Model Checking and UPPAAL

Radek Mařík

Czech Technical University
Faculty of Electrical Engineering
Department of Telecommunication Engineering
Prague CZ

October 17, 2023

Outline

Outline

Guaranteeing Correct System Behavior [Cam10]

- The complexity of software systems increases,
- Errors lead to losses
  - financial,
  - on human lives.
- Typically, an issue with critical security systems
  - aircraft,
  - satellites,
  - medical devices.

Requirements [Cam10]

- Demonstration that the requirements are
  - correct,
  - complete,
  - accurate,
  - consistent,
  - testable.

Quality Assurance Methods [Cam10]

- **Testing and Simulation** provides only probabilistic security.
- **Runtime Verification** . . . A technique that combines formal verification with program runtime.
- **Formal verification** . . . a technique based on formal methods built with mathematically based languages that allow systems to be specified and verified.
  - **Specifications** . . . system requirements written in mathematical language.
  - **Verification** . . . a formal proof that the system meets the requirements.

Principles of Formal Verification [Cam10, Če09]

Inputs

- (mathematical) system model,
  - a formal model M,
- specification of system requirements,
  - a formula φ of certain temporal logic,

Verification

- Verification that the system meets the specifications.
  - deciding whether M is a model of the formula φ, i.e. $M \models \varphi$

Typology of Formal Verification [Cam10]

Techniques

- **Static analysis** . . . verify the behavior of the program without having to run it.
  - **Abstract static analysis** . . . based on *abstract interpretation* using approximation abstract representations to verify the approximate properties of complex systems
    - analysis of pointers in modern compilers.
  - **Model validation** . . . a full scan of available program states.
  - **Limited model validation** . . . complete traversal of available program states only to a certain depth.
- **Proof of sentences** . . . finding a proof of a property, where the system and its properties are expressed as formulas in some mathematical logic.

Solvability of Temporally Logical Formalisms [Hol06]

Model validation

- We ask if the system meets the required property.
- I.e. for a structure representing a system, it is necessary to determine whether it is a model of a given formula.
- usable for verification of existing programs.

Formula Fulfillment

- The problem of deciding whether there is a model of a given formula.
- usable for automatic program synthesis.

Outline

Model Checking [Cam10]

Principle

- building the final system model,
- check whether the required property is complied with the model,
- based on a full state space search.

Basic properties

- operations with huge search spaces,
- the answer is "yes" or "no", in the negative case the system provides
  - a counterexample, i.e. running a system that does not match a property.
- software system specification analysis.

Model Checking in Practice [Cam10]

Application

- hardware verification (circuits),
- protocol verification,
- software system specification analysis.

Model Checking Approaches [Cam10]

Temporal verification of models

- use of temporal logic (expression of time),
- systems modeled as transition systems with a finite number of states.

Automatic approach

- specifications and a model expressed as automata,
- both automata are compared
  - language inclusion,
  - refining ordering,
  - observational equivalence.

Advantages / Disadvantages of Model Checking [Cam10]

Advantages

- full automation,
- high speed,
- possibility to verify even partial specifications,
- produces counterexamples.

Disadvantages

- a state explosion problem,
  - binary decision diagrams (BDD),
  - tools are able to handle systems with $100 - 200$ state variables
  - it is possible to handle systems with 10^{120} states.

Model Checking Methods Extensions [Bie08]

Removing of finiteness

- continuous variables,
- continuous time,
- working with probability,
- parameterization of the size or number of components,
- replacement of finite automata with stack automata.

Temporal Logic [Bie08]

Investigation of sequential or temporal system behavior

- reactive, distributed or parallel systems,
- A. Pnueli was the first to point out this idea,

Verified properties

- **Security** . . . A property that specifies that a particular bug or catastrophic state is not reachable.
  - all reachable states satisfy a certain invariant.
- **Liveness** . . . something happens once,
- **Fairness** . . . ,

Outline

State Space [Če09]

