Temporal Logics

Radek Mařík

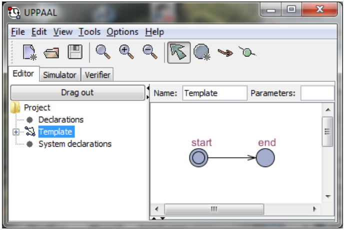
Czech Technical University
Faculty of Electrical Engineering
Department of Telecommunication Engineering
Prague CZ

November 23, 2020

Outline

Outline

Automaton Creation [UPP09]



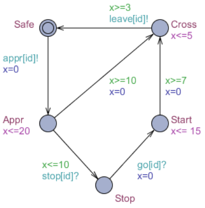
Automaton

- Starting position (double circle)
- "Add Location" to add a position
- "Selection Tool" for naming the position
- "Add Edge" to add an edge, bend the edges with the mouse around the ends
- the lower table "Position" and "Description" for error analysis

System Composition [UPP09]

System

- **System** . . . a network of parallel timed automata (processes).
- **Process** . . . an instance of a parameterized pattern.



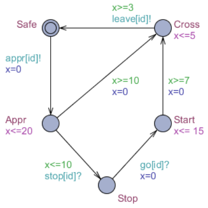
Process

- **Position** . . .
  - name,
  - invariants
- **Edges** . . .
  - guard conditions ($x >= 7$),
  - synchronization ($go[id]?$),
  - assignment ($x = 0$),

System Composition [UPP09]

System

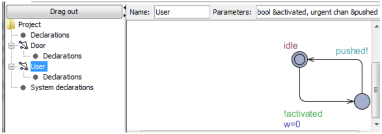
- **System** . . . a network of parallel timed automata (processes).
- **Process** . . . an instance of a parameterized pattern.

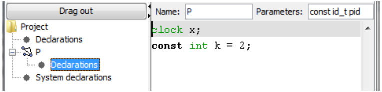


Process

- **Position** . . .
  - name,
  - invariants
- **Edges** . . .
  - guard conditions ($x >= 7$),
  - synchronization ($go[id]?$),
  - assignment ($x = 0$),

(Automaton) Template Description [UPP09]





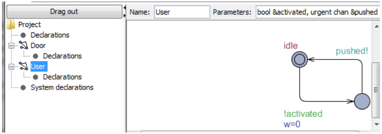
Parameterized timed automaton

- name,
- parameters,

Local declarations

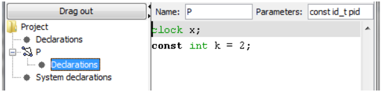
- variables,
- synchronization channels,
- constants

(Automaton) Template Description [UPP09]



Parameterized timed automaton

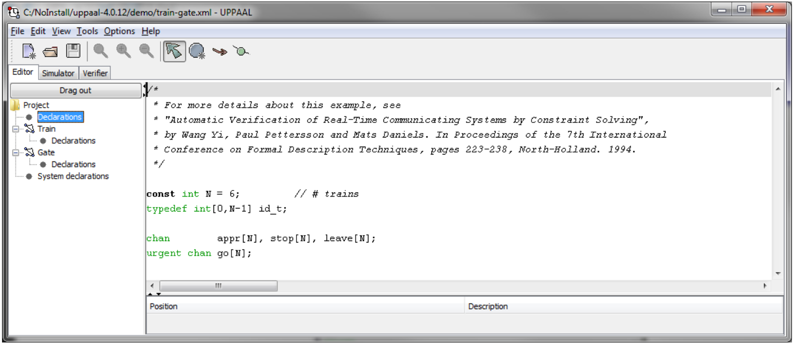
- name,
- parameters,



Local declarations

- variables,
- synchronization channels,
- constants

System Description [UPP09]



Global Declarations

- global integer variables,
- global clock,
- synchronization channels,
- constants

System Definitions [UPP09]

```
bool activated1, activated2;
urgent chan pushed1, pushed2;
urgent chan closed1, closed2;

Door1 = Door(activated1, pushed1, closed1, closed2);
Door2 = Door(activated2, pushed2, closed2, closed1);
User1 = User(activated1, pushed1);
User2 = User(activated2, pushed2);

system Door1, Door2, User1, User2;
```

Process Assignment

- a process instance declaration,
- patterns with fully/partially specified parameters,

System Definition

- a list of system processes,

System Definitions [UPP09]

```
bool activated1, activated2;
urgent chan pushed1, pushed2;
urgent chan closed1, closed2;

Door1 = Door(activated1, pushed1, closed1, closed2);
Door2 = Door(activated2, pushed2, closed2, closed1);
User1 = User(activated1, pushed1);
User2 = User(activated2, pushed2);

system Door1, Door2, User1, User2;
```

Process Assignment

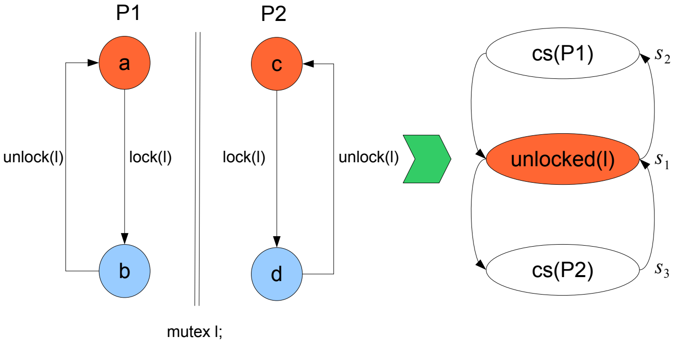
- a process instance declaration,
- patterns with fully/partially specified parameters,

System Definition

- a list of system processes,

Outline

Transitions between Configurations in Kripke's structure [Voj10]



Path in Kripke's structure [Voj10]

Path

- **Path** π ... in Kripke's structure M is an infinite sequence of states $\pi = s_0 s_1 s_3 \ldots$ such that, $\forall i \in N..R(s_i, s_{i+1})$.
- $\Pi(M, s)$ ... a set of all paths in M that start in v $s \in S$
- Suffix π^i of the path $\pi = s_0 s_1 s_3 \ldots s_i s_{i+1} s_{i+2}$ is a the path $\pi^i = s_i s_{i+1} s_{i+2}$ starting in s_i.
- $s_i = \pi[i]$

Concept of Time [Voj10]

Time Abstraction

- **Logical time** . . . works with *(partial) ordering* of states/events in system behavior.
- **Physical time** . . . *measurement* of time elapsed between two states/events.

Time in Model Verification

- **Linear time** . . . allows you to express only about a given *linear path* in state space.
  - On all paths, x must be followed by y.
  - On all paths, x must be followed by y or z.
- **Branching time** . . . allows to quantify (existentially and universally) possible futures starting with a given state. The state space is observed as an expanded *infinite tree*.
  - There is a path where the following next state is x.

Concept of Time [Voj10]

Time Abstraction

- **Logical time** . . . works with *(partial) ordering* of states/events in system behavior.
- **Physical time** . . . *measurement* of time elapsed between two states/events.

