Verification of Service Level Agreements with Markov Reward Models

Farrel Lifson
Data Network Architecture Lab
Dept of Computer Science, University of Cape Town

***Abstract*— We review the usage of Markov reward models and it's associated performability logics in the determination of Service Level Agreements. We examine the removal of zero state rewards from Markov reward models and suggest a refinement to prevent the modification of the logical properties of the model. We also introduce an alternate methodology to be able to do model checking using a suitable number to approximate zero. We observe the error that this introduces and observe some performance issues it introduces to certain underlying model checking techniques.**

I. Introduction

Model checking refers to verifying whether certain properties, expressed in a formal logic, hold in system models. The models are usually state transition systems such as a Continuous Time Markov Chain (CTMC). There are a number of logics that can be used to specify properties, but Continuous Reward Logic (CRL)[4] is the most established and the one considered in this paper.

The techniques used for verification place limitations on the properties of the model, such as non-positive rewards[4] where the limitations are due to the nature of the algorithm or the verification of both reward and time bounds[1] where solutions are computationally expensive.

II. Markov Reward Models

A Markov Reward Model (MRM) is a Continuous Time Markov Chain with an associated reward structure. The reward structure, ρ, is a function that assigns to each state s a reward $\rho(s)$. When residing for t time units in state s, the system acquires a total reward of $\rho(s) \cdot t$, which is added to a cumulative reward. For usage in model checking a MRM also has a labelling function, $\mathbf{L}$, which maps to each state a set of atomic propositions which that state satisfies.

III. Model Checking and Continuous Reward Logic

Continuous Stochastic Reward Logic (CSRL) is a logic defined in order to represent constraints on the model with both time and reward bounds. The syntax of CSRL is comprised of State and Path formula which are defined as

State-formulas $\Phi ::= a|\neg\Phi|\Phi \vee \Phi|\mathcal{S}_{\trianglelefteq p}(\Phi)|\mathcal{P}_{\trianglelefteq p}(\varphi)$

- $\mathcal{S}_{\trianglelefteq p}(\Phi)$**:** probability that Φ holds in steady state is $\trianglelefteq p$
- $\mathcal{P}_{\trianglelefteq p}(\varphi)$**:** probability that paths fulfill φ is $\trianglelefteq p$

Path-formulas $\varphi ::= \mathsf{X}^I_J\Phi|\Phi\bigcup^I_J\Phi$

- $\mathsf{X}^I_J\Phi$**:** next state reached at time $t \in I$ and reward with $r \in J$ and fulfills Φ
- $\Phi\bigcup^I_J\Psi$**:** Φ holds along the path until Ψ holds at time $t \in I$ and reward $r \in J$

However the verification of both is computationally complex[1], but the verification of only the time constraints is possible. When dealing strictly with time we use a temporal sub-logic of CSRL which is referred to as Continuous Stochastic Logic (CSL). Conversely there is also a sub-logic which deals only with rewards, the previously mentioned Continuous Reward Logic (CRL).

Verification of CSL[2] is done using a number of different techniques involving

- graph theory to reduce the complexity of the model
- numerical techniques such solving of linear equations[3].

Verification of properties described in CRL do not need any other specialised techniques other than the ones mentioned for CSL. CSRL has a property defined by the *Duality Theorem*[4] which allows us to transform the MRM in such a way that we can swap the reward and time constraints of the logic. The justification behind this is that we can regard the progress of time as the accumulation of rewards and the same in reverse. The transformation of the MRM is done in the following way where $\rho(s) \neq 0$:

1) Transform MRM $\mathcal{M} = ((S, \mathbf{R}, L), \rho)$ into $\mathcal{M}^{-1} = ((S, \mathbf{R}', L), \rho')$ with
   - $\mathbf{R}'(s, s') = \mathbf{R}(s, s')/\rho(s)$ - rescale transition rates by reward
   - $\rho'(s) = 1/\rho(s)$ - invert the reward structure
2) Transform CRL state formula Φ into Φ^{-1} by swapping reward and time bounds : $\mathsf{X}_J^I \rightarrow \mathsf{X}_I^J$ and $\bigcup_J^I \rightarrow \bigcup_I^J$

