Overview Lecture #18

⇒ Summary of LTL model checking

- Branching temporal logic
- Syntax and semantics of CTL
Summary of LTL model checking (1)

- LTL is a logic for formalizing path-based properties

- Expansion law allows for rewriting until into local conditions and next

- LTL-formula $\varphi$ can be transformed algorithmically into NBA $A_\varphi$
  - this may cause an exponential blow up
  - algorithm: first construct a GNBA for $\varphi$; then transform it into an equivalent NBA

- LTL-formulae describe $\omega$-regular LT-properties
  - but are less expressive as $\omega$-regular languages
Summary of LTL model checking (2)

- $\mathcal{T}S \models \varphi$ can be solved by a nested depth-first search in $\mathcal{T}S \otimes A_{\neg \varphi}$
  - time complexity of the LTL model-checking algorithm is linear in $\mathcal{T}S$ and exponential in $|\varphi|$

- Fairness assumptions can be described by LTL-formulae

  - the model-checking problem for LTL with fairness is reducible to the standard LTL model-checking problem

- The LTL-model checking problem is PSPACE-complete

- Satisfiability and validity of LTL amounts to NBA emptiness-check

- The satisfiability and validity problem for LTL are PSPACE-complete
Overview Lecture #18

- Summary of LTL model checking

⇒ Branching temporal logic

- Syntax and semantics of CTL
Linear and branching temporal logic

- **Linear** temporal logic:
  
  “statements about (all) paths starting in a state”
  
  \[- s \models \Box (x \leq 20) \text{ iff for all possible paths starting in } s \text{ always } x \leq 20 \]

- **Branching** temporal logic:
  
  “statements about all or some paths starting in a state”
  
  \[- s \models \forall \Box (x \leq 20) \text{ iff for all paths starting in } s \text{ always } x \leq 20 \]
  \[- s \models \exists \Box (x \leq 20) \text{ iff for some path starting in } s \text{ always } x \leq 20 \]
  \[- \text{ nesting of path quantifiers is allowed} \]

- Checking $\exists \varphi$ in LTL can be done using $\forall \neg \varphi$
  
  \[- \ldots \text{ but this does not work for nested formulas such as } \forall \Box \exists \Diamond a \]
Linear versus branching temporal logic

- **Semantics** is based on a branching notion of time
  - an infinite tree of states obtained by unfolding transition system
  - one “time instant” may have several possible successor “time instants”

- **Incomparable expressiveness**
  - there are properties that can be expressed in LTL, but not in CTL
  - there are properties that can be expressed in most branching, but not in LTL

- Distinct **model-checking algorithms**, and their time complexities

- Distinct treatment of **fairness assumptions**

- Distinct **equivalences** (pre-orders) on transition systems
  - that correspond to logical equivalence in LTL and branching temporal logics
Transition systems and trees

Transition system:
- $s_0$: $\{ x \neq 0 \}$
- $s_1$: $\{ x = 0 \}$
- $s_2$: $\{ x = 0 \}$
- $s_3$: $\{ x = 1, x \neq 0 \}$

Transition tree:
- $(s_0, 0)$
  - $(s_1, 1)$
    - $(s_2, 2)$
    - $(s_3, 2)$
      - $(s_3, 3)$
      - $(s_2, 3)$
        - $(s_3, 3)$
        - $(s_2, 4)$
        - $(s_3, 4)$

### #18: Computation tree logic

#### Model checking

<table>
<thead>
<tr>
<th>“behavior” in a state $s$</th>
<th>path-based: $\text{trace}(s)$</th>
<th>state-based: computation tree of $s$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>temporal logic</strong></td>
<td>LTL: path formulas $\varphi$</td>
<td>CTL: state formulas $\exists \varphi$</td>
</tr>
<tr>
<td></td>
<td>$s \models \varphi$ iff $\forall \pi \in \text{Paths}(s). \pi \models \varphi$</td>
<td>$\forall \varphi$</td>
</tr>
<tr>
<td><strong>complexity of the model checking problems</strong></td>
<td>PSPACE–complete</td>
<td>PTIME</td>
</tr>
<tr>
<td></td>
<td>$\mathcal{O}(</td>
<td>TS</td>
</tr>
</tbody>
</table>

# Computation tree logic

<table>
<thead>
<tr>
<th>complexity of the model checking problems</th>
<th>PSPACE–complete</th>
<th>PTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(</td>
<td>TS</td>
<td>\cdot 2^{</td>
</tr>
</tbody>
</table>

#### Implementation relation

- trace inclusion and the like (proof is PSPACE-complete)
- simulation and bisimulation (proof in polynomial time)

