UPPAAL in 1995*

Johan Bengtsson† Kim G. Larsen‡
Fredrik Larsson† Paul Pettersson† Wang Yi†

ABSTRACT UPPAAL[1] is a tool suite for automatic verification of safety and bounded liveness properties of real-time systems modeled as networks of timed automata [12, 9, 4], developed during the past two years. In this paper, we summarize the main features of UPPAAL in particular its various extensions developed in 1995 as well as applications to various case-studies, review and provide pointers to the theoretical foundation.

1 Introduction

UPPAAL is a tool suite for automatic verification of safety and bounded liveness properties of real-time systems modeled as networks of timed automata extended with data variables [12, 9, 4], developed during the past two years. In this paper, we summarize the features of UPPAAL in particular the various extensions developed in 1995 as well as applications to various case-studies, review and provide pointers to the theoretical foundation.

In developing an automatic verification tool, there are two main issues to be considered: a *user interface* which should be easy to use and a *model-checker* which should be efficient. UPPAAL consists of a graphical user interface based on Autograph, that allows system descriptions to be defined graphically and a model-checker that combines *on-the-fly* verification with a *symbolic* technique reducing the verification problem to that of solving simple *constraint systems* [12, 9]. The current version of UPPAAL is able to check for invariant and reachability properties, in particular whether certain combinations of control-nodes of timed automata and constrains on variables are reachable from an initial configuration. Bounded liveness properties can be checked by reasoning about the system in the context of a testing automata. In order to facilitate debugging, the model-checker will report a *diagnostic trace* in case the verification procedure terminates with a negative answer [10].

*This work has been supported by the European Communities (under CONCUR2 and REACT), NUTEK (Swedish Board for Technical Development) and TFR (Swedish Technical Research Council)

†Department of Computer Systems, Uppsala University, SWEDEN. E-mail: {johanb, fredrikl, paupet, yi}@docs.uu.se

‡BRICS (Basic Research in Computer Science, Center of the Danish National Research Foundation), Aalborg University, DENMARK. E-mail: kgl@iesd.auc.dk

[1]The current version of UPPAAL is available on the World Wide Web via the UPPAAL home page `http://www.docs.uu.se/docs/rtmv/uppaal`.

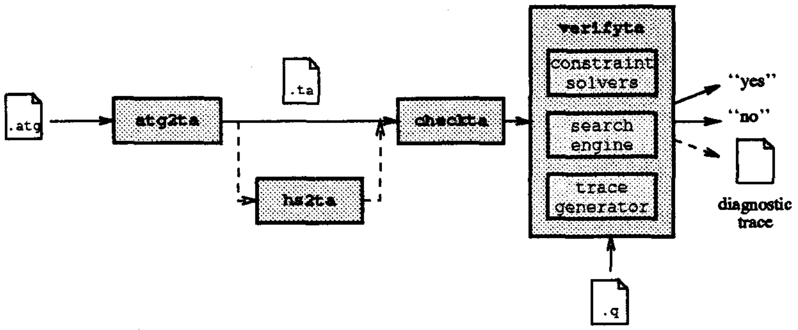

FIGURE 1. Overview of UPPAAL

The current version of UPPAAL is implemented in C++. An overview of UPPAAL is shown in Figure 1.

atg2ta A compiler from the graphical representation (`.atg`) of a network of timed automata, to the textual representation in UPPAAL (`.ta`).

hs2ta A filter that automatically transforms linear hybrid automata where the speed of clocks is given by an interval into timed automata [11], thus extending the class of systems that can be analyzed by UPPAAL.

checkta Given a textual representation (in the `.ta`-format) of a network of timed automata, **checkta** performs a number of simple but in practice useful syntactical checks.

verifyta A model-checker that combines *on-the-fly* verification with constraint solving techniques [12, 9].

2 Extensions in 1995

The UPPAAL model for real-time systems is networks of timed automata with data variables. For detailed descriptions of the model, we refer to [9, 4]. The model-checking algorithms implemented in UPPAAL are developed in [12, 9]. During the past year, we have applied UPPAAL to a number of case-studies reviewed in next section. To meet requirements arising from the case studies, the UPPAAL model and model-checker have been further extended with new features. In the following, we summarize the new features of UPPAAL developed during 1995:

Committed Locations. UPPAAL adopts hand-shaking synchronization between components in a network. The very recent case-study on the verification of Philips Audio Control Protocol with bus-collisions shows that we need to further extend the UPPAAL model with *committed locations* to model behaviors such as atomic broadcasting in real-time systems. The notion of committed locations is introduced in [3]. Our experiences with UPPAAL show that the notion of committed locations implemented in UPPAAL is not only useful in modeling real-time systems but also yields significant reductions in time- and space-usages in verifying such systems.

Urgent Actions. In order to model progress properties UPPAAL uses a notion of *maximal delay* that requires discrete transitions to be taken within a certain time bound. However, in some examples, e.g. the Manufacturing Plant [6], synchronization on certain channels should happen immediately. For this reason the UPPAAL model was extended with *urgent channels*, on which processes should synchronize whenever possible [4]. The notion of urgent channels (also known as *urgent actions* in the literature) has been implemented in both HYTECH and KRONOS.