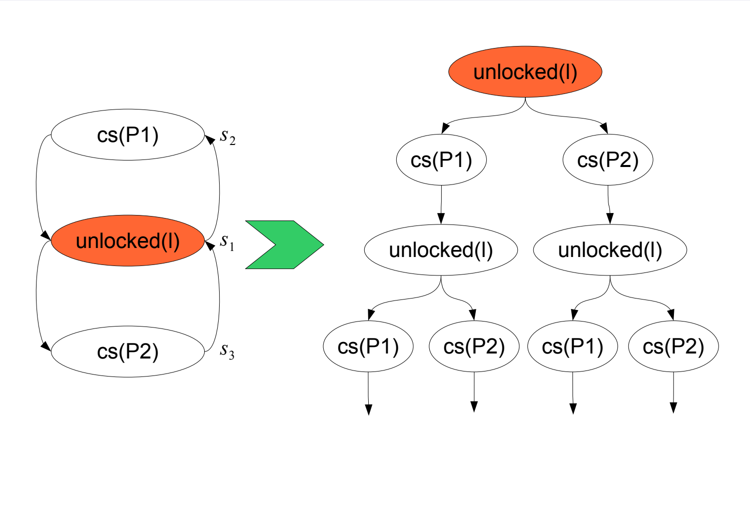
Time in Model Verification

- **Linear time** . . . allows you to express only about a given *linear path* in state space.
  - On all paths, x must be followed by y.
  - On all paths, x must be followed by y or z.
- **Branching time** . . . allows to quantify (existentially and universally) possible futures starting with a given state. The state space is observed as an expanded *infinite tree*.
  - There is a path where the following next state is x.

Outline

Computation Tree [Voj10]

Describes the properties of the processing progress.



CTL* Formula [Voj10]

Consists of

- atomic statements
- logical connectors
- path quantifiers
- temporal operators

CTL* Quantifiers and Operators [Wik10, Voj10]

Path Quantifiers

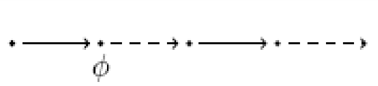
describe the branching structure of a computation tree

- E . . . there exists a processing path from the given state.
- A . . . for all processing paths from the given state.

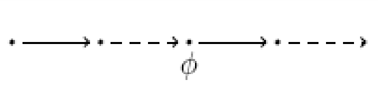
Temporal Operators

determine the properties of a given path in the computation tree

- $X\varphi$ (next time, $\bigcirc$). . . the property φ is fulfilled in the second (next) state of the path.



- $F\varphi$ (in future, $\Diamond$). . . the property φ is valid in a state of the path.



CTL* Quantifiers and Operators [Wik10, Voj10]

Path Quantifiers

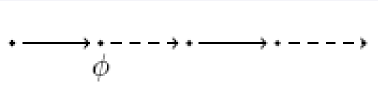
describe the branching structure of a computation tree

- E ... there exists a processing path from the given state.
- A ... for all processing paths from the given state.

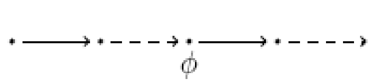
Temporal Operators

determine the properties of a given path in the computation tree

- $X\varphi$ (next time, $\bigcirc$)... the property φ is fulfilled in the second (next) state of the path.



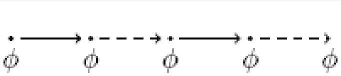
- $F\varphi$ (in future, $\Diamond$)... the property φ is valid in a state of the path.



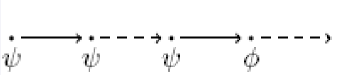
CTL* Operators [Wik10, Voj10]

Temporal Operators

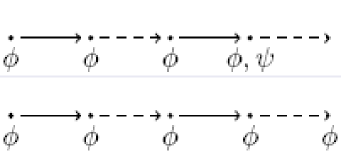
- $G\varphi$ (globally, $\Box$)... The property φ is satisfied in all states of the given path.



- $\psi U \varphi$ (until)... The property φ is valid in some path state, and the property ψ is valid at least in all previous states of this path.



- $\psi R \varphi$ (release)... The property φ must be valid until (and including) the state when the ψ property becomes satisfied, if such a state exists.



CTL* Syntax [Voj10]

Let AP be a nonempty set of atomic propositions.

Syntax of state formulas that are true in a given state

- If $p \in AP$, then p is a state formula.
- If φ a ψ are state formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$ are state fomulae.
- If φ is a path formula, potom $E\varphi$ a $A\varphi$ are state formulae.

Syntax of path formulae that are true in states along a specific path

- If φ is a state formula, then φ is also a path formula.
- If φ and ψ are path formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U \psi$ a $\varphi R \psi$ are path formulae.

CTL* is the set of state formulae generated by the above rules.

CTL* Syntax [Voj10]

Let AP be a nonempty set of atomic propositions.

Syntax of state formulas that are true in a given state

- If $p \in AP$, then p is a state formula.
- If φ a ψ are state formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$ are state fomulae.
- If φ is a path formula, potom $E\varphi$ a $A\varphi$ are state formulae.

Syntax of path formulae that are true in states along a specific path

- If φ is a state formula, then φ is also a path formula.
- If φ and ψ are path formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U \psi$ a $\varphi R \psi$ are path formulae.

CTL* is the set of state formulae generated by the above rules.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $M, s \models p$ iff $p \in L(s)$.
  - $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.
  - $M, s \models \varphi_1 \vee \varphi_2$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $M, s \models \varphi_1 \wedge \varphi_2$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $M, s \models E\psi_1$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $M, s \models A\psi_1$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $M, s \models p$ iff $p \in L(s)$.
  - $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.
  - $M, s \models \varphi_1 \vee \varphi_2$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $M, s \models \varphi_1 \wedge \varphi_2$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $M, s \models E\psi_1$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $M, s \models A\psi_1$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $M, s \models p$ iff $p \in L(s)$.
  - $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.
  - $M, s \models \varphi_1 \vee \varphi_2$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $M, s \models \varphi_1 \wedge \varphi_2$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $M, s \models E\psi_1$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $M, s \models A\psi_1$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg\varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg\varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s).M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s).M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg\varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s).M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s).M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg\varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg\varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg\varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Semantics [Voj10]

