Recent extensions to the stochastic process algebra tool CASPA

Martin Riedl and Johann Schuster and Markus Siegle
Universität der Bundeswehr München
{martin.riedl, johann.schuster, markus.siegle}@unibw.de

I. Introduction

Development of the stochastic process algebra tool CASPA started in 2003 [7]. CASPA is fully symbolic, representing state sets and transition systems by multi-terminal binary decision diagrams (MTBDD). In 2006, a symbolic model checking algorithm was developed for CASPA, verifying quantitative requirements formulated by means of the stochastic temporal logic SPDL [4]. The present note describes two recent extensions of CASPA: Firstly, for better usability, a GUI for graphical model specification has been built. Secondly, the CASPA input language has been extended by immediate actions, a feature that had often been missed, e.g. when using CASPA in a tool chain with the tool OpenSESAME [6] for the evaluation of high-availability systems.

II. Immediate transitions

Up to now, CASPA supported a purely Markovian process algebra. Immediate transitions had to be approximated by Markovian transitions with very large rates. From the numerical point of view, this can lead to ill-conditioned matrices, as the rates within the matrix differ by some orders of magnitude. Therefore the CASPA input language has been extended by immediate transitions. In connection with an elimination algorithm, this concept leads to considerably smaller state spaces than the mentioned Markovian approximation.

A. Modelling language

The CASPA modelling language is a stochastic process algebra. Fig. 1 gives a small running example of the CASPA language, including the specification of some immediate transitions. In lines (2)-(4) of the example, some constants are defined. Line (2) sets the integer constant `max` to 12, to be used within process parameters or guards. A constant `alpha` is specified in line (3), that can be used as rate of Markovian transitions. Similarly, the weight `small` defined in line (4) can be used for immediate transitions. Line (6) defines a System consisting of the processes `P(max)` and `Q` composed in parallel: The processes have to synchronise by the actions `up` or `down`. In the composition of the processes the actions `up` and `down` are hidden, so no further processes can synchronise with `System` by means of `up` and `down`. The constant `max` is the initial value of the parameterised Process `P`. The behaviour of `P` is defined in lines (7)-(9), where `P(n[max])` defines the parameter `n` and sets its range to $0 \leq \texttt{n} \leq \texttt{max}$. The term `[n<max,n>0]` defines a guard

```
(1)   /******** constants ********/
(2)   int max      = 12;
(3)   rate alpha   = 5;
(4)   weight small = 2;
(5)   /******** system specification ********/
(6)   System := hide up,down in (P(max) |[up, down]| Q)
(7a)  P(n[max]) := [n<max,n>0] -> (a,alpha); ( (* up,small *); P(n+1)
(7b)                                        + (* down,n*10 *);P(n-1) )
(8)                [n=0]       -> (bottom,4); P(n+1)
(9)                [n=max]     -> (top,3); P(n-1)
(10)  Q  := (pre,10); ( ((* up,1 *); Q1) + ((* down,1 *); Q2) )
(11)  Q1 := (back,40);Q
(12)  Q2 := (again,10);(* down,1 *);Q
```

Fig. 1. Textual specification example

for the subsequent transition. By `(a,alpha)` the Markovian transition with rate `alpha` and action label `a` is defined. It is followed by the choice (+) between two alternative branches `up` and `down`, where e.g. in `(*up,small*); P(n+1)` the immediate action `up` with its corresponding weight `small` is succeeded by process `P` with parameter changed to `n+1`. In a similar way the remaining behaviour of `P` is defined in lines (8) and (9). Process `Q`, which has no process parameter, and its subprocesses `Q1` and `Q2` are specified by lines (10)-(12).

B. Weighted approach

CASPA parses the input language and directly generates an MTBDD representation of the underlying weighted ESLTS (extended stochastic labelled transition system). In the following, the term ESLTS will be used as a synonym for the MTBDD encoding the Markovian and immediate transitions.

The synchronisation semantics are the same as for TIPP [3]. When two processes synchronise via some immediate action with label `l`, the resulting weight of the synchronised step is the product of the weights of the synchronised transitions. Similar to the synchronisation of Markovian transitions, `l` transitions without a synchronisation partner are disabled in the combined transition system. The weights of all other (unsynchronised) transitions remain the same. When the entire closed model is built, weights are transformed into probabilities. For every state s_i with $\mid J \mid$ emanating immediate transitions with weights w_{ij}, $j \in J$, the probabilities are calculated as $p_{ij} = \frac{w_{ij}}{\sum_{j \in J} w_{ij}}$. It should be mentioned that the resulting probabilities depend not only on the ratios of weights within one submodel but also on their absolute values.

C. Elimination of immediate transitions

Once the weighted ESLTS has been transformed to an ESLTS with probabilities, the immediate transitions may be

eliminated. The remaining purely Markovian transition system can then be passed to standard numerical solvers for the analysis phase.

The elimination process is performed by a semi-symbolic elimination algorithm. Semi-symbolic in this context means that every vanishing state is encoded symbolically. The transitions to be changed are then extracted and redirected symbolically. By processing one vanishing state after another, this algorithm is able to eliminate also loops of immediate transitions by rescaling them with a geometric series argument.

