02224 Real-time systems

Model Checking CTL and TCTL

Michael R. Hansen

DTU Compute
Department of Applied Mathematics and Computer Science

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 questions 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 for expressing requirements.

Syntax of CTL

The syntax of CTL is described by the grammar:

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

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 is functions until then: $\mathrm{work}\,\mathrm{AU}\,\mathrm{break}$.

Semantics

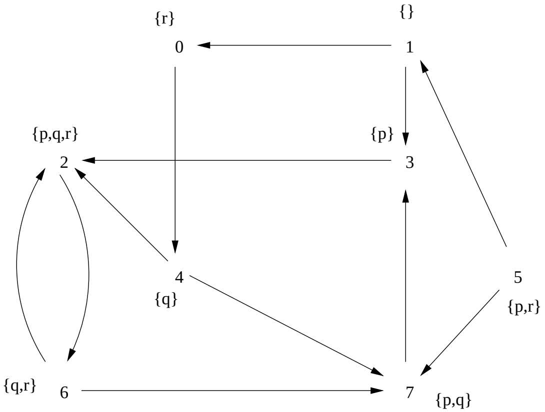
CTL formulas are interpreted over Kripke structures (or automata) $K = (V, E, L, I)$, where

- 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



The formula

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

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

CTL Model checking: basic idea

Given Kripke structure: $K = (V, E, L, I)$ and formula ϕ, we want to determine the set of states

$$\mathrm{Sat}(\phi) \subseteq V$$

where ϕ is true.

The algorithm follows the structure of ϕ working bottom-up:

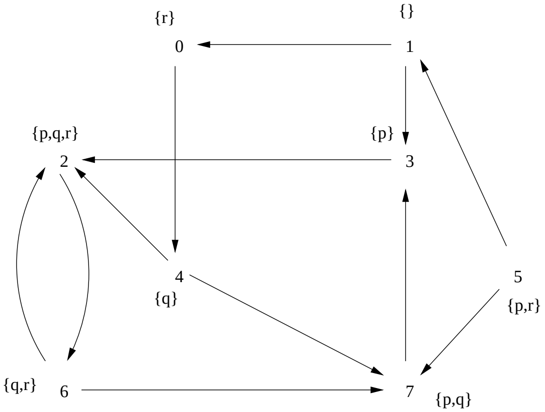
- Find $\mathrm{Sat}(a)$ for every atomic propositions a occurring in ϕ.
- Find $\mathrm{Sat}(\psi_1)$ for every subformula ψ_1 of ϕ containing precisely one operator.
- Find $\mathrm{Sat}(\psi_2)$ for every subformula ψ_2 of ϕ containing precisely two operators.
- $\cdots$
- Find $\mathrm{Sat}(\phi)$ on the basis of knowing $\mathrm{Sat}(\psi)$, for each immediate subfurmula ψ of ϕ.

Remember that it suffices to consider a small adequate set of operators, such as $\neg, \wedge, \mathrm{EX}, \mathrm{AF}, \mathrm{EU}$.

CTL Model checking: Propositional part

$$
\begin{aligned}
\mathrm{Sat}(a) &= \{s \in V \mid a \in L(s)\} \\
\mathrm{Sat}(\neg\phi) &= V \setminus \mathrm{Sat}(\phi) \\
\mathrm{Sat}(\phi \wedge \psi) &= \mathrm{Sat}(\phi) \cap \mathrm{Sat}(\psi)
\end{aligned}
$$

Example: Propositional part



- $\text{Sat}(p \Leftrightarrow q) = \{0, 1, 2, 7\}$
- $\text{Sat}(p \Leftrightarrow r) = \{1, 2, 4, 5\}$
- $\text{Sat}(\neg(p \Leftrightarrow q)) = \{3, 4, 5, 6\}$
- $\text{Sat}((p \Leftrightarrow r) \wedge \neg(p \Leftrightarrow q)) = \text{Sat}(p \Leftrightarrow r) \cap \text{Sat}(\neg(p \Leftrightarrow q)) = \{4, 5\}$

CTL Model checking: $\mathrm{EX}\ \phi$

$$\mathrm{Sat}(\mathrm{EX}\ \phi) \quad = \quad \{s \in V \mid s \text{ has a successor in } \mathrm{Sat}(\phi)\}$$

CTL Model checking: $\mathrm{AF}\ \phi$

$\mathrm{Sat}(\mathrm{AF}\ \phi)$ can be found by a marking algorithm:

- Mark all states in $\mathrm{Sat}(\phi)$
- Repeat

  if an unmarked state $s \in V$ has all its successors marked,

  then mark s

  until no new marked state is found.

