ALLOS - A TOOL TO SOLVE MARKOVIAN MODELS

Stanley R. M. Oliveira[1], Maria I. C. Cabral[2], Edilson Ferneda[2], Marcos A. G. Brasileiro[3]
stanley@dsc.ufpb.br, izabel@dsc.ufpb.br, edilson@dsc.ufpb.br, mbrasile@brufpb2.bitnet

[1] CNPTIA/EMBRAPA
Rod. D. Pedero I - Km 143,6 (SP65)
13089-500 - Campinas, SP
Brazil
Phone: ++55 19 240-1073
FAX: ++55 19 240-2007

[2] DSC/CCT/UFPB
Avenida Aprígio Veloso s/n
58.109-970 Campina Grande, PB
Brazil
Phone/FAX: ++55 83 333-1698

[3] DEE/CCT/UFPB
Avenida Aprígio Veloso s/n
58.109-970 Campina Grande, PB
Brazil
Phone: ++55 83 333-1992
FAX: ++55 83 333-1650

ABSTRACT

This work presents a friendly tool usable to solve models for networks of queues using Markov chains. The state space generation process for a specific model is automaticaly performed from a given initial state, according to the rules that describe the behavior of that model. The use of this tool is adequate for solving systems models which exhibit resource contention, as for example, computer systems, computer networks, manufacturing systems, process control and other which can be modeled using the networks of queues paradigm.

Keywords: Modeling, Markov Chains, Performance Evaluation, Automatic State Generation.

1. INTRODUCTION

Modeling and analysis for computer systems have been lately impacted by researchers who wish to predict the behavior of such systems **[1]**.

Systems which exhibit resource contention (as for example, computer systems, computer networks, manufacturing systems and process control) can be modeled using the networks of queues paradigm **[2,3]**. A network of queues is a system in which multiples queues operate assynchronously and concurrently, interconnected according to a specific topology. Each system of queue is composed by customers, a set of servers and a service discipline **[2,4]**.

Solving models of networks of queues is not an easy task, becoming restrained only to researchers who have the knowledge of the performance evaluation techniques found in the literature. Analytical techniques can be used in order to solve these models. These techniques are economic and efficient. However, they sometimes demand a simplification of the model due to its complexity. Numerical, approximate techniques can also be used, but their computational cost can be a limiting factor.

One of the analytical alternatives to solve networks of queues is the use of markovian processes. A process is named markovian if, from a given state, the future bahavior of the process is independent of the past events and depends upon only on the present state of the process **[1,4]**. This class of models is widely used in analysis and modeling of computer and communication systems and is sufficiently suitable in the majority of the cases **[5]**.

There are two major classes of markovian models, continuous times and discret times models **[1,2]**. In the first case, the state of the system can change at any instant of time; on the second case, the state of the system can change only at specific points of time. A markovian model which presents a discret state space is named a Markov chain.

Traditionally, either analytical techniques or general purpose simulation languages, such as *GPSS* **[6]** and *SIMULA* **[7]**, were used for modeling systems of networks of queues. Nowadays, the trend is the use of integrated environments that offer approximate and analytical solutions and provide a suitable way for developing powerful models, allowing the users to build, modify and represent models of the real world. Examples of some of these environments are *SAVAD* **[8.9]**, *RESQ* **[10]**, *BONes* and *OPNET* **[11]**.

In order to build such environments, this paper proposes structured techniques and facilities of interation with the computational environment which grant the software to be modular, extensible and reusable **[12,13]**.

This work presents the tool *ALLOS*, which automaticaly generates states for markovian models, solves them and presents the more important performance measures as requested by the user. The state space generation process for a specific model is automatic from a given initial state, according to the rules that describe the behavior of that model.

ALLOS is part of an intelligent simulation environment denominated *SAVAD*. This tool has been developed using the *C++* **[13]** and *PROLOG* **[14]** languages to microcomputer systems, compatible with the line IBM/PC. *ALLOS* offers an intelligent, friendly interface to the user. Therefore, the user has not to be an expert in modeling and performance evaluation of systems. The components used to describe models for networks of queues present a great flexibility, allowing the modeling of a large diversity of systems that exhibit resource contention , as for example, computer systems, computer networks, manufacturing systems and others which can be modeled using the networks of queues paradigm.

