Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 110 (2004) 169–172
www.elsevier.com/locate/entcs

Calife: A Generic Graphical User Interface for Automata Tools [1]

Bertrand Tavernier[2]

CRIL Technology - 9 - 11 rue Jeanne Braconnier
92360 Meudon la fort - France

Abstract

In this paper, we present a platform that can be used to interface tools working on automata. The Calife platform works on several automata models (transition systems, timed automata, counter automata,...) and allows to define new models and interface new tools. Several tools are currently interfaced with the platform (Uppaal, Hytech, Kronos, CMC, Coq,...) and a unique timed-automata system modelled under the Calife System Editor can be exported to all these tools.

Keywords: Graphical User Interface, Automata Models, Model-Checker, Hytech, Uppaal, Kronos, Coq.

1 Introduction

The Calife v3.0 platform is an environment developed under the GPL licence (freely downloadable at `http://calife.criltechnology.com`) allowing the specification and formal validation of systems described as synchronized product of automata.

The goal is not to provide another verification tool, but to interface (all?) existing tools working on automata in a unique environment. Tools can be grouped using model definition associated with automata classes.

[1] Partly supported by the french projects RNRT Calife and RNTL Averroes.
[2] Email: *bertrand.tavernier@criltechnology.com*

doi:10.1016/j.entcs.2004.06.004

2 Overview of the Calife Platform

The Calife platform is made of three layers:

- The system editor which allows the user to model (in a graphic form) a system as a synchronized product of automata, described in a generical formalism. The grammar of predicates used in automata is fixed using a type system defined in a "model" file (*cf* §3).
- A model compiler which checks that predicates introduced in the component are syntactically correct, w.r.t. the automata model on which the system is built (timed or hybrid automata). The compiler also checks that all variables, parameters and used functions are declared.
- A script engine (and a XSLT Processor [11]) in charge of scripts execution and code generation in the target language (*cf* §4).

The Calife platform is designed for two kinds of users. The"super-user" defines automata models and write export scripts to connect new tools (or abstractions) to the platform. That kind of user is in charge of the semantics of models w.r.t. the connected tools. The "standard-user" models systems in the editor and uses the tools interfaced to make proofs.

3 Automata and Models

XML Automata. Automata are described using the XML standard [10]. The key of interoperability between tools is to represent every predicate as an Abstract Syntax Tree (AST in short) where:

- The root is a Guard, Invariant, Updates or Activity node
- The leaves are Variable, Parameter or Constant nodes
- The intermediary nodes are associated with functions (but some usual operators are pretty-printed for flexibility reasons).

Of course, that mechanism is totally hidden to the user who defines automata in a graphical way.

Automata models. Automata classes are defined in XML Automata models [1]. The first part of models is made of a type system defining the syntax of automata. This type system consists of:

- Data type definitions. Variable types, parameter types and constants types are defined separately. Constants types are defined using a regular expression.
- Polymorphic function definitions. Functions are declared by their signatures

which are the only relevant information from a type-checking point of view.

Export Scripts. Tools integration is the second part of an XML Automata model. Every tool in the model can be used when the defined system is correctly typed with respect to the corresponding type system.

Tools are integrated using "export scripts" described in XML. Every XML node in the script is a specific action runned by the Calife script engine.

A text editor called *CalifeEdit* [1] is used to get some specific inputs from the user and to execute sub-scripts (for example for doing forward or backward analysis,...).

4 Target Code Generation

The target code generation can be splitted into two parts: generating the predicates in the target language and modelling the synchronized product in a consistent way with respect to the semantics of synchronization in the tool.

Translating predicates. That step is performed using an XSL Transformation [11]. Since every predicate is represented as an AST, the corresponding code can be generated searching through the tree from the root to the leaves and associating a simple rule for code generation with every node encountered. These rules are defined using an "xsl:template" node [11].

Modelling the synchronized product. Several techniques can be used to generate consistent code:

- direct translation of the synchronization table for tools using that king of synchronization principle (CMC [5] or Coq [6] for example). Code defining the table can be generated using a very simple XSL transformation.
- translation using an injective function which associates a single label with every synchronization vector. Every transition is replicated for each instance of its label in the table. This technique is used for tools like Kronos [4] and HYTECH [3]. Code can be generated using an XSL transformation.
- complex translation using XML pattern matching programming with Tom tool [9]. This technique is used to interface the UPPAAL tool [2] where new states must be introduced for every synchronization vector using more than one "non-epsilon" label. Complex abstractions are also made using this technique.

5 Conclusion

We summarize in the table below models and tools currently interfaced by Calife :

Models – Tools	Kronos	Hytech	Uppaal	CMC	Coq	Elan[7]
Timed Automata	×	×	×	×	×	×
Extended Timed Automata		×	×	×	×	×
Stopwatch Automata		×		×	×	
Linear Hybrid Automata		×				
P-Automata [8]		×			×	

References

[1] B. Tavernier. "Calife - User Manual". Available at http://calife.criltechnology.com

[2] "UPPAAL". Available at http://www.uppaal.com/

[3] "HyTech". Available at http://www-cad.eecs.berkeley.edu/~{}tah/hytech/

[4] "Kronos". Available at http://www-verimag.imag.fr/TEMPORISE/kronos/

[5] "CMC". Available at http://www.lsv.ens-cachan.fr/~{}fl/cmcweb.html

[6] "The Coq proof assistant". Available at http://coq.inria.fr

[7] ELAN. *ELAN*. Available at http://elan.loria.fr/.

[8] "Automates temporisés Calife". Available at http://www.loria.fr/calife

[9] "The Tom Compiler". Available at http://tom.loria.fr/

[10] Extensible Markup Language (XML) 1.0. W3C Recommendation, October 6, 2000. http://www.w3.org/TR/2000/REC-xml-20001006

[11] Extensible Stylesheet Language (XSL) Version 1.0. W3C Recommendation, October 15, 2001. http://www.w3.org/TR/xsl/