- Let be Kripke's structure $M = (S, T, \mathcal{I}, s_0, L)$ over a set of atomic propositions AP.
- For the state formula φ over AP, we denote $M, s \models \varphi$ the fact, that φ is satisfied in $s \in S$.
- For the path formula φ over AP, we denote $M, \pi \models \varphi$ the fact, that φ is satisfied along the path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 are state formulae over AP, $p \in AP$, and ψ_1, ψ_2 are path formulae over AP.
  Then, we define a relation $\models$ inductively as follows:
  - $\boldsymbol{M, s \models p}$ iff $p \in L(s)$.
  - $\boldsymbol{M, s \models \neg\varphi_1}$ iff $M, s \not\models \varphi_1$.
  - $\boldsymbol{M, s \models \varphi_1 \vee \varphi_2}$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models \varphi_1 \wedge \varphi_2}$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $\boldsymbol{M, s \models E\psi_1}$ iff $\exists \pi \in \Pi(M, s). M, s \models \psi_1$.
  - $\boldsymbol{M, s \models A\psi_1}$ iff $\forall \pi \in \Pi(M, s). M, s \models \psi_1$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
  - $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $M, \pi \models X\psi_1$ iff $M, \pi^1 \models \psi_1$.
  - $M, \pi \models F\psi_1$ iff $\exists i \geq 0.M, \pi^i \models \psi_1$.
  - $M, \pi \models G\psi_1$ iff $\forall i \geq 0.M, \pi^i \models \psi_1$.
  - $M, \pi \models \psi_1 U \psi_2$ iff $\exists i \geq 0.M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i.M, \pi^j \models \psi_1$.
  - $M, \pi \models \psi_1 R \psi_2$ iff $\forall i \geq 0.(\forall 0 \leq j < i.M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi} \models \boldsymbol{\varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi} \models \neg\boldsymbol{\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi} \models \boldsymbol{\psi_1} \vee \boldsymbol{\psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi} \models \boldsymbol{\psi_1} \wedge \boldsymbol{\psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi} \models \boldsymbol{X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi} \models \boldsymbol{F\psi_1}$ iff $\exists i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi} \models \boldsymbol{G\psi_1}$ iff $\forall i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi} \models \boldsymbol{\psi_1 U \psi_2}$ iff $\exists i \geq 0. M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i. M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi} \models \boldsymbol{\psi_1 R \psi_2}$ iff $\forall i \geq 0. (\forall 0 \leq j < i. M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0. M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i. M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0. (\forall 0 \leq j < i. M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0. M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i. M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0. (\forall 0 \leq j < i. M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0.M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0.M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0.M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i.M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0.(\forall 0 \leq j < i.M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0. M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i. M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0. (\forall 0 \leq j < i. M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0. M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i. M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0. (\forall 0 \leq j < i. M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0. M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i. M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0. (\forall 0 \leq j < i. M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0. M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0. M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i. M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0. (\forall 0 \leq j < i. M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Path Semantics [Voj10]

- To continue defining the relation $\models$:
  - $\boldsymbol{M, \pi \models \varphi_1}$ iff $M, s_0 \models \varphi_1, s_0 = \pi[0]$.
  - $\boldsymbol{M, \pi \models \neg\psi_1}$ iff $M, \pi \not\models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 \vee \psi_2}$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models \psi_1 \wedge \psi_2}$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
  - $\boldsymbol{M, \pi \models X\psi_1}$ iff $M, \pi^1 \models \psi_1$.
  - $\boldsymbol{M, \pi \models F\psi_1}$ iff $\exists i \geq 0.M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models G\psi_1}$ iff $\forall i \geq 0.M, \pi^i \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 U \psi_2}$ iff $\exists i \geq 0.M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i.M, \pi^j \models \psi_1$.
  - $\boldsymbol{M, \pi \models \psi_1 R \psi_2}$ iff $\forall i \geq 0.(\forall 0 \leq j < i.M, \pi^j \not\models \psi_1 \Rightarrow M, \pi^i \models \psi_2$.

CTL* Basic Operators [Voj10]

- All CTL* operators can be derived from $\vee$, $\neg$, X, U, and E:
  - Let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$)
  - $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$,
  - $F\varphi \equiv trueU\varphi$,
  - $G\varphi \equiv \neg F \neg \varphi$,
  - $\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$,
  - $A\varphi \equiv \neg E \neg \varphi$.

CTL* Basic Operators [Voj10]

- All CTL* operators can be derived from $\vee$, $\neg$, X, U, and E:
  - Let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$)
  - $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$,
  - $F\varphi \equiv true U \varphi$,
  - $G\varphi \equiv \neg F \neg\varphi$,
  - $\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$,
  - $A\varphi \equiv \neg E \neg\varphi$.

CTL* Basic Operators [Voj10]

- All CTL* operators can be derived from $\vee$, $\neg$, X, U, and E:
  - Let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$)
  - $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$,
  - $F\varphi \equiv true U \varphi$,
  - $G\varphi \equiv \neg F \neg\varphi$,
  - $\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$,
  - $A\varphi \equiv \neg E \neg\varphi$.

CTL* Basic Operators [Voj10]

- All CTL* operators can be derived from $\vee$, $\neg$, X, U, and E:
  - Let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$)
  - $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$,
  - $F\varphi \equiv true U \varphi$,
  - $G\varphi \equiv \neg F \neg \varphi$,
  - $\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$,
  - $A\varphi \equiv \neg E \neg \varphi$.

CTL* Basic Operators [Voj10]

- All CTL* operators can be derived from $\vee$, $\neg$, X, U, and E:
  - Let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$)
  - $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$,
  - $F\varphi \equiv true U \varphi$,
  - $G\varphi \equiv \neg F \neg \varphi$,
  - $\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$,
  - $A\varphi \equiv \neg E \neg \varphi$.

CTL* Basic Operators [Voj10]

- All CTL* operators can be derived from $\vee$, $\neg$, X, U, and E:
  - Let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$)
  - $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$,
  - $F\varphi \equiv true U \varphi$,
  - $G\varphi \equiv \neg F \neg \varphi$,
  - $\varphi R \psi \equiv \neg(\neg\varphi U \neg\psi)$,
  - $A\varphi \equiv \neg E \neg \varphi$.

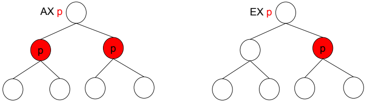
Outline

CTL Syntax [Voj10]

- CTL is a sublogic of CTL*
  - path formulae are limited to $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U\psi$, and $\varphi R\psi$,
  - where φ and ψ are state formulae.
- Therefore only 10 modal CTL operators:
  - AX and EX
  - AF and EF
  - AG and EG

CTL Syntax [Voj10]

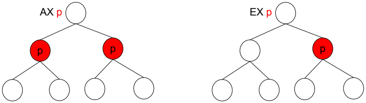
- CTL is a sublogic of CTL*
  - path formulae are limited to $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U\psi$, and $\varphi R\psi$,
  - where φ and ψ are state formulae.
- Therefore only 10 modal CTL operators:
  - AX and EX



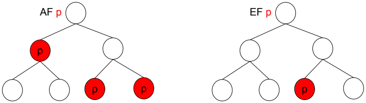
  - AF and EF
  - AG and EG

CTL Syntax [Voj10]

- CTL is a sublogic of CTL*
  - path formulae are limited to $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U \psi$, and $\varphi R \psi$,
  - where φ and ψ are state formulae.
- Therefore only 10 modal CTL operators:
  - AX and EX