The duality theorem then states:

$$Sat^{\mathcal{M}}(\Phi) = Sat^{\mathcal{M}^{-1}}(\Phi^{-1})$$

where $Sat^{\mathcal{M}}(\Phi)$ is the set of states in the MRM $\mathcal{M}$ that satisfy state formula Φ. This Duality property allows us to verify a CRL statement by transforming the statement into CSL and then verifying this against the CTMC that is produced by transforming the MRM. When we only deal with reward bounds in the state formula (CRL), then the transformed formula will deal only with time bounds (CSL). In cases when there are both time and reward bounds, as in CSRL, then there is no advantage to the transformation. Note, however, that the transformation is restricted to non-zero rewards for otherwise, if $\rho(s) = 0$ then $\mathbf{R}(s, s')/\rho(s)$ will be infinite.

IV. Service Level Agreements

One of the more appropriate practical applications of performability is that of Service Level Agreements(SLA), a contract between two parties where one party guarantees a certain level of service for some task. The party providing the contract could easily estimate a service level by simple approximation of his current computing power, without taking into account failure of components, differing load levels and so forth. Using a formalism such as MRM and combining that with CRL the provider is able to better model their service ability as well as verify whether they can meet or exceed the service levels agreed upon.

The provider therefore has the ability to model and verify both performance coupled with the reliability of the system.

A. Interaction between Service Level Management and Markov Reward Models

According to the IT Information Library (ITIL) guidelines[5] service management can be broken up into a number of different sections each dealing with certain facets of the management and planning processes. The natural area for MRM to be used would be in the *Development* phase. The Development phase is further split into *Service Design* and *Service Build & Test* units, which is a natural fit for the way we will use MRMs.

During the Service Design phase, a suitable MRM would be specified and using analysis tools (in our case primarily ETMCC) these MRM representing the system would be checked (the Service Build & Test phase). The process is cyclic so that either the model can be modified to meet the testing criteria or the criteria themselves can be modified if the model is not flexible.

V. Limitations to Model Checking with Markov Reward Models

In the symbolic solutions of model checking, the area where we are most concerned, a number of limitations are present such as the presence of states with zero reward, or the lack of multiple rewards. It is these limitations and their implications in constructing models that need further exploration. Added to the limits of the model itself, is the expressiveness of the logic used to specify the constraints. We are interested in looking at whether the logic can sufficiently express properties that will be useful in the constructing and verification of SLAs.

A. Zero Reward States

One way of eliminating zero rewards suggested by Beaudry[6] is to make all zero reward states absorbing. This severely limits the model as once a zero state is reached the verification process is halted. In models which enter zero reward states early this will place a abound on the length of time for which we can verify the model. Ciardo et al.[7] expanded on the work of Beaudry by defining a new MRM where the zero reward states are removed from the MRM and replaced with a probabilistic switch without changing the behaviour of the MRM. Wholesale removal of zero reward states however can be dangerous in model checking as we run the risk of modifying the *logical property* of the MRM.

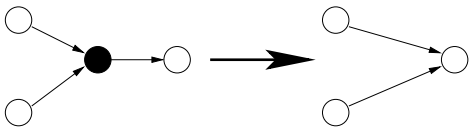

Fig. 1. Removal of a zero reward state and loss of logical property from a simple Markov chain.

Figure 1 shows a simple Markov chain with states which satisfy either of the two atomic propositions, $black$ or $white$. If the $black$ state has zero reward and is removed, it is possible that a CRL statement that was valid previously (for instance any CRL statement using $white \bigcup black$) is no longer valid in the transformed chain as $black$ no longer exists and as such the absorbing state will never be reached.

B. A Refinement to the Algorithm

In the process of model checking, the requirements, expressed as statements in CRL, are explicitly known in advance. By examining the requirements beforehand we can decide which zero reward states can safely be removed without affecting the logical properties of the MRM.

