Towards Compositional Approach for Parametric Model Checking in Software Product Lines

Vinícius Nunes, Danilo Mendonça, Genaína Rodrigues, Vander Alves
Computer Science Department
University of Brasília – Darcy Ribeiro Campus
70910-900, Brasília, DF, Brazil
{viniciusuriel, dfmendonca}@gmail.com, {genaina, valves}@cic.unb.br

Abstract

Parametric model-checking allows the use of a single model to obtain properties values from different configurations through an arithmetic formula. Formulas with hundreds of operands and operators can be evaluated at runtime in milliseconds on current computers. Nevertheless, those formulas may not scale to devices with limited resources. This work-in-progress addresses this problem by presenting a compositional parametric model checking approach able to produce partial factorized formulas. This approach simplifies the model checking by verifying smaller models separately instead of a unique large model since the effort to recombine the formulas is as simple as text replacement.

1 Introduction

Software dependability evaluation is an important issue, especially when it comes to critical systems. Estimating software reliability early in the development cycle allows us to make important decisions in design phase. However, ensuring dependability is not a trivial problem. Model checking can be used to estimate the reliability of a software through models that represent the behaviour of the system. By performing sensitivity analysis of the components, it is possible to identify which components are most critical in the software quantitative assessment.

Parametric model checking is a technique that allows late evaluation of parameters in a model. This technique outputs an arithmetical formula parametrized with parameters defined in the model. This flexibility can be used to deal with variabilities of Software Product Lines (SPLs) directly in the models avoiding the infeasible work of building different models for different products of a SPL as showed in our previous work and in some other related ones [5, 2].

PARAM is a tool for probabilistic parametric model checking. We call PARAM model one that uses this variant of the PRISM language. The PARAM tool takes as input a PARAM model and a PCTL expression and produces as output an arithmetical formula with the parameters defined in the model. Through the evaluation of these parameters it is possible to obtain the values that answer the queries in the given Probabilistic Computation Tree Logic (PCTL) [4].

Evaluating a formula at runtime brings to light some practical issues related with the formula size. Some authors have already warned that the excessive use of parameters in the model can lead tools to actually not make the model checking and just output a formula representing all the computation [3, 2]. In our last work, [5], we showed that wrong choices in modeling strategy can lead models to output large formulas and detailed the PARAM tool process of obtaining the arithmetical formula from a parametric model emphasizing the modelling decisions and relating it to its actual impact on the formula size in a quantitative way.

Formulas with hundreds of operands and operators can be evaluated at runtime in milliseconds on current computers. The same is not true when it comes to mobile devices with limited resources. This position paper addresses this problem by presenting a compositional parametric model checking approach (CPMC) able to reduce the formula size by producing partial factorized formulas and removing redundancies observed in fully expanded formula.

We follow a modular approach by dividing the software behaviour specification in independent parts such that each part can be modelled and checked separately producing independent formulas. We also show how these formulas can be composed to generate the partial factorized formula corresponding to the model checking of the entire SPL. This approach simplifies the model checking by verifying smaller models separately instead of a unique large model since the effort to recombine the formulas is as simple as text replacement.
To the best of our knowledge, the most related work deals with incremental model checking in SPL [1]. This technique allows changes in the features of the SPLs without needing to recheck the entire model. By tracking features dependencies, the technique can determine which part of the model needs to be rechecked. This technique can only be applied when adding conservative features, i.e., features that do not remove behavior from the system.

CPMC is also possible to take the advantages of incremental model checking in certain cases. Although we do not restrict the type of changes in the features (conservative or non conservative), the changes in the feature variability must be tracked from the feature model to the documentation in order to determine the sequence diagram affected and to determine which models needs to be rechecked. In the next section we further detail this approach.

2 The CPMC Approach

CPMC uses as input UML sequence and activity diagrams. These diagrams must be designed in such a way that the activity diagram represents a broader view of a scenario and each activity is further detailed by sequence diagrams. Thus, there are two abstraction levels for a given scenario. The first level, which provides the broader view, is referred to as activity level; the second level, which details each activity, is referred to as component level since the sequence diagrams describes interaction among components. Figure 1 presents an activity diagram with one of its activities detailed by several sequence diagrams.

Figure 1. Sequence diagrams detailing an activity

We call arrow path a path that connects an activity to another one, possibly passing through some decision nodes (diamond shape). For example, the third activity of Figure 1 presents two leaving arrow paths: one arriving at the fourth activity and another arriving at final state. Each leaving arrow path represents a different ending point of the component level model of the previous activity.

At the first level, each activity arrow path represents an independent atomic execution with associated reliability formula. Each execution is considered atomic in the sense that it cannot be further detailed at that level (activity level). The reliability value of each activity is calculated using the reliability of the components according to its underlying sequence diagrams. At the component level, sequence diagrams related with each activity are used to build a model as described in [5]. Each different success final state of a given component-level model is considered a different execution of its corresponding activity.

