></td>
</tr>
<tr>
<td>Statenets - an Alternative Modelling Mechanism for Performance Analysis</td>
<td>87</td>
</tr>
<tr>
<td>L Lewis</td>
<td></td>
</tr>
<tr>
<td>From Batch to Distributed Image Processing: Remote Sensing for Mineral Exploration 1972-1992</td>
<td>95</td>
</tr>
<tr>
<td>N Pendock</td>
<td></td>
</tr>
<tr>
<td>Galileo: Experimenting with Graphical User Interfaces</td>
<td>99</td>
</tr>
<tr>
<td>R Apteker and JM Bishop</td>
<td></td>
</tr>
<tr>
<td>Placing Processes in a Transputer-based Linda Programming Environment</td>
<td>109</td>
</tr>
<tr>
<td>PG Clayton, FK de-Heer-Menlah, EP Wentworth</td>
<td></td>
</tr>
<tr>
<td>Accessing Subroutine Libraries on a Network</td>
<td>117</td>
</tr>
<tr>
<td>PH Greenwood and PH Nash</td>
<td></td>
</tr>
<tr>
<td>A Multi-Tasking Operating System Above MS-DOS</td>
<td>122</td>
</tr>
<tr>
<td>R Foss, GM Rehmet and RC Watkins</td>
<td></td>
</tr>
<tr>
<td>Using Information Systems Methodology to Design an Instructional System</td>
<td>126</td>
</tr>
<tr>
<td>BC O'Donovan</td>
<td></td>
</tr>
<tr>
<td>Managing Methods Creatively</td>
<td>131</td>
</tr>
<tr>
<td>G McLeod</td>
<td></td>
</tr>
<tr>
<td>A General Building Block for Distributed System Management</td>
<td>141</td>
</tr>
<tr>
<td>P Putter and JD Roos</td>
<td></td>
</tr>
</tbody>
</table>