In the case of the *CRL until operator* ($\bigcup$) the relevant logical component of the statement is $\Phi \bigcup \Psi$. We are only concerned with states that satisfy Φ, as Ψ is regarded as a halting condition, i.e., once a Ψ state is reached all further states are unimportant. Zero reward states which satisfy Φ can safely be removed as they have no affect on the cumulative reward and their removal would not change the logical properties of the model. By preventing the removal of a zero reward state if it satisfies Ψ, and allowing the removal if it satisfies Φ, we can prevent the logical properties of the MRM from being altered.

VI. An Alternative Approach

As an alternative to eliminating zero reward states, when doing so is not desirable, we suggest replacing these states with a small positive number, $\epsilon << 0$ which we will use to approximate zero in the model. However the addition of ϵ will add an error to the cumulative reward earned due to the fact that some time t spent in a zero reward state an extra $\epsilon \cdot t$ reward is accumulated. We note that the magnitude of the error accumulated can be shown to be small enough such that the accuracy of the model is insignificantly affected and an example illustrating this will follow in Section VII.

If we add ϵ to all states in the MRM, regardless of whether the state is a zero reward state or not, then the accumulated error will always be $\epsilon \cdot t$. Using this method ensures that although the error will be larger than if ϵ affected only zero reward states, the error can be explicitly determined and it is guaranteed to have an upper bound.

VII. Example

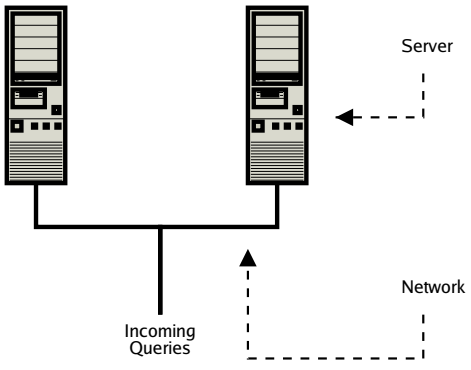

Fig. 2. The configuration of the system used in the example.

To illustrate the concepts mentioned so far we make use of a small system illustrated in Figure 2 comprising of a simple

VoIP network of three SIP (Session Initiation Protocol)[8] servers, 2 proxy servers, of which one is a backup, and a connecting network. When the main proxy server is running it can route 40 calls per hour to each server, however the backup server can only route $\frac{3}{4}$ as many. The proxy servers fail on average once a week and take 2 hours to repair. A SIP server fails on average once a day but only takes 30 minutes to repair. The network fails once every 1 hour on average due to congestion, however it only takes 10 minutes on average to become decongested. The state descriptor (m, n) indicates that the main proxy is functioning as well as n servers.

In Figure 3 we depict a Markov chain derived from Figure 2. On this Markov chain we define the reward structure as follows:

- If the main server is operational: $40 \times \#servers$
- If the main server has failed but the backup is operational: $30 \times \#servers$
- Otherwise 0 rewards are earned

The addition of the reward structure then completes the definition of a MRM.

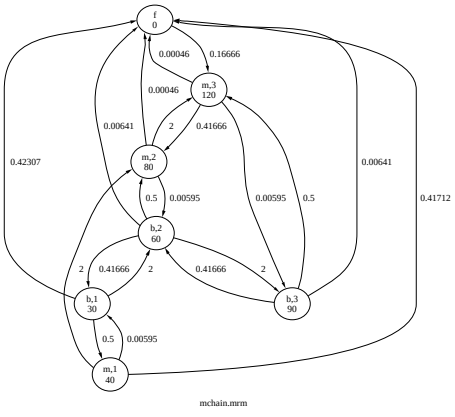

Fig. 3. A Markov chain extracted from the system description.

