Implementing Symbolic Models for Value Passing in TwoTowers

Marco Bernardo

Università di Torino, Dip. di Informatica, Corso Svizzera 185, 10149 Torino, Italy
`bernardo@di.unito.it`

Abstract. We describe an extension of TwoTowers, a software tool for the functional and performance analysis of concurrent and distributed systems modeled in EMPA, in which a symbolic model based support for data driven computations is implemented and we recall its advantages both from the expressiveness and the analysis standpoint.

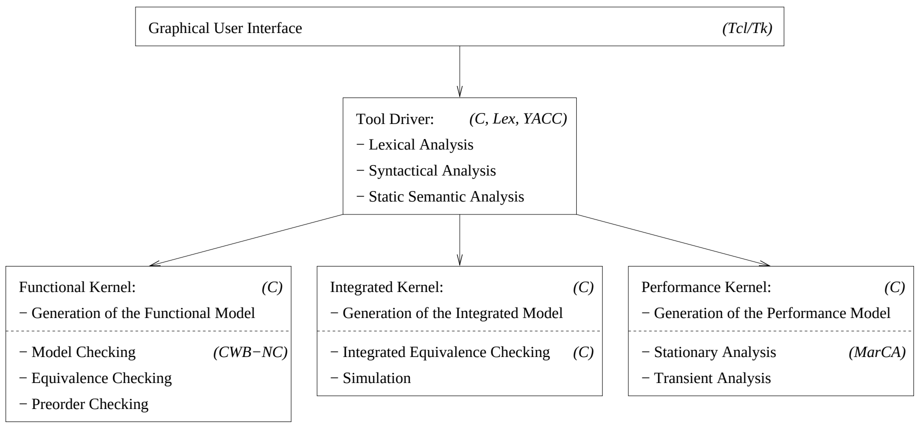
1 Introduction

TwoTowers [4] is a software tool for the functional and performance analysis of concurrent and distributed systems compositionally modeled in the stochastically timed process algebra EMPA [3]. The purpose of TwoTowers is similar to that of other stochastically timed process algebra based tools, such as the PEPA Workbench [7] and the TIPP-tool [8]. However, besides the difference in the expressive power of the algebras used by these tools and the fact that TwoTowers also implements a multiparadigm approach involving generalized stochastic Petri nets [3] (which however is outside the scope of this paper), the underlying philosophies are quite different. Since many functional properties of a system may be examined independently of performance properties and vice versa, TwoTowers uses two existing tools that we have retargeted to EMPA for the analysis of these two different types of properties. CWB-NC [6] provides a variety of different types of analysis (e.g., model checking, equivalence checking, preorder checking) of the functional behavior of systems, while MarCA [11] conducts steady state and transient performance analysis. Exploiting CWB-NC and MarCA has been advantageous both from our point of view and from the point of view of the users. We did not need to write code for implementing most of the analysis routines, thereby making the development of TwoTowers easier and faster, and we provided users with a full range of automated techniques implemented in widely used tools.

Unfortunately, when using the version of TwoTowers described in [4], a large class of systems cannot be dealt with. Such systems are those where data plays a fundamental role, so it is not possible to abstract from data in the modeling process. Think e.g. of a system that receives messages and undertakes different activities depending on the contents of the received messages. In this paper we briefly describe our effort to implement in TwoTowers a support, based on improved symbolic models, for data driven computations and we recall the advantages gained both from the expressiveness and the analysis viewpoint.

2 Implementing Value Passing in TwoTowers

TwoTowers, whose architecture is depicted in Fig. 1, is composed of 20,000 lines of code divided into a graphical user interface, a tool driver, an integrated kernel, a functional kernel, and a performance kernel. The modifications necessary to deal with systems exhibiting data driven computations are mainly concerned with the tool driver and the integrated kernel.



