Model checking time-dependent behavior

Istvan Majzik
majzik@mit.bme.hu

Budapest University of Technology and Economics
Dept. of Measurement and Information Systems

Motivation: Verification of real-time controllers

- Controllers: Time-dependent, state-based, event driven behavior
  - Time is spent in states
  - Conditions (guards) of transitions refer to time
  - Typical implementation: Timers measuring time by counting clock ticks
  - Actions to reset timers
- Typical properties to be checked
  - Satisfying deadlines:
    Reaching a given state in a given time interval
    - E.g., on request, a reply is received in (given) time
    - E.g., message that was sent is received in favorable time
  - Satisfying safety properties in given time interval:
    A property holds in each state that is reachable in a given time interval
    - E.g., the behavior is safe during a mission

Extensions of “classic” temporal logics

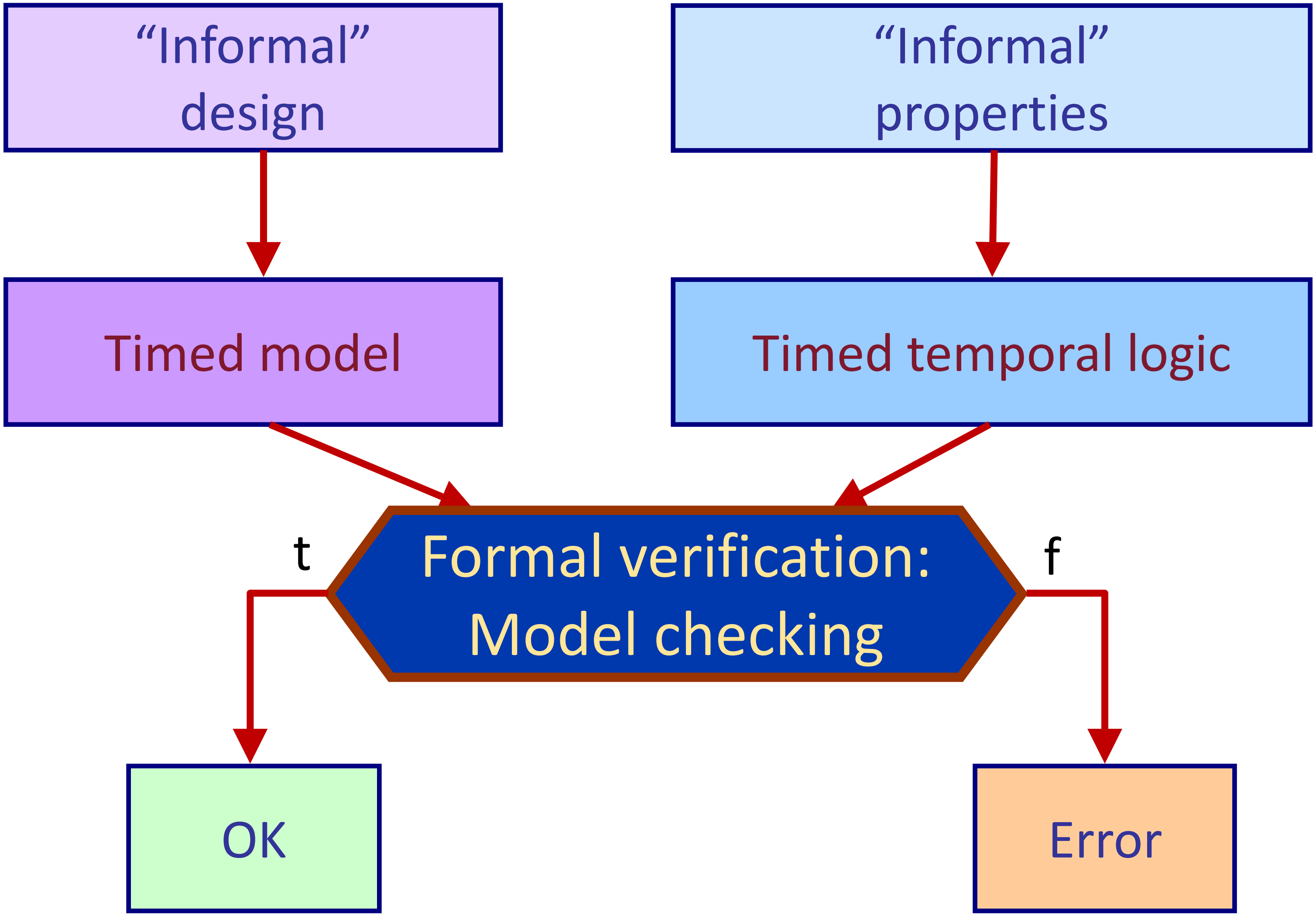
Timed temporal logics (“real-time” logics):

- Requirements of real-time systems
  - The properties refer to clock variables
  - Handling of time intervals

Other extensions: Stochastic logics

- Probability and timing related requirements:
  - E.g.: if the current state is Error then the probability that this condition holds after 2 time units as well, is less than 30%
- Extension of CTL:
  - Interpreted over Continuous-time Markov chains (not a Kripke structure)
  - Probability criteria for state reachability (steady state), path execution
  - Timing criteria (time intervals) for operators X and U

Goal: Formal verification of timed properties



The modeling approach

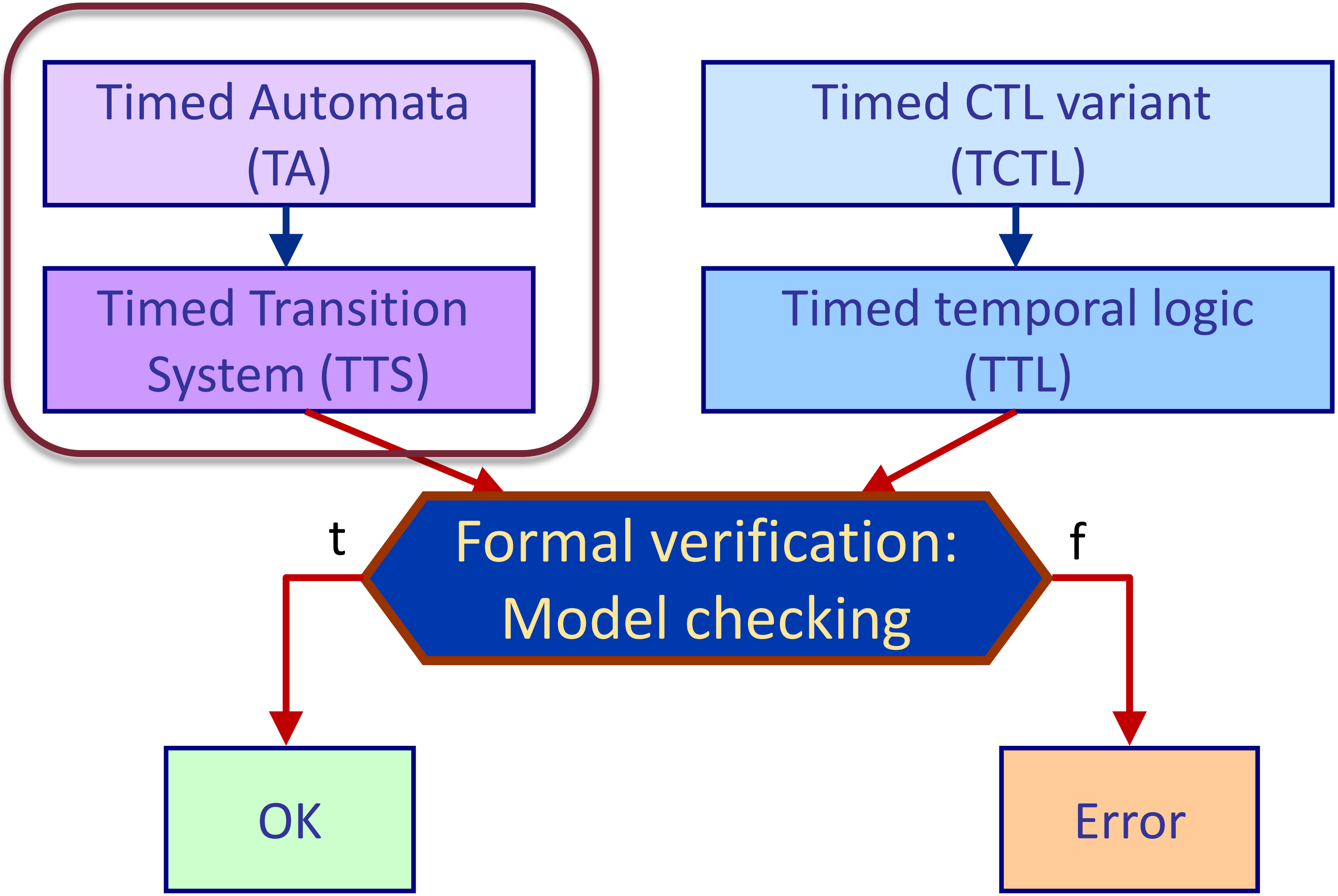
- "Engineering" model $\rightarrow$ Low-level formal model
  - The mapping to low-level formal model gives formal semantics to the engineering model
  - Model checking is performed on the formal model
- Similar approach:
  - UML statecharts $\rightarrow$ Kripke structure (KS)
  - Checking CTL properties on KS
- Model checking timed properties on timed model:
  - Timed Automata (TA) $\rightarrow$ Timed Transition System (TTS)
  - Timed CTL (TCTL) variant $\rightarrow$ Timed Temporal Logic (TL)