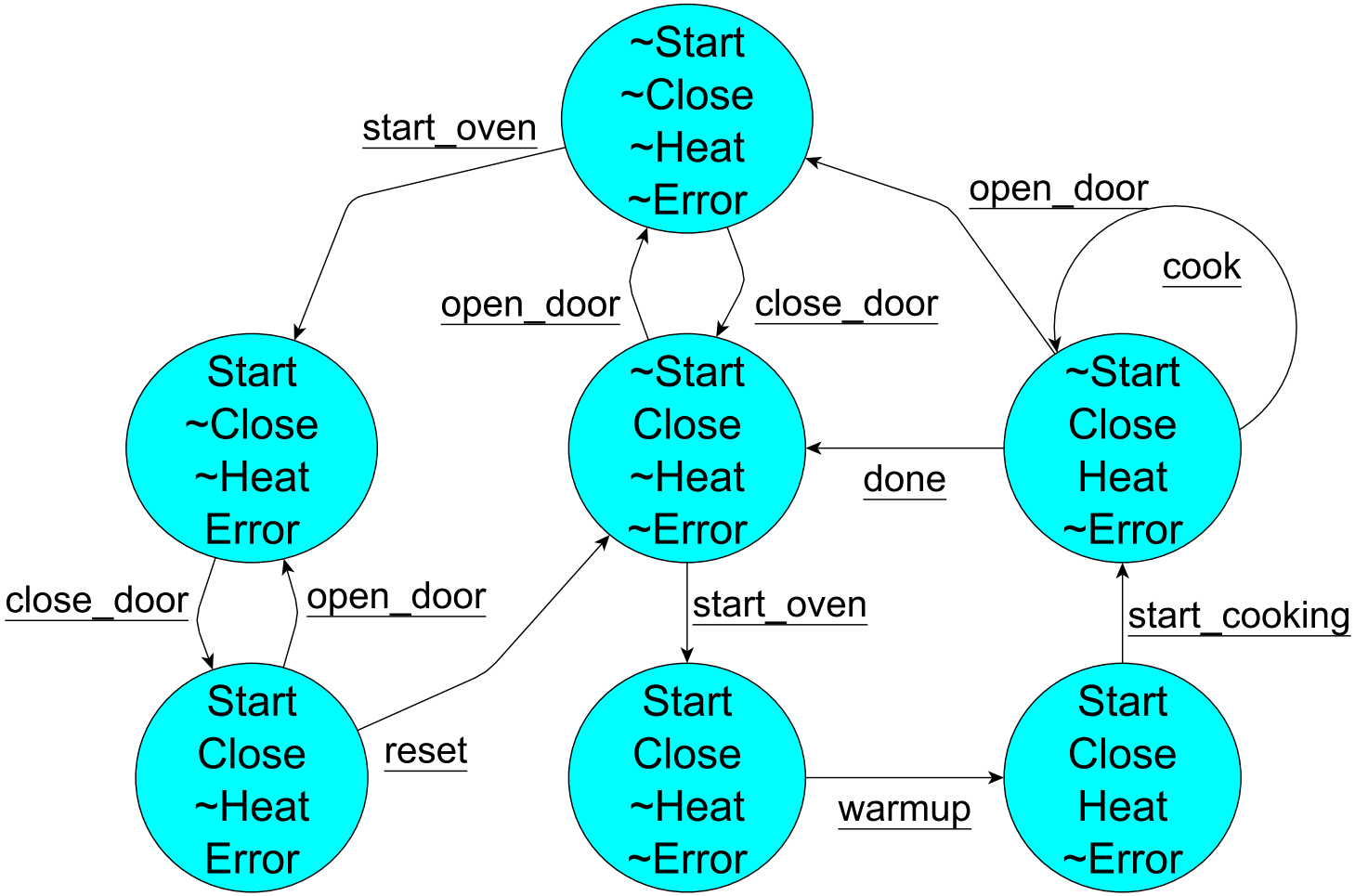
Applicable to finite state spaces only

- Only those model parameters that are specified can be verified.
- The state space can be formalized using atomic statements and Kripke's structure

Atomic statements

- a basic statement describing the system
  - expressions,
  - constants,
  - predicate symbols.
- Each atomic statement is algorithmically decidable based on a given state.
- Status . . . evaluation of all variables.

Kripke's Structure - Microwave Owen [?]



Kripke's Structure [Če09]

Kripke's structure is a type of nondeterministic finite state machine.

Kripke's structure

- A set of atomic propositions AP is given.
- Kripke's structure is a triple $(S, T, \mathcal{I})$, where
  - S is a finite set of states,
  - $T \subseteq S \times S$ is a transition relation,
  - $\mathcal{I} : S \to 2^{AP}$ is an interpretation of AP.

Extended Kripke's structure

- is a quadruple $(S, T, \mathcal{I}, s_0)$, where
  - $(S, T, \mathcal{I})$ is Kripke's structure,
  - s_0 is the initial state.

Kripke's transition system [Če09]

If we have a given set of Act actions executable by a program, we can extend Kripke's structures to indicate a transition.

Kripke's transition system

- is quintuple $(S, T, \mathcal{I}, s_0, L)$, where
  - $(S, T, \mathcal{I}, s_0)$ is an extended Kripke structure,
  - $L : T \to Act$ is a markup function.

Outline

Tool in a Nutshell [UPP10]

Tool integrating platforms

- for modeling,
- simulation,
- and verification,
- of real systems.

Development teams

- **Upp**sala University, Sweden,
- **Aal**borg University, Denmark.

Tool in a Nutshell [UPP10]

Tool integrating platforms

- for modeling,
- simulation,
- and verification,
- of real systems.

Development teams

- **Upp**sala University, Sweden,
- **Aal**borg University, Denmark.

System models [UPP10]

Model Properties

- a set of nondeterministic processes
- with the final control structure and
- real clock,
- communicating via channels or
- shared variables

Implementation [UPP10]

Main design criteria

- performance,
  - search engine *in flight*
  - symbolic techniques
- easy to use.
- diagnostic record
  - can be generated by a verifier and played by a simulator

Availability

- The first version in 1995
- The current version is 4.0.15
- graphical interfaces are implemented in Java
- verifier is implemented in C ++
- available for Linux, SunOS, MS Windows (95/98/NT/2000/XP/Vista/7/10)

Implementation [UPP10]

Main design criteria

- performance,
  - search engine *in flight*
  - symbolic techniques
- easy to use.
- diagnostic record
  - can be generated by a verifier and played by a simulator

Availability

- The first version in 1995
- The current version is 4.0.15
- graphical interfaces are implemented in Java
- verifier is implemented in C ++
- available for Linux, SunOS, MS Windows (95/98/NT/2000/XP/Vista/7/10)

Industrial Studies [UPP10]

Case Studies

- audio/video protocol
  - communication between audio/video components using a single bus
- bounded retransmission protocol,
- collision avoidance protocol
  - media based on Eternet
- car clutch controller,
- audio component control protocol (Philips)
- TDMA (Time Division Multiple Access) start-up mechanism protocol
  - synchronization of 3 communicating stations from any initial state.

Typical applications

- real-time controller,
- communication protocols.

Industrial Studies [UPP10]

Case Studies

- audio/video protocol
  - communication between audio/video components using a single bus
- bounded retransmission protocol,
- collision avoidance protocol
  - media based on Eternet
- car clutch controller,
- audio component control protocol (Philips)
- TDMA (Time Division Multiple Access) start-up mechanism protocol
  - synchronization of 3 communicating stations from any initial state.

Typical applications

- real-time controller,
- communication protocols.

Outline

System Components [UPP10]

Description language

- a language of nondeterministic conditional statements
- simple data types (bounded integers, arrays, etc.)
- network of automata with clocks and data variables.

Simulator

- investigation of possible dynamic runs of a system,
- detection of model defects before its verification,
- allows an analysis of run records leading to unwanted states.

Model Verifier

- examination of all possibilities of dynamic behavior of the model,
- check invariants and liveness by searching the state space,
- reachability of symbolic states represented by constraints.

System Components [UPP10]

Description language

- a language of nondeterministic conditional statements
- simple data types (bounded integers, arrays, etc.)
- network of automata with clocks and data variables.

Simulator

- investigation of possible dynamic runs of a system,
- detection of model defects before its verification,
- allows an analysis of run records leading to unwanted states.

Model Verifier

- examination of all possibilities of dynamic behavior of the model,
- check invariants and liveness by searching the state space,
- reachability of symbolic states represented by constraints.

System Components [UPP10]

Description language

- a language of nondeterministic conditional statements
- simple data types (bounded integers, arrays, etc.)
- network of automata with clocks and data variables.

