Towards Accurate Probabilistic Models using State Refinement

Paulo H. Maia[†] Jeff Kramer[†] Sebastian Uchitel[‡] Nabor C. Mendonça[⋆]

[†]Imperial College London
London, UK
{pmaia@doc.ic.ac.uk,
j.kramer@imperial.ac.uk

[‡]U. of Buenos Aires, Argentina &
Imperial College London, UK
su2@doc.ic.ac.uk

[⋆]Universidade de Fortaleza
Fortaleza, CE, Brazil
nabor@unifor.br

ABSTRACT

Probabilistic models are useful in the analysis of system behaviour and non-functional properties. Reliable estimates and measurements of probabilities are needed to annotate behaviour models in order to generate accurate predictions. However, this may not be sufficient, and may still lead to inaccurate results when the system model does not properly reflect the probabilistic choices made by the environment. Thus, not only should the probabilities be accurate in properly reflecting reality, but also the model that is being used. In this paper we identify and illustrate the problem showing that it can lead to inaccuracies and both false positive and false negative property checks. We propose state refinement as a technique to mitigate this problem, and present a framework for iteratively improving the accuracy of a probabilistically annotated behaviour model.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software / Program Verification—*model checking, statistical methods, verification.*

General Terms

Design, Languages, Theory, Verification.

Keywords

Probabilistic model checking, behaviour model, accuracy, refinement.

1. INTRODUCTION

Probabilistic models are useful in the analysis of system behaviour and non-functional properties. They can extend the classical finite state machine formalism, or variations of it, with probabilistic transitions. Formal verification of such models typically involves checking quantitative properties, defined using a temporal logic language, such as PCTL, and can be carried out via simulation, numerical analysis or by using a probabilistic model checker, such as PRISM [4].

There is a significant body of work on probabilistic modelling formalisms, their properties and efficient algorithms for analysing them. However, the problem of constructing probabilistic behaviour models has received less attention. To aid engineers in building probabilistic behaviour models, some approaches (e.g., [1, 6, 7]) assume the existence of a non-probabilistic behaviour model and develop techniques that annotate transitions with probabilities based on additional information such as operational profiles, execution traces, and probability estimations. Such kinds of approaches have been used to probabilistically reason about system behaviour in a number of software-related activities, such as system reliability prediction [1, 6] and model-based testing [7].

Regardless of the annotation techniques, these approaches consider the quality of the information (or lack thereof) used to generate the annotations as the main threat to the accuracy of the resulting annotated model. For instance, the reliability of estimations and the size of the execution trace sample space will impact the accuracy of the annotated model.

However, in this paper we show that *even when we use an accurate information source to compute probabilities for annotations, and have a model that captures precisely the non-probabilistic behaviour of the system, inaccurate results can be obtained due to the structure of the non-probabilistic model being annotated.* The essence of the problem is that the annotation processes are heavily influenced by the structure of the model being annotated and may produce different results when annotating non-probabilistic behaviour models that are behaviourally equivalent but structurally different, which may cause the analysis to produce inaccurate predictions (false positive or false negative results on probabilistic properties). We show that the notion of state refinement can explain that phenomenon and that it can be used to mitigate the problem. We show that state refinement is guaranteed to preserve or increase the accuracy of the initial annotated model. Based on this, we present a framework for iteratively improving the accuracy of a probabilistically-annotated behaviour model.

The rest of the paper is organised as follows. Section 2 introduces the motivating system that will be used throughout the paper to illustrate our concepts and techniques. The concept of state refinement and its properties is presented in

ESEC-FSE'09, August 23–28, 2009, Amsterdam, The Netherlands.

Section 3, while Section 4 describes our proposed framework. Conclusions and future work are presented in Section 5.

2. MOTIVATING EXAMPLE

In this section, we describe a set of experiments that use traces from a simulation or alternatively from observation of the use of an existing system. We show that depending on the structure of the behaviour model that is annotated with the observed probabilities, the resulting analysis and model checking may produce different results from the actual values derived from the traces.

Let us assume the existence of a system behaviour model (BM), which models the actual behaviour of the system under analysis, and the set of traces generated by a simulation of the system. The simulation is fed by an environment probabilistic behaviour model (EPBM) which dictates the usage profile. Using the simulation traces, we generate (using an Annotator tool) annotations for BM giving the occurrence probabilities of each transition so as to produce a probabilistically annotated behaviour model (PABM). This PABM is then analysed with respect to a set of desired properties using the PRISM model checker [4]. The results are compared with those produced by a Counter tool, which counts the total number of simulation traces that satisfy that property and calculates the probability of each property holding. The difference between these results, referred to as Δ, indicates the accuracy of the initial PABM.