A. Observing the error magnitude

To observe the size of the error introduced by the ϵ-reward technique we used Monte Carlo simulation on the CTMC illustrated in Figure 3. Figure 4 illustrates the growth of error over a 1000 simulations with ϵ replacing the reward in previously zero reward states. Each simulation exited after 1200 units had passed. The lower and upper bounds of the error are clear in the left hand graph with the size of the error being a function of ϵ and the total time spent in the zero reward states. In the right hand side graph we have reduced the rate at which we enter the failure state to more ideal conditions of one failure every 3 months with a 6 hour repair time. The accumulated error in this case is very small, around 2.5×10^{-5}, as the total time spent in the zero reward state, and therefore the magnitude of the error, is greatly reduced.

The straight line graphs in Figure 4 reflect the largest possible error $\epsilon \cdot t$. If there are cases where the total time in a zero reward state becomes very large we can expect the upper error bound to tend towards this line. We note that in Figure 4 that although the largest possible error grows at a rate greater than the upper bound of the lower graph, it is orders of magnitude smaller than the total accumulated reward at that point in time, with the total accumulated reward in the example used reaching In cases where it is not possible to compute the error due to the residence time in zero reward states, we can easily calculate $\epsilon \cdot t$ and guarantee that the error be less.

B. Numerical Complexity

The ϵ-reward technique has, however, an affect on the underlying numerical methods used in model checking. For our research we use the Erlangen-Twente Markov Chain Checker (ETMCC)[3], a symbolic model checker, which uses the Fox-Glynn[9] method to compute the Poisson probabilities of the underlying Markov chain. The affect on the algorithm is such that the greater the orders of magnitude between the largest and smallest elements in the rate matrix the longer the algorithm must run, and more importantly, the more memory is needed. Figure 5 shows the number of iterations needed by the tool in order to solve sample CRL queries performed on the model described in Section VII-A. For small ϵ the number of iterations needed begins to increase quite significantly once ϵ is smaller than 0.0001.

In our example, when attempting to make ϵ smaller by another order of magni-

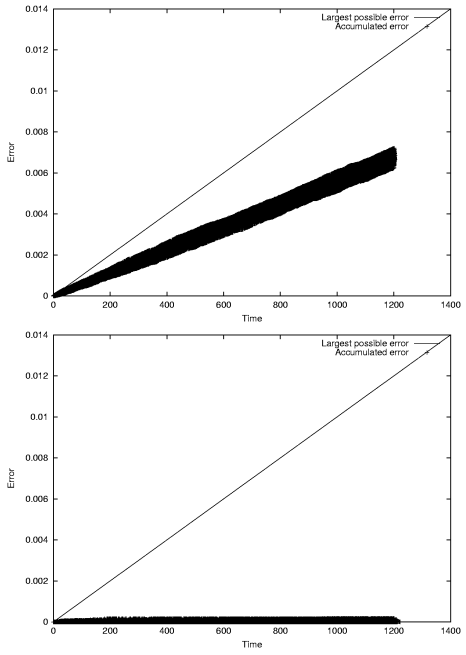

Fig. 4. The growth of error after the introduction of ϵ. The top figure resides for a longer period of time in ϵ states than the one on the bottom.

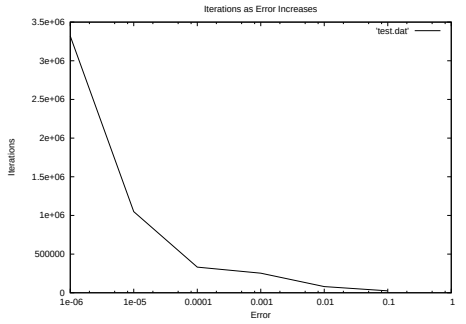

Fig. 5. The number of iterations required to solve a Markov chain with ϵ.

tude, we run into memory space problems. Note however, that the execution time remained reasonable, despite the increasing number of iterations, and that memory was the primary constraint. We performed the verification on a 1.4Ghz AMD Athlon processor with 256MB RAM.

VIII. Applying CRL Constraints

Once we have a MRM (comprising a CTMC and associated reward structure) we can apply constraints expressed in Continuous Reward Logic to the MRM. We need to ensure that the CRL expression models as accurately as possible the constraints we wish to impose.

