Markovian Component Modeling

F. Dankar and G.V. Bochmann
School of Information Technology and Engineering (SITE), University of Ottawa
Email: fdankar@site.uottawa.ca, bochmann@site.uottawa.ca

Abstract

We consider the following problem: For a real-time probabilistic system S consisting of two submodules M_1 and M_2, the specification of the global system S is given, as well as part of the specification of M_1 and part of the specification of M_2 (the possible traces are known but not their probabilities nor their timing delays). We need to fully determine of M_1 and M_2 in a way to "best approximate" the composition $M_1 \times M_2$ to S.
In this paper, a solution of this problem in the context of continuous time Markov chains (CTMC) is given.

1. Introduction

Modern systems tend to be more and more complex, hence resulting in large and complex models. The notion of decomposing a system into smaller subsystems is well known and results in smaller and easier to model components.

An instance of this problem is submodule construction. The submodule construction problem consists of constructing the specification of a submodule X when the specifications of system S and all its submodules except X are given. The construction should be done in a way such that the composition of the known submodules with X conforms to S. Submodule construction was first formulated and treated in [7], where the specifications are given as execution sequences and trace equivalence is the conformance relation. Some work was done using labeled transition systems as a model for the specification and observational equivalence as the conformance relation [6,9] and other work was done for I/O finite state machines as a model for the specifications and quasi-equivalence as conformance relation [4,8].

The problem that we deal with consists of determining the performance of all submodules of a real time probabilistic system S when the specification of system S is known as well as the traces of all its submodules. The exact problem we deal with is stated as follows:

A continuous-time Markov chain (CTMC) S is designed as a composition of two CTMC submodules M_1 and M_2. The specification of S is given as well as the traces for M_1 and M_2 (in M_1 and M_2 possible transitions are known, but not their probabilities nor their timing delays, which are expressed in terms of transition rates). We need to determine the unknown rates in M_1 and M_2 such that $M_1 \times M_2$ behaves at least "as good as" S. Note that, given S and M_1 the transitions in M_2 can be constructed using submodule construction [1,7].

To design S as a product of two components M_1 and M_2, some internal interactions are required between M_1 and M_2 for the purpose of synchronization. When the product of M_1 and M_2 is made these internal interactions are hidden and they become internal delays.

Some of the challenges faced while determining the unknown rates in M_1 and M_2, are:

- Removing the internal delays from the composition $M_1 \times M_2$ (we use an algorithm defined in [3]).
- Proving that the characteristics of $(M_1 \times M_2)$ is not altered after removing its internal delays. To do that, we define two equivalences, namely "average simulation equivalence" and "average delay equivalence", that are preserved with the removal of internal delays.
- Prove some important properties for the equivalences.

In the coming section we will present some basic definitions needed in the course of the paper, then in Section 3 we will present a motivating example to better explain the purpose of the paper. In Sections 3 and 4 we will proceed with the solution steps. Then in Section 5 two examples are presented.

2. RTFSM, CTMC and DTMC

2.1. RTFSM

A real-time finite state machine RTFSM M= (S, $\sum$, Δ, T, P, s_0) is defined as follows:

- $\sum$ is a set of events,
- S is a finite set of states
- $\Delta \subseteq S x \sum x S$ is the transition relation satisfying the following condition: Given $s \in S$, the set $\{ a \in \sum ; (s, a, s') \in \Delta$, for some $s' \in S\}$ is finite. Note: This is referred to as the local finite branching property.
- T: $S \rightarrow Q$(rational numbers) is the exact time spent in every state.
- P: $\Delta \rightarrow [0,1]$ is the probability function, which assigns a probability to every transition. The following condition applies, for all states $s \in S$, $\Sigma_{\text{all transitions } t \subset \Delta \text{ from } s} P(t)=1$.
- s_0 is the initial state

2.2. CTMC

We follow the same ideas as in [10]:
A Continuous Time Markov Chain (CTMC) is a tuple (S, $\sum$, Δ, μ, s_0) where:

- $\sum$, S, Δ and s_0 are defined as in Section 2.1
- μ: $\Delta \rightarrow [0,\infty)$ is the transition rate function

- The exit rate of a state s is the sum of the rates of all the activities enabled from s, i.e. $\mu(s)= \Sigma_{e \in \Sigma} \mu(s,e,s')$, for all s'.
- The rate of going from state s to state r is denoted by $\mu(s,r)= \Sigma_{e \in \Sigma} \mu(s,e,r)$
- Given that we are in a particular state x, with exit rate $\mu(x)$, the probability that a certain transition (e, $\mu(e)$) occurs is:
  $P_e = \mu(e) / \mu(x)$, Moreover $\Sigma_{\text{all transitions from } x} \mu(e) / \mu(x)=1$.
  Hence, an alternative definition for a CTMC would give the exit rate r_x for every state x together with the probability for each transition e from x. The rate of the transition e from x would be then: $r_e=P_e r_x$.
- If $\mu(x)$ is the rate of state x, then the amount of time spent on average in state x is $1/\mu(x)$. So a CTMC and an RTFSM differ in the delay time at their states, In an RTFSM it is constant, while in a CTMC it is not constant, it is rather random with given exponential distribution.
- The probability of going from state s to state r is denoted by $P(s,r)= \Sigma_{e \in \Sigma} P(s,e,r)$ where P(s,e,r) is the probability of going from state s to state r through transition e.

