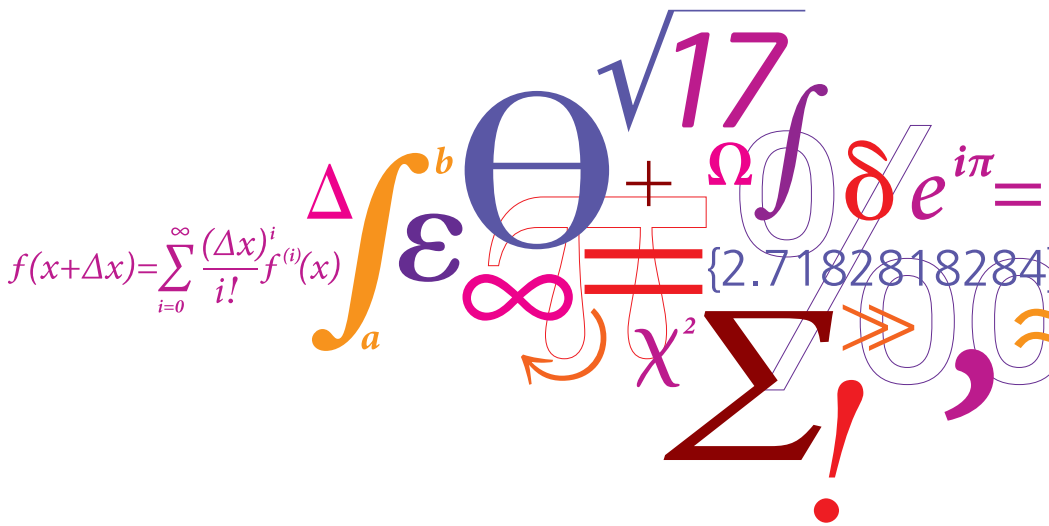
02224 Real-time systems

Computation Tree Logic and UPPAALs query language

Michael R. Hansen



DTU Compute
Department of Applied Mathematics and Computer Science

Operational models

An operational model like a timed-automata model is good at, for example

- system analysis through simulation
- prototype development, code generation, and hardware synthesis

but bad for supporting declarative requirements like:

- Every message will be answered within 20 ms
- Two trains will not be at an intersection simultaneously
- . . .

The model checking problem

A model checking problem has the form:

$$M \models \phi?$$

where

- M is typically an operational system model, e.g. given by a network of timed automata
- ϕ is a formula in a logic for expressing requirements unambiguously in a declarative manner

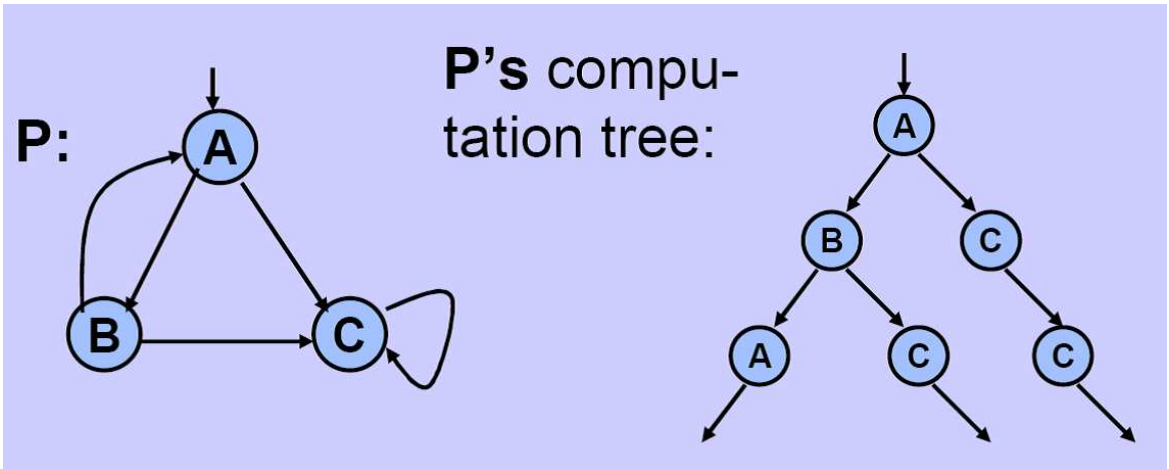
The answer to the question is YES if every behavior of M satisfies ϕ, and otherwise NO.

Model checking provides guarantees

Here we shall use Computation Tree Logic (CTL) and the Timed version Timed CTL as for expressing requirements.