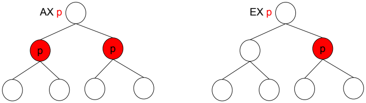
  - AF and EF



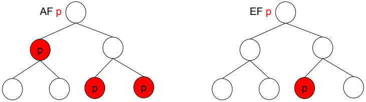
  - AG and EG

CTL Syntax [Voj10]

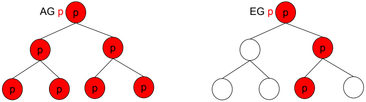
- CTL is a sublogic of CTL*
  - path formulae are limited to $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U\psi$, and $\varphi R\psi$,
  - where φ and ψ are state formulae.
- Therefore only 10 modal CTL operators:
  - AX and EX



  - AF and EF



  - AG and EG

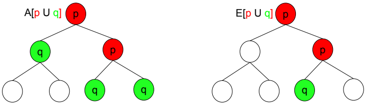


CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU
  - AR and ER
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\varphi \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\varphi \equiv E[trueU\varphi]$
  - $\boldsymbol{AG}\varphi \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\varphi \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\varphi \boldsymbol{U}\psi] \equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A}[\varphi \boldsymbol{R}\psi] \equiv \neg E[\neg\varphi U\neg\psi]$
  - $\boldsymbol{E}[\varphi \boldsymbol{R}\psi] \equiv \neg A[\neg\varphi U\neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU

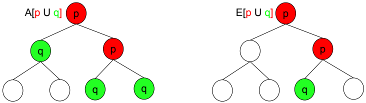


  - AR and ER

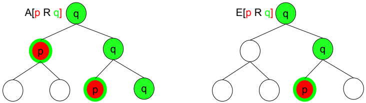
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\varphi \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\varphi \equiv E[trueU\varphi]$
  - $\boldsymbol{AG}\varphi \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\varphi \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\varphi \boldsymbol{U} \psi] \equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A}[\varphi \boldsymbol{R} \psi] \equiv \neg E[\neg\varphi U \neg\psi]$
  - $\boldsymbol{E}[\varphi \boldsymbol{R} \psi] \equiv \neg A[\neg\varphi U \neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



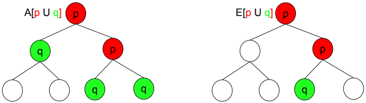
  - AR and ER



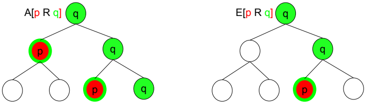
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $AX\varphi \equiv \neg EX\neg\varphi$
  - $EF\varphi \equiv E[trueU\varphi]$
  - $AG\varphi \equiv \neg EF\neg\varphi$
  - $AF\varphi \equiv \neg EG\neg\varphi$
  - $A[\varphi U\psi]$ $\equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $A[\varphi R\psi] \equiv \neg E[\neg\varphi U\neg\psi]$
  - $E[\varphi R\psi] \equiv \neg A[\neg\varphi U\neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



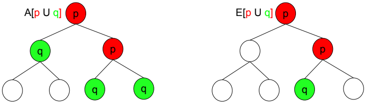
  - AR and ER



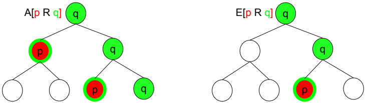
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\varphi \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\varphi \equiv E[trueU\varphi]$
  - $\boldsymbol{AG}\varphi \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\varphi \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\varphi \boldsymbol{U} \psi] \equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A}[\varphi \boldsymbol{R} \psi] \equiv \neg E[\neg\varphi U\neg\psi]$
  - $\boldsymbol{E}[\varphi \boldsymbol{R} \psi] \equiv \neg A[\neg\varphi U\neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



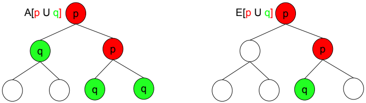
  - AR and ER



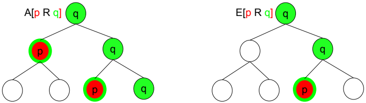
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\varphi \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\varphi \equiv E[true U \varphi]$
  - $\boldsymbol{AG}\varphi \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\varphi \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\varphi \boldsymbol{U} \psi] \equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A}[\varphi \boldsymbol{R} \psi] \equiv \neg E[\neg\varphi U \neg\psi]$
  - $\boldsymbol{E}[\varphi \boldsymbol{R} \psi] \equiv \neg A[\neg\varphi U \neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



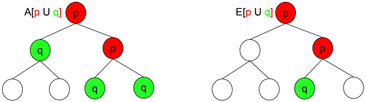
  - AR and ER



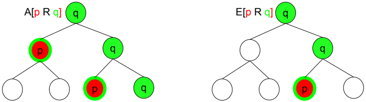
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\varphi \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\varphi \equiv E[trueU\varphi]$
  - $\boldsymbol{AG}\varphi \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\varphi \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\varphi \boldsymbol{U} \psi] \equiv \neg E[\neg\psi U(\neg\varphi \land \neg\psi)] \land \neg EG\neg\psi$
  - $\boldsymbol{A}[\varphi \boldsymbol{R} \psi] \equiv \neg E[\neg\varphi U \neg\psi]$
  - $\boldsymbol{E}[\varphi \boldsymbol{R} \psi] \equiv \neg A[\neg\varphi U \neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



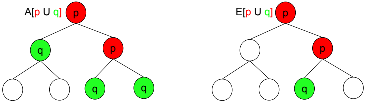
  - AR and ER



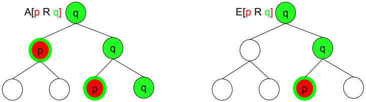
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX\varphi} \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF\varphi} \equiv E[trueU\varphi]$
  - $\boldsymbol{AG\varphi} \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF\varphi} \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A[\varphi U\psi]}$ $\equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A[\varphi R\psi]} \equiv \neg E[\neg\varphi U\neg\psi]$
  - $\boldsymbol{E[\varphi R\psi]} \equiv \neg A[\neg\varphi U\neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



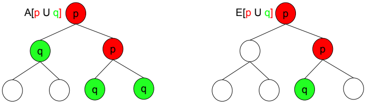
  - AR and ER



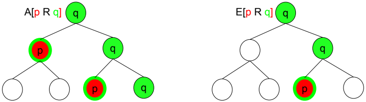
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\varphi \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\varphi \equiv E[trueU\varphi]$
  - $\boldsymbol{AG}\varphi \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\varphi \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\varphi \boldsymbol{U}\psi]$ $\equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A}[\varphi \boldsymbol{R}\psi] \equiv \neg E[\neg\varphi U\neg\psi]$
  - $\boldsymbol{E}[\varphi \boldsymbol{R}\psi] \equiv \neg A[\neg\varphi U\neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