2.3. DTMC

A discrete time Markov chain (DTMC) is a tuple (S, $\sum$, Δ, P, s_0) where:

- $\sum$, S, Δ, P and s_0 are defined as in Section 2.1

So a DTMC is an RTFSM, but in a DTMC, the delay time spent at any state is the same and is equal to the time unit, i.e., at each unit of time, a transition takes place. For details see [2]

2.4. Traces

An execution sequence E of a CTMC M is an alternating finite sequence of states and events with their corresponding rates, of the form:

$E = s_0 \xrightarrow{(e1,r1)} s_1 \xrightarrow{(e2,r2)} \ldots s_n$ where $r_i = \mu(e_i, s_i, s_{i+1})$.

We use the following terminology:

- The trace of E written tr(E), is the sequence of actions: $e_0, e_1, \ldots$ appearing in E.
- Probability of E, or P(E), is the probability of getting the execution sequence E which is the product of the probabilities of all the transitions in E: $P(E) = r_1/\mu(s_0) \ldots r_n/\mu(s_{n-1})$.
- The average delay time of E, or ADT(E), is the average total time spent before we enter the final state s_n, of E, which is the summation of the ADT spent in each state: $ADT(E) = 1/\mu(s_0) + \ldots + 1/\mu(s_{n-1})$

2.5. Composition

Let M_1 and M_2 be two CTMC, where:
$M_1= (S_1, \sum_1, \Delta_1, \mu_1, s_0^1)$ and $M_2= (S_2, \sum_2, \Delta_2, \mu_2, s_0^2)$, we will adopt the definition of a composition from [2]:
The composition $M_1 \times M_2$ is a CTMC M defined as follows:
$M= (S1 \times S2, \sum_1 \cup \sum_2, \Delta, \mu, (s_0^1, s_0^2))$ where:

- Δ is the set of all $((s_1,s_2), e, (s'_1,s'_2))$ such that if $e \in \Sigma_1$ then $(s_1, e, s'_1) \in \Delta_1$ otherwise $s_1=s'_1$, and if $e \in \Sigma_2$ then $(s_2, e, s'_2) \in \Delta_2$ otherwise $s_2=s'_2$
- $\mu((s_1,s'_1), e, (s_2,s'_2)) = \mu_1(s_1, e, s)$ if $e \in \Sigma_1/\Sigma_2$
  $= \mu_2(s_2, e, s'_2)$ if $e \in \Sigma_2/\Sigma_1$
  $= [\mu_1(s_1, e, s'_1). \mu_2(s_2, e, s'_2)/(\mu_1(s_1). \mu_2(s_2))]$ $(\min(\mu_1(s_1). \mu_2(s_2))$ if $e \in \sum_1 \cap \sum_2$

We note that the rate of the shared activity (third case above) is reflecting the rate of the slower participant. The resulting system is a CTMC, for a proof or a more thorough discussion refer to [2].

2.6. Hiding

Let M be a CTMC with alphabet Σ, and let $L \subseteq \Sigma$, we define M/L to be a CTMC that behaves exactly like

M except that all the symbols that belong to L are hidden, i.e. we replace the occurrence of symbols from L by the invisible symbol i, and hence we can interpret the activity as an internal delay.

If $M=(S, \sum, \Delta, \mu, s_0)$, then $M/L=(S, \sum \cup \{i\}/L, \Delta', \mu', s_0)$ where

- $(s,e,s') \in \Delta'$ if $(s,e,s') \in \Delta$
- $(s,i,s') \in \Delta'$ if there exists $e \in L$ with $(s,e,s') \in \Delta$
- $\mu'(s,e,s') = \mu(s,e,s')$ and $\mu'(s,i,s') = \Sigma_{e \in L} \mu(s,e,s')$

2.7. Probability and Steady State Probability Distributions

Let $M= (S, \sum, \Delta, \mu, s_0)$ be a CTMC. We denote by

- $v_s(t)$ the probability of being in state s at time t, for some $s \in S$.
- v(t) the probability distribution vector of M at time t.
- Π, the stationary distribution of M (if it exists), which is: $\Pi = \lim_{t \to \infty} v(t)$

Remark: With some minor changes (like taking 1/T instead of μ), the definitions above apply for an RTFSM.

3. Approach to Solving the Problem

In this section, we first present an example and then explain our approach to finding the performance parameters of several system components in order to satisfy the behavior specified for the overall system.

3.1. Motivating Example

User requests arrive a a certain rate and a server processes them. The server either fails or succeeds in processing. If it fails the whole system stops. If a new request arrives while the server is busy processing a previous request, the server apologizes and ignores the new request. If another request arrives while the server apologizes to the previous one, the server does not see the new request. The time required for a new request to arrive is a random variable with exponential distribution and with rate = ¼, see Figure 1.

We assume that the system S is designed as a composition of two components M and N, see Figure 2, (refer to [1] for details about constructing submodules from a given specification in the context of finite state machines). Both systems M and N fail at the same time because they rely on a third system that may fail (example: power supply). The rates of machine M are known, while those of machine N are not. We need to determine the rates of machine N so that the composition of M and N behaves "better" than S. Before discussing what we mean by "better", we present a brief explanation about the behavior of M and N and their composition, followed by a statement of the problem to be solved.