Models for time-dependent behavior

Timed Transition Systems

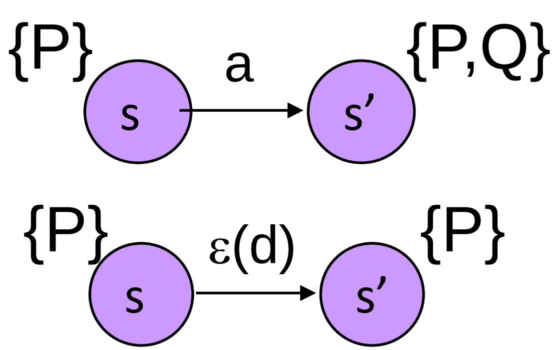
Timed Automata

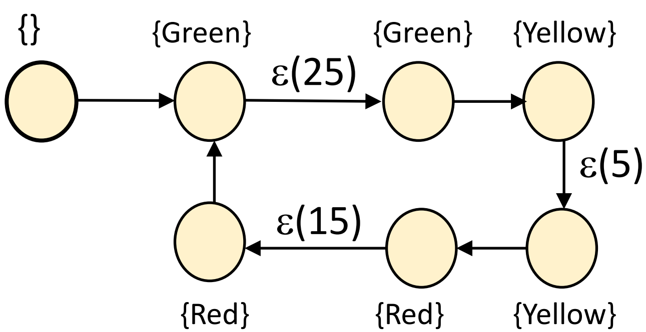
Overview of the approach



Low-level model: Timed Transition System (TTS)

- Notation (properties of states and transitions):
  - Atomic propositions: $AP = \{P, Q, \ldots\}$
  - Atomic actions: $Act = \{a, b, c, \ldots\}$
  - Delay actions: $\Delta = \{\varepsilon(d) \mid d \in R_{\geq 0}\}$
- Definition of TTS: $TTS = (S, s_0, \rightarrow, V)$ where
  - S set of states
  - s_0 initial state
  - $\rightarrow \subseteq S \times L \times S$, where $L \in Act \cup \Delta$ (Δ delay action is included)
  - $V: S \rightarrow 2^{AP}$ labeling of states





Engineering model: Timed Automaton (TA)

- Automaton (states, transitions) + clock variables
  - Concurrent (system) clocks
  - These all increase with the same pace
  - The clock value can be inspected in guards and invariants
  - The clocks can be reset in actions, independently from each other
- Notation for clocks:
  - $C = \{x, y, z, \ldots\}$ clocks
  - $B(C)$ expressions on clocks, $g \in B(C)$ is a clock expression
    - Syntax: $g ::= x \sim n \mid x - y \sim n \mid g \wedge g$
      where $\sim \in \{\leq, \geq, ==, <, >\}$,
      and n non-negative integer (constant)

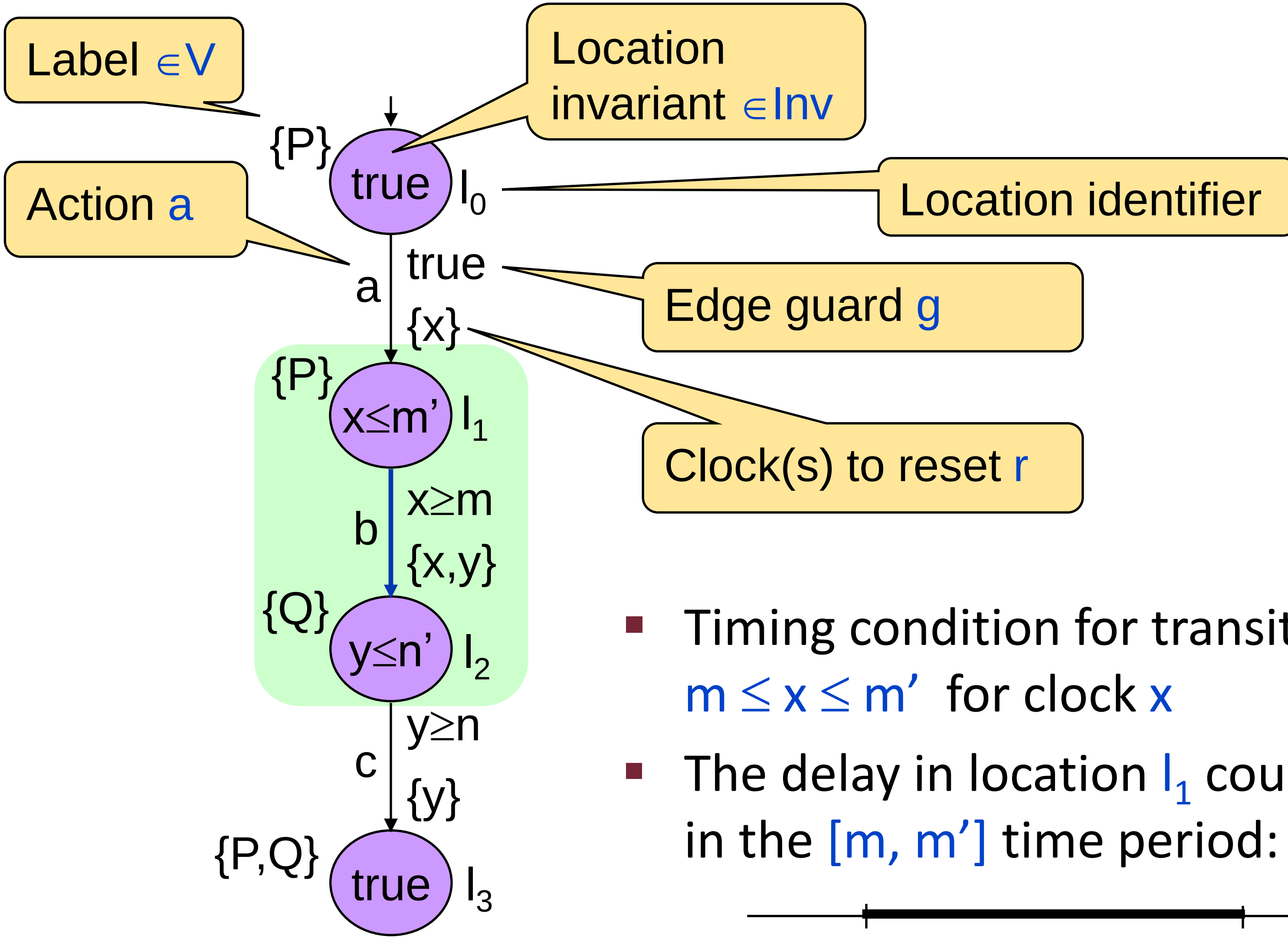
Formal definition of Timed Automaton

- $TA = (N, l_0, E, Inv, V)$ with Act, AP, C where
  - N control locations (will be part of the state)
  - $l_0 \in N$ initial location – here the value of clocks is 0
  - $E \subseteq N \times B(C) \times Act \times 2^C \times N$ set of edges, where an edge is

    $$l \xrightarrow{g,a,r} l'$$

    where
    - g: clock expression – guard condition
    - a: action – activity
    - r: clock set – clocks that are reset
  - $Inv: N \rightarrow B(C)$ clock invariants
    - Limiting the time spent in a control location
  - $V: N \rightarrow 2^{AP}$ labeling (local conditions in control locations)