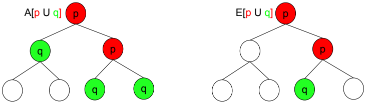
  - AR and ER



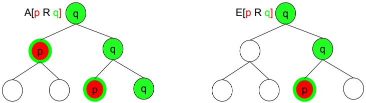
- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\varphi \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\varphi \equiv E[true U \varphi]$
  - $\boldsymbol{AG}\varphi \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\varphi \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\varphi \boldsymbol{U} \psi]$ $\equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A}[\varphi \boldsymbol{R} \psi] \equiv \neg E[\neg\varphi U \neg\psi]$
  - $\boldsymbol{E}[\varphi \boldsymbol{R} \psi] \equiv \neg A[\neg\varphi U \neg\psi]$

CTL Modal Operators [Voj10]

- Modal CTL operators:
  - AU and EU



  - AR and ER



- There are 3 basic CTL modal operators - EX, EG, and EU:
  - $\boldsymbol{AX}\boldsymbol{\varphi} \equiv \neg EX\neg\varphi$
  - $\boldsymbol{EF}\boldsymbol{\varphi} \equiv E[true U\varphi]$
  - $\boldsymbol{AG}\boldsymbol{\varphi} \equiv \neg EF\neg\varphi$
  - $\boldsymbol{AF}\boldsymbol{\varphi} \equiv \neg EG\neg\varphi$
  - $\boldsymbol{A}[\boldsymbol{\varphi U \psi}] \equiv \neg E[\neg\psi U(\neg\varphi \wedge \neg\psi)] \wedge \neg EG\neg\psi$
  - $\boldsymbol{A}[\boldsymbol{\varphi R \psi}] \equiv \neg E[\neg\varphi U\neg\psi]$
  - $\boldsymbol{E}[\boldsymbol{\varphi R \psi}] \equiv \neg A[\neg\varphi U\neg\psi]$

Outline

LTL Syntax [Voj10]

- LTL is a sublogic of CTL*
  - It only allows formulas of the form $A\varphi$, in which state subformulae are atomic propositions.
  - LTL formula is created according to the following grammar:
    - $\varphi ::= A\psi$ (A is often omitted)
    - $\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid X\psi \mid F\psi \mid G\psi \mid \psi U \psi \mid \psi R \psi$,
    - where $p \in AP$.
  - LTL provides expressions about specific paths in a given Kripke's structure
    - i.e. ignores branching

LTL, CTL, CTL* [Voj10]

- LTL and CTL cannot be compared:
  - For example, CTL cannot express the LTL formula $A(FGp)$
  - For example, LTL cannot express the CTL formula $AG(EFp)$
- CTL* covers both LTL and CTL
  - disjunction $(A(FGp)) \vee (AG(EFp))$ cannot be expressed in either LTL or CTL.

Outline

BNF grammar of specification language [UPP10]

BNF grammar

- $A[\,]Expression$
- $E <> Expression$
- $E[\,]Expression$
- $A <> Expression$
- $Expression --> Expression$

Notes

- No expression can have side effects.
- The expression $process.location$ tests whether a certain process is in a given position.

BNF grammar of specification language [UPP10]

BNF grammar

- $A[]Expression$
- $E <> Expression$
- $E[]Expression$
- $A <> Expression$
- $Expression --> Expression$

Notes

- No expression can have side effects.
- The expression $process.location$ tests whether a certain process is in a given position.

Examples of Specification Language [UPP10]

BNF grammar

- $A[]1 < 2$
  - Invariantly $1 < 2$
- $E <> p1.cs and p2.cs$
  - True, if the system can reach a state in which processes $p1$ and $p2$ are in their position cs
- $A <> p1.cs imply not p2.cs$
  - Invariantly process $p1$ in position cs implies that process $p2$ **is not** in position cs.
- $A[] not deadlock$
  - Invariantly, the process does not contain a deadlock.

Examples of Specification Language [UPP10]

BNF grammar

- $A[]1 < 2$
  - Invariantly $1 < 2$
- $E <> p1.cs and p2.cs$
  - True, if the system can reach a state in which processes $p1$ and $p2$ are in their position cs
- $A <> p1.cs imply not p2.cs$
  - Invariantly process $p1$ in position cs implies that process $p2$ **is not** in position cs.
- $A[] not deadlock$
  - Invariantly, the process does not contain a deadlock.

Examples of Specification Language [UPP10]

BNF grammar

- $A[]1 < 2$
  - Invariantly $1 < 2$
- $E <> p1.cs and p2.cs$
  - True, if the system can reach a state in which processes $p1$ and $p2$ are in their position cs
- $A <> p1.cs imply not p2.cs$
  - Invariantly process $p1$ in position cs implies that process $p2$ **is not** in position cs.
- $A[] not deadlock$
  - Invariantly, the process does not contain a deadlock.

Examples of Specification Language [UPP10]

BNF grammar

- $A[]1 < 2$
  - Invariantly $1 < 2$
- $E <> p1.cs and p2.cs$
  - True, if the system can reach a state in which processes $p1$ and $p2$ are in their position cs
- $A <> p1.cs imply not p2.cs$
  - Invariantly process $p1$ in position cs implies that process $p2$ **is not** in position cs.
- $A[] not deadlock$
  - Invariantly, the process does not contain a deadlock.

Outline

Conditions over clocks [BDL05]

- C ...clock set
- $B(C)$ ...a set of conjunctions over simple conditions of type
  - $x \bowtie c$
  - $x - y \bowtie c$
  - where
    - $x, y \in C$,
    - $c \in \mathbb{N}$,
    - $\bowtie \in \{<, \leq, =, \geq, >\}$

Query Language [BDL05]

- **State formulae** . . . describe individual states.
- **Path formulae** . . . are evaluated along model paths and traces.
  - reachability,
  - safety,
  - liveness.

State Formulae [BDL05]

- an expression that can be evaluated for a given state without having to analyze the behavior of the model.
- a superset of guards, i.e. it has no side effect,
- unlike guards, the use of disjunctions is not limited.
- Test whether the process is in the given position . . . $P.\ell$
  - P . . . process
  - ℓ . . . position
- **deadlock** . . .
  - a special state formula, which is fulfilled for all blocked states,
  - A state is blocked if there is no action transition from that state or any delayed state successor.