CTL Model checking: $\phi \,\mathrm{EU}\, \psi$

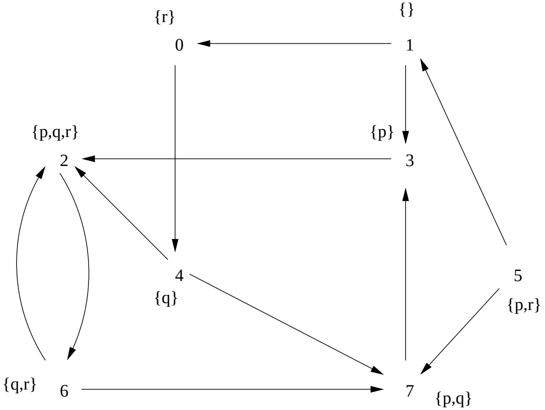
$\mathrm{Sat}(\phi \,\mathrm{EU}\, \psi)$ can be found by a marking algorithm:

- Mark all states in $\mathrm{Sat}(\psi)$
- Repeat

    if an unmarked state $s \in \mathrm{Sat}(\phi)$ has a successor marked,
    then mark s

  until no new marked state is found.

Example: Modality part



- $\mathrm{Sat}((p \Leftrightarrow r) \wedge \neg(p \Leftrightarrow q)) = \{4, 5\}$
- $\mathrm{Sat}(\mathrm{EF}\ ((p \Leftrightarrow r) \wedge \neg(p \Leftrightarrow q))) = \{4, 5, 0, 1\}$

Result

It is decidable whether a Kripke $K = (V, E, L, I)$ satisfies a CTL formula ϕ.

Complexity of decision procedure: $O(|\phi| \cdot (|V| + |E|))$.

The size of the Kripke structure is, however, exponential in the number of parallel components.

Timed CTL: informally

The syntax for Timed CLT is obtained from the CTL syntax by

- deletion of $\mathrm{EX}\,\phi$ and $\mathrm{AX}\,\phi$ formulas,
- allowing clock constraints α as atomic formulas, and
- adding a freeze quantifier $z\ in\ \phi$, where z is a clock called the freeze identifier.

A formula $z\ in\ \phi$ is true is a state s iff ϕ is true in s for $z = 0$.

A TCTL formula is interpreted over a timed transition system (see Lecture 2) for a timed automaton, where a state has the form (l, v)

- l is a timed-automaton location, and
- v is a clock evaluation

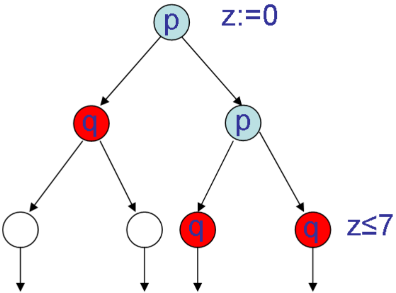
and a transition can be a

- delay transition of the form $(l, v) \xrightarrow{d} (l, v + d)$, or a
- discrete transition of the form $(l, v) \longrightarrow (l', v')$

Example: $\phi \, \mathrm{AU}_{\leq k} \, \psi$

z *in* $(p \, \mathrm{AU} \, (z \leq 7 \wedge q))$ also written $p \, \mathrm{AU}_{\leq 7} \, q$

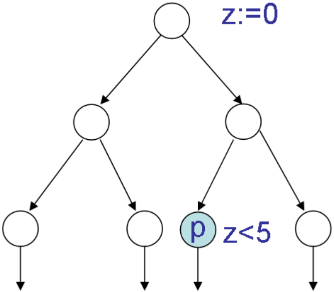
p holds continuously until q holds within 7 time units for all paths:



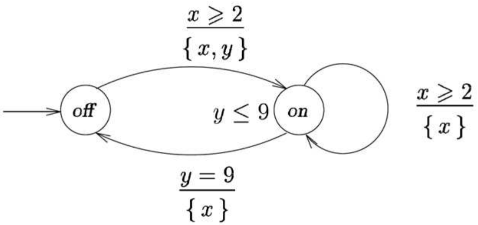
Example: $\mathrm{EF}_{<k}\,\psi$

z *in* $\mathrm{EF}\,(z < 5 \wedge p))$ also written $\mathrm{EF}_{<5}\,p$

p becomes true within 5 time units along some path:



Further examples



- The light is always off for at least 2 minutes: (off AU $x \geq 2$)
- The light is sometimes off for at least 3 minutes: (off EU $x \geq 3$)
- If the light is on, it will inevitably switch off within 9 seconds: AG (on $\Rightarrow$ $\mathrm{AF}_{\leq 9}$off)

Model checking for TCTL

A timed transition system has infinitely (even uncountably) many states due to the clock evaluations.

Is model checking possible for timed automata?

Yes. Alur and Dill (in the early 90ties) showed how to partition the infinite number of clock evaluations into a finite collection of equivalence classes having the following property:

Equivalent clock evaluations satisfy the same TCTL formulas

Some observations

Notation:

1. Let $\lfloor r \rfloor$, for $r \in \mathbb{R}$, denote the largest integer that is at most r
2. Let $frac(r) = r - \lfloor r \rfloor$ denote the fractional part of r

Observe

- Atomic clock constraints compare clocks with natural numbers e.g. $x < c$
- $v \models x < c$ iff $\lfloor v(x) \rfloor < c$ — the fractional part is not relevant.
- $v \models x \leq c$ iff either $\lfloor v(x) \rfloor < c$ or $\lfloor v(x) \rfloor = c$ and $frac(v(x)) = 0$ — the fractional part is not relevant.
- $v \models g$ depends just on the integer part on clock values and on whether fractional parts are 0.