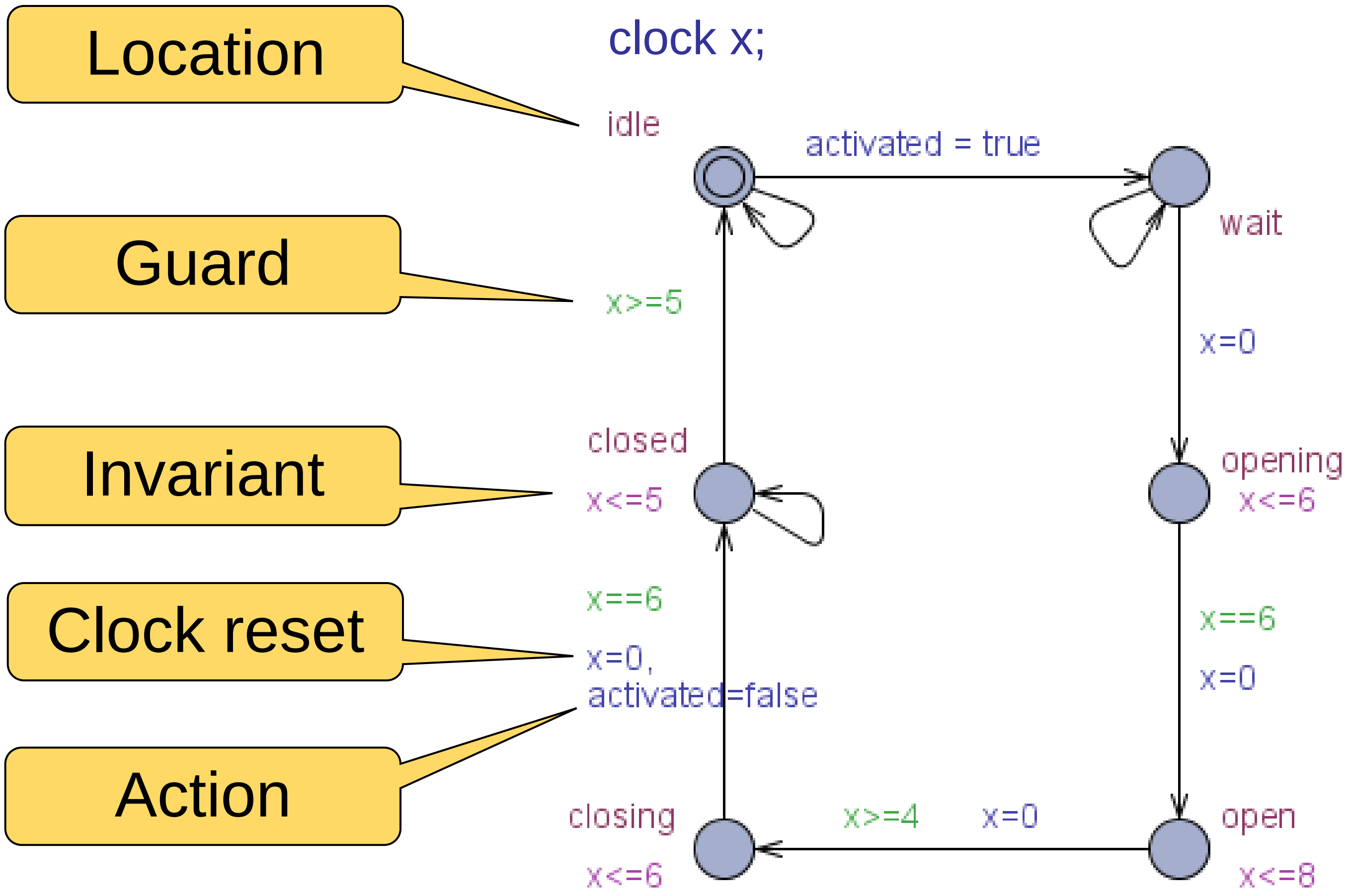
Example: Notations in a TA model

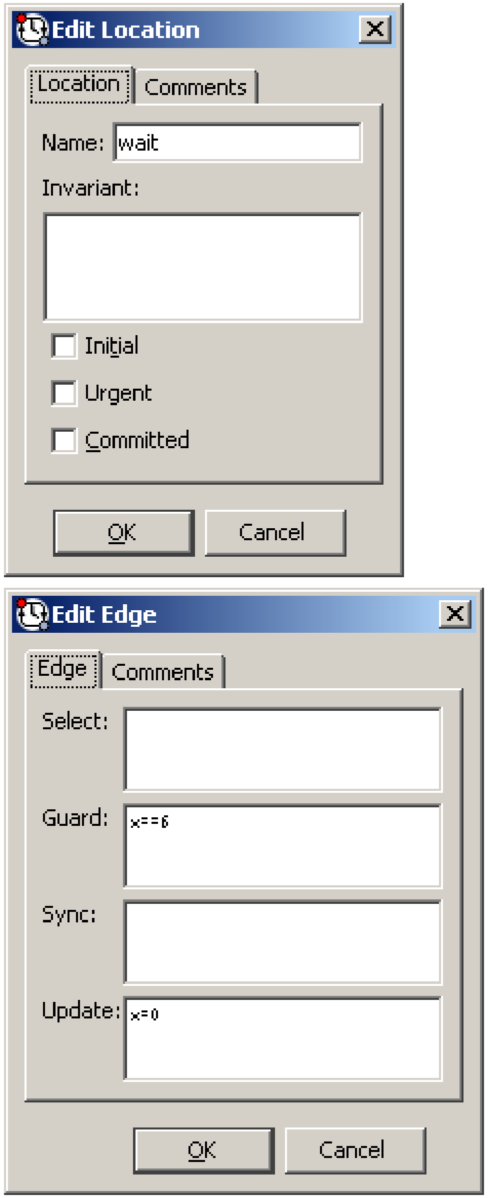


- Timing condition for transition $l_1 \rightarrow l_2$: $m \leq x \leq m'$ for clock x
- The delay in location l_1 could be in the $[m, m']$ time period:

Recap: Timed automaton in UPPAAL

Example: revolving door





Informal semantics of a Timed Automaton

- Initial state:
  - Initial location is active, all clocks are set to 0
- Delay:
  - The values of clocks are increased (at the same pace)
  - The maximum time that can be spent in a control location is determined by the location invariant
- Firing of a transition:
  - Transition (on an edge) is enabled if
    - Source location is active
    - Guard condition is satisfied
    - Clock resets satisfy the invariant of the target location
    - Synchronization (if any – see later) is possible
  - Transition that is enabled may fire (random selection)
    - Action (variable assignment) executed
    - Clocks that were reset become 0
    - The target location of the edge becomes active

Formal semantics of TA: Notations

- Notations for formalizing the semantics:
  - $u: C \to N$ clock valuation
    - $u(x)$ is the value of clock x
  - $u+d$ increasing the clock valuation for all clocks by d
    - The new value of clock x is $u(x)+d$
  - uv: merging clock valuations for sets of clocks, where u and v are clock valuations and K, C are independent: $C \cap K=0$
    - $uv(x)=u(x)$ if $x \in C$
    - $uv(x)=v(x)$ if $v \in K$
  - $[C' \to 0]u$ for all clocks $x \in C'$ the valuation becomes 0, otherwise remains the same
  - $g(u)$ is the evaluation of a guard g in case of valuation u
- State of TA: (l, u) control location and clock valuation
  - Valuation of integer variables is similar (not given separately)

Formal semantics of TA: Mapping to TTS

- The semantics of a TA is a $TTS=(S, s_0, \rightarrow, V)$ where
  - S set of states, where each state is in form (l,u)
  - $s_0 = (l_0,u_0)$ initial state
  - $\rightarrow \subseteq S \times L \times S$ is defined in the following way:
    - $(l,u) \rightarrow^a (l',u')$ is possible, if there exist r and g such that

$$l \xrightarrow{g,\, a,\, r} l' \quad \text{edge exists between the locations,}$$
$$g(u) \quad \text{guard evaluates to true,}$$
$$u' = [r \rightarrow 0]u \quad \text{clock resets occur}$$

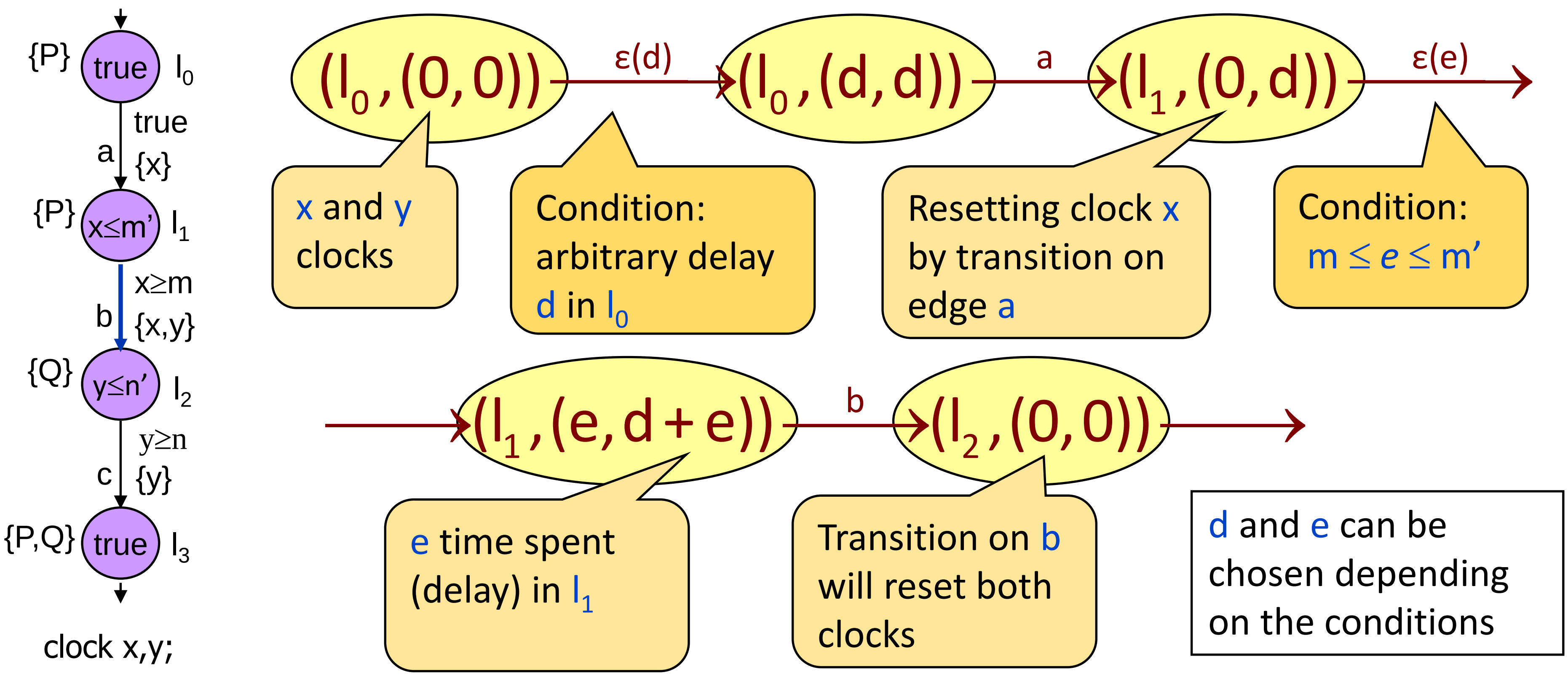
    - $(l,u) \rightarrow^{\varepsilon(d)} (l',u')$ is possible, if

$$l = l' \quad \text{control location does not change,}$$
$$u' = u + d \quad \text{time spent is } d\text{,}$$
$$Inv(u') \quad \text{clock invariant holds}$$