Outline

Reachability [BDL05]

- the simplest feature,
- asks if there is a possibility that the given state formula φ is satisfied in every reachable state.
- i.e. there is a path from the initial state such that φ will be fulfilled once along this path.
- check the basic properties of the model
  - that at least the basic behavior can be achieved,
  - an example of a communication protocol with one transmitter and one receiver
    - it is possible to send a message by transmitter at all.
    - The message is eventually received by the receiver.
- in UPPAAL: `E<>`φ

Safety [BDL05]

- anything bad will never happen
- an example of a nuclear power plant model
  - the operating temperature is always (invariantly) below a certain threshold,
  - the container will never melt
- a variant: something is not possible to happen at all
- an example of playing a game
  - The safe state is that if we can still play the game, then there is no way to lose it.
- in UPPAAL:
  - is formulated positively
  - let φ be a state formula
  - `A[]` $\varphi \equiv \neg E \Diamond \neg \varphi$ . . . φ should be true in all reachable states
  - `E[]` φ . . . there is a path along which φ is always true

Liveness [BDL05]

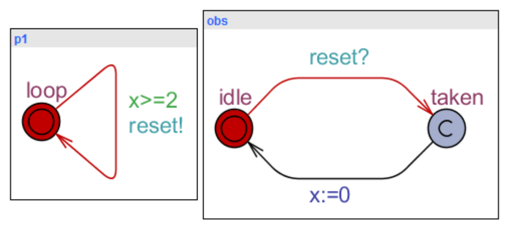
- something will eventually happen one day
- examples
  - pressing the *on* button on the remote control will cause the TV to turn on once.
  - in the communication protocol model:
    any message sent will be received eventually.
- in UPPAAL:
  - `A<>`$\varphi \equiv \neg E\Box\neg\varphi$ ... φ will always be fulfilled eventually,
  - φ `-->` $\psi \equiv A\Box(\varphi \Rightarrow A\Diamond\psi)$ ...
    whenever φ is fulfilled, then ψ will be met eventually.

Outline

Observer [BDL05]

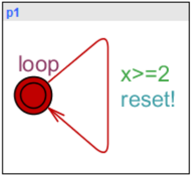
- an additional automaton
- detects events without having to change the model itself.

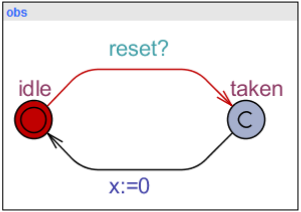
Example



- clock reset detection
- extra clock reset (x:=0)

Initial Variant of Example [BDL05]





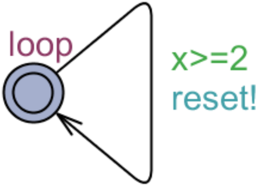

```
// Place global declarations here.
clock x;
chan reset;
```

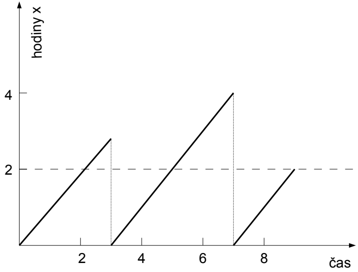
```
// Place template instantiations here.
p1 = P1();
obs = Obs();

// List one or more processes to be comp
system p1, obs;
```

- The goal is to stay in position if the (invariant) condition on the clock applies and then leave the position.
- Option 1: no invariant

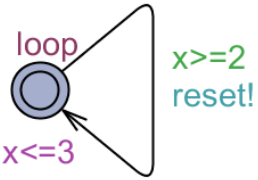
1. Variant of the Example [BDL05]

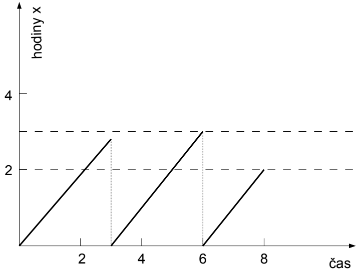




- The goal is to stay in position if the (invariant) condition on the clock applies and then leave the position.
- Option 1: no invariant
- `A[] obs.taken imply x>=2`
- `E<> obs.idle and x>3`

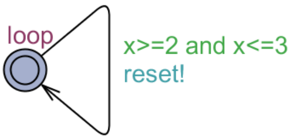
2. Variant of the Example [BDL05]

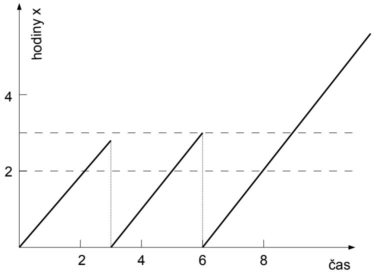




- The goal is to stay in position if the (invariant) condition on the clock applies and then leave the position.
- Option 2: with invariant
- `A[] obs.taken imply (x>=2 and x<=3)`
- `E<> obs.idle and x>2`
  - `E<> obs.idle and x>3` . . . is not satisfied
- `A[] obs.idle imply x<=3`

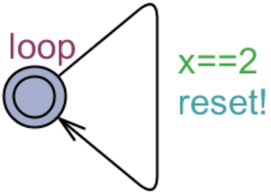
3. Variant of the Example [BDL05]

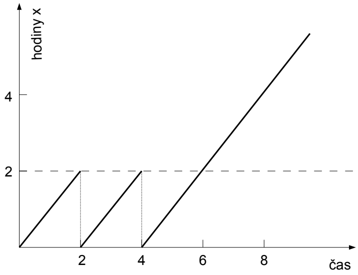




- The goal is to stay in position if the (invariant) condition on the clock applies and then leave the position.
- Option 3: without invariant with guards
- `A[] x>3 imply not obs.taken` ...deadlock occurs
- `A[] not deadlock` ...is not satisfied

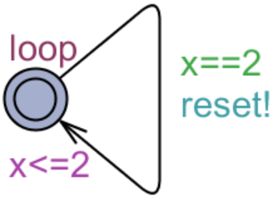
4. Variant of the Example [BDL05]

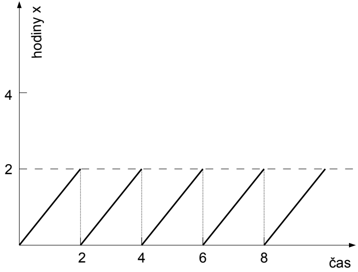




- The goal is to stay in position if the (invariant) condition on the clock applies and then leave the position.
- Option 4: without invariant with equality guards
- `A[] x>2 imply not obs.taken` ...deadlock occurs
- `A[] not deadlock` ...is not satisfied

5. Variant of the Example [BDL05]

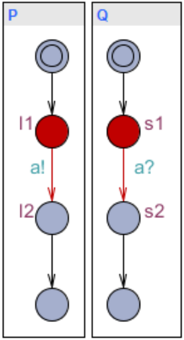




- The goal is to stay in position if the (invariant) condition on the clock applies and then leave the position.
- Option 5: with invariant and with equality guards
- `A[] obs.taken imply x==2`
- `E<> obs.idle and x>2` . . . is not satisfied
- `A[] obs.idle imply x<=2`

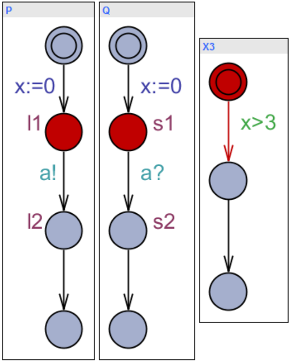
Outline

Example 1, processes P, Q [Dav05]



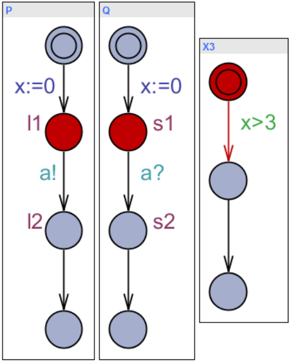
- The goal is to make the sync transition as soon as possible.
- i.e. as soon as both P and Q automata are ready (simultaneously in positions l_1 and s_1).
- How to choose a model when they get into positions at different time instants?

