Tampere Verification Tool

Heikki Virtanen, Henri Hansen, Antti Valmari, Juha Nieminen, and Timo Erkkilä

Tampere University of Technology, Institute of Software Systems
PO Box 553, FIN-33101 Tampere, FINLAND
hansen@cs.tut.fi

Abstract. Tampere Verification Tool (TVT) is a collection of programs for automated verification of concurrent and reactive systems. TVT has its roots in process algebras and explicit state space exploration, but in addition to actions, our formalism allows use of state-based information in the form of truth-valued state propositions. Furthermore, it contains three types of state proposition-like notions to support on-the-fly verification, and one state proposition to exploit partially defined processes. TVT supports compositional state space construction, stubborn sets and visual verification.

1 Introduction

The story of Tampere Verification Tool (TVT) started at the beginning of 1990's, when CFFD semantics was first introduced [10]. CFFD describes an abstraction of the behaviour of a system. It is the weakest congruence relation with respect to the composition operators of process algebras that preserves both stuttering-insensitive properties specified in linear temporal logic and deadlocks [4]. Based on CFFD, the "Advanced Reachability Analysis" (ARA) tool [9] was developed. ARA supports LOTOS as the modelling language, compositional construction of a system, visual verification, and CFFD equivalence comparison. ARA was used in the industry even a decade later [7].

Eventually, ARA became difficult to maintain, LOTOS proved ill-suited for verification applications [6], and many new verification ideas emerged. When Nokia offered us a contract for developing a new verification tool in 1999, we started the development of TVT. TVT was intended to be used both in Nokia Research Center and as a platform for developing and testing new verification ideas in Tampere University of Technology. TVT has been made freely available for academic use under the Nokia Open Source licence and can be downloaded from [8].

2 Key Features

TVT is a collection of non-interactive[1] command-line programs. The programs are used for manipulating and analysing behaviours of systems. The formalism

[1] The only exception is the visualisation tool which has a graphical user interface.

used as a model of the behaviour of a process is a *labelled state transition system* or an *LSTS*.

An LSTS is a labelled transition system (LTS), where the states can be labelled with truth-valued propositions. The actions of an LSTS are used to talk about how the components in a system interact with each other and the environment. In other words, the LTS-part of an LSTS describes the behaviour of the system. The propositions have been added to exploit the fact that in state-based models it is easier to express properties that depend on the global state of the system [1].

Perhaps the most important operation in TVT is the parallel composition of LSTSs. It implements a flexible parallel composition operator [6] combining parallel composition, multiple renaming, hiding and restriction. The parallel composition is controlled by a set of synchronisation rules given in a file. The file describes how actions of processes synchronise and how values of state propositions of the result are evaluated from the values of state propositions of the component processes [1]. During the construction of the state space, the program can do on-the-fly verification as discussed in [2] and in other ways, and reduction using the stubborn set method [12]. Partially defined processes, with cut states marking the pruning points [5] can also be used to further reduce the state space or to avoid the modelling of uninteresting parts of a system.

Other programs of the tool include reduction algorithms and conversions between different representations of LSTSs. Currently the tool supports reductions that preserve CFFD-semantics and strong bisimulation, but it is possible to add support for other semantics as well. The LSTS file format has been designed with that in mind.

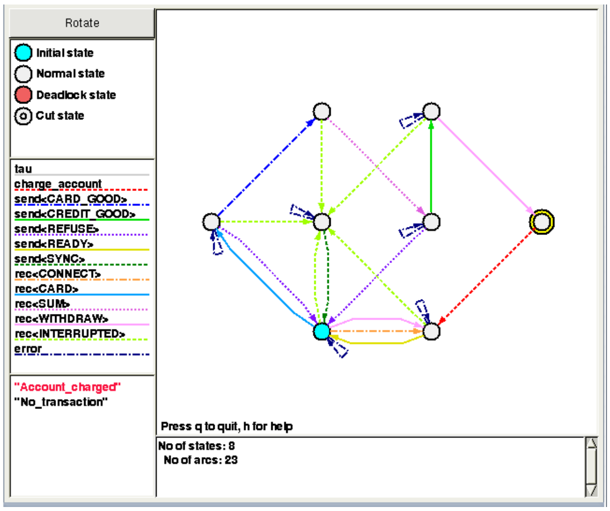
There are programs for visualisation and comparisons of LSTS representations of systems. The theory of visual verification has been explored in [13,11].

Essentially, TVT is a framework for development of tools and methods for explicit state space exploration, with emphasis on process algebra. Real-time or hybrid methods are not included.

3 Modelling Issues

To support compositionality in full scale, TVT has two modelling languages: one for describing LSTSs and another for describing communication between LSTSs. The languages used in modelling are close to the structure of LSTSs and synchronisation rules. The only significant difference is the possibility to use local variables when defining processes. The compiler "unfolds" the variables by duplicating states and instantiating action names with data values. This resembles a lot the unfolding of a coloured Petri net into an ordinary Petri net [3].

In addition to the compositional bottom-up construction of a state space, the pre-congruence property of CFFD semantics can be used to reduce the state space even further. One can replace any component with a more deterministic one without invalidating the correctness of a correct system. In the verification phase we may use a small specification process instead of an actual component if we know that the component will be an implementation of that specification.



```
Press q to quit, h for help
No of states: 8
 No of arcs: 23
```

Fig. 1. Visualisation of a component behaviour

As an example of the use of TVT, consider a system consisting of a bankteller, a bank, and the communications links between them. In Figure 1 we see one component of this system, the bank. This component models the behaviour of the computer system in a bank that communicates with a bankteller machine. The actions "rec" and "send" have one parameter. In the figure, we have an explicit representation, where the values of the parameter have already been unfolded. One proposition ("Account_charged") has been selected and the state in which it is true is highlighted with a double circle.

