# Timed Systems and Uppaal

Renato Neves

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

# Table of Contents

# Uppaal

Editor

- Templates and instantiations
- Global and local declarations
- System definition

Simulator

- Viewers: automata animator

Verifier (we will explore this later on)

- Answers questions about the system at hand

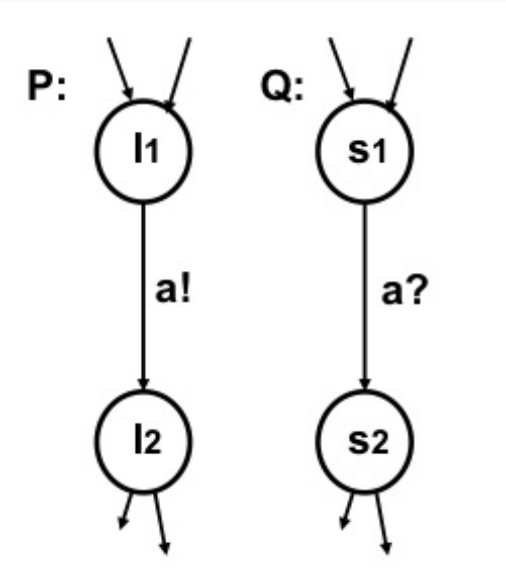
# Extensions of timed automata provided by Uppaal

- data expressions over bounded integer variables (e.g `int[2..45] x`) allowed in guards, assigments, and invariants
- rich set of operators over integer and Booleans, including bitwise operations, arrays, and initialisers
- non-standard types of synchronisation
- non-standard types of location

# Extension: Broadcast Synchronisation

- Declared as `broadcast chan channelname`
- a sender can synchronise with an arbitrary number of receivers
- any receiver than can synchronise in the current state must do so
- broadcast sending is never blocking (the send action can occur even with no receivers).

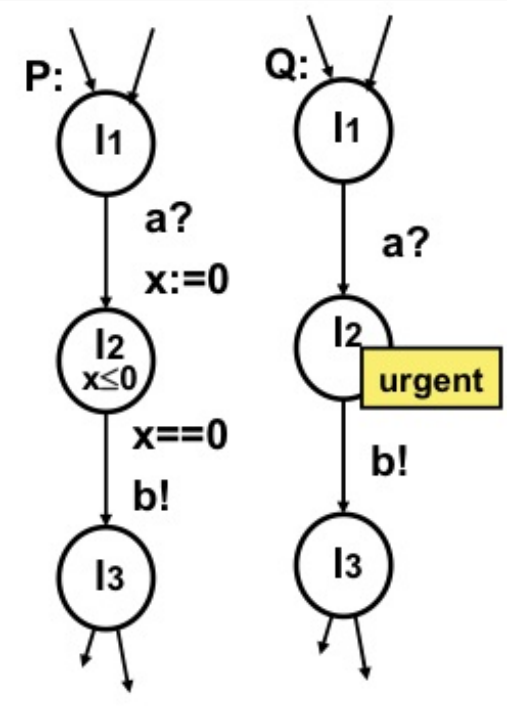
# Extension: Urgent Synchronisation




Declared as `urgent chan channelname`
Both edges need to be taken as soon as they are ready (simultaneously in locations $\ell_1$ and $s_1$). In other words, no delay allowed if a synchronisation transition on an urgent channel is enabled. Note the problem cannot be solved with invariants because locations $\ell_1$ and $s_1$ can be reached at different moments

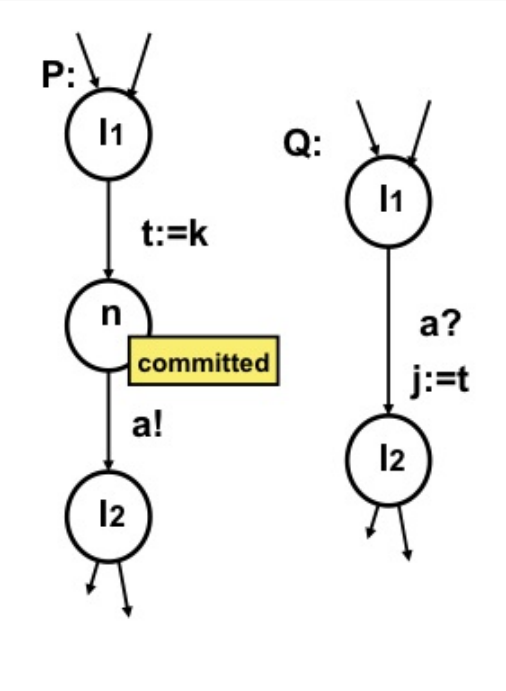
- Edges using urgent channels for synchronisation cannot have time constraints (i.e. clock guards)