The running example used in this paper is based on an application called TeleAssistance (TA), a distributed system for medical assistance, originally described in [3]. It consists in a web service process for remote assistance of patients. Once the process is started by the patient, it offers three choices: *sending the patient's vital parameters*, *sending a panic alarm by pressing a button* and *stopping the application*. The first message contains the patient's vital parameters that are forwarded to the Medical Laboratory service (LAB), where the data will be analysed. The LAB replies by sending one of the following results: *change drug*, *change doses* or *send an alarm*. The latter message triggers the intervention of a First-Aid Squad (FAS) by sending an alarm to the FAS. When the patient presses the panic button, the application also generates an alarm that is sent to the FAS. Finally, the patient may decide to stop the TA service. TA can fail in the following situations: sending an alarm to the FAS, receiving the data analysis from the LAB, or sending a change dose or change drug message to the patient. In all cases, the system goes to a final state indicating that a failure has occurred. Figure 1(a) depicts the TA behaviour model.

Let us suppose we are interested in validating the following properties with respect to the TA application:[1]

R1: The probability that no failures ever occurred is greater than 0.7

R2: After a changeDrug or changeDose has occurred, the probability that the next message received by the TA generates an alarm which fails is less then 0.015

R3: The probability that no failures ever occurred and the user has sent the vital parameters or pressed the panic button at least once is grater than 0.65

[1]The first two properties presented here were also originally described in [3], while the third one is an adaptation of other properties described therein.

Table 1: Probabilities of the environment controlled actions in the PABMs of Experiments 1 and 2

Action	Experiment 1	Experiment 2
vitalParamMsg	0.6	0.30
pButtonMsg	0.3	0.37
stopMsg	0.1	0.33

Table 2: Results for Experiment 1

Property	PRISM	COUNTER	Δ
R1	0.75299	0.753	0.00001
R2	0.01595	0.016	0.00005
R3	0.64400	0.630	0.01400

We conduct two different validation experiments, as described below.

Experiment 1. This experiment used the BM shown in Figure 1(a), which is the same one used in [3]. The EPBM is also the same used in that work and it is depicted by Figure 1(b). In that environment model, the user always chooses each of the options provided by the system with the same probability at each interaction cycle. As we can see from Table 1, the probabilities of the environment controlled actions (*vitalParamsMsg*, *pButtonMsg*, and *stopMsg*) in the PABM generated for this experiment represents the same probabilistic choices made by the user.

Table 2 shows the analysis results for this first experiment. We can see that Δ is small for all three properties analysed, which indicates that the provided BM is accurate with respect to that particular EPBM.

Experiment 2. This experiment is similar to the first one, with the exception that we used a different EPBM, in which the user chooses each option with different probabilities in each of three interactions with the system. We can see from Table 1 that this time the probabilities of the environment controlled actions in the generated PABM are not consistent with the environment probabilistic choices.

Table 3 shows the analysis results for this experiment. Here we can see that Δ has increased for properties R2 and R3, specially the latter, when compared to the results obtained in the first experiment. In addition, according to this experiment, PRISM indicates that property R2 is valid, while R3 is violated. However, the Counter indicates that both these results are false, and in fact R2 is violated, while R3 is valid. Therefore, this experiment has generated both a false positive and a false negative.

The above experiments show that if we use a behaviour model that properly represents the probabilistic choices made

Table 3: Results for Experiment 2

Property	PRISM	COUNTER	Δ
R1	0.92899	0.929	0.00001
R2	0.01500	0.031	0.01600
R3	0.59500	0.821	0.22600

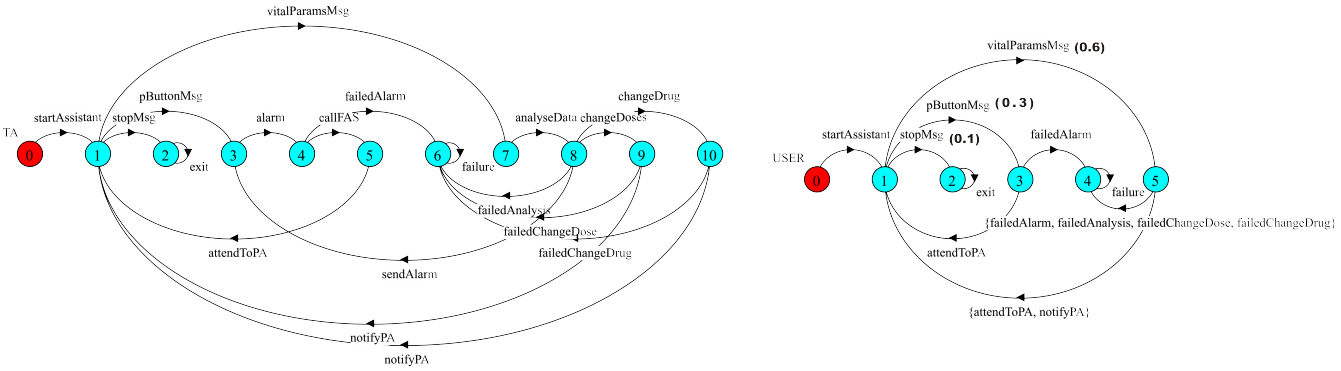

Figure 1: BM for the TeleAssistance application (left) and the EPBM used in Experiments 1 and 2 (right).