It is important to emphasize that markovian models have caused impact in the researches concerning the area of performance evaluation of systems which exhibit resource contention, since that simulation programs, when compared to mathematical models, are more expensive to build, are error-prone, difficult to be validated and costly in terms of CPU execution cycles **[1]**. Besides, a markovian model can be altered (e.g., truncating the state space) so that this model becomes more treatable.

The decline of costs for computer memory, the increase of the processing capacity and the advance of techniques to solve models allow that markovian processes with a large number of states be solved, reducing the strong restriction imposed by markovian modeling that is the state explosion **[1,15]**.

In this paper we describe the tool *ALLOS* and we present an example of its application in modeling. In the section 2, we present the modeling process using *ALLOS*, emphesizing the automatic state generation process. A numerical example is shown in the section 3, where we point out the solution and validation of this tool and, finally, in the section 4, we present our conclusions.

2. THE MODELING PROCESS USING *ALLOS*

2.1. Modeling Elements

The *ALLOS* interface allows the user, in an interactive fashion, to describe his model and to store/present attributes and parameters of elements used in the modeling. It validates automaticaly the modeling elements during the construction of the model and validates it globally before submiting it to an evaluation. The set of available elements allows the construction of models for networks of queues in a simple way. A brief description of such elements follows. More details can be found in **[8,9]**.

Customers: They are temporary entities that circulate through model nodes (elements) requesting a specific service. The path followed by a customer is named its *route*. Customers in a same class have same attributes (e.g. service distribution probability, queuing discipline, priority level, etc).

Workstations: They represent resources in a queuing model. There is a queuing discipline associated with each workstation. ALLOS uses the queuing discipline FCFS (First Come First Served). In adittion, each workstation is composed of one or more servers (entities attending customers). Two types of workstations were defined in ALLOS: single server and multiple servers. Each queue associated with a single server or multiple servers may have a length of up to K customers units (storage capacity), where K must be specified by the user.

Control Points: They are nodes that control the flow of customers within a network of queues. There are four types of control-points, designed to facilitate the task of computer networks modeling, particularly those presenting integration of services:

- **Multiplication Point:** Enables a replication of customers arriving to this node. Its structure is composed of one input and two or more outputs each one eventually belonging to a different route.
- **Fusion Point**: Makes the fusion of one or more customers belonging to the same class, to compose a single customer.
- **Synchronization Point**: This element is used to block customers in a queuing system model while waiting for a certain condition to be satisfied; after that, customers are freed according to a liberation discipline.
- **Scheduler Point**: Schedules customers (waiting in different queues) according to a liberation discipline where each non empty queue may try to liberate one or more customers from its queue. Consequently, collisions may occur with this discipline. In such cases those collision customers wait for a a new liberation according to a specific algorithm.

Sources: For open queue networks, source represents the process of customers arriving into a system. Generally, the arrival process is described in terms of the probability distribution of customer interarrival times.

Sinks: They are nodes (in open queue networks) where customers leave the model.

Classes: Usually, while building a model for a network of queues, one needs to describe classes of customers traversing the model. Associated with each customer class there are a name and one or more routes.

2.2. Automatic State Generation

Models of real world systems generally originate a Markov chain with a large number of states. A problem not less important refers to transition rates and automatic state generation **[1]**. It is very important that the user can specify the model in a friendly way and that the performance measures of interest can also be observed automaticaly, not demanding an user knowledge about details of the mathematical representation.

In many models for computer systems, most of the time is spent in a number relatively small of states in comparison with the total number of model states. For such models, several performance measures of interest can be calculated from a number relatively small of states **[1]**.

The basic idea of dynamic exploration techniques is to develop algorithms to guide the generation of transition rate matrix. The importance of a state should be eveluated in terms of its contribution to calculate the performance measures of interest, as for example, throughput, mean queue length, mean queuing time and utilization factor **[1,15]**.