III. Graphical Model Specification

In order to provide CASPA to a broader spectrum of users, a GUI for graphical model specification has been implemented [2]. It has been developed in a model-driven style using Eclipse Graphical Modeling Framework (GMF) [1] to stay abreast of changes due to the constant evolvement of CASPA. A model can be specified graphically in a two-layered approach: the system layer and the process layer. On **system layer** the concurrent processes can be defined and composed by synchronisation nodes and hiding nodes in a directed acyclic graph with the system node as its root (see Fig. 2). By

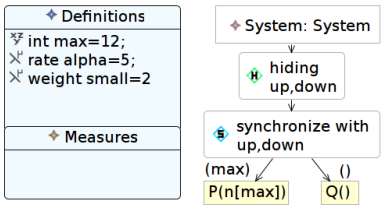

Fig. 2. Graphical Specification on System Layer

selecting a process node at system layer, a window is opened representing the **process layer** for specifying a sequential process similar to a labelled transition system or automaton. States and transitions can be defined, the latter with weights (immediate transitions) or rates (Markovian transitions). Two consecutive transitions can be separated using unnamed states. Absorbing states represent a "behaviour" that is either the same process (e.g. with modified parameter values) or a stop state. In Fig. 3, the two sequential processes P and Q from Sec. II are shown.

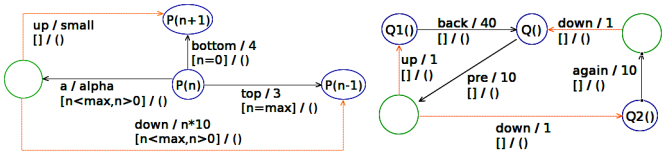

Fig. 3. Graphical Specification on Process Layer

Assuming we selected `P(n[max])` in Fig. 2, a window would appear with the left-hand side graph of Fig. 3. Here we can see the transition system of process `P`. From state `P(n)` the Markovian transition `a` with rate `alpha` leads to an unnamed state, from where there is a choice between the immediate transitions `up` (with weight `small` leading to `P(n+1)`) and `down` (with weight `n·10` and guard `[n<max,n>0]` leading to `P(n-1)`). The Markovian transition `top` with rate `3` and guard `[n=M]` also disembogues in `P(n-1)` and the Markovian transition `bottom` with rate `4` leads to state `P(n+1)`.

A complete syntax check and partial semantic validation on the model can be performed. The GUI also provides its own XML-based format to preserve all layout information. An import mechanism has been implemented to load existing textual model specifications. As the CASPA core component does not implement the new XML-based format as a second input language, an export mechanism outputs the model in CASPA textual language syntax. Furthermore, a panel has been created to ease and support the parameter settings and options for analysing a model: Parameters can be set by checkboxes or value fields and the analysis methods can be chosen from a menu. The output of CASPA is redirected to an assistant window that shows the analysis results in a text field and occurring error informations in a separate list box.

IV. Conclusion

In this note we have presented two new features for CASPA: An extension of the modelling language to handle immediate transitions and a GUI for graphical model specification as a front end to the CASPA core application. Future work on the tool is to implement an improved hybrid elimination algorithm that consists of a purely symbolic preprocessing step as proposed in [5] and a semi-symbolic postprocessing step that only has to handle the *exceptional* cases, i.e. immediate loops. Also, a debugging routine is being developed that is capable of calculating paths to deadlock states.

Acknowledgment

The authors would like to thank Jens Bachmann who implemented the CASPA GUI front end as part of his diploma thesis.

References

[1] Eclipse Graphical Modeling Framework, http://www.eclipse.org/modeling/gmf/.

[2] Bachmann, J. Entwurf und Implementierung eines graphischen Modelleditors und einer Benutzerschnittstelle für das Werkzeug CASPA. Diploma thesis, Lehrstuhl für Entwurf von Rechen- und Kommunikationssystemen an der Universität der Bundeswehr München, August 2007.

[3] Hermanns, H., and Rettelbach, M. Syntax, Semantics, Equivalences, and Axioms for MTIPP. *Proc. of PAPM'94, Arbeitsberichte des IMMD 27*, 4 (1994), 71–87.

[4] Kuntz, M. *Symbolic Semantics and Verification of Stochastic Process Algebras*. Dissertation, Lehrstuhl 7 des Instituts für Informatik der FAU Erlangen, 2006.

[5] Siegle, M. *Behaviour analysis of communication systems: Compositional modelling, compact representation and analysis of performability properties*. Shaker Verlag, Aachen, 2002.

[6] Walter, M., Siegle, M., and Bode, A. OpenSESAME: The Simple but Extensive, Structured Availability Modeling Environment. *Reliability Engineering and System Safety 93*, 6 (2007), 857–873.

[7] Werner, E. Leistungsbewertung mit Multi-terminalen Binären Entscheidungsdiagrammen. Diploma thesis, Lehrstuhl 7 des Instituts für Informatik der FAU Erlangen, June 2003.