ObjectCheck: A Model Checking Tool for Executable Object-Oriented Software System Designs

Fei Xie[1], Vladimir Levin[2], and James C. Browne[1]

[1] Dept. of Computer Sciences, Univ. of Texas at Austin, Austin, TX 78712, USA
{feixie,browne}@cs.utexas.edu
[2] Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974, USA
levin@research.bell-labs.com

1 Introduction

Specifying software system designs with executable object-oriented modeling languages such as xUML [1][2], an executable dialect of UML, opens the possibility of verifying these system designs by model checking. However, state-of-the-art model checkers are not directly applicable to executable object-oriented software system designs due to the semantic and syntactic gaps between executable object-oriented modeling languages and input languages of these model checkers and also due to the large state spaces of these system designs.

This paper presents ObjectCheck, a new tool that supports an approach [3] to model checking executable object-oriented software system designs modeled in xUML. The approach can be summarized as follows:

- A software system design is specified in xUML as an executable model according to Shlaer-Mellor Method [4].
- A property to be checked on the design is specified in an xUML level logic.
- The xUML model may be abstracted, decomposed, or transformed to reduce the state space that must be explored for checking the property.
- The xUML model and the property are automatically translated to a model and a query in the S/R [5] automaton language. Prior to the translation, the xUML model is reduced with respect to the property as proposed in [6].
- The S/R query is checked on the S/R model by COSPAN [5] model checker.
- If the query fails, an error track is generated by COSPAN and is automatically translated into an error report in the name space of the xUML model.

ObjectCheck provides comprehensive automation for the approach by supporting xUML level property specification, xUML-to-S/R translation and optimization, error report generation, and error visualization. ObjectCheck also provides preliminary automation for the integrated state space reduction proposed in [7].

The most closely related work to ObjectCheck are a toolset for supporting UML model checking based on Abstract State Machines [8], and the vUML

R.-D. Kutsche and H. Weber (Eds.): FASE 2002, LNCS 2306, pp. 331–335, 2002.

tool [9]. Both tools translate and verify UML models based on ad hoc execution semantics that do not include action semantics while action semantics of xUML follows a proposal for Action Semantics for UML [10], which has been finalized by OMG. ObjectCheck combines commercially supported software design environments [1][2] and model checkers with research tools to provide a comprehensive capability for model checking xUML models.

2 xUML Semantics

In xUML, a system is composed of instances of classes, which are either active, having dynamic behaviors, or passive, having no dynamic behaviors and being used to store data. There can be association and generalization relationships defined among classes. A large system can be recursively partitioned into packages, which are groups of classes closely coupled by associations and generalizations.

The execution behavior of a class instance is specified by an extended Moore state model where each state has an associated action that is executed in a run-to-completion mode upon entry to the state. State transitions are invoked by messages. State actions can be categorized as follows:

- Read or write actions that read or write attributes of class instances, or dynamically create or delete class instances;
- Computation actions that perform various mathematical calculations;
- Messaging actions that send messages to active class instances;
- Composite actions that are control structures and recursive structures that permit complex actions to be composed from simpler actions;
- Collection actions that apply other actions to collections of elements, avoiding explicit indexing and extracting of elements from these collections.

The execution behavior of an xUML model is an asynchronous interleaving of the executions of the state models of active class instances in the model.

3 Overview of ObjectCheck

To provide comprehensive automation support for model checking xUML models, ObjectCheck is structured as shown in Figure 1. Under the architecture, we selected industrial toolsets such as Bridgepoint [2] or Objectbench [11], as the xUML visual editors and COSPAN as the model checking engine. We incorporated the optimization module of SDLCheck [12] that implements Static Partial Order Reduction (SPOR) and other software specific model checking optimizations. Furthermore, we implemented the following components of ObjectCheck:

Property Specification Interface. The property specification interface enables formulation of properties to be checked on xUML models in an xUML level logic. The logic defines a set of temporal templates such as `Always` and `Eventually`. A property formulated on an xUML model consists of instantiations of these templates with propositional logic expressions over the semantic constructs of the xUML model.

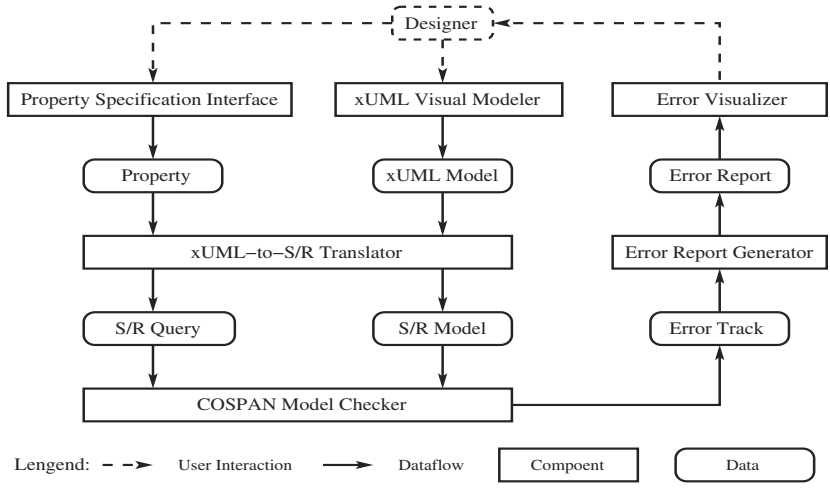

Fig. 1. Architecture of ObjectCheck

xUML-to-S/R Translator. The translator inputs an xUML model and a property to be checked on the model and outputs an S/R model and a query to be checked on the S/R model. Details of the translation can be found in [3].