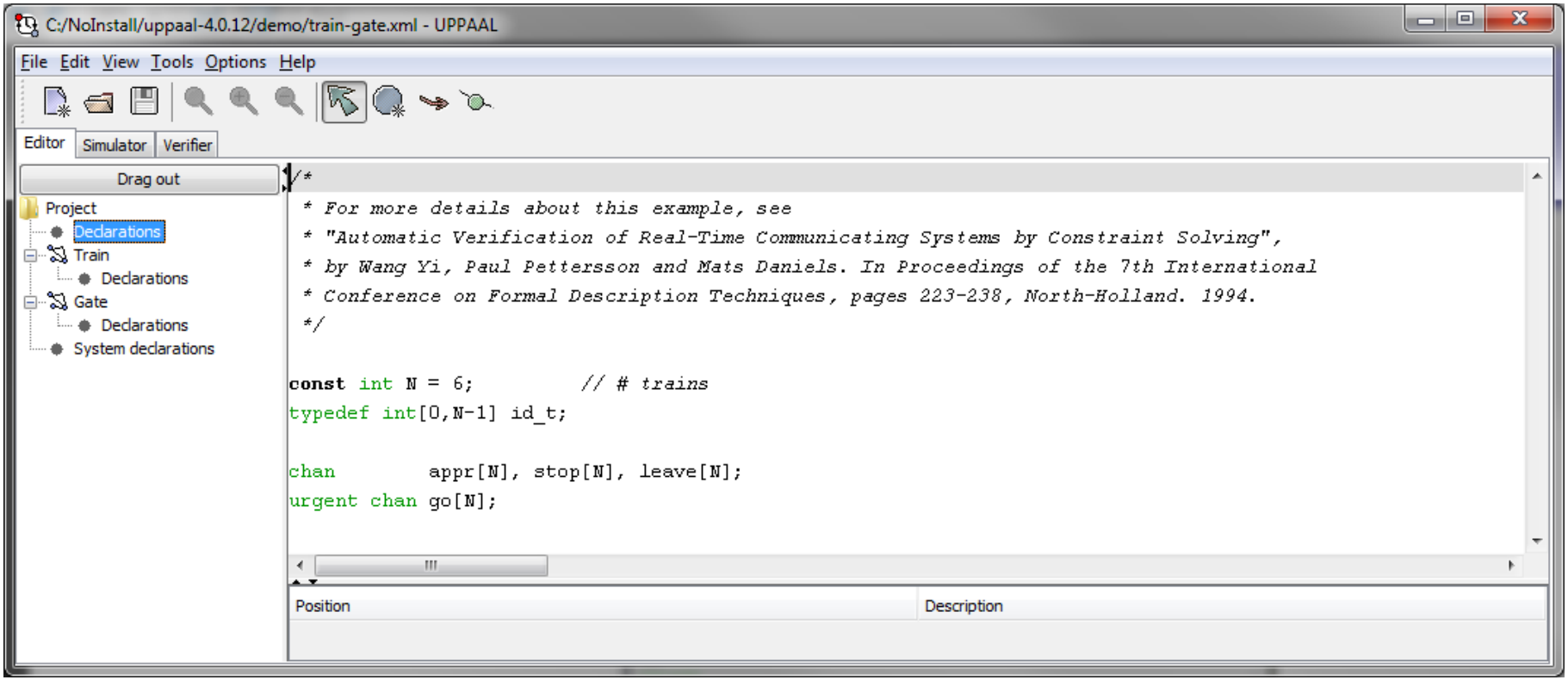
Simulator

- investigation of possible dynamic runs of a system,
- detection of model defects before its verification,
- allows an analysis of run records leading to unwanted states.

Model Verifier

- examination of all possibilities of dynamic behavior of the model,
- check invariants and liveness by searching the state space,
- reachability of symbolic states represented by constraints.

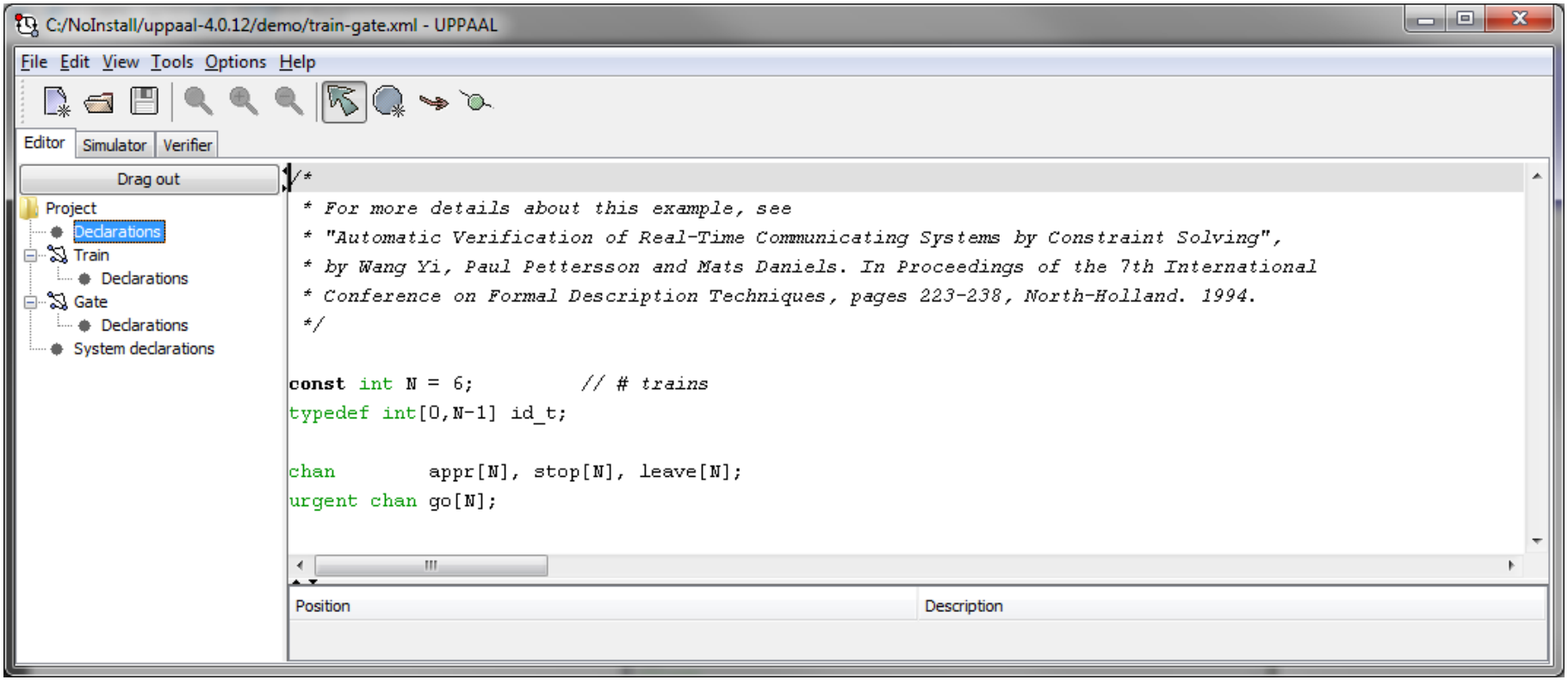
System Editor [UPP10]