  - $V(l,u) = V(l)$ is the labeling of states

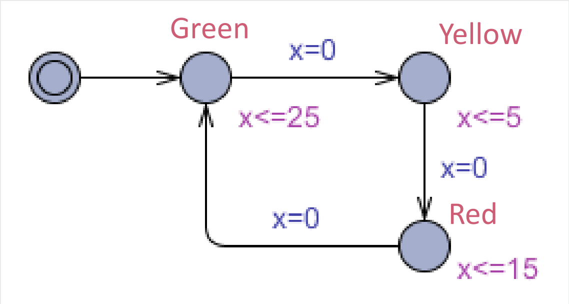
Example of the formal semantics of TA

- The semantics of a TA determines a set of TTS
  - Guards and invariants make various delays possible: possible delays are in (multidimensional) ranges
- The TTS is defined in case of the example TA as follows:



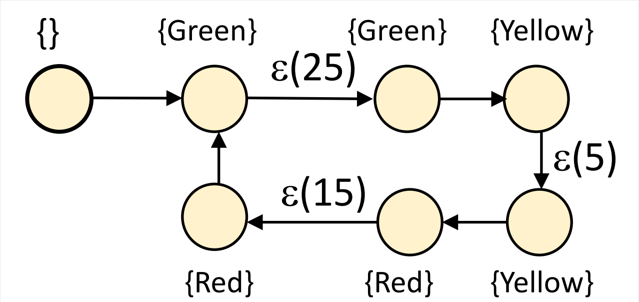
Summary: Formal semantics of TA

Timed Automaton (TA) —Mapping→ Timed Transition System (TTS)



TA=(N, l_0, E, Inv, V), where

- Set of control location: N
- Initial control location: l_0
- Edges with guards, actions and clocks resets:
  $E \subseteq N \times B(C) \times Act \times 2^C \times N$
- Location invariants: Inv: $L \rightarrow B(C)$
- Location labeling: V: $N \rightarrow 2^{AP}$



TTS = (S, s_0, $\rightarrow$, V) where

- Set of states: S as (l,u)
- Initial state: s_0
- Transitions: $\rightarrow \subseteq S \times A \times S$,
  $A = Act \cup \{\varepsilon(d) \mid d \in \Re_{\geq 0}\}$
  - Action transitions: Act labels
  - Delay transitions: $\varepsilon(d)$
- Labeling: V: $S \rightarrow 2^{AP}$

Composition of Timed Automata

- Composition of TA: Network of automata
  - Synchronization among automata
    - Transitions executed simultaneously (rendezvous)
    - Synchronous communication: Sending and receiving on a channel
  - Definition of the composition (synchronization):
    - Which are the transitions that are executed simultaneously?
    - Description: by an f synchronization function,
      that is defined on actions (this way implicitly on transitions)
    - Example: c! are c? are synchronized, f(c!, c?)=0 – corresponding transitions are executed simultaneously, resulting in “no action”
  - $TA_1 \mid_f TA_2$ composition:
    - Its semantics is given as a TTS $\leftarrow$ derived from composition of TTSs
    - Before that: Let us define the composition of TTSs

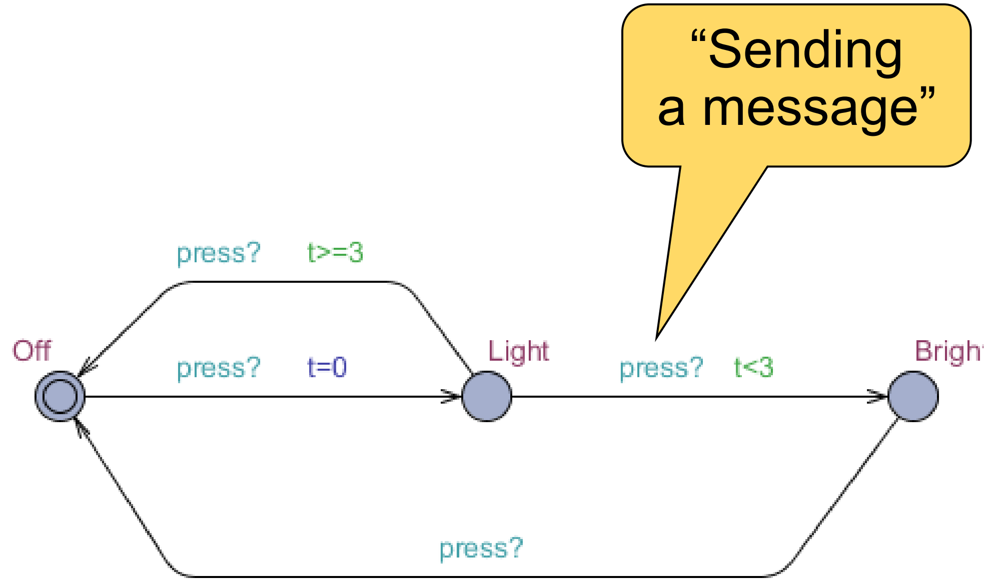
Recap: Synchronization in UPPAAL

Declarations:

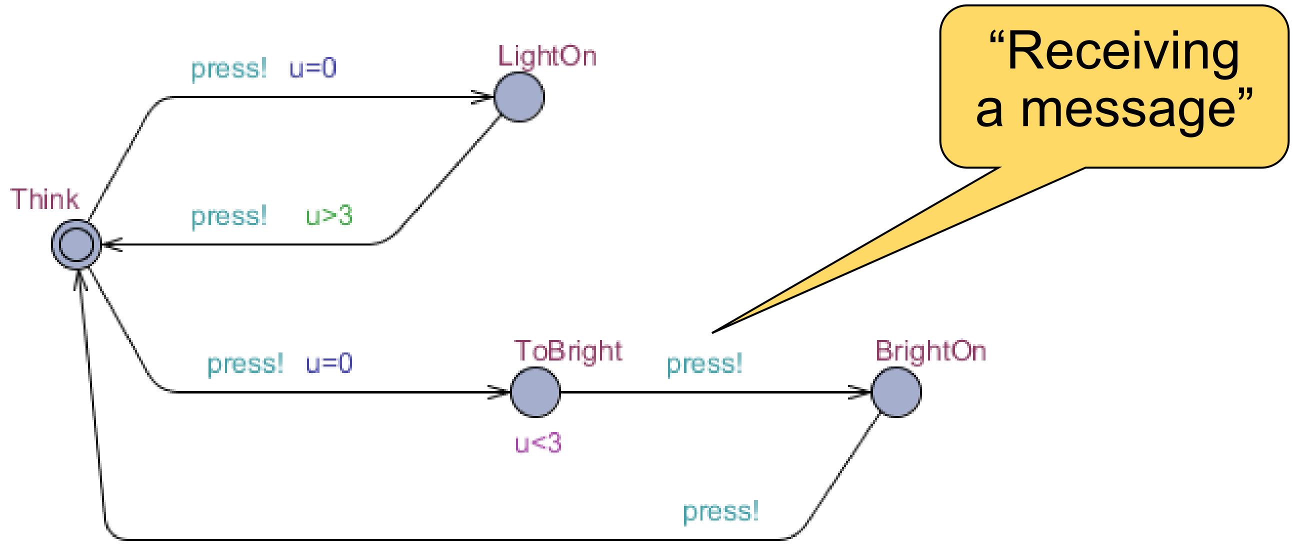
clock t, u;

chan press;

Switch:



User:



Background: Parallel composition of TTSs

- Parameter: synchronization function f
  - $f: (Act \cup \{0\}) \times (Act \cup \{0\}) \rightarrow Act \cup \{0\}$
    where 0 denotes a missing action (also when no transition is taken)
- Definition: Composition $TTS_1 \mid_f TTS_2 = TTS_0$,

  where $TTS_1 = (S_1, s_{1,0}, \rightarrow_1, V_1)$ and $TTS_2 = (S_2, s_{2,0}, \rightarrow_2, V_2)$
  resulting in $TTS_0 = (S, s_0, \rightarrow, V)$
  - $(s_1 \mid_f s_2) \in S$ (pairs of states are composed)
  - $s_0 = (s_{1,0} \mid_f s_{2,0}) \in S$ (initial state)
  - $\rightarrow$ is defined inductively (transitions in TTS_0):
    - $(s_1 \mid_f s_2) \rightarrow^e (s'_1 \mid_f s'_2)$ if $s_1 \rightarrow^a_1 s'_1$ and $s_2 \rightarrow^b_2 s'_2$ and $f(a,b)=e$
    - $(s_1 \mid_f s_2) \rightarrow^{\varepsilon(d)} (s'_1 \mid_f s'_2)$ if $s_1 \rightarrow^{\varepsilon(d)} s'_1$ and $s_2 \rightarrow^{\varepsilon(d)} s'_2$
  - $V(s_1 \mid_f s_2) = V_1(s_1) \cup V_2(s_2)$ (union of labeling)

Semantics of the parallel composition of TA

- Notation: $TA_1 \mid_f TA_2$ network of automata
- Semantics of $TA_1 \mid_f TA_2$ is a $TTS_0 = TTS_1 \mid_f TTS_2$ where
  - Semantics of TA_1 is TTS_1, semantics of TA_2 is TTS_2
    - $TA_1 \mid_f TA_2$ is not an automaton, but $TTS_1 \mid_f TTS_2$ is a TTS
    - Note: It is possible to construct such $TA_1 \otimes TA_2$ product automation, that for the semantics of $TA_1 \otimes TA_2$: $TTS_{TA1 \otimes TA2} \sim TTS_1 \mid_f TTS_2$, i.e., these are bisimulation equivalent (the definition of bisimulation: see later)
- The f synchronization function in case of UPPAAL TA:
  - f(a!,a?)=0 synchronized actions (a! "sending" and a? "receiving")
  - f(a,0)=a action of the first automaton only
  - f(0,a)=a action of the second automaton only

Summary: Semantics of the parallel composition of TA

Network of timed automata

$TA_1 \mid_f TA_2$

Mapping →

Timed transition system

$TTS_0 = TTS_1 \mid_f TTS_2$

- Defining synchronized actions:
  $f\colon (Act \cup \{\}) \times (Act \cup \{\}) \to Act$
- TA_1 semantics:
  $TTS_1 = (S_1, s_{1,0}, \to_1, V_1)$
- TA_2 semantics:
  $TTS_2 = (S_2, s_{2,0}, \to_2, V_2)$

$TTS_0 = TTS_1 \mid_f TTS_2 = (S, s_0, \to, V)$, where

- States: (s_1, s_2) pairs, $s_1 \in S_1$, $s_2 \in S_2$
- Initial state: $s_0 = (s_{1,0}, s_{2,0})$
- Transitions: $\to$ defined as:
  - Action transition: $(s_1, s_2) \to^{e} (s_1', s_2')$
    if $s_1 \to^{a}{}_1\, s_1'$ and $s_2 \to^{b}{}_2\, s_2'$ and $f(a,b)=e$
  - Delay transition: $(s_1, s_2) \to^{\varepsilon(d)} (s_1', s_2')$
    if $s_1 \to^{\varepsilon(d)} s_1'$ and $s_2 \to^{\varepsilon(d)} s_2'$
- Labeling of states:
  $V(s_1, s_2) = V_1(s_1) \cup V_2(s_2)$

Strange behavior of timed automata

Time convergence

Timelock

Zenoness

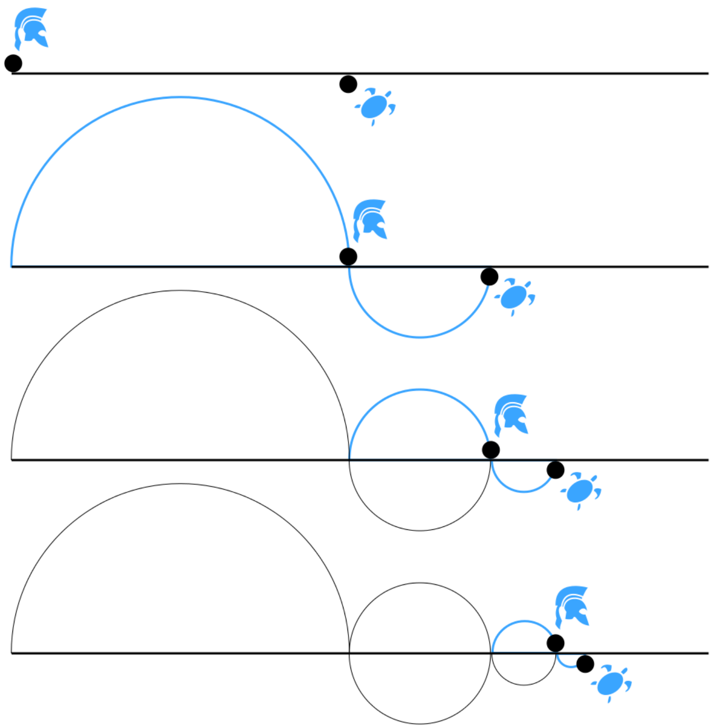
Overview

- Strange behavior: “Unrealistic” execution paths, these may complicate the model checking
  - Time convergence: Infinite sequence of delays converges towards a constant delay
  - Timelock: Time cannot progress to infinity
  - Zenoness: Performing infinitely many actions in finite time
- Handling these paths:
  - Time convergent paths must not be generated as counter-examples by model checking (these are not “fair” paths)
  - Timelock and zenoness can be avoided by proper construction of the model (imposing delays)

Background: Zeno paradox and convergent series

Zeno paradox: Race of Achilles

- The quicker runner (Achilles) gives the slower runner (tortoise) a head start
- In the race, the quicker runner can never overtake the slower
  - The quicker must first reach the point where the slower started
  - In the meantime the slower moved along
  - And so on, so that the slower always holds a lead



From wikipedia: https://en.wikipedia.org/wiki/Zeno%27s_paradoxes

Convergent series (in mathematics):

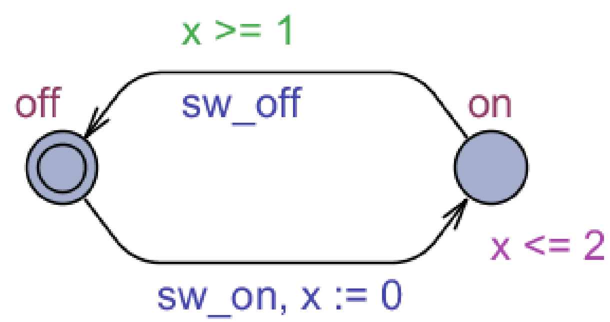
- Sequence of infinite partial sums has a finite limit:

$$L = \sum_{n=0}^{\infty} a_n.$$

- Example: $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^n} + \cdots.$

Time convergence

- Example automaton:



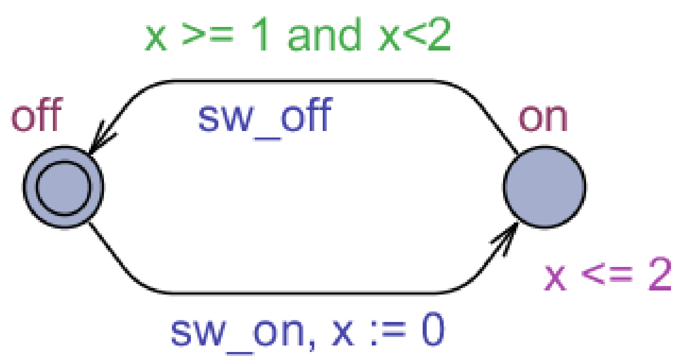
- Example path in its TTS: valid but not realistic

$$\langle off, 0\rangle \xrightarrow{\varepsilon(1/2)} \langle off, 1-2^{-1}\rangle \xrightarrow{\varepsilon(1/4)} \langle off, 1-2^{-2}\rangle \xrightarrow{\varepsilon(1/8)} \langle off, 1-2^{-3}\rangle \;\ldots\ldots$$

- Time convergent path (in general):
  - Infinite sequence d_1, d_2, … of delays,
    where $d_1+d_2+\ldots$ converges to d (constant)
- Time divergent path:
  - The sum of delays converges to infinity

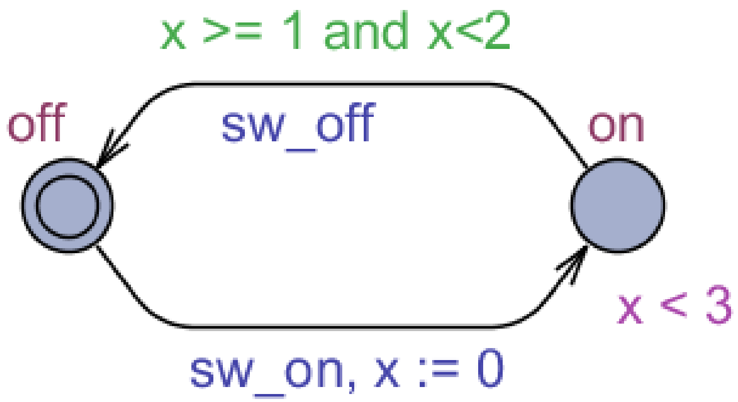
Timelock

- A location contains a timelock if there is no time divergent path from that location
  - There is no path on which the time can progress to infinity
  - Terminal location is not necessarily a timelock
    - If location invariant is true then the time can progress in that location to infinity
- Example automaton with timelock:
  - (on, 2) is reachable, and there is no divergent path



