Tool Environment for Validation and Verification of Real-Time Systems

E-mail: uppaal@docs.uu.se
http://www.uppaal.com/

UPPAAL is an integrated tool environment for modeling, simulation and verification of real-time systems, developed jointly by Uppsala University, Sweden and Aalborg University, Denmark. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels or shared variables [14, 9]. Typical application areas include real-time controllers and communication protocols in particular, those where timing aspects are critical.

UPPAAL consists of three main parts: a description language, a simulator and a model-checker. The description language is a non-deterministic guarded command language with simple data types (e.g. bounded integers, arrays, etc.). It serves as a modeling or design language to describe system behavior as networks of automata extended with clock and data variables. The simulator is a validation tool which enables examination of *possible* dynamic executions of a system during early design (or modeling) stages and thus provides an inexpensive mean of fault detection prior to verification by the model-checker which covers the *exhaustive* dynamic behavior of the system. The model-checker is to check invariant and liveness properties by exploring the state-space of a system, i.e. reachability analysis in terms of symbolic states represented by constraints.

The two main design criteria for UPPAAL have been *efficiency* and *ease of usage*. The application of *on-the-fly* searching technique has been crucial to the efficiency of the UPPAAL model-checker. Another important key to efficiency is the application of a *symbolic* technique that reduces verification problems to that of efficient manipulation and solving of constraints [14, 7, 11, 10]. To facilitate modeling and debugging, the UPPAAL model-checker may automatically generate a *diagnostic trace* that explains why a property is (or is not) satisfied by a system description. The diagnostic traces generated by the model-checker can be loaded automatically to the simulator, which may be used for visualization and investigation of the trace.

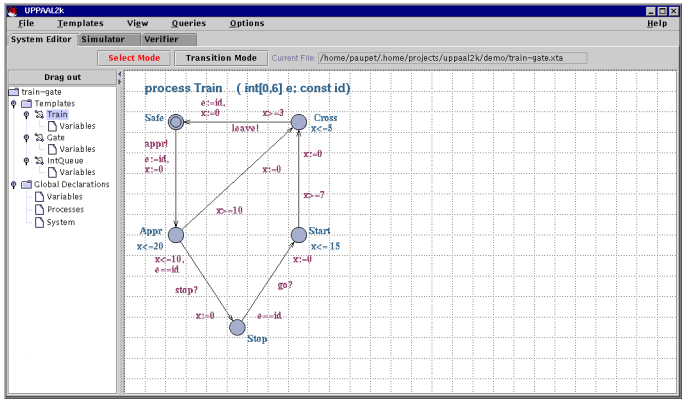

Figure 1: The system editor of UPPAAL.

Since its first release in 1995, UPPAAL has been applied in a number of case studies (see next section for a short summary). To meet requirements arising from the case studies, the tool has been extended with various features. The current version of UPPAAL was released in September 1999. It is implemented in Java and C++, and is available for Linux, SunOS and Windows 95/98/NT. The features of UPPAAL include:

- A graphical system editor allowing graphical descriptions of systems.
- A graphical simulator which provides graphical visualization and recording of the possible dynamic behaviors of a system description, i.e. sequences of symbolic states of the system. It may also be used to visualize traces generated by the model-checker.
- A requirement specification editor that also constitutes a graphical user interface to the verifier of UPPAAL.
- A model-checker for automatic verification of safety and bounded-liveness properties by reachability analysis of the symbolic state-space.
- Generation of diagnostic traces in case verification of a particular real-time system fails. The diagnostic traces may be automatically loaded and graphically visualized using the simulator.

The system editor, the simulator, and the requirement specification editor of UPPAAL are shown in Figure 1, 2 and 3, respectively.

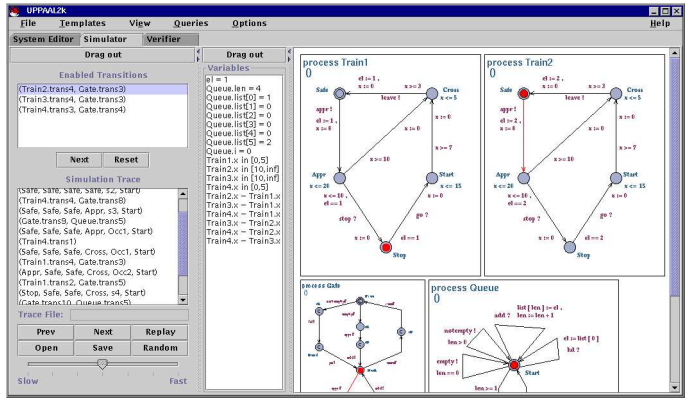

Figure 2: The simulator of UPPAAL.