Editor

- creation of graphical and textual description of systems

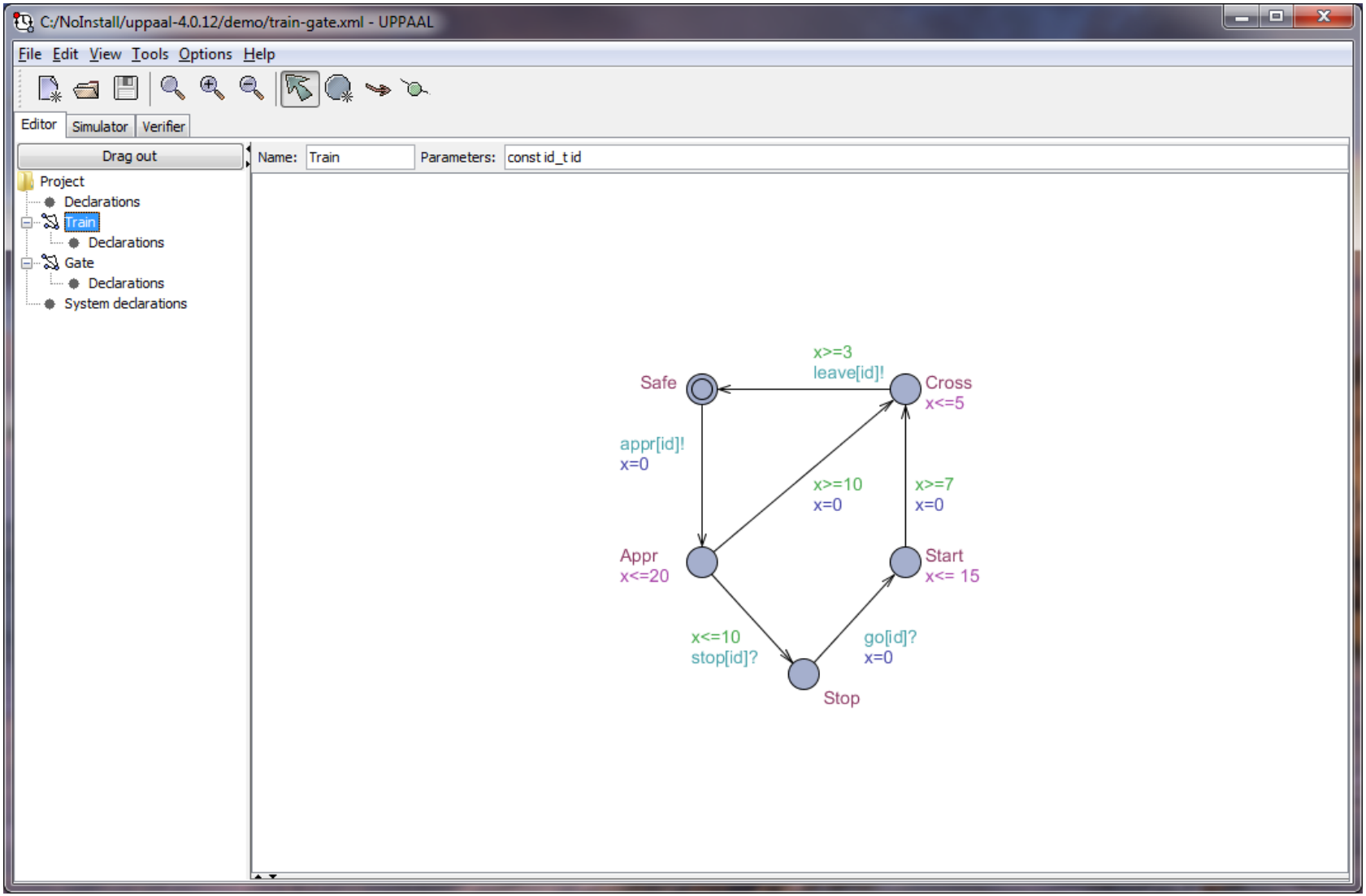
System Editor [UPP10]