Error Report Generator. When an S/R query fails on an S/R model, COSPAN generates an error track specifying an execution trace inconsistent with the query. The error report generator compiles an error report in xUML notations from the error track. The error report consists of an execution trace of the corresponding xUML model, which violates the corresponding xUML level property.

Error Visualizer. To facilitate debugging an error found by COSPAN in an xUML model, an error visualizer is provided, which generates a test case from the error report and reproduces the error by executing the xUML model with the test case in a simulator included in the xUML visual editor.

State Space Reduction. ObjectCheck supports powerful state space reduction algorithms. Localization Reduction [5] and Symbolic Model Checking (SMC) are performed by COSPAN. The xUML-to-S/R translator makes use of the optimization module of SDLCheck, which has been modified to reflect xUML semantics. The module transforms an xUML model to reduce the model checking complexity of the resulting S/R model and, in particular, it implements SPOR that reduces the set of possible interleavings of executions of state models (which, otherwise, all should be explored in the model checking phase) by eliminating the interleavings irrelevant to the property to be checked. Currently, a state space

reduction manager is being developed, which, together with other components of ObjectCheck, implements the integrated state space reduction proposed in [7].

4 Applications

ObjectCheck has been successfully applied in model checking the xUML models of a number of interesting examples such as a robot controller system, which is previously reported in [13], and an online ticket sale system. An illustration of applying ObjectCheck to the online ticket sale system follows.

The xUML model of the online ticket sale system is composed of instances of three classes: Dispatcher, Agent, and Ticket Server. The system processes concurrent ticketing requests submitted by customers. A liveness property to be checked on the model is that after an agent is assigned to a customer, eventually the agent will be released.

The xUML model is translated into two S/R models: one with SPOR off and the other with SPOR on. The S/R query corresponding to the liveness property was checked on the two S/R models by COSPAN with two different state space searching algorithms: Explicit State Enumeration and SMC. The computational complexities of the four model checking runs are compared in Table 1. Both

Table 1. Comparison of Model Checking Complexities

SPOR	SMC	Memory Usage	Time Usage
Off	Off	Out of Memory	–
Off	On	113.73M	44736.5S
On	Off	17.3M	6668.3S
On	On	74.0M	1450.3S

SPOR and SMC lead to significant reduction on the model checking complexity. The combination of SPOR and SMC leads to less running time, but requires more memory than applying SPOR only. It can be observed that no single reduction algorithm alone can achieve an overwhelming advantage over other reduction algorithms. Therefore, various combinations of state space reduction algorithms on various types of xUML models have to be studied for better combinations.

5 Conclusions

ObjectCheck has facilitated effective model checking of non-trivial software system designs represented as xUML models. Ongoing research in state space reduction at the xUML model level shows significant promise for enabling model checking of substantial software system designs specified as xUML models.

Acknowledgement

We gratefully acknowledge Robert P. Kurshan, Natasha Sharygina, and Husnu Yenigün. This work was partially supported by TARP grant 003658-0508-1999.

References

1. Kennedy Carter: `http://www.kc.com/html/xuml.html`. Kennedy Carter (2001)
2. Project Tech.: `http://www.projtech.com/pubs/xuml.html`. Project Tech. (2001)
3. Xie, F., Levin, V., Browne, J.C.: Model Checking for an Executable Subset of UML. Proc. of 16th IEEE International Conf. on Automated Software Engineering (2001)
4. Shlaer, S., Mellor, S.J.: Object Lifecycles: Modeling the World in States. Prentice-Hall, Inc (1992)
5. Hardin, R.H., Har'El, Z., Kurshan, R.P.: COSPAN. Proc. of 8th International Conf. on Computer Aided Verification (1996)
6. Kurshan, R.P., Levin, V., Minea, M., Peled, D., Yenigün, H.: Static Partial Order Reduction. Proc. of 4th International Conf. on Tools and Algorithms for the Construction and Analysis of Systems (1998)
7. Xie, F., Browne, J.C.: Integrated State Space Reduction for Model Checking Executable Object-oriented Software System Designs. Proc. of FASE 2002 (2002)
8. Compton, K., Gurevich, Y., Huggins, J.K., Shen, W.: An Automatic Verification Tool for UML. Univ. of Michigan, EECS Tech. Report CSE-TR-423-00 (2000)
9. Lilius, J., Porres, I.: vUML: a Tool for Verifying UML Models. Proc. of the Automatic Software Engineering Conf. (1999)
10. OMG: Action Semantics for the UML. OMG (2000)
11. SES: Objectbench User Manual. SES (1996)
12. Levin, V., Yenigün, H.: SDLCheck: A Model Checking Tool. Proc. of 13th International Conf. on Computer Aided Verification (2001)
13. Sharygina, N., Kurshan, R.P., Browne, J.C.: A Formal Object-oriented Analysis for Software Reliability. Proc. of 4th International Conf. on FASE (2001)