An elegant and efficient way to dynamic exploration is the automatic state generation, where from the specification of a system model by the user, the state transition matrix is generated from a given initial state, according to the rules that describe the behavior of such model.

The language selected for implementing the automatic state generation module is *Prolog*. According to discussions in **[5]**, three features that Prolog provides make it appropriate for implementing *ALLOS*. First, *Prolog* allows the use of untyped complex data structures which allows the user full freedom in the descriptions of model states. Second, *Prolog* provides unification, a powerful form of pattern matching used in the preconditions of the rules to determine which actions can be taken and, finally, *Prolog* has backtracking search as a basic feature of the language. As more than one rule may have its preconditions satisfied at the same time, the backtracking automatically applies all possible such rules to find all reachable states.

2.3 Procedure for Automatic State Generation

Consider Figure 1 which represents the available state transitions of models that use the modelig elements presented in the section §2.1.

$$[\text{queue}_i]\left\{\begin{array}{c}\text{source}\\ \text{sink}\\ \text{workstation}\\ \text{multiplication}\\ \text{scheduler}\\ \text{synchronization}\\ \text{fusion}\end{array}\right\}[\text{queue}_j]$$

Figure 1: Modeling elements

In this convention, each modeling element can be preceded and followed by one or more queues. Exceptions are applied to elements source (it can not be preceded by a queue) and sink (it can not be followed by a queue). The notation used in this case is "[ ]" which represents a null queue.

From a state $(k_1,\dots,k_n)$, where n is the number of model queues, the set of reachable states for that model is defined as follow:

1. [ ] source $[f_j]$
If $k_{f_i} < Max_{f_i}$
Then $(k_{f_1},\dots,k_{f_j},\dots,k_{f_n}) \underset{1}{\Rightarrow} (k_{f_1},\dots,k_{f_j}+1,\dots,k_{f_n})$

2. $[f_i]$ sink [ ]
If $k_{f_i} > 0$
Then $(k_{f_1},\dots,k_{f_i},\dots,k_{f_n}) \underset{2}{\Rightarrow} (k_{f_i},\dots, k_{f_i}-1,\dots,k_{f_n})$

3. $[f_i]$ workstation $[f_j]$
If $(k_{f_i} > 0) \wedge (k_{f_j} < Max_{f_j})$
Then $(k_{f_1},\dots,k_{f_i},\dots,k_{f_j},\dots,k_{f_n}) \underset{3}{\Rightarrow} (k_{f_1},\dots,k_{f_i}-1,\dots,k_{f_j}+1,\dots,k_{f_n})$

4. $[f_i]$ multiplication $[f_{j_1},\dots,f_{j_m}]$
If $k_{f_i} > 0$
Then $(k_{f_1},\dots,k_{f_i},\dots,k_{f_{j_\alpha}},\dots,k_{f_n}) \underset{4}{\Rightarrow} (k_{f_1},\dots,k_{f_i}-1,\dots,k_{f_{j_\alpha}}+1,\dots,k_{f_n})$,
$\forall\alpha \mid 1 \le \alpha \le m, k_{f_{j_\alpha}} < Max_{f_{j_\alpha}}$

5. $[f_{i_1},\dots,f_{i_m}]$ scheduler $[f_j]$
$\forall\alpha \mid 1 \le \alpha \le m,$
If $(k_{f_{i_\alpha}} \ge SchP_{f_{i_\alpha}}) \wedge (k_{f_j} + SchP_{f_{i_\alpha}} \le Max_{f_j})$
Then $(k_{f_1},\dots,k_{f_{i_\alpha}},\dots,k_{f_j},\dots,k_{f_n}) \underset{5}{\Rightarrow} (k_{f_1},\dots,k_{f_{i_\alpha}}-1,\dots,k_{f_j}+1,\dots,k_{f_n})$