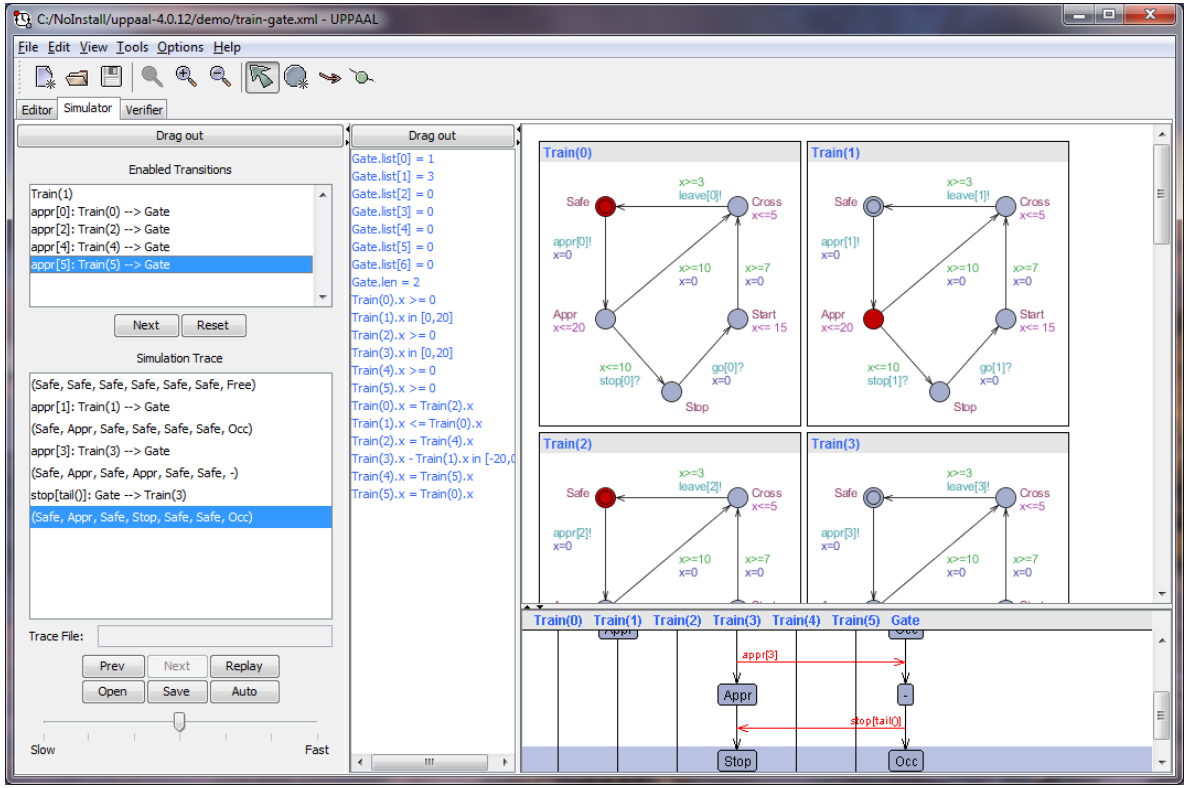
Editor

- creation of graphical and textual description of systems

Graphical System Editor [UPP10]



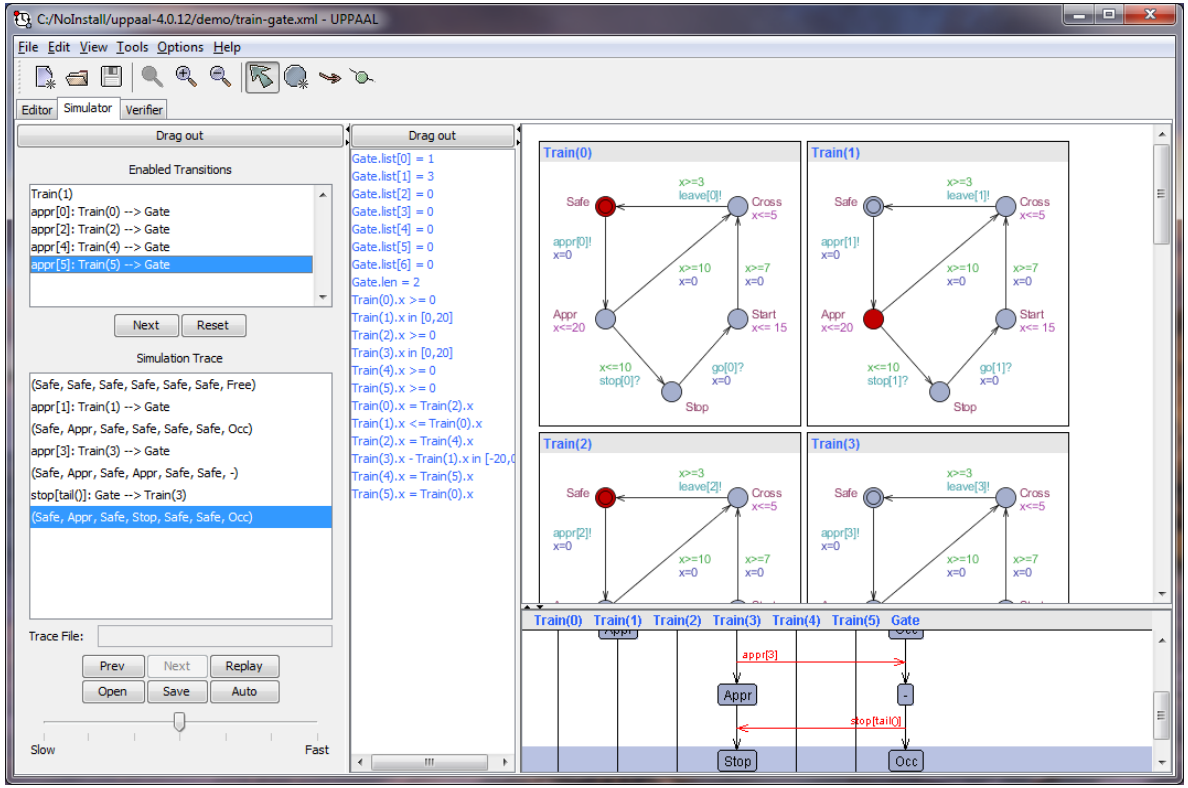
Graphical Simulator [UPP10]



Simulator

- a graphical visualization and recording of possible dynamic behavior of the system description,
- a sequence of symbolic states of the system,
- a possibility to visualize the route generated by the verifier.

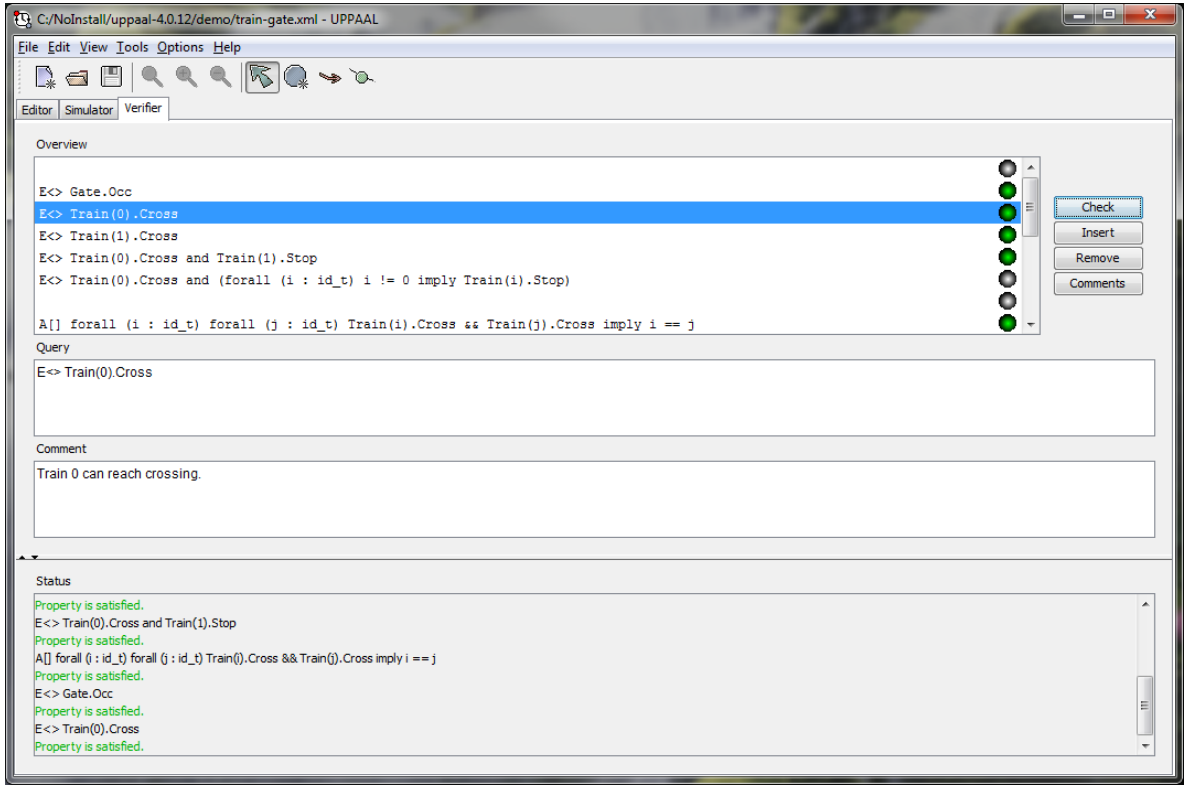
Graphical Simulator [UPP10]