Example 1, processes $P, Q, X3$ [Dav05]



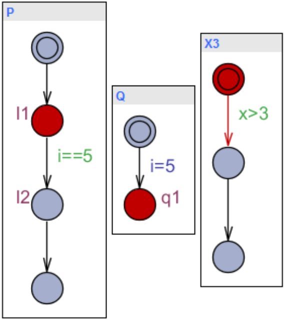
- The goal is to make the sync transition as soon as possible.
- i.e. as soon as both P and Q automata are ready (simultaneously in positions l_1 and s_1).
- How to choose a model when they get into positions at different time instants?
- **Solution:** `urgent chan a`

Example 1, processes $P, Q, X3$ [Dav05]



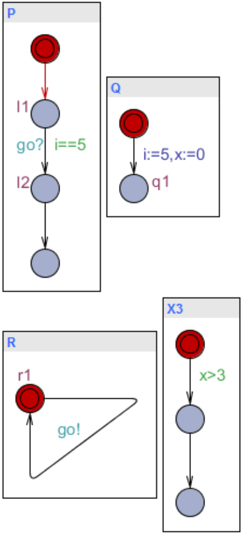
- The goal is to make the sync transition as soon as possible.
- i.e. as soon as both P and Q automata are ready (simultaneously in positions l_1 and s_1).
- How to choose a model when they get into positions at different time instants?
- **Solution:** `urgent chan a`

Example 2, processes $P, Q, X3$ [Dav05]



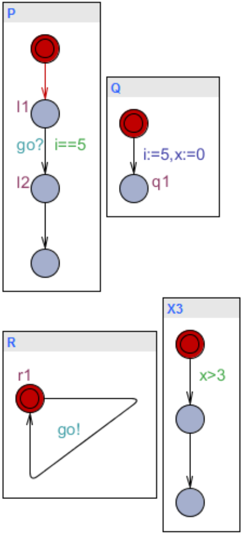
- The goal is to make a transition with the condition $i == 5$ once it is met..

Example 2, processes $P, Q, R, X3$ [Dav05]



- The goal is to make the transition with the condition $i == 5$ as soon as possible.
- i.e. as soon as both P and Q automata are ready (simultaneously in positions l_1 and s_1).
- How to choose a model when they get into positions at different time instants?
- **Solution:**
- `urgent chan go`
- another process that emits an action go!

Example 2, processes $P, Q, R, X3$ [Dav05]

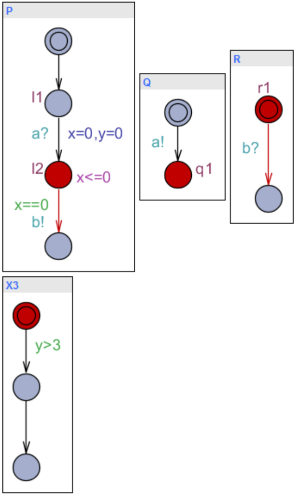


- The goal is to make the transition with the condition $i == 5$ as soon as possible.
- i.e. as soon as both P and Q automata are ready (simultaneously in positions l_1 and s_1).
- How to choose a model when they get into positions at different time instants?
- **Solution:**
- `urgent chan go`
- another process that emits an action go!

Urgent Channels [Dav05]

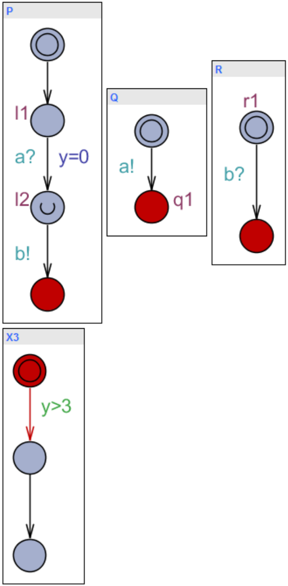
- `urgent chan hurry`
- **Semantics:**
- There is no delay if an edge with an urgent action can be executed.
-
- **Restrictions:**
  - No clock guards are allowed on the edges of the urgent action.
  - Invariants and guards on data variables are allowed.

Urgent Position using Clocks [Dav05]



- Suppose we model a simple system M that accepts packages on channel a and immediately sends them to channel b
- P_1 models the system using the clock x

Urgent Position [Dav05]



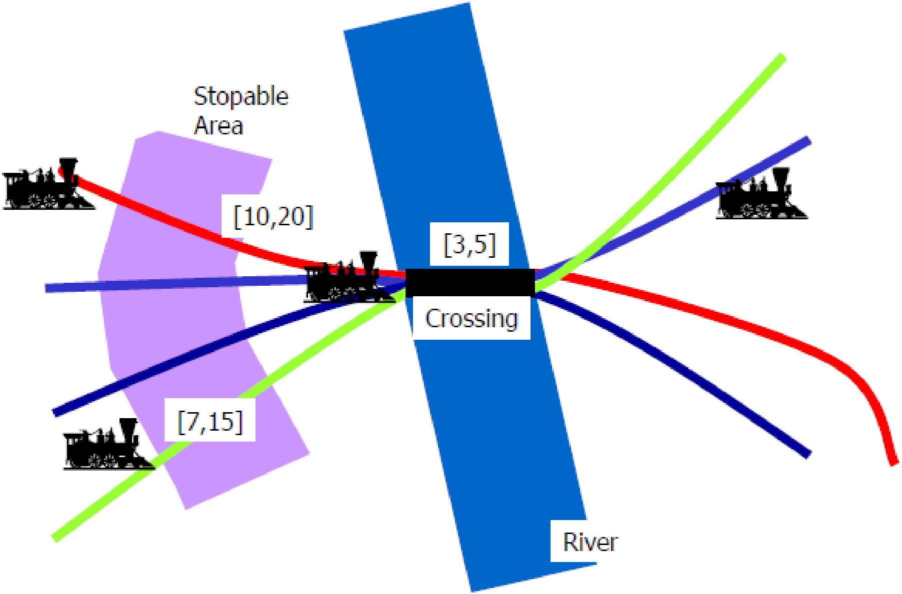
- Suppose we model a simple system M that accepts packages on channel a and immediately sends them to channel b
- P_2 models the system using an urgent position
- P_1 and P_2 have the same behavior

Urgent Position [Dav05]

- **Semantics:**
- There is no delay in the urgent position.
-
- **Note:**
    - Using urgent positions **reduces** the number of clocks in the model and thus the complexity.