Clock equivalence

Let A be a timed automaton, c_x be the largest value the clock x is compared to, and v and v' clock evaluations.

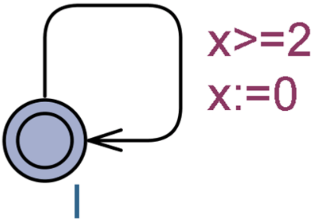
v and v' are clock equivalent (written $v \approx v'$) iff

1. for each clock x we have that either both $v(x)$ and $v'(x)$ are greater than c_x or
$$\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$$

2. for each clock x with $v(x) \leq c_x$ we have
$$frac(v(x)) = 0 \quad \text{iff} \quad frac(v'(x)) = 0$$
and

3. for all clocks x, y with $v(x) \leq c_x$ and $v(y) \leq c_y$ we have
$$frac(v(x)) \leq frac(v(y)) \quad \text{iff} \quad frac(v'(x) \leq frac(v'(y))$$

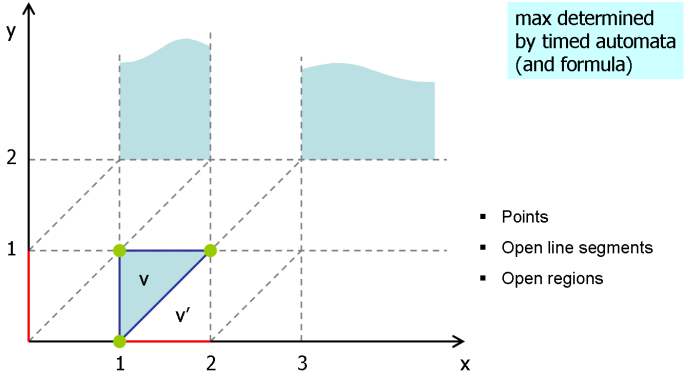
Example: Clock regions



This simple automaton has 6 clock regions (equivalence classes):

$$[x = 0], [0 < x < 1], [x = 1], [1 < x < 2], [x = 2], [x > 2]$$

Example: Geometric interpretation of clock regions

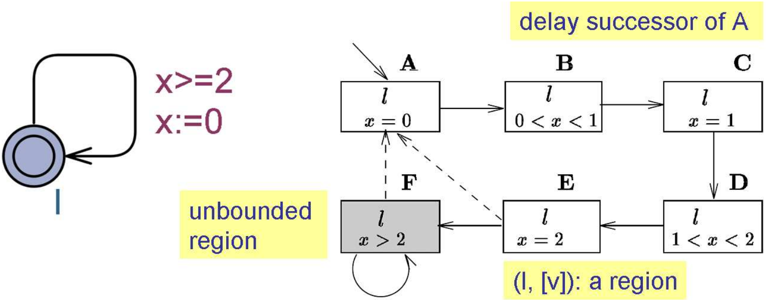


- Points
- Open line segments
- Open regions

There are 60 regions:

- 12 closed areas
- 6 open areas
- 7 open line segments
- 23 closed line segments
- 12 corner points

Example: Region automaton



In the region automaton a state has the form $(l, [v])$, where

- l is a timed-automaton location and
- $[v]$ is a clock equivalence class

Result

- A region automaton has a finite number of states.
- It is decidable whether a timed automaton satisfies a TCTL formula
- The complexity of the decision procedure is exponential in the number of clocks and in
  the constants mentioned in clock constraints.

The symbolic states of Uppaal are based on Zones rather that regions.

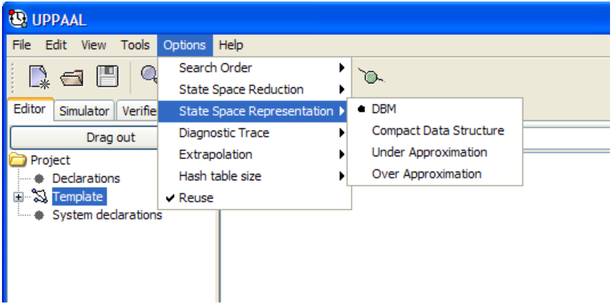
A zone is defined by a conjunction of formulas of the form:

$$x \bowtie c \qquad \text{or} \qquad x - y \bowtie c$$

where $\bowtie \in \{<, \leq, = \geq, >\}$.

A zone-based representation is typically much more compact that a region-based one.

Uppaal Verification Options



- Search order: Breadth first or Depth first
- State Space Reduction: None, Conservative, Aggressive
- State Space Representation: DBM, Compact, Under approximation, Over approximation