Graphical User Interface (Tcl/Tk)
Tool Driver: (C, Lex, YACC)
− Lexical Analysis
− Syntactical Analysis
− Static Semantic Analysis
Functional Kernel: (C)
− Generation of the Functional Model
− Model Checking (CWB−NC)
− Equivalence Checking
− Preorder Checking
Integrated Kernel: (C)
− Generation of the Integrated Model
− Integrated Equivalence Checking (C)
− Simulation
Performance Kernel: (C)
− Generation of the Performance Model
− Stationary Analysis (MarCA)
− Transient Analysis

Fig. 1. Architecture of TwoTowers

It is well known that data driven computations can be modeled in process algebras like EMPA by introducing the capability for processes of passing values to one another [10]. Thus, the tool driver of the new version of TwoTowers has been enhanced to accept system descriptions in EMPA$_{\rm vp}$ [3], a value passing extension of EMPA. A system description in EMPA$_{\rm vp}$ is a sequence of parametrized constant definitions of the form

$$A(t_1\ f\_par_1, \dots, t_n\ f\_par_n; t'_1\ l\_var_1, \dots, t'_m\ l\_var_m) \stackrel{\Delta}{=} E$$

where formal parameters and local variables are declared. Formal parameters are bound to actual parameters whenever a parametrized constant invocation of the form $A(a\_par_1, \dots, a\_par_n)$ occurs. Local variables, instead, are used to get values from other parallel processes via synchronization.

Term E above is compositionally expressed using operators such as action prefix ($<\alpha, \tilde{\lambda}>.F$), alternative composition ($F_1 + F_2$), and parallel composition ($F_1 \parallel_S F_2$). As far as actions are concerned, besides the usual distinction among exponentially timed ($\tilde{\lambda} \in \mathbf{R}_+$), immediate ($\tilde{\lambda} = \infty_{l,w}$) and passive ($\tilde{\lambda} = *$) actions, in EMPA$_{\rm vp}$ we have also the distinction among unstructured (including the invisible τ), input ($\alpha = a?(x)$) and output ($\alpha = a?(e)$) actions. Whenever an output action of a process synchronizes with an input action of another process, the former process passes the values carried along with its output action to the latter process, which can then store the received values for subsequent use (by employing a parametrized constant invocation) or look at them immediately in order to decide which action to undertake next. To accomplish this, EMPA$_{\mathrm{vp}}$ comes equipped with a conditional operator of the form "if (β, p) then E_1 else E_2". Its meaning is given by the internal immediate choice $<\tau, \infty_{l,p}>.E_1 + <\tau, \infty_{l,1-p}>.E_2$ governed by the boolean expression β. In practice, in case of simulation the boolean expression β is considered to solve the choice because we are able to evaluate it, while in case of numerical analysis the (guess about the) probability p that β holds true is used.

We recall that all the formal parameters and local variables declared in the header of a constant definition have a type t defined by the following syntax

$$t ::= \mathrm{int} \mid \mathrm{real} \mid \mathrm{bool} \mid \mathrm{list}(t) \mid \mathrm{array}(\mathit{length}, t)$$

Thus, the new version of the tool driver, besides parsing EMPA$_{\mathrm{vp}}$ specifications, implements routines for the type checking of parametrized constant invocations and structured action synchronizations. Additionally, the tool driver performs other static checks which prevents each formal parameter from being referenced within input actions and each local variable from being referenced before occurring in an input action (which should synchronize with an output action thereby setting the local variable).

The new version of the integrated kernel, instead, implements the semantic rules for EMPA$_{\mathrm{vp}}$. These rules produces compact versions of symbolic models called symbolic transition graphs with lookahead assignment (STGLA) [3], which are an improvement of the symbolic models proposed in [9]. These models are called symbolic because the variables occurring within the actions labeling their transitions are kept symbolic, i.e. they are not replaced by values, thereby avoiding infinite branching structures. In a STGLA, states are in correspondence with EMPA$_{\mathrm{vp}}$ terms and have a set of free variables (which correspond to formal parameters and local variables of parametrized constants), while transitions are labeled with triples each of which is composed of an action, a boolean guard enabling/disabling the transition (see the boolean expressions in conditional operators), and a set of assignments to be performed in the derivative state as a consequence of structured action synchronizations and/or parametrized constant invocations.