Applications

UPPAAL has been applied in a number of (industrial) case-studies. In this section we briefly review a selection of them.

Audio/Video Protocol: This is an audio control protocol highly dependent on real-time. The protocol, which is developed by Bang & Olufsen, is to transmit messages between audio/video components over a single bus. Though it was known to be faulty, the error was not found using conventional testing methods. In [5], UPPAAL is applied to automatically produce an error-trace that reveals the error. Furthermore, a correction is suggested and proved using UPPAAL.

Bounded Retransmission Protocol: The protocol was proposed and studied at *COST 247, International Workshop on Applied Formal Methods in System Design*. It is based on the alternating bit protocol over a lossy communication

channel, but allows for a bounded number of retransmissions. In [4], it is reported that a number of properties of the protocol have been automatically verified with UPPAAL. In particular, it is shown that the correctness of the protocol is dependent on correctly chosen time-out values.

Collision Avoidance Protocol: This protocol is implemented on top of an Ethernet-like medium such as the CSMA/CD protocol. It is to ensure an upper bound on the communication delay between the network nodes. In [6], the protocol is designed and proved correct using UPPAAL. The two main established properties show that the protocol is collision-free, and it does ensure an upper bound on the user-to-user communication delay (assuming a perfect medium).

Gearbox Controller: In [12], UPPAAL is applied in an industrial case-study, to the design and analysis of a prototype gearbox controller for vehicles by Mecel AB (a Swedish company developing control systems for vehicle industries). The gearbox controller is a component in the real-time system that controls a modern car.

In the design of the controller, the simulation tool of UPPAAL is applied to validate the system behavior. The correctness of the gear-box controller design is established by automatic verification of 46 properties derived from informal requirements specified by Mecel AB.

Philips Audio-Control Protocol: This protocol is developed and implemented by Philips to exchange control information between components in audio equipment using Manchester encoding. The correctness of the encoding relies on timing delays between signals. In [8] the protocol is modeled and verified using UPPAAL.

In [3], a version of the protocol extended with bus collision detection is analyzed using UPPAAL. The model in this case-study is significantly larger than the original version since several new components (and variables) are introduced, and existing components are modified to deal with bus collisions.

TDMA Protocol Start-Up Mechanism: In [13], a formal verification of the start-up algorithm of a TDMA (Time Division Multiple Access) protocol is reported. It is checked using UPPAAL that an ensemble of three communicating stations becomes synchronized and operational within a bounded time from an arbitrary initial state, given a clock-drift corresponding to $\pm 10^{-3}$. Furthermore, an upper time-bound for the start-up to complete was derived.

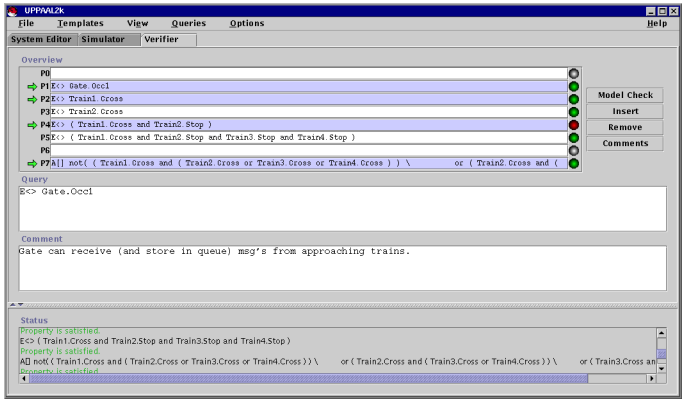

Figure 3: The requirement specification editor of UPPAAL.

Organisation

UPPAAL is developed in collaboration between the Department of Information Technology at Uppsala University (UPP) in Sweden and the Department of Computer Science at Aalborg University (AAL) in Denmark.