by the environment, this model is accurate and, consequently, we can expect more accurate predictions. This was observed in Experiments 1. On the other hand, using a behaviour model that does not take into account all the different probabilistic choices made by the environment may result in inaccurate predictions, as it happened for Experiment 2. We also conducted other experiments that corroborate those results. For more details about those experiments, the reader is referred to [5].

Therefore, in order to obtain accurate predictions, it is necessary that the system behaviour model has a representative structure. By representative we mean that the structure of the behaviour model has to accurately reflect and be consistent with the probabilistic choices made by the environment.

3. STATE REFINEMENT

We define *state refinement* as the process by which a behaviour model is transformed into a bisimilar model by expanding a particular state, i.e., by introducing in the new model a copy of that state and its transitions such that we obtain two bisimilar states. The effect of it is a partition on the state set of the refined model such that each partition is abstracted by one state in the abstract model. Consequently, the refined and the abstract models are not isomorphic, since the former has a different structure from the latter. The aim of state refinement is to find a behaviourally equivalent model that improves the structure of the model being refined, making it more realistic with respect to the real system behaviour. We denote by $P_1 \sqsubseteq P_2$ the fact that the model P_1 P_1 is state refined by the model P_2.

Given two models P_1 and P_2, $P_1 \sqsubseteq P_2$, we say that a state s of P_1 is an *abstract state* if there exists at least two bisimilar states to s in P_2. State refinement is applied to abstract states, since they are the states to be expanded in order to generate more realistic models.

The efficacy of state refinement in achieving more representative models is supported by two main properties:

- Preservation or improvement of the accuracy: *by state refining a model we always improve the accuracy or we do not obtain worse predictions.* This is explained by the fact that when a model is refined, it contains more information than the original model. In the worse case, the same set of probabilities used to annotate the original model is repeated for the bisimilar states of the refined one, generating the same prediction for a property. However, as the probabilities were misrepresented in the first model, they tend to be more precise in the new one, making that model yielding more precise results than the original one.

- Convergence: *there is at least one sequence of refinement steps that always generates an accurate model and that, at a certain point, no more refinement will be necessary.* One model can have several refinements, since it can have many abstract states. Furthermore, the same abstract set can be refined in different ways. Then, by the previous property, at least one of the refined models will be more accurate than the abstract one. If we choose the most accurate refined model and refine it, then it will generate a more accurate model yet. By repeating the process successively, we will always obtain a better model until we reach a model that produces results as accurate as the real probabilistic system behaviour model. At that moment, no more refinement is necessary and an accurate model is obtained.

Therefore, there is a real gain in accuracy when applying state refinement to a model. For more details about those properties, including formal proofs, the reader is referred to [5].

4. THE FRAMEWORK

We propose a framework to iteratively refine a provided behaviour model until we obtain an accurate model. The framework, shown in Figure 2, is divided into three phases. In Phase 1, we obtain a probabilistic behaviour model, represented as a probabilistic LTS (PLTS), for the system under analysis by annotating the occurrence probability of each transition on the provided system behaviour model, represented as an LTS. The probabilities can come from different sources, such as estimates made by an expert on the system, traces generated by monitoring the system execution or an operational profile.

Phase 2 takes the probabilistic model of Phase 1 and verifies its accuracy with respect to a group of properties provided by the framework user, called the *benchmark*. Each property is verified using a type of probabilistic verification, such as numerical analysis or probabilistic model checking. In order to verify whether the probabilistic model is accurate, the framework assumes that an *oracle* provides the true value for each property of the benchmark. An oracle can be, for instance, an expert on the system, such as the developer or the user, who provides values for each property based on their knowledge, or a set of traces of the system, obtained from logging the system execution. The accuracy

verification is carried out by comparing the result obtained from the probabilistic verification to the result provided by the oracle, regarding the same property.

If the model is inaccurate, it is fed into Phase 3, where state refine produces new models. These refined models then feed Phase 1 and the whole process is carried out again for each one, when the most accurate model among them is determined. If it is still inaccurate, then the process is repeated. The process terminates when a sufficiently accurate model is obtained.