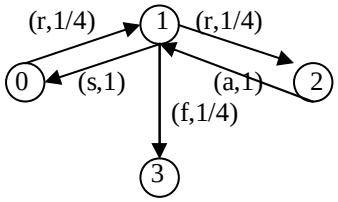

Figure 1. CTMC S, where r represents the request seen by the server, f failure, s success and a the ``apologies I'm busy`` message

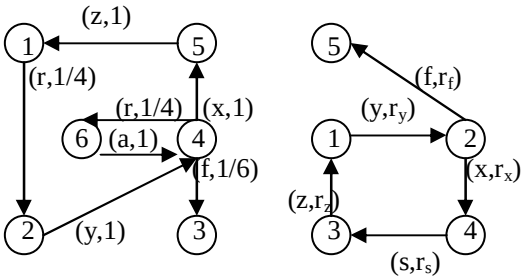

Figure 2. Submodules M and N

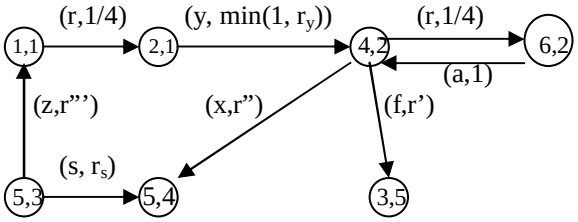

Figure 3. The composition K=MxN

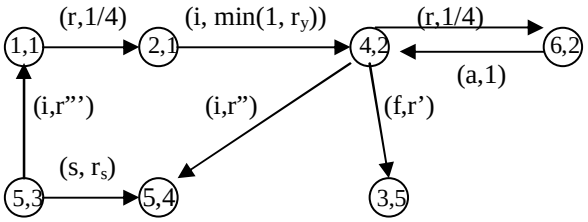

Figure 4. Composition L=MxN/x,y,z with x,y,z hidden

Components M and N run in parallel and synchronize via some events: M gets the requests and N processes them and sends a success message if it succeeds in processing. At any time during the

processing, both machines may fail. While processing a request, machine M may receive another request, which is rejected by N by sending an "apologies, I'm busy" message. N and M synchronize on actions x, y, z and f (where x, z and y are some internal actions common to both machines) to ensure that:

1. Every request that is processing either succeeds or both machines fail.
2. N starts processing a request only when no other request is being processed by machine N.

In Figure 3, $r' = \min(17/12, r_f + r_s).r_f/(r_f + r_s).2/17$
And $r'' = \min(17/12, r_f + r_s).r_x/(r_f + r_s).12/17$
And $r''' = \min(1, r_z)$
x, z and y are internal actions, they are considered as internal delays so we replace them by empty moves as shown in Figure 4.

3.2. Designing Performance Parameters for Composed Systems

CTMC L is a system constructed with the intention to do the same work specified in specification S. Since we cannot always achieve a system with identical behavior, we are looking for a system with equal or better behavior than the specification (we will define later what we mean by better). Therefore the problem here consists of determining the rates of subsystem M and N in a way to satisfy the above statement. To be able to solve this problem we need to**:**

1- **Remove the internal delays from CTMC L:** we should obtain a CTMC F that has a "similar" structure to L, but no internal delays. We resort to this construction because comparing F and S is more convenient than comparing L and S (due to the absence of internal delays). In the next section we will present an overview of the algorithm used to remove the internal delays, also called i-moves; a detailed explanation of the algorithm can be found in [3].

2- **Define equivalences:** after removing the i-moves from CTMC L, see Figure 7, we get a CTMC F that is equivalent to L according to two equivalences which we call *average delay simulation equivalence*, and *average delay equivalence*. For some systems, we are interested in their behavior after they reach the steady state (example: systems that run indefinitely), so we require a certain equivalence between F and L that preserves some useful steady state properties; here the average simulation equivalence is useful. In other systems, we are interested in their behavior starting from the initial state and describing the probabilities and delay times for doing certain actions. In this context, the average delay equivalence is useful.

3- **Define some criteria for comparison:** For every system, some criteria are considered to be better than others, for example, one of our main concerns in a given system might be the speed, so the faster the model the better, or the criteria might be lower failure rate, so the less the model fails the better. Sometimes, if the system has limited storage capacity, we might be interested in having the least data lost per unit of time on average, so the less data lost in the model the better…etc. These requirements are problem-dependant and we will call them "criteria for comparison". In the particular problem above we consider the following criteria: A better system is a system with less probability of failure, moreover, we require faster processing for requests on average (i.e. less average delay time for a given request before a success occurs).

3.3. Removing i-Moves

Definition 1. A CTMC M has a n-cycle of i-moves if there exists a state x in M, where we can navigate from x and come back to x with a trace consisting of exactly n i-moves.

Definition 2. Suppose M is a CTMC, we denote by P_t^x the probability of a transition t from state x (see Figure 5).

We proceed with removing the i-moves as follows: For each i-move in M from state x to state y, (we will take Figure 5 as our case), which is not part of an n-cycle, do the following

- Add to state x (Figure 5) all transitions that can be done from state y, in our case they are: $b_1,\ldots,b_m$, these new transitions have x as a starting state but go to the same state they used to go to from y.