The people involved with development and application are Wang Yi (Professor, UPP), Kim G. Larsen (Professor, AAL), Gerd Behrmann (Ph.D., AAL), Paul Pettersson (Ph.D., UPP), Alexandre David (Post Doc, AAL), Emmanuel Fleury (Ph.D., AAL), Brian Nielsen (Ph.D., AAL), Arne Skou (Ph.D., AAL), John Håkansson (Ph.D. Student, UPP), Jacob Illum Rasmussen (Ph.D. Student, AAL), Pavel Krcál (Ph.D. Student, UPP), Ulrik Larsen (Ph.D. Student, AAL), Didier Lime (Post Doc., AAL), Marius Mikucionis (Ph.D. Student, AAL), and Leonid Mokrushin (Ph.D. Student, UPP).

Further Information

UPPAAL has a home page on World Wide Web, `http://-www.uppaal.com/`, containing pointers to the published material on UPPAAL and complete information for installation.

Detailed informal descriptions of UPPAAL can be found in the papers "Tutorial on UPPAAL" [2], "UPPAAL in a Nutshell" [9], and "UPPAAL - Now, Next, and Future" [1].

References

[1] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. UPPAAL - Now, Next, and Future. In F. Cassez, C. Jard, B. Rozoy, and M. Ryan, editors, *Modelling and Verification of Parallel Processes*, number 2067 in Lecture Notes in Computer Science, pages 100–125. Springer–Verlag, 2001.

[2] Gerd Behrmann, Alexandre David, , and Kim G. Larsen. A tutorial on UPPAAL. In *Proc. of 4*th *International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, number 3185 in Lecture Notes in Computer Science, 2004.

[3] Johan Bengtsson, W.O. David Griffioen, Kåre J. Kristoffersen, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Verification of an Audio Protocol with Bus Collision Using UPPAAL. In Rajeev Alur and Thomas A. Henzinger, editors, *Proc. of the 8th Int. Conf. on Computer Aided Verification*, number 1102 in Lecture Notes in Computer Science, pages 244–256. Springer–Verlag, July 1996.

[4] P.R. D'Argenio, J.-P. Katoen, T.C. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In *Proc. of the 3*rd *Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, number 1217 in Lecture Notes in Computer Science, pages 416–431. Springer–Verlag, April 1997.

[5] Klaus Havelund, Arne Skou, Kim G. Larsen, and Kristian Lund. Formal Modeling and Analysis of an Audio/Video Protocol: An Industrial Case Study Using UPPAAL. In *Proc. of the 18*th *IEEE Real-Time Systems Symposium*. IEEE Computer Society Press, December 1997.

[6] Henrik E. Jensen, Kim G. Larsen, and Arne Skou. Modelling and Analysis of a Collision Avoidance Protocol Using SPIN and UPPAAL. In *Proc. of 2*nd *Int. Workshop on the SPIN Verification System*, pages 1–20, August 1996.

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

[8] Kim G. Larsen, Paul Pettersson, and Wang Yi. Diagnostic Model-Checking for Real-Time Systems. In *Proc. of Workshop on Verification and Control of Hybrid Systems III*, number 1066 in Lecture Notes in Computer Science, pages 575–586. Springer–Verlag, October 1995.

[9] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, October 1997.

[10] Kim G. Larsen, Carsten Weise, Wang Yi, and Justin Pearson. Clock difference diagrams. *Nordic Journal of Computing*, 6(3):271–298, 1999.

[11] Fredrik Larsson, Kim G. Larsen, Paul Pettersson, and Wang Yi. Efficient Verification of Real-Time Systems: Compact Data Structures and State-Space Reduction. In *Proc. of the 18*th *IEEE Real-Time Systems Symposium*, pages 14–24. IEEE Computer Society Press, December 1997.

[12] Magnus Lindahl, Paul Pettersson, and Wang Yi. Formal Design and Analysis of a Gear-Box Controller. In *Proc. of the 4*th *Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, number 1384 in Lecture Notes in Computer Science, pages 281–297. Springer–Verlag, March 1998.

[13] Henrik Lönn and Paul Pettersson. Formal Verification of a TDMA Protocol Startup Mechanism. In *Proc. of the Pacific Rim Int. Symp. on Fault-Tolerant Systems*, pages 235–242, December 1997.

[14] Wang Yi, Paul Pettersson, and Mats Daniels. Automatic Verification of Real-Time Communicating Systems By Constraint-Solving. In Dieter Hogrefe and Stefan Leue, editors, *Proc. of the 7*th *Int. Conf. on Formal Description Techniques*, pages 223–238. North–Holland, 1994.