#### Fairness

- no special techniques
- special techniques needed

© JPK
Branching temporal logics

There are various branching temporal logics:

- Hennessy-Milner logic
- Computation Tree Logic (CTL)
- Extended Computation Tree Logic (CTL*)
  - combines LTL and CTL into a single framework
- Alternation-free modal $\mu$-calculus
- Modal $\mu$-calculus
- Propositional dynamic logic
Overview Lecture #18

• Summary of LTL model checking

• Branching temporal logic

⇒ Syntax and semantics of CTL
Computation tree logic

modal logic over infinite trees [Clarke & Emerson 1981]

- **Statements over states**
  - $a \in AP$
  - $\neg \Phi$ and $\Phi \land \Psi$
  - $\exists \varphi$
  - $\forall \varphi$

- **Statements over paths**
  - $\mathbf{\Box} \Phi$
  - $\Phi \mathbf{U} \Psi$

$\Rightarrow$ note that $\mathbf{\Box}$ and $\mathbf{U}$ *alternate* with $\forall$ and $\exists$

- $\forall \mathbf{\Box} \mathbf{\Box} \Phi$ and $\forall \exists \mathbf{\Box} \Phi \notin \text{CTL}$, but $\forall \mathbf{\Box} \forall \mathbf{\Box} \Phi$ and $\forall \mathbf{\Box} \exists \mathbf{\Box} \Phi \in \text{CTL}$
Derived operators

potentially $\Phi$: $\exists \diamond \Phi = \exists (\text{true} \cup \Phi)$

inevitably $\Phi$: $\forall \diamond \Phi = \forall (\text{true} \cup \Phi)$

potentially always $\Phi$: $\exists \Box \Phi := \neg \forall \diamond \neg \Phi$

invariantly $\Phi$: $\forall \Box \Phi = \neg \exists \diamond \neg \Phi$

weak until:

$\exists (\Phi W \Psi) = \neg \forall ( (\Phi \land \neg \Psi) \cup (\neg \Phi \land \neg \Psi) )$

$\forall (\Phi W \Psi) = \neg \exists ( (\Phi \land \neg \Psi) \cup (\neg \Phi \land \neg \Psi) )$

the boolean connectives are derived as usual
Visualization of semantics

∀ red

∃ red

∃ (yellow U red)

∃ ◇ red

∀ ◇ red

∀ red

∀ (yellow U red)
Example properties in CTL
Semantics of CTL \textit{state}-formulas

Defined by a relation $\models$ such that

$$s \models \Phi \text{ if and only if formula } \Phi \text{ holds in state } s$$

- $s \models a$ iff $a \in L(s)$
- $s \models \neg \Phi$ iff $\neg (s \models \Phi)$
- $s \models \Phi \land \Psi$ iff $(s \models \Phi) \land (s \models \Psi)$
- $s \models \exists \varphi$ iff $\pi \models \varphi$ for \textit{some} path $\pi$ that starts in $s$
- $s \models \forall \varphi$ iff $\pi \models \varphi$ for \textit{all} paths $\pi$ that start in $s$
Semantics of CTL path-formulas

Define a relation $\models$ such that

$\pi \models \varphi$ if and only if path $\pi$ satisfies $\varphi$

\[ \pi \models \Box \Phi \quad \text{iff} \quad \pi[1] \models \Phi \]
\[ \pi \models \Phi \cup \Psi \quad \text{iff} \quad (\exists j \geq 0. \pi[j] \models \Psi \land (\forall 0 \leq k < j. \pi[k] \models \Phi)) \]

where $\pi[i]$ denotes the state $s_i$ in the path $\pi$
Transition system semantics

- For CTL-state-formula $\Phi$, the *satisfaction set* $\text{Sat}(\Phi)$ is defined by:

$$\text{Sat}(\Phi) = \{ s \in S \mid s \models \Phi \}$$

- $TS$ satisfies CTL-formula $\Phi$ iff $\Phi$ holds in all its initial states:

$$TS \models \Phi \quad \text{if and only if} \quad \forall s_0 \in I. s_0 \models \Phi$$

  - this is equivalent to $I \subseteq \text{Sat}(\Phi)$

- **Point of attention:** $TS \not\models \Phi$ and $TS \not\models \neg \Phi$ is possible!

  - because of several initial states, e.g., $s_0 \models \exists \Box \Phi$ and $s'_0 \not\models \exists \Box \Phi$
Example of CTL semantics
Infinitely often

\[ s \models \forall \square \forall a \quad \text{if and only if} \quad \forall \pi \in Paths(s) \text{ an } a\text{-state is visited infinitely often} \]
Weak until