In the case of service level agreements we wish to know how much work will be performed before the system has some sort of failure (a failure being defined as when no work is done, not just if the main server fails). In Figure 3 a failure is indicated if the state satisfies atomic proposition f. We are not concerned with whether only the main server has failed (indicated by a state that satisfies b) but rather when both the main and backup servers fail which occurs when a state satisfies $\neg(b \vee m)$ which is equivalent to the simpler statement $\neg f$. We would like to test the fact that if we do fail we will have transacted 250 SIP requests and that there is a greater than 95% chance that this happens. The CRL expression to express this is

$$\mathcal{P}_{<0.05}(\neg f \bigcup_{250} f)$$

which can be informally translated as that there is a less than a 5% chance of failure occurring before 250 requests have been processed.

Another constraint that we might wish to check is that of repair. Here we specify that their should be a 98% chance that the main server is repaired before 50 transactions are processed by the backup server. The CRL statement to do this would be expressed as

$$\mathcal{P}_{<0.02}(b \bigcup_{50} m)$$

If we are interested in testing the probability of a network failure happening from a state when everything is fully functional we would use the $\wedge$ operator to indicate the extra constraint of the number of servers available before failure. This would be expressed on the left hand side of the Until operator as shown by the CRL statement

$$\mathcal{P}_{<0.001}((m \wedge 3) \bigcup f)$$

If any of the above constraints are not met, than the system designer has the option of either readjusting the paramaters of the model to meet the requirements or by weakening the requirements so that the model is satisfied.

With this small model we are limited by the number of atomic propositions available to us. In larger models with more atomic propositions describing a number of properties of the system the number of constraints that can be applied increases significantly.

IX. Conclusion

We have shown the usage of Markov Reward Models through the modelling of a simple system. It is clear that in order to be able to extract useful results from the model it is vital that an accurate, detailed model of the system is needed. Both the underlying structure of the Markov chain, especially the failure and repair rates of the system, and the reward structure need to reflect the behavior of the system as accurately as possible.

We have addressed the concern that removing zero reward states from Markov reward models can change the logical property of the model. Doing so can cause previously valid requirements to no longer hold on the transformed model. By taking into account the requirements before the removal of zero reward states we can guarantee that the logical characteristics will not be affected.

We introduced an alternate approach to the elimination of zero reward states by replacing zero awards with ϵ, a suitably small number. This removes the need for the removal of zero state rewards, however it does introduce a cumulative error which we quantify via simulation.

References

[1] B. Havekort, L. Cloth, H. Hermanns, J.-P. Katoen, and C. Baier, "Model checking performability properties," *Proceedings of the International IEEE Conference on Dependable Systems and Networks*, pp. 103–112, 2002.

[2] C. Baier, J.-P. Katoen, and H. Hermanns, "Approximate symbolic model checking of continuous-time markov chains," in *International Conference on Concurrency Theory*, 1999, pp. 146–161. [Online]. Available: citeseer.nj.nec.com/baier99approximate.html

[3] H. Hermanns, J.-P. Katoen, J. Meyer-Keyser, and M. Siegle, "A tool for model checking markov chains," *Software Tools For Technology*, 1999.

[4] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen, "On the logical characterisation of performability properties," *Automata, Languages and Programming*, pp. 780–792, 2000. [Online]. Available: citeseer.nj.nec.com/article/baier00logical.html

[5] IT Information Library, "Service management." [Online]. Available: http://www.itil.org/itil_e/itil_e_040.html

[6] M. Beaudry, "Performance-related reliability measures for computing systems," *IEEE Transactions on Computers*, vol. 27, pp. 540–547, June 1978.

[7] G. Ciardo, R. Marie, B. Sericola, and K. Trivedi., "Performability analysis using semi-markov reward processes," *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1251–1264, 1990. [Online]. Available: citeseer.nj.nec.com/ciardo90performability.html

[8] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session initiation protocol," 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2543.txt

[9] B. L. Fox and P. W. Glynn, "Computing poisson probabilities," *Communications of the ACM*, vol. 31, no. 4, pp. 440–445, 1988.