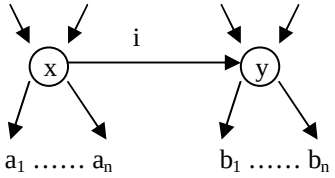

Figure 5. A general case of i-move

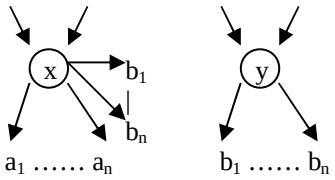

Figure 6. States x and y after removing the i-move

- The probabilities of the new transitions from x are:

$P_{bj}^x = P_i^x P_{bj}^y$. In other words, it is the probability of going through the i-transition from state x to state y, then going through the b_j transition.

- The new ADT of x denoted by D_x' becomes: $D_x' = P_i^x(1/r_x+1/r_y)+(1- P_i^x)1/r_x = P_i^x/r_y+1/r_x = D_y+ P_i^x D_x$, where D_x is the initial ADT of state x
  In other words, the new ADT of x, would be the average of the ADT of x multiplied by the probability of leaving through one of the a_i's added to the ADT spent in x then in y consecutively multiplied by the probability of doing transition i. (before doing one of the b_i's).

On the other hand, if we have several interconnected loops of i-moves (like the case in Figure 6a), then for each state x, we look at all the possible loops connected to x, and we calculate the probability for each non i-transition possible from x, as well as the ADT of x, by taking into consideration that while in x, we can traverse any of these loops many times before doing a non i-transition. For a detailed construction, see [3].

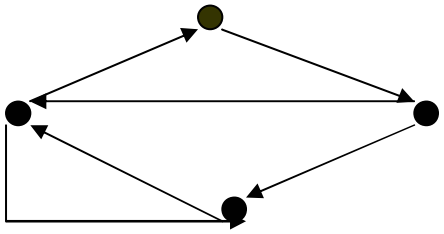

Figure 6a. All moves represented are i-moves.

4. Equivalences and Steady State Behavior

4.1. Equivalences

Definition 3. Two CTMC $M_1= (S_1, \sum, \Delta_1, \mu_1, s_0^1)$ and $M_2= (S_2, \sum, \Delta_2, \mu_2, s_0^2)$ are said to be **equal** if we can find a labeling *f* which maps the states of M_2 to the states of M_1 such that:
$\Delta_1=f(\Delta_2)$, $s_0^1 = f(s_0^2)$, $S_1 = f(S_2)$ and $\mu_1=f(\mu_2)$ (where f(x) represents x with the occurrence of any state $s \in S_2$ replaced by *f*(s))

Definition 4. Let M_1 and M_2 be two CTMC where:
$M_1= (S_1, \sum, \Delta_1, \mu_1, s_0^1)$ and $M_2= (S_2, \sum, \Delta_2, \mu_2, s_0^2)$, and let $M_1'= (S_1, \sum, \Delta_1', \mu_1', s_0^1)$ and $M_2'= (S_2, \sum, \Delta_2', \mu_2', s_0^2)$ be the CTMCs we obtain by removing the i-moves from M_1 and M_2 , respectively. We say that M_1 and M_2 are **average delay simulation equivalent**, written

$$M_1 \cong M_2 ,$$

if M_1' and M_2' are equal.

Definition 5. Let M_1 and M_2 be two CTMC where:
$M_1= (S_1, \sum, \Delta_1, \mu_1, s_0^1)$ and $M_2= (S_2, \sum, \Delta_2, \mu_2, s_0^2)$, Then M_1 and M_2 are said to be **average delay equivalent**, if:

- When E is a trace that is possible through M_1 then it is possible through M_2 and vice versa.
- Let $e \in \Sigma$, then after performing trace E in M_1 and M_2, the probability of having e as the next visible (non i-transition) is the same in both M_1 and M_2
- After performing E, the delay time needed on average before a visible transition occurs is the same for both M_1 and M_2.

Proposition 1. If $M_1 \cong M_2$ then M_1 is average delay equivalent to M_2. (See [3] for a proof)

4.2. Steady State Behavior of Equivalent CTMC's

As we have seen in the example in Section 3, the actual system we built is system L, while the system we are using for comparison with S is system F. So what can we say about the similarities of these two systems?

Theorem 1. Let $M=(S, \sum \cup \{i\}, \Delta, \mu, s_0)$ be a CTMC with i-moves and $N=(S, \sum, \Delta, \mu, s_0)$ be the CTMC obtained from M by removing the i-moves according to the algorithm described in Section 3. Then the rate of a transition e, for any $e \in \Sigma$, in steady state, is the same for M and N.
For a proof, see Appendix A

Corollary 1. If $M \cong N$ then the rate of doing a transition e, for any $e \in \Sigma$, in steady state, is the same for M and N.