6. $[f_{i_1},\dots,f_{i_m}]$ synchronization $[f_{j_1},\dots,f_{j_p}]$
If $(\forall\alpha \mid 1 \le \alpha \le m, k_{f_{i_\alpha}} \ge SynP1_{f_{i_\alpha}}) \wedge$
$(\forall\beta \mid 1 \le \beta \le p, k_{f_{j_\beta}} + SynP2_{f_{j_\beta}} \le Max_{f_{j_\beta}})$
Then $(k_{f_1},\dots,k_{f_{i_\alpha}},\dots,k_{f_{j_\beta}},\dots,k_{f_n}) \underset{6}{\Rightarrow}$
$(k_{f_1},\dots,k_{f_{i_\alpha}} - SynP1_{f_{i_\alpha}},\dots,k_{f_{j_\beta}} + SynP2_{f_{j_\beta}},\dots,k_{f_n})$

7. $[f_{i_1},\dots,f_{i_m}]$ fusion $[f_j]$
If $(\forall\alpha \mid 1 \le \alpha \le m, k_{f_{i_\alpha}} \ge 0) \wedge (k_{f_j} < Max_{f_j})$
Then $(k_{f_1},\dots,k_{f_{i_\alpha}},\dots,k_{f_j},\dots,k_{f_n}) \underset{7}{\Rightarrow} (k_{f_1},\dots,k_{f_{i_\alpha}}-1,\dots,k_{f_j}+1,\dots,k_{f_n})$,
$\forall\alpha \mid 1 \le \alpha \le m$

Where: k_i is the queue lenth i, Max_i is the maximum queuing length i, $SchP_i$ is the parameter associated with the liberation of customers of queue i at the scheduler point, $SynP1_i$ is the parameter associated with the liberation of customers of queue i at the synchronization point, $SynP2_j$ is the parameter associated with the liberation of customers at the synchronization point to the queue j, and where $E_a \overset{1}{\underset{i}{\Rightarrow}} E_b$, or simply $E_a \underset{i}{\Rightarrow} E_b$, represents the existence of one direct transition refering to modeling element i, between the states E_a e E_b.

Let E_a e E_b be two states. We say that E_b is *reachable* from E_a ($E_a \overset{*}{\Rightarrow} E_b$) if and only if there is a finite number of transtions between E_a and E_b ($E_a \Rightarrow E_{a+1} \Rightarrow \dots \Rightarrow E_b$). The transition from the state E_a to the state E_b, is considered *valid* ($E_a \overset{val}{\Rightarrow} E_b$) if and only if $\nexists E_c \mid E_b \overset{*}{\underset{i}{\Rightarrow}} E_c, 5 \leq i \leq 7$. So, the generation of a valid transition is done according to the following rule:

```
If $\exists E_c \mid ((E_b \overset{*}{\underset{i}{\Rightarrow}} E_c, 5 \leq i \leq 7) \wedge \nexists E_d \mid ((E_c \underset{i}{\Rightarrow} E_d, 5 \leq i \leq 7))$
    Then $E_a \underset{val}{\Rightarrow} E_c$
    Else $E_a \underset{val}{\Rightarrow} E_b$, since that $\nexists E_c \mid E_b \overset{*}{\underset{i}{\Rightarrow}} E_c, 5 \leq i \leq 7$
```

That rule can be still enriched through consideration of restrictions. For example, the restriction $\sum_{i=1,n} k_i = population$, where k represents the queue length i and n is the total number of model queues, define a model for closed network of queues.

Therefore, given a network of queues F, an initial state E_0 is defined generically as:

```
If F is open (contains elements of type *source*)
    Then $E_0 = (0,\ldots,0)$
    Else $E_0 = (P,0,\ldots,0)$
```

where P represents the population of the closed chain. If $P > K_1$, then the excess will pass to K_2, and so on while $P < K_i$. That initial state can also be defined differently in the presence of restrictions.

Finally, the Markov chain generation procedure is defined as:

A set of states generated from a unique initial state E_0,

```
builds-states([E|Es]) :-
    If E was not yet built
        Then  $X = \{State : E \underset{val}{\Rightarrow} State\}$
              builds-states($Es \circ X$)
        Else  builds-states($Es$)
```

where $A \circ B$ represents a concatenation of the lists A and B.