Simulator

- a graphical visualization and recording of possible dynamic behavior of the system description,
- a sequence of symbolic states of the system,
- a possibility to visualize the route generated by the verifier.

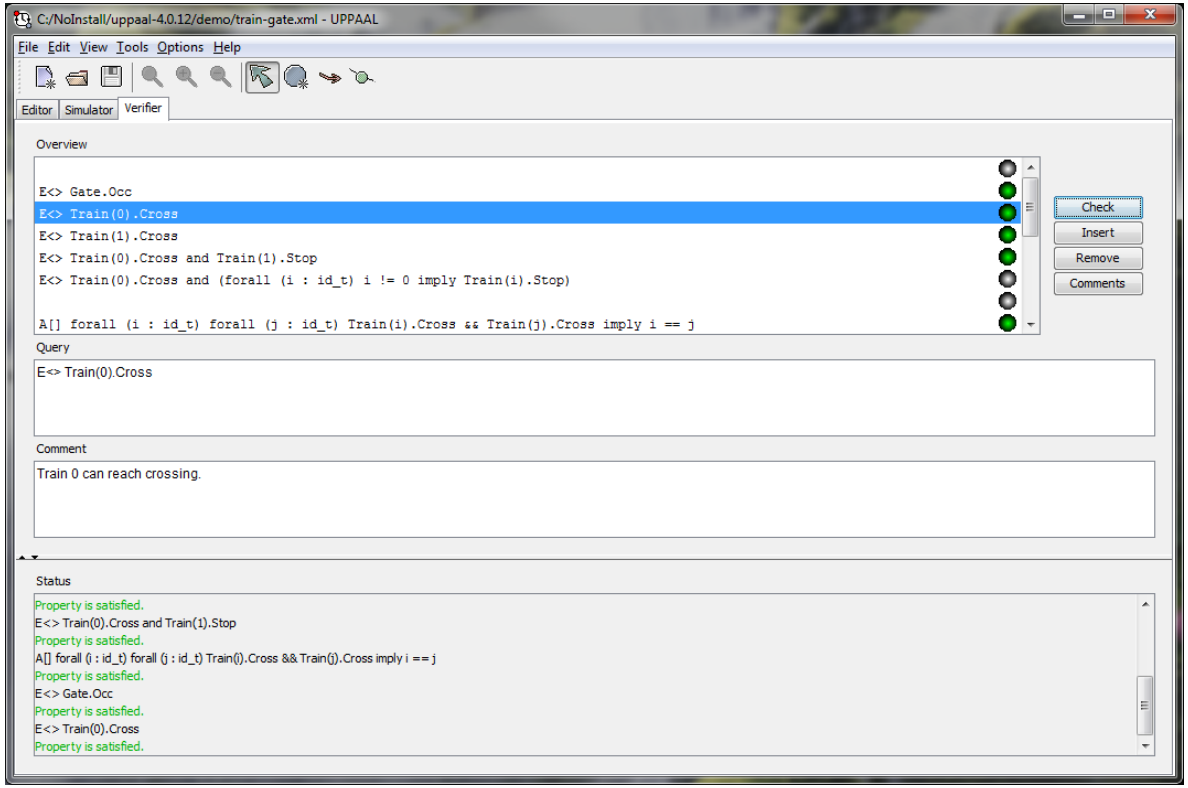
Verifier [UPP10]



Verifier

- Requirements Specification Editor
- Model verifier machine
  - automatic verification of liveness and bounded liveness using reachability in a symbolic state space.

Verifier [UPP10]



Verifier

- Requirements Specification Editor
- Model verifier machine
  - automatic verification of liveness and bounded liveness using reachability in a symbolic state space.

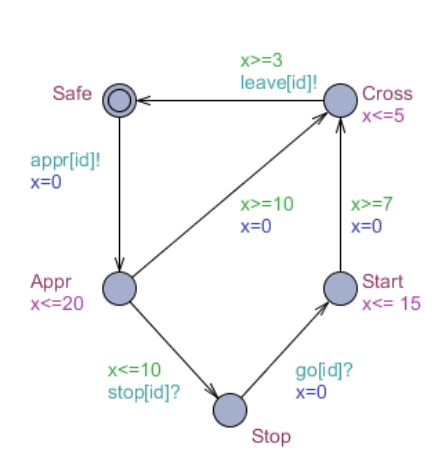
Outline

Default Principles [UPP09]

Model

- Timed automaton
  - a finite state machine with clocks,
  - time is continuous,
  - The clock measures the progress of time.

Process Patterns - Automata



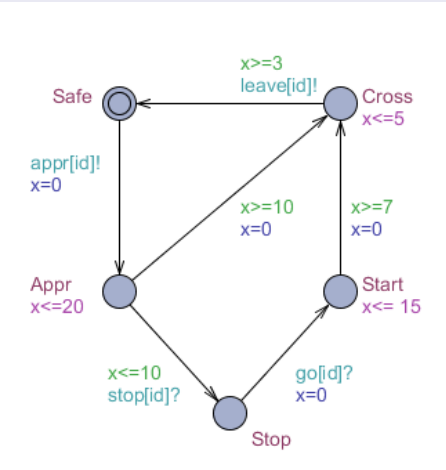
- positions and edges,
- symbolic variables and constants as parameters,
- local variables and clocks,
- the given process is an instance of the pattern.