Besides implementing the generation of compact versions of STGLA according to the semantic rules for EMPA$_{\mathrm{vp}}$, which is complicated by the fact that the dot notation and localities must be used to avoid variable name clashes as explained in [3], the new version of the integrated kernel contains a routine inspired by the algorithm of [9] to check two EMPA$_{\mathrm{vp}}$ specifications for Markovian bisimulation equivalence. Furthermore, the routine for the simulative analysis of performance has been enhanced with the capability of evaluating expressions and executing assignments labeling transitions in the correct order [3].

3 Advantages of Value Passing

The extension of TwoTowers described above yields three advantages.

First of all, value passing allows systems with data driven computations to be expressed. As an example, in [1] we have been able to measure, with the new version of TwoTowers, the cell loss ratio, the mean queue length, and the link utilization for an ATM switch implementing the explicit rate marking mechanism for ABR traffic.

Second, by means of the value passing machinery, and in particular the presence of data expressions, it is possible in our Markovian framework to model systems whose activities have generally distributed durations. As an example, in [5, 2] we have been able to predict and compare via simulation the QoS of several adaptive mechanisms for the transmission of packetized audio over the Internet, where transmission delays were assumed to follow a Gaussian distribution.

Third, the compactness of the generated STGLA can be exploited at analysis time. As an example, in [3] we have modeled the alternating bit protocol both in pure EMPA and in EMPA_{vp}. It turned out that the STGLA underlying the latter model had only 50% of the states of the former model.

References

1. A. Aldini, M. Bernardo, R. Gorrieri, *"An Algebraic Model for Evaluating the Performance of an ATM Switch with Explicit Rate Marking"*, in Proc. of PAPM '99, Prensas Universitarias de Zaragoza, pp. 119-138, 1999
2. A. Aldini, M. Bernardo, R. Gorrieri, M. Roccetti, *"A Simulative Analysis of Internet Audio Mechanisms Using Formal Methods"*, in Proc. of ESS '99, SCS International, pp. 281-288, 1999
3. M. Bernardo, *"Theory and Application of Extended Markovian Process Algebra"*, Ph.D. Thesis, University of Bologna (Italy), 1999 (`http://www.di.unito.it/~bernardo/`)
4. M. Bernardo, R. Cleaveland, S. Sims, W. Stewart, *"TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems"*, in Proc. of FORTE/PSTV '98, Kluwer, pp. 457-467, 1998
5. M. Bernardo, R. Gorrieri, M. Roccetti, *"Formal Performance Modelling and Evaluation of an Adaptive Mechanism for Packetized Audio over the Internet"*, in Formal Aspects of Computing 10:313-337, 1999
6. W.R. Cleaveland, S. Sims, *"The NCSU Concurrency Workbench"*, in Proc. of CAV '96, LNCS 1102:394-397, 1996
7. S. Gilmore, J. Hillston, *"The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling"*, in Proc. of PERFORMANCE TOOLS '94, LNCS 794:353-368, 1994
8. H. Hermanns, V. Mertsiotakis, M. Rettelbach, *"A Construction and Analysis Tool Based on the Stochastic Process Algebra TIPP"*, in Proc. of TACAS '96, LNCS 1055:427-430, 1996
9. Z. Li, H. Chen, *"Computing Strong/Weak Bisimulation Equivalences and Observation Congruence for Value-Passing Processes"*, in Proc. of TACAS '99, LNCS 1579:300-314, 1999
10. R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989
11. W.J. Stewart, *"Introduction to the Numerical Solution of Markov Chains"*, Princeton University Press, 1994