3. Numerical Example

According to Figure 2, we choose an open network with one multiplication point (applied in some cases to broadcasting models). The modeling elements for this example are one source, three workstations each one of them with a single server, a multiplication point, and two sinks. This model presents two routes (route1 and route2). The customers of route1 leaving the workstation Serv1 are duplicated at Mult_P, and end up at Sink1, while their duplicates follow route2 starting at Mult_P and going to Sink2, where they leave the system.

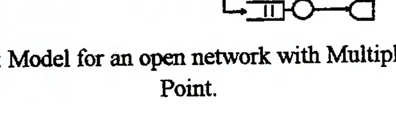

Figure 2: Model for an open network with Multiplication Point.

We now summarize the specification of each one of model elements, that could be submitted by an *ALLOS* user:

Source:
Name: Source1
Probability distribution: exponential
Average: 1.0

Workstations:
Name: Serv1
Kind of server: single
Queue length (k): 3
Queuing discipline: FCFS
Service distribution: exponencial
Average: 0.5 (package/msegs)

Name: Serv2
Kind of server: single
Queue length (k): 3
Queuing discipline: FCFS
Service distribution: exponencial
Average: 0.5 (package/msegs)

Name: Serv3
Kind of server: single
Queue length (k): 3
Queuing discipline: FCFS
Service distribution: exponencial
Average: 0.5 (package/msegs)

Sinks:
Name: Sink1

Name: Sink2

Multiplication Point:
Name: Mult_P

Class:
Name: class1
Priority: no priority (priority: 0)

Routes:
Name: route1
Class: class1
Nodes sequence:
Source1 >> Serv1 >> Mult_P >> Serv2 >> Sink1

Name: route2
Class: class1
Nodes sequence: Mult_P >> Serv3 >> Sink2

The space state associated with the model is shown in Figure 3.

1 [0,0,0]	2 [1,0,0]	3 [2,0,0]	4 [0,1,1]	5 [1,1,1]	6 [0,0,1]
7 [0,1,0]	8 [3,0,0]	9 [1,0,1]	10 [1,1,0]	11 [2,1,1]	12 [0,2,2]
13 [2,0,1]	14 [0,1,2]	15 [2,1,0]	16 [0,2,1]	17 [1,2,2]	18 [3,1,1]
19 [1,1,2]	20 [0,0,2]	21 [3,0,1]	22 [1,2,1]	23 [0,2,0]	24 [3,1,0]
25 [2,2,2]	26 [0,3,3]	27 [1,0,2]	28 [2,1,2]	29 [0,2,3]	30 [1,2,0]
31 [2,2,1]	32 [0,3,2]	33 [1,3,3]	34 [3,2,2]	35 [2,0,2]	36 [0,1,3]
37 [1,2,3]	38 [3,1,2]	39 [2,2,0]	40 [0,3,1]	41 [1,3,2]	42 [3,2,1]
43 [2,3,3]	44 [1,1,3]	45 [0,0,3]	46 [3,0,2]	47 [2,2,3]	48 [1,3,1]
49 [0,3,0]	50 [3,2,0]	51 [2,3,2]	52 [3,3,3]	53 [1,0,3]	54 [2,1,3]
55 [3,2,3]	56 [1,3,0]	57 [2,3,1]	58 [3,3,2]	59 [2,0,3]	60 [3,1,3]
61 [2,3,0]	62 [3,3,1]	63 [3,0,3]	64 [3,3,0]		

Figure 3: State space for the open network model with mutiplication point.

From the automatic state generation for the state space which will compose the model's transition rate matrix, *ALLOS* solves the model using the method presented in [3]. Other methods to solve transition rate matrix can be found in [1].

As an example of performance measures of interest which can be obtained, we have the utilization factor in each workstation (server), the mean answer time in such workstation and the throughput:

— Utilization factor in each workstation (Serv1, Serv2, Serv3) = 0.498266

— Mean queue length, including job in service (Serv1, Serv2, Serv3) = 0.852150

— Throughput for the open network model with multiplication point = 0.911754

The results above were compared with [16], in order to valid the tool *ALLOS* which is part of intelligent simulation environment entitled *SAVAD*.