# Extension: Urgent Location




- Time cannot progress in an urgent location
- Both models are equivalent,
- but the use of urgent locations reduces the number of clocks in a model and simplifies analysis

# Extension: Committed Location




- Delay is also not allowed and one of the (possibly several) committed locations must be exited in the next instant, i.e. the next transition of the whole system must involve an outgoing edge of at least one of the committed locations
- Our aim is to pass the value $k$ to variable $j$ (via global variable $t$)
- location $n$ is committed to ensure that no other automaton can change $t$ before the assignment $j := t$

# Table of Contents

# Properties: Expression and Satisfaction

**The Satisfaction Problem**

Given a timed automaton $ta$ and a property $\phi$ show that

$$\mathcal{T}(ta) \models \phi$$

- in which logical language shall $\phi$ be specified?
- how is $\models$ defined?

# The Logical Language

A variant of Computation Tree Logic (CTL) with two types of formulae

- state: for describing properties of states in $\mathcal{T}(ta)$
- path: for describing properties of paths in $\mathcal{T}(ta)$

# State Formulae

Clock constraints which are used for guards and invariants and similar constraints over integer variables:

`x >= 8`, `i == 8`, and `x < 2`, ...

Additionally,

- $ta.\ell$, which tests current location: $(\ell, \eta) \models ta.\ell$ provided $(\ell, \eta)$ is a state in $\mathcal{T}(ta)$
- `deadlock`:
  $(\ell, \eta) \models \forall_{d \in \mathbb{R}_{\geq 0}}$. there is no transition from $\langle \ell, \eta + d \rangle$

# Path Formulae

$$\Pi ::= A\Box\,\Psi \mid A\Diamond\,\Psi \mid E\Box\,\Psi \mid E\Diamond\,\Psi \mid \Phi \rightsquigarrow \Psi$$

$$\Psi ::= ta.\ell \mid g_c \mid g_d \mid \texttt{not}\ \Psi \mid \Psi\ \texttt{or}\ \Psi \mid \Psi\ \texttt{and}\ \Psi \mid \Psi\ \texttt{imply}\ \Psi$$

where

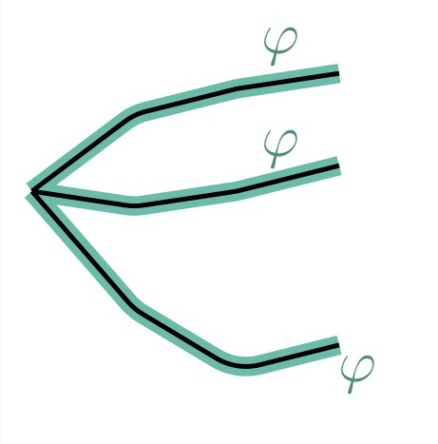
- $A$, $E$ quantify (universally and existentially, resp.) over paths
- $\Box$, $\Diamond$ quantify (universally and existentially, resp.) over states in a path

also notice that

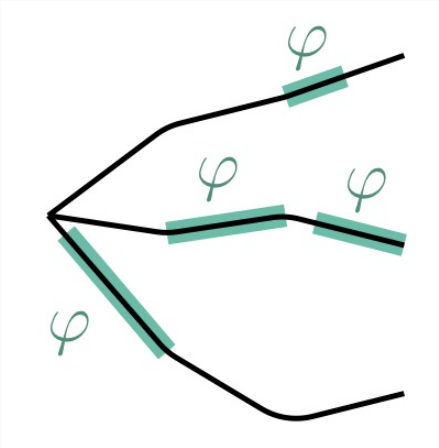
$$\Phi \rightsquigarrow \Psi \stackrel{\text{abv}}{=} A\Box\,(\Phi \Rightarrow A\Diamond\,\Psi)$$