Using the compositional approach inherent in TVT, the state space of the system that contains about 11 million states and 67 million transitions can be constructed and reduced to contain only the behaviour of the system visible to the user at the bankteller in less than a minute. The result is shown in Figure 2, drawn using the TVT visualisator.

Each program in TVT implements just one operation. This makes it possible to combine operations in various orderings, perhaps reducing intermediate results to avoid state explosion. Other programs like `make` may be used to automate the composition and include suitable reductions for different purposes, e.g., one for visualisation and another one for producing a component for use as a part of a bigger system.

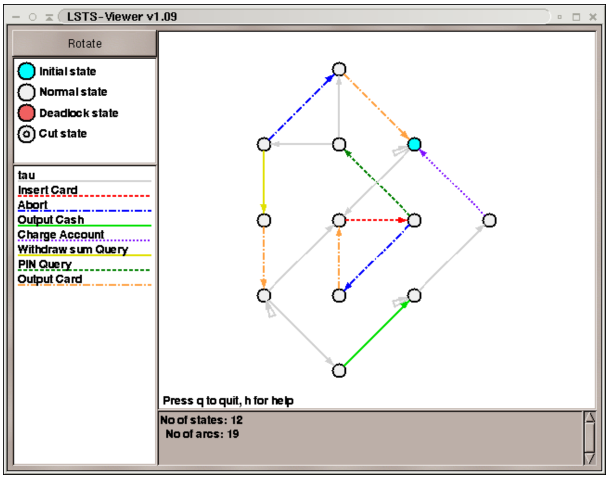


Fig. 2. Visualisation of the whole system behaviour

4 Summary

Tampere Verification Tool (TVT) is a framework for classical verification based on exhaustive and reduced state space exploration methods, on-the-fly verification, and visualisation. A key feature of TVT is compositional bottom-up construction of the state space which is an effective method to combat the state explosion problem.

The formalism used in TVT, labelled state transition system (LSTS), is derived from a pure action-based formalism, labelled transition system (LTS), by adding support for state propositions. These state propositions are used for the modelling of properties more naturally associated with states than with actions and as markers for error conditions in on-the-fly verification.

As such TVT can be used as a computer-aided software engineering tool when designing concurrent systems. In addition, TVT is a framework for implementation of software for new verification methods.

Acknowledgements. The development of TVT was supported by Nokia Research Center, The Technology Development Centre of Finland and Emil Aaltonen Foundation.

References

1. H. Hansen, H. Virtanen, and A. Valmari. Merging state-based and action-based verification. In *Proceedings of the third international conference on Application of Concurrency to System Design (ACSD 2003)*, pp. 150–156, IEEE, 2003.

2. J. Helovuo and A. Valmari. Checking for CFFD-preorder with tester processes. In *Tools and Algorithms for the Construction and Analysis of Systems, 6th International Conference, TACAS 2000*, number 1785 in Lecture Notes in Computer Science, pp. 283–298, Berlin, March 27–31 2000. Springer-Verlag.
3. K. Jensen *Coloured Petri Nets Volume 1: Basic Concepts, Analysis Methods and Practical Use.* EATCS Monographs on Theoretical Computer Science, 234 pages. Springer-Verlag, 1992.
4. R. Kaivola and A. Valmari. The weakest compositional semantic equivalence preserving nexttime-less linear temporal logic. In *Proceedings of CONCUR '92, Third International Conference on Concurrency Theory*, number 630 in Lecture Notes in Computer Science, pp. 207–221. Springer-Verlag, 1992.
5. A. Kangas and A. Valmari. Verification with the undefined: A new look. In Thomas Arts and Wan Fokkink, editors, *Proceedings of Eighth International Workshop on Formal Methods for Industrial Critical Systems (FMICS'03)*, volume 80 of *ENTCS.*
6. K. Karsisto. *A New Parallel Composition Operator for Verification Tools.* PhD thesis, Tampere University of Technology, 2003.
7. S. Leppänen and M. Luukkainen. Compositional verification of a third generation mobile communication protocol. In Ten-Hwang Lai, editor, *Proceedings of the 1st Workshop on Distributed System Validation and Verification*, pp. E118–E125, 2000.
8. Tvt-project home page. http://www.cs.tut.fi/ohj/VARG/TVT/
9. A. Valmari, J. Kemppainen, M. Clegg, and M. Levanto. Putting advanced reachability analysis techniques together: the ARA tool. In *Proceedings of Formal Methods Europe '93: Industrial-Strength Formal Methods*, number 670 in Lecture Notes in Computer Science, pp. 597–616. Springer-Verlag, 1993.
10. A. Valmari and M. Tienari. An improved failures equivalence for finite-state systems with a reduction algorithm. In *Proceedings of Protocol Specification, Testing and Verification XI*, pp. 3–18. North Holland, 1991.
11. A. Valmari, H. Virtanen, and A. Puhakka. Context-sensitive visibility. In Cleaveland R. and Garavel H., editors, *FMICS'02 7th International ERCIM Workshop on Formal Methods for Industrial Critical Systems*, volume 66 of *ENCTS*, pp. 201–217.
12. A. Valmari. Stubborn set methods for process algebras. In *Proceedings of POMIV'96, Workshop on Partial Order Methods in Verification*, volume 29 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 213–231. American Mathematical Society, July 1996.
13. A. Valmari and M. Setälä. Visual verification of safety and liveness. In *Proceedings of Formal Methods Europe '96: Industrial Benefit and Advances in Formal Methods*, number 1051 in Lecture Notes in Computer Science, pp. 228–247. Springer-Verlag, 1996.