Outline

Example Idea [BDL05]



Example Specifications [BDL05]

Textual Specifications

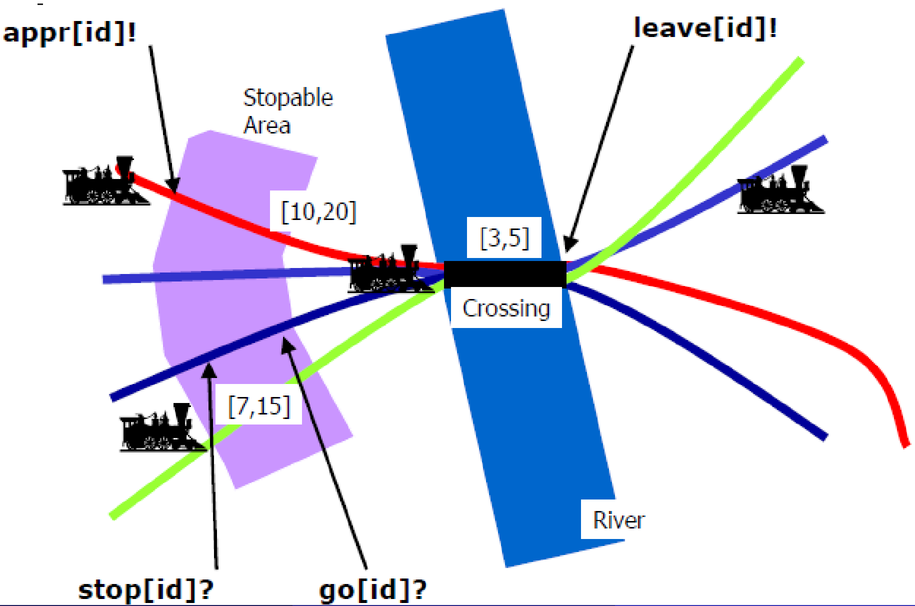
- Bridge access control for several trains.
- A bridge as a critically shared resource can only be crossed by one train.
- The system is defined as several trains and a controller.
- The train cannot be stopped immediately, it also takes time to start.

Timing and Communication [BDL05]

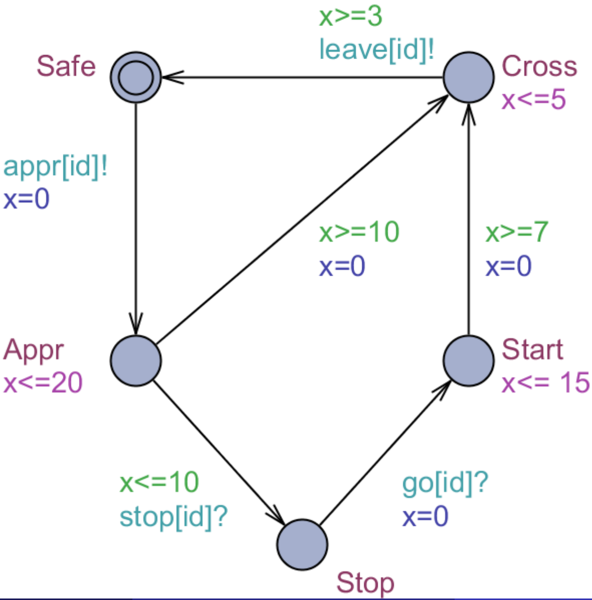
Time constraints and communication

- The train sends a **appr!** signal on time when it arrives at the bridge.
- Then the train has 10 time units to receive a stop signal,
  - this allows a safe stop in front of the bridge.
- After these 10 time units, it takes another 10 units for the train to reach the bridge if it is not stopped.
- If the train is stopped, the train will start after it receives the **go!** signal from the bridge controller.
- When the train leaves the bridge, it sends a signal **leave!**.

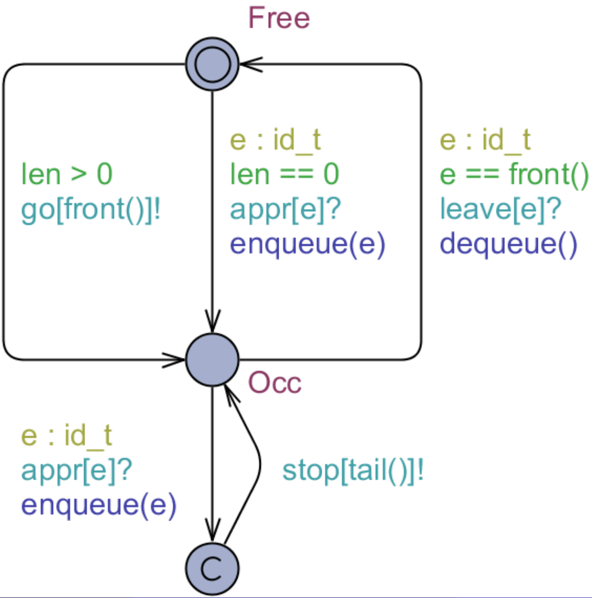
Synchronization Signals [BDL05]



Train Template [BDL05]



Bridge Controller Template [BDL05]



Model Verification [BDL05]

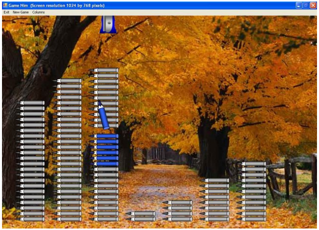
- `E<> Gate.Occ`
- `E<> Train(0).Cross`
- `E<> Train(1).Cross`
- `E<> Train(0).Cross and Train(1).Stop`
- `E<> Train(0).Cross and (forall (i : id_t) i != 0 imply Train(i).Stop)`
- `A[] forall (i : id_t) forall (j : id_t) Train(i).Cross && Train(j).Cross imply i == j`
- `A[] Gate.list[N] == 0`
- `Train(0).Appr --> Train(0).Cross`
- `Train(1).Appr --> Train(1).Cross`
- `Train(2).Appr --> Train(2).Cross`
- `Train(3).Appr --> Train(3).Cross`
- `Train(4).Appr --> Train(4).Cross`
- `Train(5).Appr --> Train(5).Cross`
- `A[] not deadlock`

Outline

Simple Variant NIM

- NIM is a game based on logic and strategy.
- 2 players are playing.
- The player removes one to MAX (2) items (matches, protons) from the series during his turn.
- The player who removes the last thing wins.

Classic Variant NIM



- NIM is a game based on logic and strategy.
- 2 players are playing.
- The players remove objects from different piles/rows.
- The player must remove at least one object during his turn.
- On his turn, the player removes any number of objects, all of which belong to one pile.
- Basic variants of the game:
  - **Normal** . . . The player who removes the last thing wins.
  - **Loss** . . . The player who removes the last thing loses.

Literatura 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.

[Dav05] UPPAAL tutorial at rtss'05 (), December 2005.

[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.

[Voj10] Tomas Vojnar. Formal analysis and verification. Lecture handouts, http://www.fit.vutbr.cz/study/courses/FAV/public/, August 2010.

[Wik10] Linear temporal logic. http://en.wikipedia.org/wiki/Linear_temporal_logic, November 2010.