Diagnostic Traces. Ideally, a model-checker should be able to report diagnostic information whenever the verification of a particular real-time system fails. UPPAAL reports such information by generating a *diagnostic trace* from the initial state to a state violating the property. The usefulness of this kind of information was shown during the debugging of an early version of Philips Audio-Control Protocol [10].

3 Case-Studies

UPPAAL was applied to a number of case-studies and benchmark examples during 1995, including: several versions of Fischers Protocol [1], two version of Philips Audio-Control Protocol [5, 10, 3], a Steam Generator [2], a Train Gate Controller [7], a Manufacturing Plant [6], a Mine-Pump Controller [8] and a Water Tank [11].

In terms of complexity, Philips Audio-Control Protocol with bus-collision is the most serious case-study where UPPAAL is applied so far. The protocol is developed by Philips to exchange information between components (e.g. amplifier, tuner, CD-player, etc.) in one of their high-end audio sets. In [10] Philips Audio-Control Protocol without bus-collision [5] was verified using UPPAAL. In the verification of the protocol, the *diagnostic model-checking* feature of UPPAAL was used for detecting and correcting several errors in an early description of the protocol[2]. Recently a version of Philips Audio-Control Protocol with *two* senders and with *bus-collision handling* was verified using UPPAAL. The result is reported in [3]. This case study is comprehensive compared with previous verification efforts of real-time and hybrid systems described in the literature. During this case-study UPPAAL was extended with *committed locations*, allowing efficient modelling of broadcast communication[3].

[2]UPPAAL installed on a Sparc Station 10 running SunOS 4.1.4, with 32 MB of primary memory verifies that the received bit stream is guaranteed to be identical to the sent bit stream in 3.6 seconds.

[3]The verification of Philips Audio-Protocol with Bus Collision was carried out using an extended version of UPPAAL installed on a SGI ONYX machine.

4 Future Extensions

In this paper we have summarized the main features of UPPAAL in particular its recent extensions as well as applications to various case-studies.

Our experience with UPPAAL during the past years shows that in verifying real-time systems, space-consuming is a more serious problem than time-consuming as a verification process must store not only control-nodes searched but also possible clock values associated with the control-nodes. We have introduced the notion of committed locations which is useful in modeling real-time behaviors, and also yields significant reduction in memory-usage. As future work, we shall further develop techniques for minimizing memory-usage. Future work also includes extending the current model-checker of UPPAAL to check bounded liveness properties of [10] and implementing the newly developed compositional model-checking technique of [9].

5 REFERENCES

[1] Martin Abadi and Leslie Lamport. An Old-Fashioned Recipe for Real Time. *Lecture Notes in Computer Science*, 600, 1993.

[2] J.-R. Abrial. Steam-boiler control specification problem. June 1995. International Seminar on Methods for Semantics and Specification.

[3] Johan Bengtsson, David Griffioen, Kåre Kristoffersen, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Verification of an Audio Protocol with Bus Collision Using UPPAAL. Submitted for publication, 1996.

[4] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL—a Tool Suite for Automatic Verification of Real–Time Systems. In *Proc. of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems*, Lecture Notes in Computer Science, October 1995.

[5] D. Bosscher, I. Polak, and F. Vaandrager. Verification of an Audio-Control Protocol. In *Proc. of FTRTFT'94*, volume 863 of *Lecture Notes in Computer Science*, 1994.

[6] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *Proc. of the 16th IEEE Real-Time Systems Symposium*, pages 66–75, December 1995.

[7] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. A Users Guide to HYTECH. Technical report, Department of Computer Science, Cornell University, 1995.

[8] Mathai Joseph, Alan Burns, Andy Welling, Krithi Ramamritham, Jozef Hooman, Steve Schneider, Zhiming Liu, and Henk Schepers. *Real-time Systems Specification, Verification and Analysis*. Prentice Hall, 1996.

[9] Kim G. Larsen, Paul Pettersson, and Wang Yi. Compositional and Symbolic Model-Checking of Real-Time Systems. In *Proc. of the 16th IEEE Real-Time Systems Symposium*, pages 76–87, December 1995.

[10] Kim G. Larsen, Paul Pettersson, and Wang Yi. Diagnostic Model-Checking for Real-Time Systems. In *Proc. of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems*, Lecture Notes in Computer Science, October 1995.

[11] A. Olivero, J. Sifakis, and S. Yovine. Using Abstractions for the Verification of Linear Hybrids Systems. In *Proc. of CAV'94*, volume 818 of *Lecture Notes in Computer Science*, 1994.

[12] Wang Yi, Paul Pettersson, and Mats Daniels. Automatic Verification of Real-Time Communicating Systems By Constraint-Solving. In *Proc. of the 7th International Conference on Formal Description Techniques*, 1994.