5. Examples:

5.1. Example 1

We will first proceed with the solution for the problem proposed in Section 3: To start with, we need to remove the i-moves from CTMC L. Figure 7 represents the graph obtained from L by removing the i-moves following the algorithm defined in Section 3:

In Figure 7, we have indicated probabilities instead of rates where α=r'+1/4+r". And the ADT for each state is given below:
ADT(1,1) =1/4
ADT(2,1)=1/min(1,r_y)+
1/(r'+1/4+r")+(r"/(1/4+r'+r")).1/ r_s
ADT(4,2)= (r_s+r")/[(r'+1/4+r")r_s]
ADT(5,5)= r"'+1/4
ADT(6,2)=1

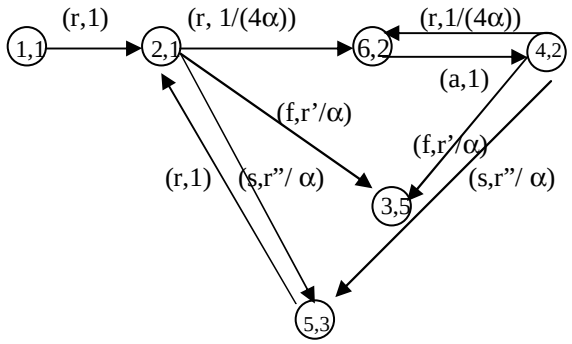

Figure 7. CTMC F obtained by removing the i-moves

Remark for this example, we cannot simplify F anymore, the aggregation as defined in [2] is not possible for F. While in Example1 we got the same number of states as in the given specification, here the number of states are larger, and cannot be simplified.

We now analyze the problem: The criteria of comparison in this problem is concerned with transition probabilities of the systems to be compared as well as ADT. Since the "average delay equivalence" conserves these criteria then we are safe in comparing F and S instead of L and S.

The first part of the analysis is concerned with comparing the probability of failure, while the second part deals with the average time for request processing.

1- The probability of failure, P_f, for a given request r

In F and S, if a given request r starts processing, no matter how many new requests arrive, the machine has to go into either fail or success options. Therefore the probability of failure P_f for a given request r, can be measured as:

(probability of failure of r given we are in the processing state) /(probability of success of r given we are in the processing state)

so (P_f in S)= (¼ / 1)= ¼

(P_f in F)= (r'/α)/(r"/α) = r'/ r" = [min(17/12,r_f +r_x).[r_f/(r_f + r_x)].2/17] / [min(17/12,r_f +r_x).[r_x/(r_f + r_x)].12/17] = 2r_f /12 r_x.

We need **(P_f in F) < (P_f in S)**, which leads to

2r_f /12 r_x < ¼,

r_f < 3/2 r_x

2- The ADT needed for a request r to succeed

- In S: Let r be a request that the machine is processing. During the processing, new requests might arrive; each with rate ¼, at any time the processing might succeed with a probability of ½ (see Figure 1). So, a success might happen at first with probability 1 and ADT= 3/2 (average delay time at state 1), or it might happen after 1 new request arrive and is refused by a sorry message with probability ¼ and ADT =5/2+3/2 (average delay time in state 1 then state 2 then state 1 again) and so on… So, the total ADT before a success is: (d in S)=1.3/2+1(3/2+5/2)+…. + 1.(3/2+n.5/2)+….
- In F: By similar reasoning, the ADT before a success occurs is: (d in F)= (r"/α) [1/min(1, r_y) +1/α+r"/ αr_s] +……..+ (r"/α)(1/4α) [1/min(1, r_y) +1/α+r"/ (αr_s)+(1+(r_s+r")/αr_s]+….+ (r"/α)(1/4α)^n [1/min(1, r_y) +1/α+r"/ (αr_s)+n(1+(r_s+r")/αr_s]
- We need (d inF)<(d in S), this is achieved if the general term in the sequence (d in F)< the general term in (d in S), i.e. if

(r"/α)(1/4α)^n [1/min(1, r_y) +1/α+r"/ (αr_s)+n(1+(r_s+r")/αr_s]<3/2+n.5/2

for all n

One solution is obtained by taking: r_f = r_x =1 and r_y =1/20 r_s =3.

5.2. Example 2

We assume that a queue can hold a maximum of two elements, elements arrive and are stored in the queue and remain there until they are removed, any element that needs to be stored after the queue is full is lost. The time before an elements is removed is a random variable with exponential distribution and rate = 2. The time before an element arrives is also a random variable with exponential distribution and rate =1, this is shown in Figure 8, where *put* represents the arrival of an element, *get* the removal of an element and s a 'sorry full' message, the state number represents the number of elements in the queue.

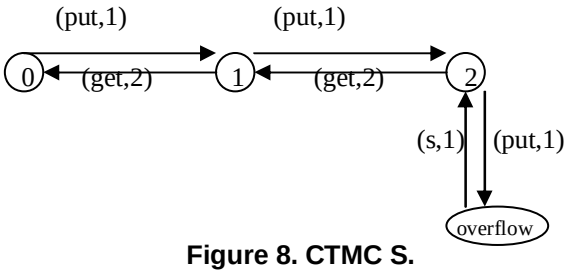

Figure 8. CTMC S.

- **Criteria for comparison**: A better system is a system with fewer elements lost per unit of time on average.

In system S, to get the average number of elements lost per time unit, we need to calculate the rate of getting a put request, for that we need to calculate the steady state probability.