Example: Timelock with time convergent path

- Example automaton:

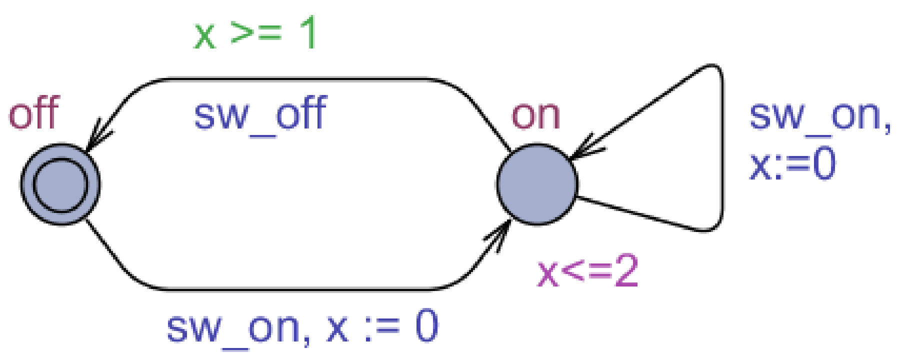


  - In its TTS (on, d) is timelock if $2 \leq d < 3$
  - Time convergent path to timelock:

$$\langle on, 2\rangle\ \langle on, 2.9\rangle\ \langle on, 2.99\rangle\ \langle on, 2.999\rangle\ \langle on, 2.9999\rangle\ \ldots$$

Zenoness

- Zeno path:
  - Time convergent, but at the same time infinitely many $a \in Act$ actions can be executed
- Example automaton:



Zeno paths:

sw_on loop without delay

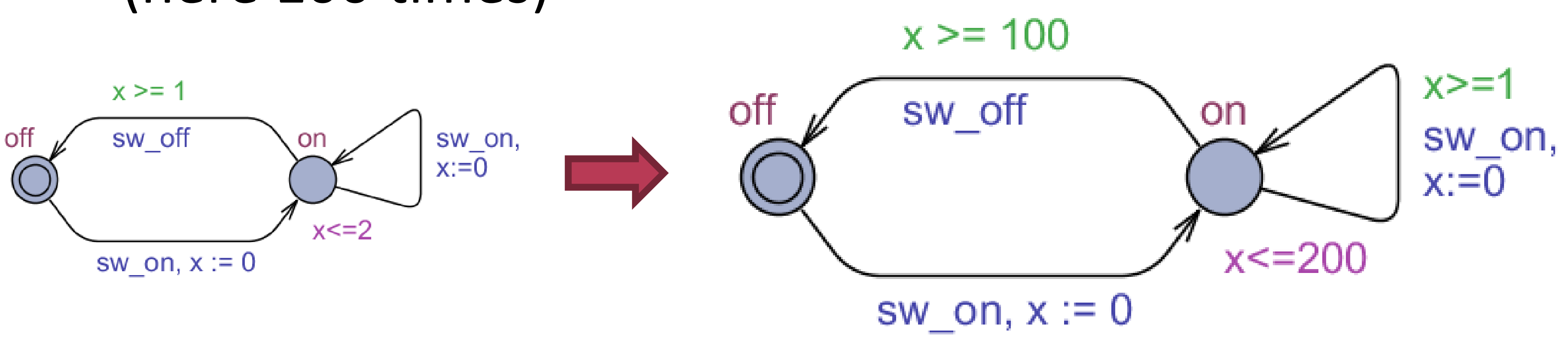
$$\langle \mathit{off}, 0\rangle \xrightarrow{\text{sw\_on}} \langle on, 0\rangle \xrightarrow{\text{sw\_on}} \langle on, 0\rangle \xrightarrow{\text{sw\_on}} \langle on, 0\rangle \xrightarrow{\text{sw\_on}} \ldots$$

$$\langle \mathit{off}, 0\rangle \xrightarrow{\text{sw\_on}} \langle on, 0\rangle \xrightarrow{0.5} \langle on, 0.5\rangle \xrightarrow{\text{sw\_on}} \langle on, 0\rangle \xrightarrow{0.25} \langle on, 0.25\rangle \xrightarrow{\text{sw\_on}} \ldots$$

sw_on loop with delays but their sum converges to 1:
0.5 + 0.25 + 0.125 + …

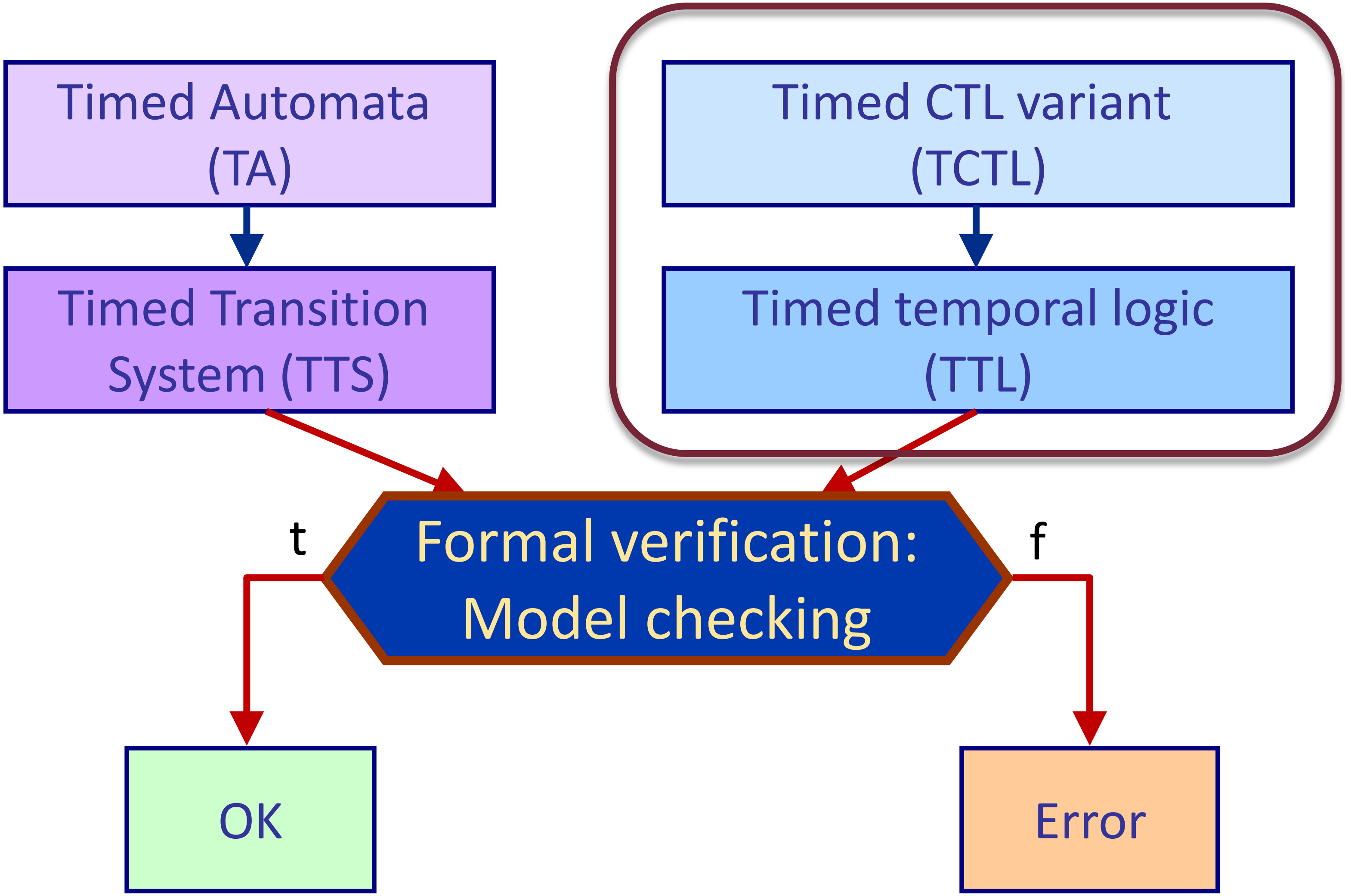
Avoiding Zeno paths

- In case of the previous example automaton:
  - Imposing (minimal) delays between successive sw_on actions (this way time will progress)
- Example: The modified automaton model
  - The minimal delay is 1 unit (in case of integer clocks)
  - The given application-specific delays are increased (here 100 times)



Formalizing properties: Timed temporal logics

Overview of the approach



Introduction of a Timed Temporal Logic

- Expectations:
  - Use clock variables in the logic (intuitive)
  - Recursion is allowed in the definition of its semantics
  - Formalize the typical safety and liveness properties on TA
  - Decidable (properties can be checked)
- Notation:
  - K set of formula clocks
    - Used in the property formula only (if model clocks are not known)
    - Their rate is the same as the rate of the model clocks
  - Id identifiers (in TL formula to include recursion)
    - $Z \in Id$ variable
    - Z can be assigned a formula: $Z := \varphi$
    - $D(Z)$ denotes the assignment: $D(Z) = \varphi$, if Z was assigned φ