Figure 2 illustrates the process of obtaining the factorized formula in CPMC. This process encompasses the following steps:

1. build the PARAM models at component level corresponding the activities. This step takes as input the feature model, UML diagrams, SPL CK and any additional documentation needed to interpret diagrams;

2. build the PARAM model at activity level. This step takes as input the same inputs as the first step except the SPL related documentation;

3. model check each component-level model to obtain its corresponding formulas;

4. model check the activity level model to obtain the fully parameterized formula representing the factorization;

5. the formulas obtained in the third step are combined as defined in the formula obtained in the fourth step resulting in the factorized formula.

Note that the step 2 does not take documentation related with variability as input, since CPMC does not deal with variability at activity diagram level.

The activity level model referred on step 2 is modeled with the following rules:

1. Each arrow path has a corresponding parameter \( v \), such that \( v \in V \) where \( V \) is the set of arrow paths parameters representing its associated success probability.

2. The activity diagram is modelled as a single PARAM model.

3. Each activity \( a \in A \) such that \( A \) is the set of activities, is modeled as a single state.
4. Each arrow path \((a_s, a_d) \in A \times A\), where \(a_s\) is the source activity and \(a_d\) is the destination activity of the arrow path, is mapped to a PARAM transition leaving the state associated with the activity \(a_s\) and arriving at the state associated with activity \(a_d\). Each PARAM transition is labelled with one different parameter \(v\) given by the \(L: A \times A \rightarrow V\) labelling function such that \(L((a_s, a_d)) = v\).

5. For each state of the PARAM model associated with an activity \(a_s\), we add a transition representing the chance of failure associated with the activity given by \(1 - (\sum_{(a_s, a_d) \in A \times A} L((a_s, a_d)))\).

Each activity comprises one or more atomic executions, thus there is no need to detail its execution on a separated PARAM module with a single state (Rules 2 and 3). Each different activity arrow path leads to a different next activity. Therefore a state on the activity level model has a leaving transition for each different final state of its corresponding component-level model and each of such transitions is labelled with a different parameter (Rules 1, 4). Each arrow path has a corresponding formula generated with the sequence diagrams associated with the source activity of the arrow path that will be used as replacement to its corresponding parameter in the step 5 of the process (see Figure 2). Each activity has an associated chance of failure representing the aggregated chance of failure of all arrow paths of an activity (Rule 5).

For example, Figure 3 and Listing 1 present an activity diagram and its corresponding activity level model. Each state is labeled with a action to better illustrate the correspondence between the model and the diagram.

**Listing 1. Activity Level PARAM Model**

```plaintext
dtmc

param double capture;
param double situation;
param double qosgoal1;
param double qosgoal2;

```

**Figure 2. Compositional Parametric Model Checking Process**

2.1 Rationale

The assumption behind CPMC is that factorization leads to simple formulas by reducing redundancies of parameters in the formula. This section addresses the two main questions related with this assumption: (1) how does CPMC
generate a factorization of the formula? and (2) why can the factorization lead to simpler formulas?

The factorization is achieved by the hierarchical parameterization of the SPL model. At high level, activities are independent, so parametric modelling can be performed. The sequence diagrams associated with each activity rely on the same assumptions of the approach presented in [5], thus they can be modelled that way. CPMC proposes a way to split the SPL model in smaller models representing sequence diagrams grouped by activities and recombine them. This grouping represents the partial factorization obtained in the compositional approach.

![Diagram](image)

(a) Non-Compositional Parametric Model Checking Approach

![Diagram](image)

(b) Compositional Parametric Model Checking Approach

Figure 4. Approaches Comparison

Figures 4(a) and 4(b) present examples of the non-compositional approach presented on [5] and of the proposed compositional approach for the same SPL. Z, W, X are parameters associated with non-mandatory features of the SPL.

Notably the non-compositional approach has a simplified process to obtain the parametric formula with fewer steps and fewer models. One of the motivations of that approach is to build a single model to represent the entire SPL, although CPMC requires more models it does not invalidate the assumptions of this motivation, since the number of models is not tied to the number of configurations of the SPL.

In CPMC, the formula obtained from activity diagram \((A + BC)\) represents a factorization of the formula presented on Figure 4(a) \((Z + W \times X + 3 \times W)\) and the formulas obtained from sequence diagrams are mapped to parameters of the factorized formula.

This example illustrates how the factorization can reduce the formula size. The formula presented on Figure 4(a) has four operations (two multiplications and two sums), on the other formula presented on Figure 4(b) has only three operations (two sums and one multiplication). Additionally, the second formula avoids the repetition of the parameter \(W\) which reduces the effort of evaluation of this parameter since it has only one occurrence. Note that we count the number of operations as the number of operators explicitly expressed in the formulas and that the actual number of operations executed depends on the evaluation strategy.

Observe that the formulas generated by both approaches uses the same parameters and results in the same values for the same evaluation of its parameters.

3 Conclusion

This paper presented our work-in-progress compositional parametric model checking approach to support runtime evaluation of SPL, e.g., at devices with limited computational resources. Our hypothesis is that CPMC is applicable to any SPL since the factorized result can be expanded to its full-size form by applying the distributive property which gives flexibility to the SPL application engineer. For future work, we plan to carry out case studies in order to fully assess the appropriateness of our proposed CPMC approach versus a NCPMC one and potentially automate the process.

References