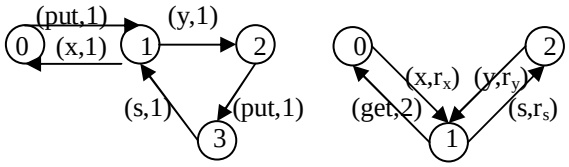

Figure 9. CTMC M and N

We assume that the system is designed as a composition of two components M and N, (see Figure 9), that synchronize on cetain events:
M fills up the queue when elements arrive.
N empties the queue when elements are removed.
N and M synchronize on actions x,y, and s to ensure that:

- No more than two elements are present in the queue at a given time
- The removal of an element is done only if the queue is nonempty
- Both send a "sorry full" message when an element arrives and the queue is full.

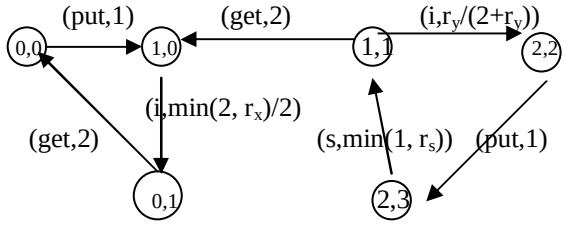

Figure 10. CTMC K.

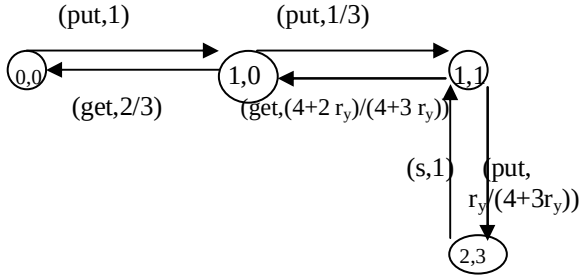

Figure 11. CTMC F.

CTMC F is obtained by removing i-moves and then discarding the nonconnected components. The numbers next to the transition labels is the probability of the transition, and the ADTs are as follows:
ADT(0,0)= 1
ADT(1,0)= 1/3+(2/(min2,r_x))
ADT(1,1)= $2(3r_y^2+7r_y+4)/(4+3r_y)^2$
ADT(2,3)= min(1,r_s)

In the remaining of this section, we need to compare the performance of CTMC S and CTMC L Now, we need to find the unknown rates in N so that, M composed with N performs better than S. We proceed as follows:

- compose M and N, then hide events x and y (as discussed in Section 2) and obtain CTMC L, shown in Figure 10.
- then we remove the i-moves from L to get the CTMC F, shown in Figure 11. From Theorem 1, we have L≅F.
- Then we find the unknown rates so that F performs better than S in terms of average number of elements lost per unit of time.according to the criteria defined above, that is according to the average number of elements lost per unit of time, C_S and C_L, respectively.

The steady state probability of S, denoted by Π_S, gives for every state s in S, the probability of being in state s in equilibrium. So to calculate C_S, we need to calculate the steady state probability, in particular the probability of being in state 2 in equilibrium multiplied by the rate of getting an element:

The steady state probability of system S is π_s=[1/2,1/4,1/8,1/8]. The elements lost per unit of time are:

C_S =π(2).(rate of going out of state 2 through f)= 1/8.1=1/8

We know that L and F have same number of elements lost per unit of time (from Theorem 1) hence $C_F = C_L$. We calculate C_F as follows:
Denote by x= $(4+2r_y)(4+3r_y)/6(3r_y^2+7r_y+4)$ and y= $(4+3r_y)r_y/6(3r_y^2+7r_y+4)$
The steady state probability of F is
π_F=[6x/(2x-y+1), 3x/(2x-y+1), 6x/[3x(2x-y+1)], (1-x-y)3x/[x(2x-y+1)]
and hence the average number of elements lost per unit of time:

C_F = π(2). (rate of going out of state 2 through f)=2/[(2x-y+1)]. $r_y/(4+3r_y)$ this should be smaller than C_S

By replacing x and y by their values, we get:
$2/\{(2[(4+2r_y)(4+3r_y)/6(3r_y^2+7r_y+4)]-[(4+3r_y)r_y/6(3r_y^2+7r_y+4)]+1)\}. r_y/(4+3r_y)<1/8$
One solution could be achieved by taking: r_y = ½, $r_x \geq 2$, $r_s \geq 1$. In this case we get:
C_F = 49.5/559.625= 0.088452<1/8= C_S

So, on average, the machine L obtained performs better than the original specification S.

6. Conclusion

The main concern in this paper was to decompose a continuous time Markov chain into smaller and easier to model chains. The same work could be applied to an

RTFSM and to discrete time Markov chains. As future work, we are trying to generalize these ideas to any real time probabilistic system regardless of the distribution of time delays in its states.

7. References

[1] G. Bochmann, "Submodule Construction-the inverse of composition", *Technical Report*, University of Ottawa, 2001.

[2] Bremaud P., *Markov Chains Gibbs Fields, Monte Carlo Simulation, and queues*, Springer Verlag, 1999.

[4] J. Drissi, G v. Bochmann, "Submodule Construction for systems of I/O Automata", *Submitted to publication, www.site.uottawa.ca/~bochmann* .

[3] F. Dankar, G.V. Bochmann, "Removing Internal Delays from a Continuous Time markov Process", to appear in *Proceedings of the 4th International Symposium for Mathematical Modelling.*

[5] E Haghverdi, H Ural, "An Algorithm for Submodule Construction", *Technical Report of the Department of Computer Science*, University of Ottawa, 1996.