4. Conclusions

As for the user, it is desirable that a tool for modeling networks of queues be characterized by transparency of the details both the mathematical definitions of the modeled system and solution techniques, so that the user will not need to be an expert in these techniques. In this sense, *ALLOS* with its friendly interface, can easily be used for solving model networks of queues by means of Markov chains.

ALLOS is part of the intelligent simulation environment *SAVAD*. This tool automaticaly generates the states of markovian models, from a given initial state,.according to the rules that describe the behavior of the models, solves them and presents the performance measures requested by the user. The use of this tool is adequate for solving systems models which exhibit resource contention, as for example, computer systems, computer networks, manufacturing systems, process control and others which can be modeled using the networks of queues paradigm.

ALLOS has been validated through exhaustive tests, and results compared with the simulator for network of queues presented in [16].

The selection of the *Prolog* and *C++* languages allows more flexibility in the implementation of this tool. The languages also favored the adoption of design and programming methodologies granting the software to be modular and reusable, so that it easily allows extensions which can enrich its options of using.

Aiming at the widest possible dissemination, the tool *ALLOS* was designed to be used in microcomputers compatible with the line IBM/PC. Now, we intend to extend it to Unix-workstations.

References

[1] E. de S. Silva; R. R. Muntz, "Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação", *VIII Escola de Computação*, Gramado (RS), 1992.

[2] L. Kleinrock, "Queueing Systems", Vol. 1: Theory. Wiley Interscience, New York (USA), 1975.

[3] C. H. Sauer; K. M. Chandy, "Computer Systems Performance Modeling", Prentice-Hall, Englewood Cliffs, New Jersey (USA), 1981.

[4] H. Kobayashi, "Modeling and Analysis: An Introduction to System Performance Evaluation Methodology", Addison-Wesley, New york (USA), 1978.

[5] M. C. Diniz; E. de S. Silva, "Especificação e Geração de Modelos Markovianos para Análise de Desempenho e Confiabilidade de Sistemas", *Revista Brasileira de Computação*, Vol. 6, n. 3, pp. 23-42, jan/mar, 1991.

[6] T. J. Shriber, "Simulation Using GPSS", John Willey & Sons, 1974.

[7] G. M. Birtwistle et al., "Simula Begin", Auerbach Publisher, Philadelphia (USA), 1973.

[8] M. I. C. Cabral; F. A. C. Souto; H. C. Castro Filho; H. de M. Silva; M. A. G. Brasileiro, "An integrated system for modeling and evaluating models of networks of queues", in M. H. Hamza (Editor), *Proceedings of the IASTED International Conference - Modeling and Simulation - MS'94 - Pittsburgh (USA)*, IASTED, Anaheim (USA), 1994, pp. 206-209.

[9] M. M. Dias; M. A. G. Brasileiro; M. I. C. Cabral, "An Expert System for Performance Evaluation of Computer System Models", *Summer Computer Simulation Conference*, Reno (USA), 1992.

[10] C. H. Sauer; E. MacNair, "The Evaluation of the Research Queueing Package RESQ", in *Modelling Techniques and Tools for Performance Analysis*, North-Holland, 1985, pp. 5-24.

[11] M. G. McComas; A. M. Law, "Simulation Software for Communication Networks: The State of the Art", *IEEE Communication Magazine*, March, 1994, pp. 44-50.

[12] G. Booch, "Object Oriented Design", The Benjamim/ Cummings Publishing, 1991.

[13] B. Stroustrup, "The C++ Programming Language", Addison-Wesley, 1991.

[14] W. F. Clocksin; C. S. Mellish, "Programming in Prolog", Springer-Verlag, 1981.

[15] E. de S. Silva; P. M. Ochoa, "State Space Exploration in Markov Models", *Performance Evaluation Review*, Vol. 20, n. 1, June, 1992, pp. 152-166.

[16] H. C. Conceição Filho, "SIM/SAVAD - Um Simulador de Modelos de Redes de Filas". Campina Grande, Paraíba, Brazil: COPIN/CCT/UFPB, 1993. (MSc thesis).