The syntax of Timed TL

- $\varphi ::= P \mid c \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \exists\varphi \mid \forall\varphi \mid \langle a\rangle\varphi \mid [a]\varphi \mid$
  $x \text{ in } \varphi \mid x+n \sim y+m \mid Z$

  where $P\in AP$, $c\in B(K)$, $a\in Act$, $x\in K$, and $Z\in Id$, $m,n\in N$
- Temporal operators (informally):
  - $\exists\varphi$ – exists a delay such that φ holds
  - $\forall\varphi$ – for all delays φ holds
  - x in φ – by resetting x clock φ holds
  - $x+n \sim y+m$ – comparison of clock expressions
- This Timed TL can be evaluated on TTS (this way also on TA and network of TA)
  - s: (l,u) state of TTS (derived from TA)
  - (s,v) notation for TTS state and formula clock valuation v

The semantics of Timed TL (1)

- $(s,v) \models P$ for atomic proposition P iff $P \in V(s)$
  - I.e., P is included among the labels of state s
- $(s,v) \models c$ for clock expression iff $c(v)$ holds
  - I.e., in the case of clock valuation v the clock expression c is true
- $(s,v) \models \varphi_1 \wedge \varphi_2$ iff $(s,v) \models \varphi_1$ and $(s,v) \models \varphi_2$
- $(s,v) \models \varphi_1 \vee \varphi_2$ iff $(s,v) \models \varphi_1$ or $(s,v) \models \varphi_2$
- $(s,v) \models \exists\varphi$ iff $\exists d,s'$: $s \rightarrow^{\varepsilon(d)} s'$ és $(s',v+d) \models \varphi$
  - I.e., there exists a state reachable from (s,v) by a delay, in which φ holds
- $(s,v) \models \forall\varphi$ iff $\forall d,s'$: $s \rightarrow^{\varepsilon(d)} s' \Rightarrow (s',v+d) \models \varphi$
  - I.e., for all states reachable from (s,v) by delay, φ holds

The semantics of Timed TL (2)

- $(s,v) \models \langle a \rangle \varphi$ iff $\exists s': s \rightarrow^a s'$ and $(s',v) \models \varphi$
  - I.e., there exists a state reachable from (s,v) by action a, in which φ holds
- $(s,v) \models [a]\varphi$ iff $\forall s': s \rightarrow^a s' \Rightarrow (s',v) \models \varphi$
  - I.e., in all states reachable from (s,v) by action a, φ holds
- $(s,v) \models x \text{ in } \varphi$ iff $(s,v') \models \varphi$ where $v'=[\{x\}\rightarrow 0]v$
  - I.e., by resetting formula clock x, φ holds
- $(s,v) \models x+n \sim y+m$ iff $v(x)+n \sim v(y)+m$
  - I.e., comparison holds for the values of the formula clocks
- $(s,v) \models Z$ iff $(s,v) \models D(Z)$
  - I.e., the expression assigned to Z is true on (s,v)

Properties of the Timed TL

- Recap: The syntax

$$\varphi ::= c \mid P \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \exists\varphi \mid \forall\varphi \mid \langle a\rangle\varphi \mid [a]\,\varphi \mid$$
$$x \text{ in } \varphi \mid x+n \sim y+m \mid Z$$

- Low level, simple operators
  - Existential and universal operators for transitions with actions or delay
  - „Base logic" (its role is similar to the mu-calculus)
  - Expressivity is high (since recursion is allowed,
    but this construct in itself is not easy to use and not intuitive)
- Using the Timed TL
  - Definition of composite / derived operators from the simple ones
  - These are closer to intuition and practical use:
    E.g., invariants, UNTIL, $\text{UNTIL}_{<t}$, BEFORE t
  - Restrictions in model checkers (e.g., UPPAAL, KRONOS) in order to have more effective model checking algorithms

Useful expressions in the Timed TL

- INV(φ) invariant: it is the recursive expression assigned to Z, namely $\mathbf{Z := \varphi \land \forall Z \land [Act]Z}$
  - here $[Act]Z$ means $[a_1]Z \land [a_2]Z \land \ldots \land [a_n]Z$ for all $a_i \in Act$

Useful expressions in the Timed TL

- INV(φ) invariant: it is the recursive expression assigned to Z, namely $Z := \varphi \wedge \forall Z \wedge [Act]Z$

  here $[Act]Z$ means $[a_1]Z \wedge [a_2]Z \wedge \ldots \wedge [a_n]Z$ for all $a_i \in Act$

In all states that are reachable by delay transition, Z will hold

In all states that are reachable by action transition, Z will hold

These together form a general "next state" operator for both delay and action transitions

If Z holds on M, where $Z := \varphi \wedge \forall Z \wedge [Act]Z$, **then φ is invariant on M**

Useful expressions in the Timed TL

- INV(φ) invariant: it is the recursive expression assigned to Z, namely $\mathbf{Z := \varphi \wedge \forall Z \wedge [Act]Z}$

  here $[Act]Z$ means $[a_1]Z \wedge [a_2]Z \wedge \ldots \wedge [a_n]Z$ for all $a_i \in Act$

- φ_1 UNTIL φ_2 „weak until": it is Z, where $\mathbf{Z := \varphi_2 \vee (\varphi_1 \wedge \forall Z \wedge [Act]Z)}$

  φ_2 will not necessarily hold

- φ_1 UNTIL$_{<n}$ φ_2 $\equiv$ x in (($\varphi_1 \wedge x<n$) UNTIL φ_2)

  here x is evaluated after reset, this way time n is relative

- φ BEFORE n $\equiv$ true UNTIL$_{<n}$ φ

- Example: at(l_i) BEFORE t deadline property
  - It means reaching l_i location before t
  - Here notation: at(l_i) means that the automaton is at control location l_i

Simplification for effective evaluation

- Recap: The original syntax
  $\varphi ::= c \mid P \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \exists\varphi \mid \forall\varphi \mid \langle a\rangle\varphi \mid [a]\,\varphi \mid$
  $x \text{ in } \varphi \mid x+n \sim y+m \mid Z$

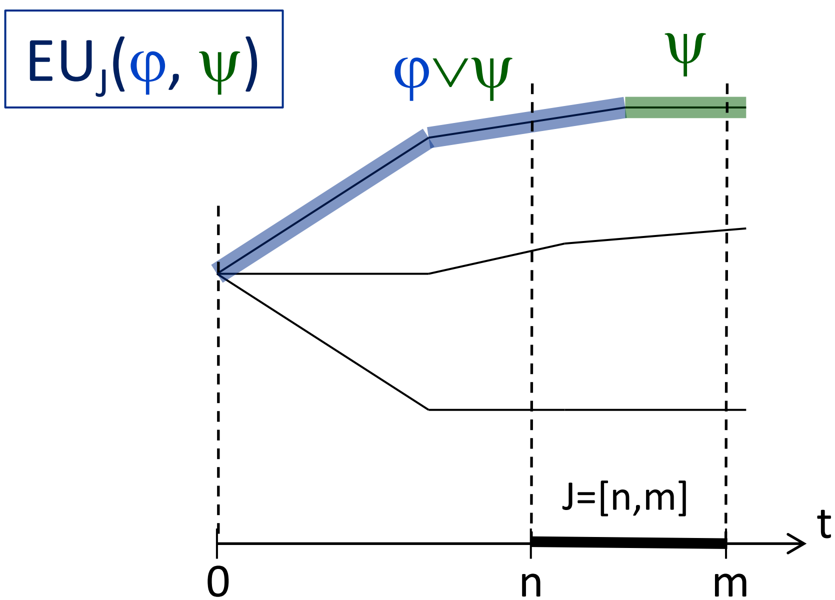
- To formalize safety and bounded liveness properties it is sufficient to restrict it as follows:
  - $\exists\varphi$ omitted (existential quantifier on delays)
  - $\langle a\rangle$ omitted (existential quantifier on actions)
  - $c\vee\varphi$ formula allowed only
  - $P\vee\varphi$ formula allowed only

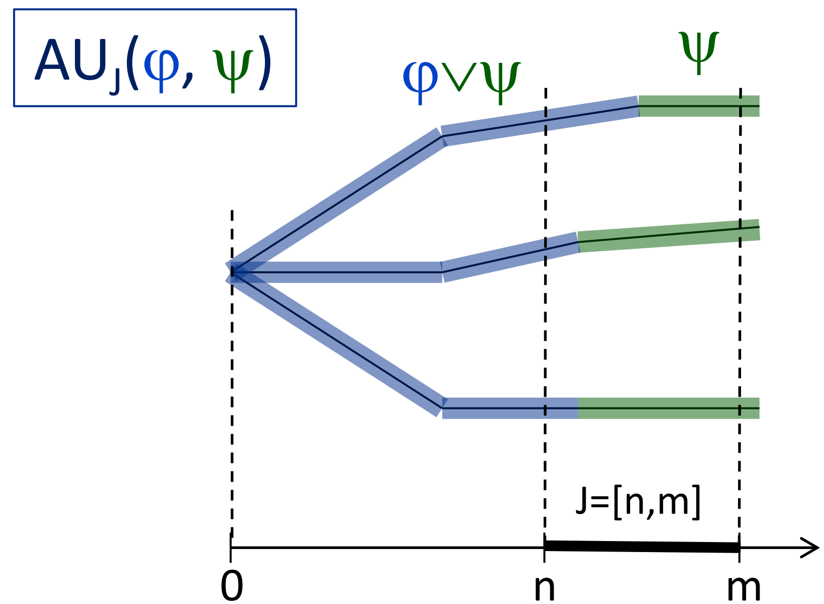
  Invariants, UNTIL, UNTIL$_{<n}$, BEFORE t can be expressed