[6] J. Hillston, "Compositional Markovian Modelling Using a Process Algebra", *Proc. Of the 2nd International Workshop on Numerical Solutions of Markov Chains.* Kluer Academic Publishers, 1995.

[7] P. Merlin, G. v. Bochmannn, "On the Construction of Submodule Specifications and Communications Protocols", *ACM Trans. On programming Languages and Systems*, Vol. 5, No 1. 1983

[8] A. Petrenko, N Yevtushenko, G. v. Bochmann, "Experiments on Nondeterministic Systems for the Reduction Relation", *IWTCS*'96.

[9] H. Qin, P. Lewis, "Factorization of Finite State Machines under Strong and Observational Equivalences", *Journal of Formal Aspects of Computing*, Vol. 3, 1991.

[10] S. Wu, S. Smolka, E. Stark, "Composition and behavior of Probabilistic I/O Automata", *Theoretical Computer Science*, Vol. 176, No. 1-2, 1997.

Appendix A

Proof for Proposition 1

Proof: We proceed in the proof as we remove each i-move.

Case1: Non cyclic i-moves: We assume that M has an i-move between states x and y, for some x,y in S, that is not part of a n-cycle, refer to Figure 5, and M' is the CTMC obtained from M by removing this i-move.

Our aim is to prove that, in M and M', the rate of doing a certain transition e in steady state denoted by P_t^e and $P'_t{}^e$ respectively, is the same. For that we need to calculate the steady state probability for M and M', then use it to find the values: P_t^e and $P'_t{}^e$.

<u>First we find the steady state probability:</u>

- let Π be the steady state probability of M, then we know that the probability of going out of a state s in the next time instance at equilibrium equals the probability of coming into s in the next time instance, refer to [2] for more details. Applying this to state x we get:

$\Pi(x)\mu(x)=\Sigma_{z\neq x}\,\Pi(z)\,\mu(z,x)$

$\Rightarrow \Pi(x)/D(x)=\Sigma_{z\neq x}\,\Pi(z)\,P(z,x)/D(z)$ where D(s) is the ADT of sate s

similarly,

$\Pi(y)/D(y) =\Sigma_{z\neq y}\,\Pi(z)\,P(z,y)/D(z)$

- Now, let M' be the CTMC obtained from M by removing the i-move from x to y, refer to Figure 6. If Π' is the steady state probability for M' and D' is its average delay time function, then:

$\Pi'(x)\mu'(x)=\Sigma_{z\neq x}\,\Pi'(z)\,\mu'(z,x)$

but the transitions coming into x have the same rate in both M and M'

$\Rightarrow \Pi'(x)/D'(x) =\Sigma_{z\neq x}\,\Pi(z)\,P(z,x)/D(z)$

$\Rightarrow \Pi'(x)/[D(x) +P_i\,D(y)]=\Sigma_{z\neq x}\,\Pi(z)\,P(z,x)/D(z)=$ $\Pi(x)/D(x)$

$\Rightarrow$ **$\Pi'(x)= \Pi(x) + \Pi(x)\,P_i\,D(y)/D(x)$**

Now, $\Pi'(y)\mu'(y)=\Sigma_{z\neq y}\,\Pi'(z)\,\mu'(z,y)$ but $\mu'(y)=)\mu(y)$

$\Rightarrow \Pi'(y)/D(y) =[\Sigma_{z\neq y,x}\,\Pi(z)\,P(z,y)/D(z)]$
$+[\Pi'(x)/D'(x)]P'(x,y)$

But $P'(x,y) = P(x,y)-P_i$,

$\Rightarrow \Pi'(y)/D(y) =[\Sigma_{z\neq y,x}\,\Pi(z)\,P(z,y)/D(z)]$
$+[\Pi'(x)/D'(x)][P(x,y)- P_i]$

But $\Pi'(x)/D'(x)= \Pi(x)/D(x)$

$\Rightarrow \Pi'(y)/D(y) =[\Sigma_{z\neq y,x}\,\Pi(z)\,P(z,y)/D(z)]$
$+[\Pi(x)/D(x)][P(x,y)- P_i]$

$\Rightarrow \Pi'(y)/D(y) =[\Sigma_{z\neq y}\,\Pi(z)\,P(z,y)/D(z)] - P_i\,\Pi(x)/D(x)$

$\Rightarrow$ **$\Pi'(y)= \Pi(y) - P_i\,\Pi(x)D(y)/D(x)$**

<u>Now, we calculate the rate of doing any transition from x and from y in the next time instant in both M and M'</u>

- Probability of doing transition a_i in the next time instance in M (refer to Figure 5) = $\Pi(x)P(a_i)/D(x)$

- Probability of doing transition a_i in the next time instance in M'(refer to Figure 6) = $\Pi'(x)P(a_i)/D'(x)$

$= P(a_i)\,[\,\Pi(x)+ \Pi(x)\,P_i\,D(y)/D(x)]/\,[D(x)+ P_i\,/D(y)]$

$= P(a_i)\,[\,\Pi(x)/D(x)]\,[D(x)+ P_i\,D(y)]/\,[D(x)+ P_i\,D(y)]$

$= P(a_i)\,[\,\Pi(x)/D(x)]$

- Probability of doing transition b_i in the next time instance in M (refer to Figure 5)