The state refinement carried out in this phase consists of choosing a specific abstract state and then expanding it. We consider an abstract state a state that starts a loop, i.e., it has more than one outgoing transition and it can be reached by following all possible paths started from it. Once we identify an abstract state, we expand it using a loop unfolding technique, which consists of introducing additional copies of the states and transitions belonging to the loop, making the loop back transitions point to those new copies.

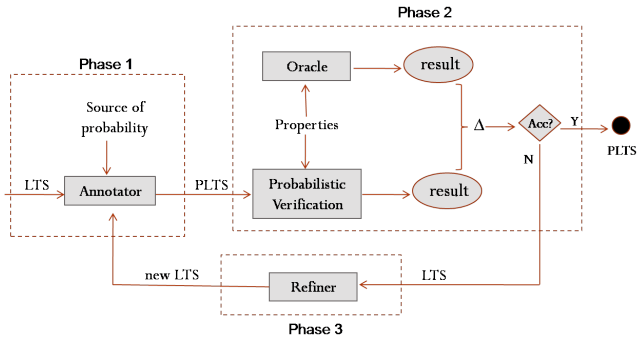

Figure 2: Framework for state refinement.

5. CONCLUSIONS AND FUTURE WORK

We conducted a case study using the TA system, the same presented in Section 2, in order to assess the gain in accuracy that the framework introduces, taking into account the number of refinement steps necessary and the size of the final model. We focused on property R2 and we used a set of 1000 traces generated randomly by the simulation of the TA as our source of probabilities. For that property, three refinement steps were needed to obtain an accurate model. However, the final model is more than eight times larger (w.r.t. the number of states) than the initial one. The scalability is a drawback of our state refinement technique, since it may lead to state space explosion. This problem is being investigated currently.

Quantitative behaviour analysis has been the focus of intensive research in the last decades, and applied to address different aspects of software behaviour within different problem domains [2, 7, 6, 1, 3]. All the those approaches share our goal of using probabilistic behaviour models to quantitatively predict system (in the case of [6] and [3]) or component (in the case of [1]) level properties. In addition, they all share our concern that having accurate models is key to improve prediction accuracy. However, none of those approaches take into account the fact that the *structure* of the target behaviour model (and not only its state and transition probabilities) may have a strong influence in the analysis results. We argue that it is important that both the probabilities be accurate and the model itself accurately reflects reality. We have shown an example which illustrates this, indicated its consequences, and proposed a framework to obtain accurate models through successive state refinements.

Our contribution is twofold. Firstly, we presented the importance of taking into account the structure of the model in the probabilistic analysis of properties. We presented the state refinement technique to mitigate inaccuracy prediction, and showed that it is guaranteed to preserve or increase the accuracy of the initial annotated model. Secondly, we provided a working framework for iteratively refining a model until an accurate model is obtained. We also introduced the concept of benchmarking of properties to guide our state refinement process. Currently, we are investigating other situations in which state refinement can be useful. In addition, we are conducting more case studies in order to validate and improve the framework.

6. ACKNOWLEDGEMENTS

This research is funded by CAPES (Brazil) under grant no. BEX 4257-05/7 and partially funded by CONICET:PIP112-00955KA4.

7. REFERENCES

[1] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik. Early prediction of software component reliability. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 111–120, Leipzig, Germany, 2008. ACM.

[2] P. R. D'Argenio, B. Jeannet, H. E. Jensen, and K. G. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *PAPM-PROBMIV '01*, pages 39–56, Aachen, Germany, 2001. Springer-Verlag.

[3] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. Model evolution by run-time adaptation. In *ICSE '09: Proceedings of the 31st International Conference on Software engineering*, Vancouver, Canada, 2009. ACM.

[4] M. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST '04: Proceedings of the The Quantitative Evaluation of Systems*, pages 322–323, Enschede, The Netherlands, 2004. IEEE Computer Society.

[5] P. H. Maia, J. Kramer, S. Uchitel, and N. C. Mendonça. An approach to obtain accurate probabilistic models using state refinement. Technical report, Department of Computing, Imperial College London, 2009.

[6] G. Rodrigues, D. Rosemblum, and S. Uchitel. Using scenarios to predict the reliability of concurrent component-based software systems. In *FASE'05 / ETAPS 2005: 8th International Conference on Fundamental Approaches to Software Engineering*, pages 111–126, Edinburgh, Scotland, 2005.

[7] J. A. Whittaker and M. G. Thomason. A markov chain model for statistical software testing. *IEEE Trans. Softw. Eng.*, 20(10):812–824, 1994.