Default Principles [UPP09]

Model

- Timed automaton
  - a finite state machine with clocks,
  - time is continuous,
  - The clock measures the progress of time.

Process Patterns - Automata



- positions and edges,
- symbolic variables and constants as parameters,
- local variables and clocks,
- the given process is an instance of the pattern.

Timed Automaton [BDL05]

Timed automaton

- is a sextuplet $(L, \ell_0, C, A, E, \mathcal{I})$, where
  - L is a set of positions,
  - $\ell_0 \in L$ is the starting position,
  - C is a set of clocks.
  - A is a set of actions, co-actions and internal τ-actions,
  - $E \subseteq L \times A \times B(C) \times 2^C \times L$ is a set of edges between positions with action, guard and a set of clocks that are reset, and
  - $\mathcal{I} : L \to B(C)$ assigns invariants to positions.

Examples

- $y := 0$ . . . reset clock y,
- $press?$ and $press!$ . . . indicate action and co-action (here a channel synchronization).

Timed Automaton [BDL05]

Timed automaton

- is a sextuplet $(L, \ell_0, C, A, E, \mathcal{I})$, where
  - L is a set of positions,
  - $\ell_0 \in L$ is the starting position,
  - C is a set of clocks.
  - A is a set of actions, co-actions and internal τ-actions,
  - $E \subseteq L \times A \times B(C) \times 2^C \times L$ is a set of edges between positions with action, guard and a set of clocks that are reset, and
  - $\mathcal{I} : L \to B(C)$ assigns invariants to positions.

Examples

- $y := 0$ . . . reset clock y,
- $press?$ and $press!$ . . . indicate action and co-action (here a channel synchronization).

Timed Automaton Clock [BDL05]

Clock

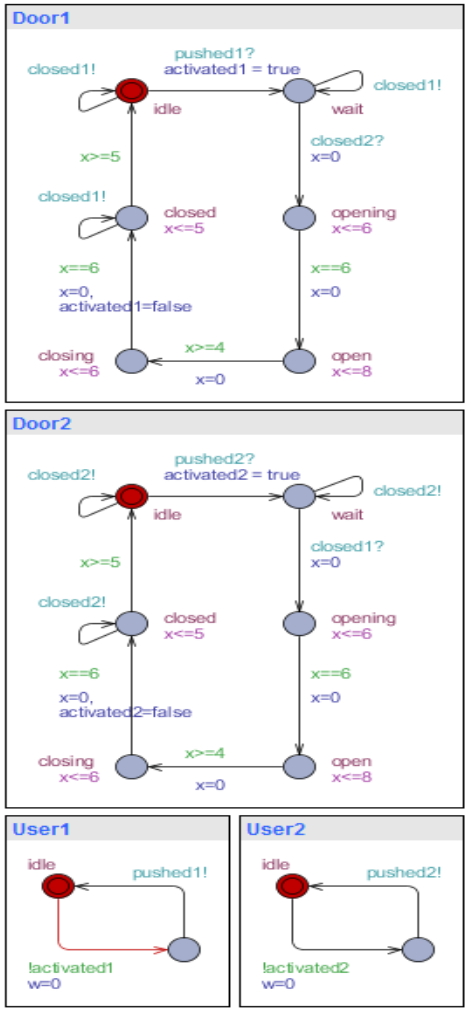
- **Clock evaluation** is a function of $u : C \to \mathbb{R}_{\geq 0}$ from a set of clocks to non-negative real numbers.
- Let $\mathbb{R}^C$ be the set of all clock evaluations.
- Let $u_0(x) = 0$ for all $x \in C$.
- Writing $u \in \mathcal{I}(\ell)$ will mean that u satisfies $\mathcal{I}(\ell)$.
- It is possible to make a transition from a given state using *action* or *delay*.

Timed Automata Semantics [BDL05]

Timed Automata Semantics

- Let $(L, \ell_0, C, A, E, \mathcal{I})$ be a timed automaton.
- Semantics . . . a transition system with label $\langle S, s_0, \rightarrow \rangle$, where
- $S \subseteq L \times \mathbb{R}^C$ is a set of states,
- $s_0 = (\ell_0, u_0)$ is the initial state,
- $\rightarrow \subseteq S \times (\mathbb{R}_{\geq 0} \cup A) \times S$ is a transition relation such that
  - $(\ell, u) \xrightarrow{d} (\ell, u + d)$ if $\forall d' : 0 \leq d' \leq d \Longrightarrow u + d' \in \mathcal{I}(\ell)$
  - $(\ell, u) \xrightarrow{a} (\ell', u')$ if $\exists e = (\ell, a, g, r, \ell') \in E$ $\mid e \in g, u' = [r \mapsto 0]u, u' \in \mathcal{I}(\ell')$,
- $u + d$ maps each clock $x \in C$ every hour to the value $u(x) + d$, for $d \in \mathbb{R}_{\geq 0}$,
- $[r \mapsto 0]u$ indicates clock evaluation, which maps every clock in r to 0 and agrees with u over $C \setminus r$.

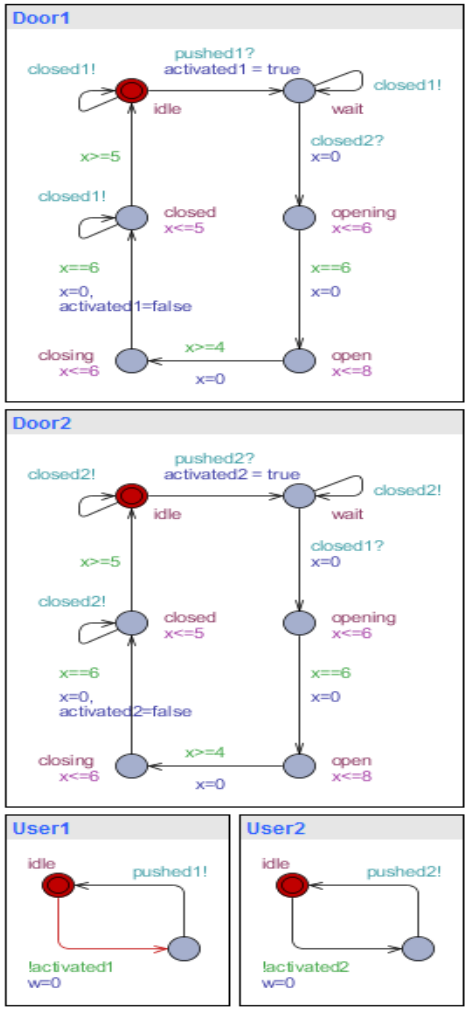
System Specifications [UPP09]



The system is a composition of parallel processes

- each process is modeled as an automaton
- the automaton has a set of positions,
- position changes are done using edges / transitions.
- The state of the system is characterized by the current
  - position of each automaton,
  - values of variables, and
  - clock status.
- transitions can be controlled using guards and synchronizations
- guard is a condition over variables and clocks specifying when a transition is possible.