Timed CTL

Timed CTL

- CTL variant with time: Timed Computational Tree Logic
- Characteristics:
  - Temporal operators are bound by time intervals
    - J = [n,m] bound, with open or closed intervals
  - Only the U “until” temporal operator is included in the syntax
    - With existential and universal quantifier on paths: EU and AU





Formal syntax of TCTL

$$\text{TCTL} ::= P \mid g \mid \varphi \wedge \psi \mid \neg \varphi \mid EU_J(\varphi, \psi) \mid AU_J(\varphi, \psi)$$

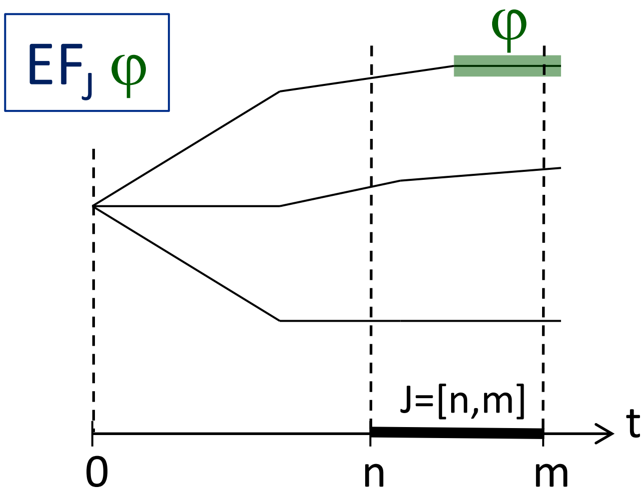
- Atomic propositions: $P \in AP$ state labels
- Clock expressions: $g \in B(C)$
- Boolean operators in case of φ and ψ formula:
  - $\varphi \wedge \psi$
  - $\neg \varphi$
- Temporal operators in case of φ and ψ formula and J bounded time interval:
  - $EU_J(\varphi, \psi)$ – there exists a path on which the following holds: ψ holds in time interval J and until that $\varphi \vee \psi$ holds
  - $AU_J(\varphi, \psi)$ – on all paths the following holds: ψ holds in time interval J and until that $\varphi \vee \psi$ holds

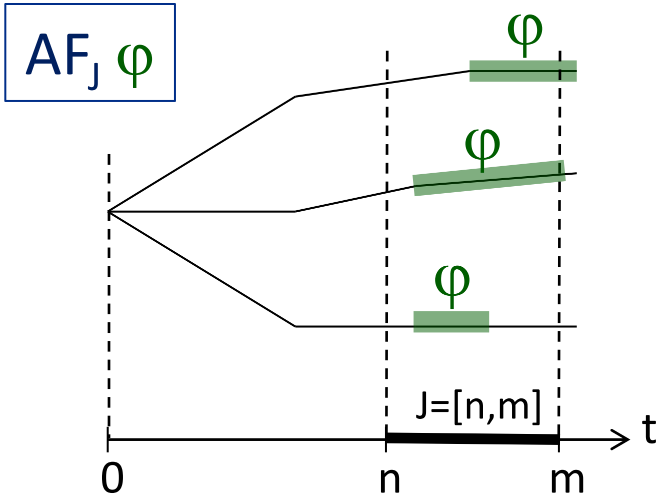
  here J is in form $[n,m]$, $(n,m]$, $[n,m)$, (n,m), also $m=\infty$ is possible

Defining derived temporal operators

$EF_J\ \varphi = EU_J(\text{true}, \varphi)$

$AF_J\ \varphi = AU_J(\text{true}, \varphi)$





$EG_J\ \varphi = \neg AF_J\ \neg\varphi$

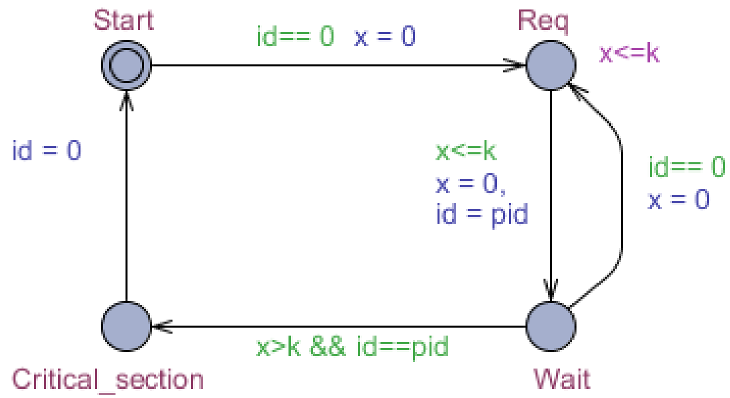
$AG_J\ \varphi = \neg EF_J\ \neg\varphi$

In case of untimed properties: $J = [0,\infty)$

The model checker KRONOS

- Using the TA formalism
- TCTL temporal logic variant
  - $\exists$<> (corresponds to EF)
  - $\forall$[] (corresponds to AG)
  - $\exists\text{<>}_{=n}$ (reachable in n time units)
  - $\forall[]_{\leq n}$ (always reached in max. n time units)
- Interesting property that is often specified:
  - $\forall[]\ \exists\text{<>}_{=1}$ true
    - In each state the time is able to progress 1 time unit
    - It is not possible that "time is stopped"

Recap: Temporal operators in UPPAAL



Model of a mutual exclusion protocol (Fischer) for automata:

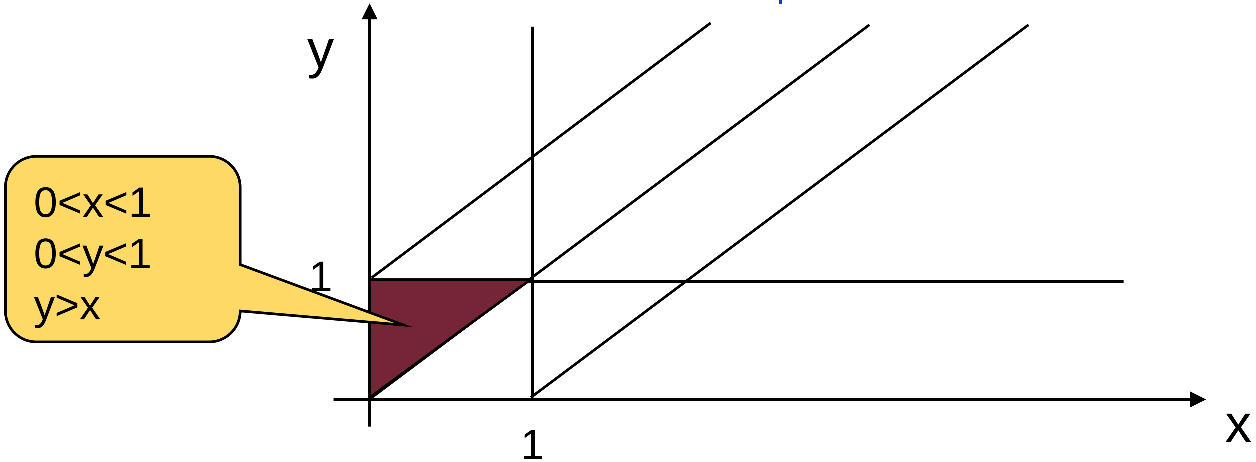
- Liveness without timing for automation P0:
  - After Wait, the critical section will eventually be reached on all paths:

    P0.Wait --> P0.Critical_section
- Timed liveness:
  - After Wait, the critical section will be reached on all paths in less that T time units:

    P0.Wait --> (P0.Critical_section and x<T)
  - Note that the x clock is reset when entering Wait

Outlook: The basic idea of model checking

- Identification of (time) regions,
  where conditions are evaluated to the same truth value
  - Conditions determined by invariants and guards in the TA
  - There are many potential delays that make a condition true
  - This way regions are formed on the clock variables
  - The truth of a Timed TL expressions is defined on the regions
- Semantics based model checking:
  - Can be solved as a constraint satisfaction problem
  - Is there a clock valuation with which φ holds?



Summary

- Motivation: Checking the models of real-time systems
- Models and mappings
  - Timed Transition System (TTS)
  - Timed Automata (TA) $\rightarrow$ TTS
  - Network of TA $\rightarrow$ TTS
- Interesting behavior in models of timed systems
  - Time convergence, timelock, zenoness (Zeno path)
- Formalizing properties
  - Timed TL
  - Timed CTL variants
- Model checking
  - Basic idea: regions are manipulated