# Path Formulae pt. II

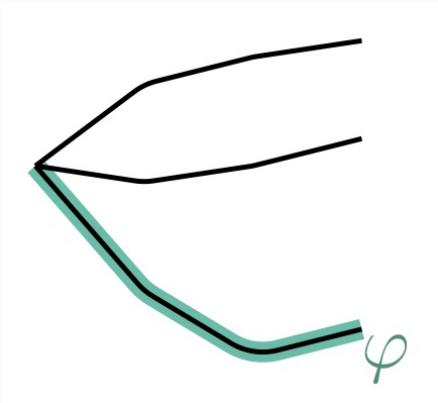
$A\Box\,\varphi$



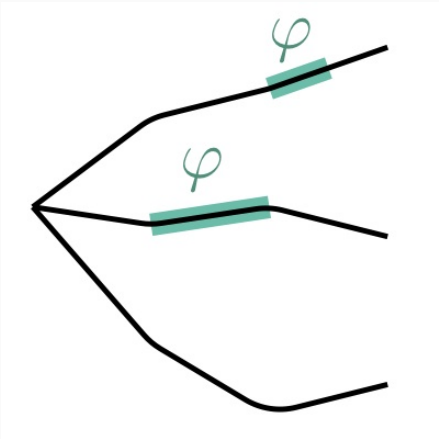

$A\Diamond\,\varphi$




$E\Box\,\varphi$

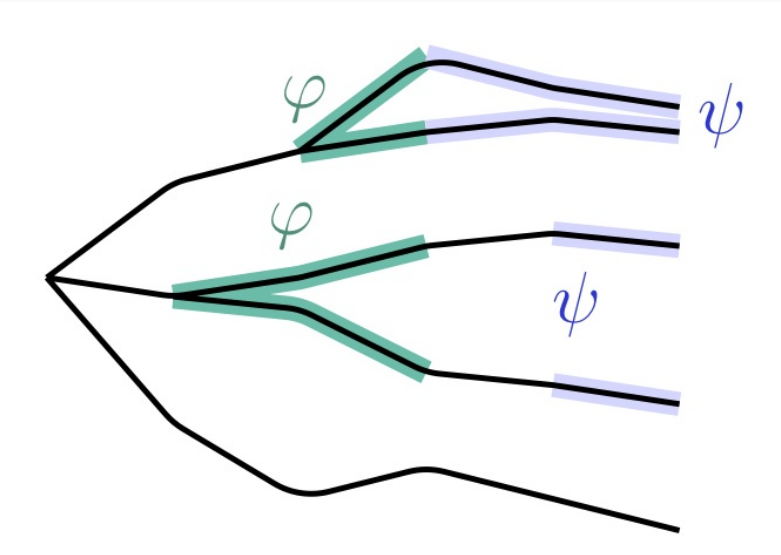



$E\Diamond\,\varphi$



# Path Formulae pt. III

$$\varphi \rightsquigarrow \psi$$




**Example**

If a message is sent, it will eventually be received –

`send(m)` $\rightsquigarrow$ `received(m)`

# Reachability Properties

$E \Diamond \phi$

Is there a path starting at the initial state such that a state formula $\phi$ is eventually satisfied?

Examples:

- Is it possible for a sender to send a message?
- Can a message possibly be received?
- Is it possible to reach a certain location of the automaton?
- ...

# Safety Properties

**$A\Box\,\phi$ and $E\Box\,\phi$**

Something bad will never happen

Example:

- In a nuclear power plant will the temperature of the core be always under a certain threshold?
- Will we ever reach a state with a deadlock?

# Liveness Properties

**$A\Diamond\,\phi$ and $\phi \rightsquigarrow \psi$**

Something good will *eventually happen*

or if *something* happens then *something else* will eventually happen

Examples:

- Always when pressing the on button the television will eventually turn on
- In a communication protocol any message that has been sent should eventually be received

# Exercises

Write the sentences below in CTL

1. The system never enters in deadlock
2. The location $\ell$ is reachable
3. In all executions we reach location $\ell$
4. If we reach location $\ell$ we will inevitably reach location $s$
5. There exists at least one execution where variable `i` is always below or equal 10
6. The two philosophers never eat at the same time