$= \Pi(y)P(b_i)/D(y)$

Probability of doing transition b_i in the next time instance in M' (refer to Figure 6)

$= \Pi'(y)P(b_i)/D'(y) + \Pi'(x)P(b_i)P_i/D'(x)$

$= \Pi'(y)P(b_i)/D(y) + \Pi'(x)P(b_i)P_i/D'(x)$ **(1)**

but, from the calculations above we get that :

$\Pi'(x)/D'(x) = \Pi(x)/D(x)$ and $\Pi'(y)/D'(y) = \Pi(y)/D(y) - P_i \Pi(x)D(y)/D(x)$

so **(1)** $\Rightarrow \Pi(y)P(b_i)/D(y) - [\Pi(x)/D(x)] P(b_i)P_i + [\Pi(x)/D(x)] P(b_i)P_i$

$= \Pi(y)P(b_i)/D(y)$

So after we remove the i-move between x and y, the rate of a transition e from x or y remains the same. But, the steady state probability for any state s where $s \neq x,y$ is the same in M and M' as well as the rate of doing transition e out of s. Hence the rate for transition e in M and M' remains the same. Moreover, removing all non cyclic i-moves from a CTMC keeps the rate of a transition fixed.

Case2:

Cyclic i-moves: Now we need to consider the case of a n-cycle of i-moves, and several interconnected loops of i-moves and compare the rate of doing a transition e, for some $e \in \Sigma$, at steady state, before and after removing the i-moves.

For the sake of simplicity, we will treat the case of a 2-cycle of i-moves only, the other cases are treated using the same reasoning.

Let M be a CTMC with a 2-cycle of i-moves, see Figure 12. Figure 13 shows CTMC M after removing its 2-cycle of i-move.

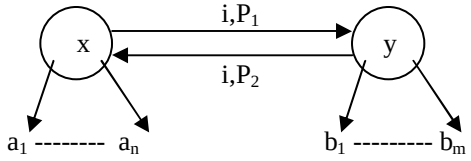

Figure 10:CTMC M

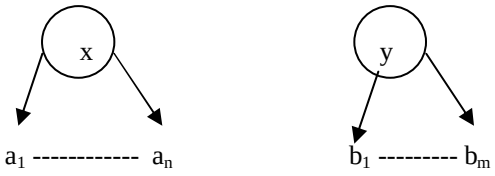

Figure 11. CTMC M'

We need to calculate the rate of transition a_1 from x or y in both M and M'.

- Let Π be the steady state probability of M, then

$\Pi(x)\mu(x) = \Sigma_{z \neq x} \Pi(z)\, \mu(z,x)$

$\Rightarrow \Pi(x)/D(x) = \Sigma_{z \neq x} \Pi(z)\, P(z,x)/D(z)$ where D(s) is the ADT of sate s

$\Rightarrow \Pi(x)/D(x) = \Pi(y)\, P(y,x)/D(y) + \Sigma_{z \neq x,y} \Pi(z)\, P(z,x)/D(z)$

similarly,

$\Pi(y)/D(y) = \Pi(x)\, P(x,y)/D(x) + \Sigma_{z \neq y,x} \Pi(z)\, P(z,y)/D(z)$

- If Π' is the steady state probability for M' and D' is its average delay time function, then:

$\Pi'(x)\mu'(x) = \Sigma_{z \neq x} \Pi'(z)\, \mu'(z,x)$

Since there is no more transitions from y to x

$\Pi'(x)\mu'(x) = \Sigma_{z \neq x,y} \Pi'(z)\, \mu'(z,x)$

but the transitions coming into x have the same rate in both M and M'

$\Rightarrow \Pi'(x)/D'(x) = \Sigma_{z \neq x,y} \Pi(z)\, P(z,x)/D(z)$

similarly

$\Rightarrow \Pi'(y)/D'(y) = \Sigma_{z \neq x,y} \Pi(z)\, P(z,y)/D(z)$

Now, we need to calculate the rate of transition a_i from x and from y in both M and M'

- Rate of doing transition a_i in M (Figure 10):

$P_t = \Pi(x)P(a_i)/D(x)$

- Rate of doing transition a_i in M'(Figure 11):

$P'_t = [P(a_i)/(1-P(y,x)P(x,y)]\Pi'(x)/D'(x) + [P(a_i)P(y,x)/(1-P(y,x)P(x,y)\Pi'(y)/D'(y)$

$= [P(a_i)/(1-P(y,x)P(x,y)]\Sigma_{z \neq x,y} \Pi(z)\, P(z,x)/D(z) + [P(a_i)P(y,x)/(1-P(y,x)P(x,y)] \Sigma_{z \neq x,y}\Pi(z)\, P(z,y)/D(z)$

$= [P(a_i)/(1-P(y,x)P(x,y)]\, [\Pi(x)/D(x) - P(y,x)\Pi(y)/D(y)] + [P(a_i)P(y,x)/(1-P(y,x)P(x,y)][\Pi(y)/D(y) - P(x,y)\Pi(x)/D(x)$

$= P(a_i)[\Pi(x)/D(x)]\, [1-P(y,x)P(x,y)]/[1-P(y,x)P(x,y]$

$= P_t$

So after we removed the 2-cycle of i-move from the CTMC, the rate of transition a_i remains the same. Hence, the proposition is proved.