Computation tree

A state of an automaton can be considered as spanning a computation tree, for example:



CTL is a logic over such computation trees.

Syntax of CTL

The syntax of CTL is described by the grammar:

$$\phi, \psi \quad ::- \quad a \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \Rightarrow \psi \qquad \text{(propositional fragment)}$$
$$\mid \mathrm{EX}\phi \mid \mathrm{AX}\phi \mid \mathrm{EF}\phi \mid \mathrm{AF}\phi \mid \mathrm{EG}\phi \mid \mathrm{AG}\phi \mid \phi\mathrm{EU}\psi \mid \phi\mathrm{AU}\psi$$

where $a \in ap$ is an *atomic proposition*.

The modalities are obtained by combining

- path quantifiers:
  - A (all) means for all path originating in the current state, and
  - E (exists) means for some part originating in the current state.

- temporal operators F, G, X and U are interpreted on a path:
  - F ϕ (finally) means for some state on the path: ϕ,
  - G ϕ (globally) means for all states on the path: ϕ,
  - X ϕ (next) means the next state on the path: ϕ, and
  - ϕ U ψ (until) means that ψ holds for some state s_j on the path, and ϕ holds for all previous states $s_i, i < j$ on that path.

Examples

- It is possible that the lamp is never on: $\mathrm{EG}\,\neg\mathrm{on}$.
- The system never deadlocks: $\mathrm{AG}\,\neg\mathrm{deadlock}$.
- It is always possible to restart: $\mathrm{AG}\,\mathrm{EF}\,\mathrm{restart}$.
- The system always eventually breaks, but it functions until then: $\mathrm{work}\,\mathrm{AU}\,\mathrm{break}$.

Semantics

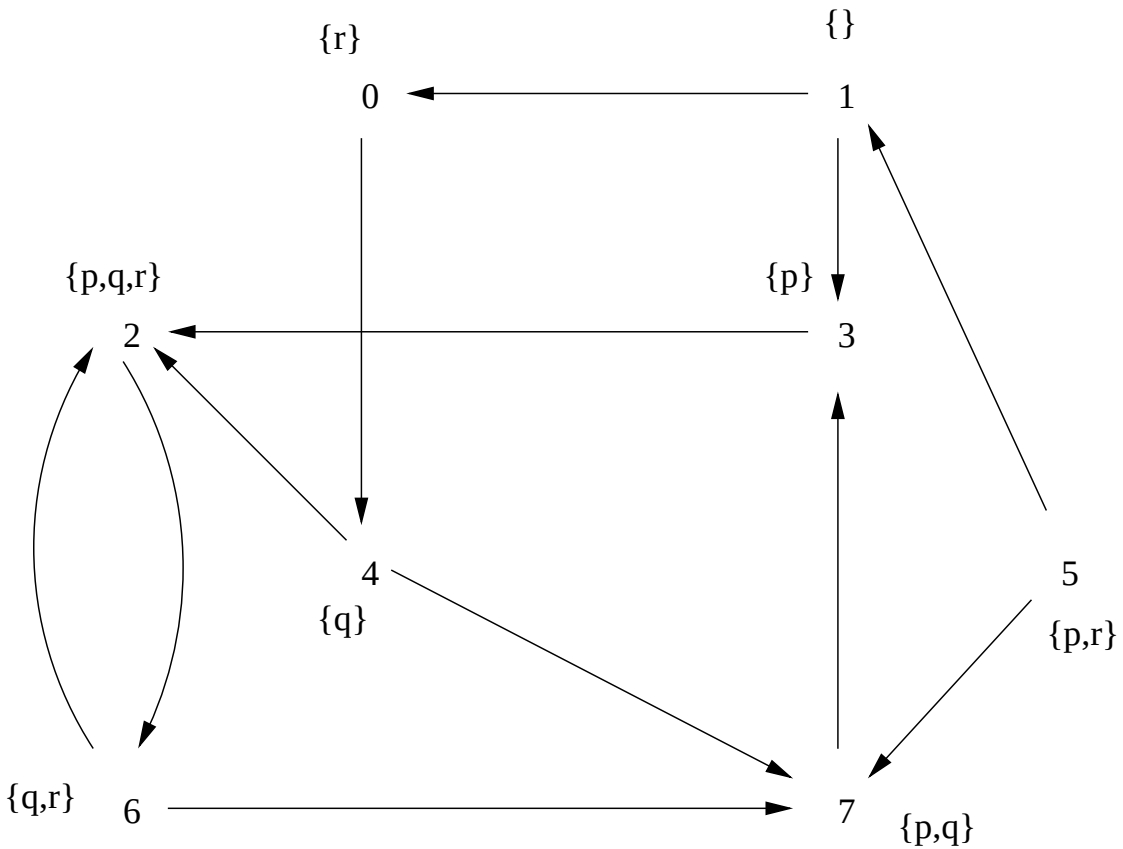
CTL formulas are interpreted over Kripke structures (or automata) $K = (V, E, L, I)$, where