Process Communication [UPP09]



Synchronization

- synchronization is a mechanism when two processes perform a simultaneous transition
  1. synchronization channel a,
  2. the first process triggers the transition with the message $a!$
  3. the second process makes the transition by receiving the message $a?$
- during the transition it is possible to assign to variables or reset the clock.

Timed Automata Network [BDL05]

Automata set

- A common set of clocks and actions.
- n timed automata $\mathcal{A}_i = (L_i, \ell_i^0, C, A, E_i, \mathcal{I}_i), 1 \leq i \leq n$
- a position vector $\overline{\ell} = (\ell_1, \cdots, \ell_n)$
- a common invariant function $\mathcal{I}(\overline{\ell}) = \wedge_i \mathcal{I}_i(\ell_i)$
- $\overline{\ell}[\ell'_i/\ell_i]$ ... i-th element ℓ_i of vector $\overline{\ell}$ is replaced by ℓ'_i

Timed Automata Network Semantics [BDL05]

Timed Automata Network

- n timed automata $\mathcal{A}_i = (L_i, \ell_i^0, C, A, E_i, \mathcal{I}_i)$
- an initial position vector $\bar{\ell}^0 = (\ell_1^0, \cdots, \ell_n^0)$
- Semantics . . . a transition system with a label $\langle S, s_0, \rightarrow \rangle$, where
- $S \subseteq (L_1 \times \cdots \times L_n) \times \mathbb{R}^C$ is a set of states,
- $s_0 = (\bar{\ell}_0, u_0)$ is the initial state,
- $\rightarrow \subseteq S \times S$ is a transition relation such that
  - $(\bar{\ell}, u) \xrightarrow{d} (\bar{\ell}, u + d)$ if $\forall d' : 0 \leq d' \leq d \Longrightarrow u + d' \in \mathcal{I}(\bar{\ell})$, and
  - $(\bar{\ell}, u) \xrightarrow{a} (\bar{\ell}[\ell'_i/\ell_i], u')$ if $\exists \ell_i \xrightarrow{\tau g r} \ell'_i$
    $\mid u \in g, u' = [r \mapsto 0]u, u' \in \mathcal{I}(\bar{\ell}[\ell'_i/\ell_i])$,
  - $(\bar{\ell}, u) \xrightarrow{a} (\bar{\ell}[\ell'_j/\ell_j, \ell'_i/\ell_i], u')$ if $\exists \ell_i \xrightarrow{c? g_i r_i} \ell'_i$ a $\ell_j \xrightarrow{c! g_j r_j} \ell'_j$
    $\mid u \in (g_i \wedge g_j), u' = [r_i \cup r_j \mapsto 0]u, u' \in \mathcal{I}(\bar{\ell}[\ell'_j/\ell_j, \ell'_i/\ell_i])$,

Outline

Language types [BDL05]

Types

- **Constants** . . . `const name value`, an integer value.
- **Restricted integer values** . . . `int [min, max] name`, the default setting is -32768 to 32768.
- **Arrays** . . . clocks, channels, constants, integer variables
  `chan c [4]; clock a [2]; int [3,5] and [7];`.
- **Initiators** . . . setting the values of integer variables and fields with integer variables
  `int i: = 2; int k [3]: = {1, 2, 3 };`.

Special Transitions [BDL05]

Controling elements

- **Binary synchronization** ... `chan c`,
  edges `c!` and `c?`, nondeterministic pair, blocking.
- **Broadcast synchronization** ... `broadcast chan c`,
  one edge `c!` with all possible `c?`, does not block.
- **Urgent synchronization** ... `urgent chan c`.
  - Delays are not allowed if a transition with the urgent channel is possible.
- **Urgent position** ... The system time cannot elapse if the system is in an urgent position.
- **Commited position**
  - **Commited state** ... at least one of the positions is commited.
  - The commited state cannot be delayed.
  - The next transition must include one output edge leading from the commited position.

Language Expressions [BDL05]

Clocks, integer variables and constants

- **Guard** . . . the result is a logical value.
- **Sync** . . . sync label `Expression!` or `Expression?` or blank.
  The result is a channel.
  It can reference integers, constants, channels.
- **Assignment** . . . comma-separated expressions.
  It can reference clocks, integer variables, constants.
  It can only assign integer values to the clock.
- **Invariant** . . . Conjunction of conditions of the form `x <e` or `x <= e`, where
  - `x` is a link to a clock,
  - `e` is calculated to an integer.

  It can refer to clocks, integer variables, constants.

References I

[BDL05] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL, updated 25th october 2005. Technical report, Department of Computer Science, Aalborg University, Denmark, October 2005.

[Bie08] Armin Biere. Tutorial on model checking, modelling and verification in computer science. In *Proc. 3rd Intl. Conf. on Algebraic Biology (AB'08). Lecture Notes in Computer Science (LNCS)*, volume 5147. Springer, 2008.

[Cam10] Alarico Campetelli. Analysis techniques: State of the art in industry and research. techreport TUM-I1008, Technische Universität München, April 2010.

[Če09] Jiří Čermák. Porovnání modelovacích schopností verifikačních nástrojù. Master's thesis, Masarykova univerzita, Fakulta informatiky, Brno, 2009.

[Hol06] Lukáš Holík. Rozhodnutelnost v temporálních logikách. Master's thesis, Masarykova univerzita, Fakulta informatiky, Brno, 2006.

[UPP09] UPPAAL 4.0: Small tutorial, November 2009.

[UPP10] Tool environment for validation and verification of real-time systems (UPPAAL pamphlet). http://www.it.uu.se/research/group/darts/papers/texts/uppaal-pamphlet.pdf, September 2010.