- V is a finite set of *vertices* (or *states*).
- $E \subseteq V \times V$ is the *transitions*. If $(v, v') \in E$, then state v has v' as a *successor state*
- $L : V \to 2^{AP}$ is a *labelling function*. If $L(v) = \{a_1, \ldots, a_m\}$, then each atomic proposition a_i (for $1 \leq i \leq m$) holds in state v.
- $I \subseteq V$ is a set of initial states

A *path* π of $K = (V, E, L, I)$ is an infinite sequence of states: $\pi = s_0 s_1 \cdots s_i s_{i+1} \cdots$ such that $(s_i, s_{i+1}) \in E, i \geq 0$.

For any path $\pi = s_0 s_1 \cdots s_i s_{i+1} \cdots$, let $\pi_k = s_k$ for $k \geq 0$.

Example: Kripke structure – initial states are left out



Example from BaierKatoen2008

The formula

$$\mathrm{EF}((p \Leftrightarrow r) \wedge \neg(p \Leftrightarrow q))$$

holds in the following states: $\{0, 1, 4, 5\}$

Semantic relations

Given Kripke structure: $K = (V, E, L, I)$ and formula ϕ, we define two semantic relations:

- $s, K \models \phi$, where $s \in V$ reads: "ϕ holds in state s"
- $K \models \phi$ reads: "ϕ holds in the Kripke structure K"

Where the last one is defined in terms of the first:

$$K \models \phi \quad \text{iff} \quad s, K \models \phi, \text{ for every initial state } s \in I$$

Definition of $s, K \models \phi$ (Propositional part)

Given Kripke structure: $K = (V, E, L, I)$.

We define $s, K \models \phi$ by structural induction on ϕ:

- $s, K \models a$ iff $a \in L(s)$
- $s, K \models \neg\phi$ iff not $s, K \models \phi$ (also written $s, K \not\models \phi$)
- $s, K \models \phi \wedge \psi$ iff $s, K \models \phi$ and $s, K \models \phi$
- $s, K \models \phi \vee \psi$ iff $s, K \models \phi$ or $s, K \models \phi$
- $s, K \models \phi \Rightarrow \psi$ iff $s, K \not\models \phi$ or $s, K \models \psi$

Other propositional cases are straightforward.

Definition of $s, K \models \phi$ (Modalities)

- $s, K \models \mathrm{EX}\,\phi$ iff for some path π with $\pi_0 = s$: $\pi_1, K \models \phi$
- $s, K \models \mathrm{AX}\,\phi$ iff for every path π with $\pi_0 = s$: $\pi_1, K \models \phi$
- $s, K \models \mathrm{EF}\,\phi$
  iff for some path π with $\pi_0 = s$: $\pi_k, K \models \phi$, for some $k \geq 0$
- $s, K \models \mathrm{AF}\,\phi$
  iff for every path π with $\pi_0 = s$: $\pi_k, K \models \phi$, for some $k \geq 0$
- $s, K \models \mathrm{EG}\,\phi$
  iff for some path π with $\pi_0 = s$: $\pi_k, K \models \phi$, for all $k \geq 0$
- $s, K \models \mathrm{AG}\,\phi$
  iff for every path π with $\pi_0 = s$: $\pi_k, K \models \phi$, for all $k \geq 0$
- $s, K \models \phi\,\mathrm{EU}\,\psi$
  iff for some path π with $\pi_0 = s$: $\pi_k, K \models \psi$, for some $k \geq 0$,
  and for all $i : 0 \leq i < k : \pi_k, K \models \phi$
- $s, K \models \phi\,\mathrm{AU}\,\psi$
  iff for every path π with $\pi_0 = s$: $\pi_k, K \models \psi$, for some $k \geq 0$,
  and for all $i : 0 \leq i < k : \pi_k, K \models \phi$

A sufficient fragment

It suffices with a small set of operators and modalities, e.g. $\neg, \wedge, \mathrm{EX}, \mathrm{EU}, \mathrm{EG}$, all the other can be derived from those.

$$
\begin{aligned}
\phi \vee \psi &\equiv \neg(\neg\phi \wedge \neg\psi) \\
\mathrm{true} &\equiv a \vee \neg a \\
\phi \Rightarrow \psi &\equiv \neg\phi \vee \psi \\
\phi \Leftrightarrow \psi &\equiv (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi) \\
\mathrm{AX}\phi &\equiv \neg\mathrm{EX}\neg\phi \\
\mathrm{EF}\phi &\equiv \mathrm{true}\,\mathrm{EU}\,\phi \\
\mathrm{AG}\,\phi &\equiv \neg\mathrm{EF}\,\neg\phi \\
\mathrm{AF}\,\phi &\equiv \neg\mathrm{EG}\,\neg\phi \\
\phi\,\mathrm{AU}\,\psi &\equiv \neg(\neg\psi\,\mathrm{EU}\,(\neg\phi \wedge \neg\psi)) \wedge \mathrm{AF}\,\psi
\end{aligned}
$$

where we exploit that $\neg$ binds tighter that any dyadic operator.

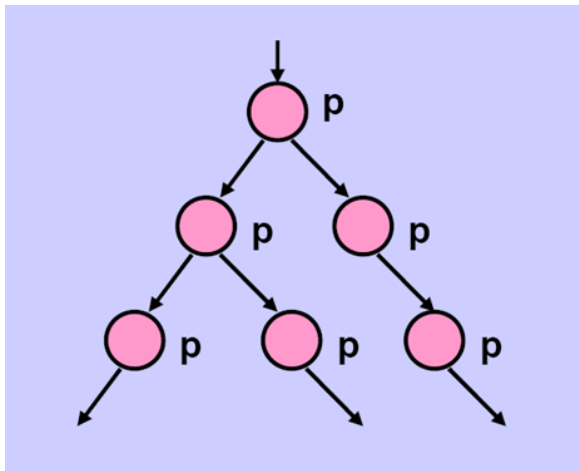
Typical form of formulas

The query language of UPPAAL allows modalities at the outermost level only.

Typical requirements can be formalized in this fragment and checked efficiently. We now illustrate these forms.

P holds invariantly

P is true in all reachable states:



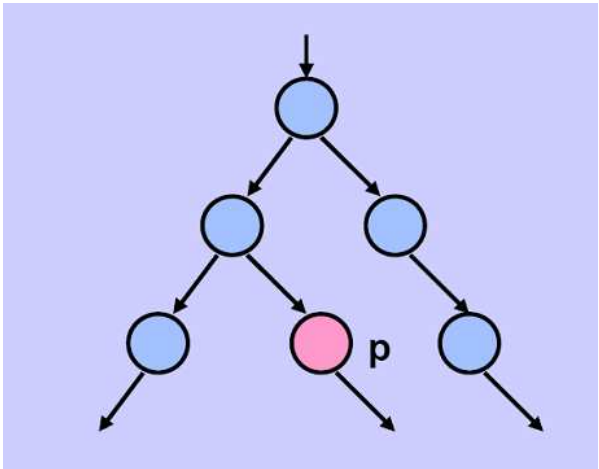
For every paths, globally *P*:

- CTL: AG *p*
- UPPAAL: `A[] p`

G is written `[]`

P is possible

P is true in some reachable state:



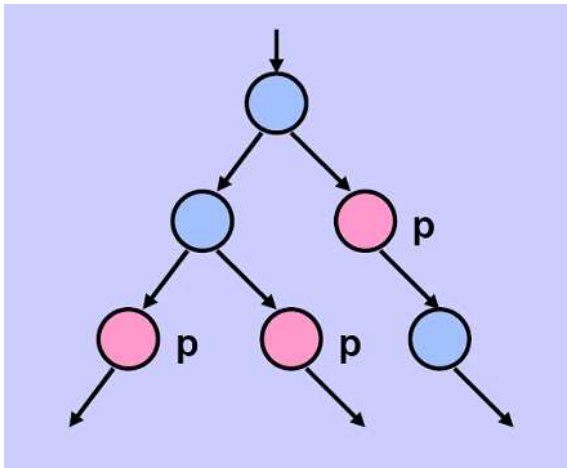
For some paths, finally *P*:

- CTL: EF *p*
- UPPAAL: E<> p

F is written <>

P holds inevitable

P is always true eventually:

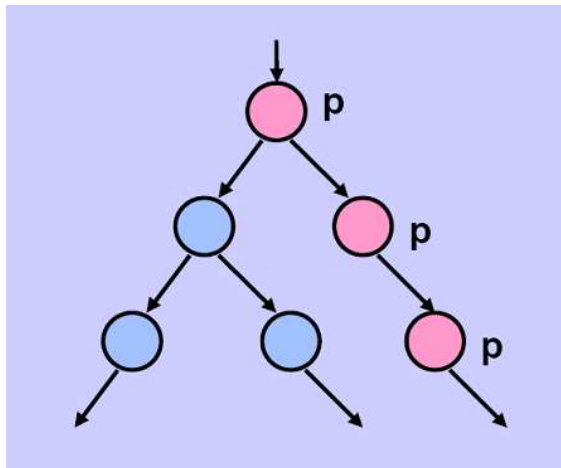


For every paths, finally *P*:

- CTL: AF *p*
- UPPAAL: A<> p

P holds potentially always

It is possible that *P* is always true:

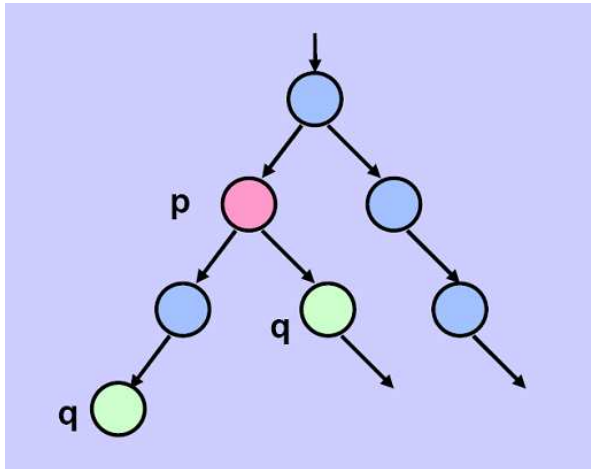


For some paths, globally *P*:

- CTL: $\mathrm{EG}\ p$
- UPPAAL: `E[] p`

P leads to *Q*

If *P* holds then eventually *Q* will hold



For every paths, globally if *P* then inevitably *Q*:

- CTL: $\mathrm{AG}\,(p \Rightarrow \mathrm{AF}q)$
- UPPAAL: `p --> q`

CTL and UPPAAL – informally

A network of timed automata has a continuous-time semantics, where states consist of pairs of process locations and clock evaluations. check last lecture

As a consequence

- the next modalities AX and EX are not meaningful
- the semantics of until modalities AU and EU must be reconsidered — we return to that later

We shall be more formal later about Timed CTL when we discuss model-checking algorithms.

Queries in UPPAAL (1)– informally

A query in UPPAAL has one of the following forms:

- `A[] p`
- `A<> p`
- `E[] p`
- `E<> p`
- `p --> q`

where *p* and *q* are state formulas.

A state formula is a side-effect free expression which can be checked in a state without considering behavior.

State formulas in UPPAAL – informally

The following are examples of state formulas:

- `P.l` – process P is in location l.
- Guards, for example `x <= 3 && y > 5` where `x` and `y` are clocks.
- Formulas involving arithmetics, for example $i1 + i2 + i3 > 7$, where $i1, i2, i3$ are integer variables.
- the formula `deadlock`
- Boolean expressions, using for example negation, disjunction, conjunction and implication on state formulas.

For more details, please consult the UPPAAL tutorial

Deadlock, Timelock and zeno-path

- A state is a deadlock state if there are no outgoing action transitions neither from the state itself or any of its delay successors
- A state contains a timelock if there is no time-divergent path starting from it.
- A path is Zeno if it is time-convergent and it performs infinitely many discrete actions

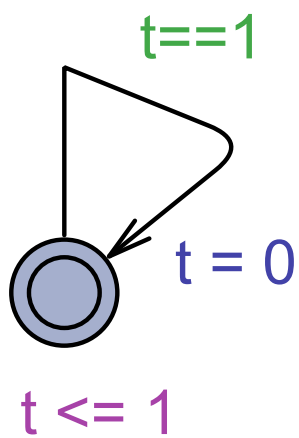
Remember:

- A path is time divergent when time growths towards infinity; otherwise it is time convergent

Trick: Spotting Anomalies in Uppaal

Timelocks and Zeno paths represent typical modelling flaws:

- They should be avoided in the models.
- Add a test automaton to the system:



  where `t` is a new clock
- Veryfy the liveness property: `t==0 --> t==1`