Computation Tree Logic

Lecture #17 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: katoen@cs.rwth-aachen.de

June 06, 2007
Overview Lecture #17

⇒ 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 do not have the same expressivity as $\omega$-regular languages.
Summary of LTL model checking (2)

- \( TS \models \varphi \) can be solved by a nested depth-first search in \( TS \otimes A_{\neg \varphi} \)
  - time complexity of the LTL model-checking algorithm is linear in \( TS \) 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 #17

- 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”

  - \( s \models \Box(x \leq 20) \) iff for all possible paths starting in \( s \) always \( x \leq 20 \)

- **Branching** temporal logic:
  
  “statements about all or some paths starting in a state”

  - \( s \models \forall \Box(x \leq 20) \) iff for all paths starting in \( s \) always \( x \leq 20 \)
  - \( s \models \exists \Box(x \leq 20) \) iff for some path starting in \( s \) always \( x \leq 20 \)
  - nesting of path quantifiers is allowed

- Checking \( \exists \varphi \) in LTL can be done using \( \forall \neg \varphi \)
  
  - . . . 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

\( s_0 \) \{ x \neq 0 \}
\( s_1 \) \{ x = 0 \}
\( s_2 \) \{ x = 0 \}
\( s_3 \) \{ x = 1, x \neq 0 \}
#17: Computation tree logic

<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>universal path quantification: $\forall \varphi$</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>$O \left(</td>
<td>TS</td>
</tr>
<tr>
<td><strong>implementation-relation</strong></td>
<td>trace inclusion and the like (proof is PSPACE-complete)</td>
<td>simulation and bisimulation (proof in polynomial time)</td>
</tr>
<tr>
<td><strong>fairness</strong></td>
<td>no special techniques</td>
<td>special techniques needed</td>
</tr>
</tbody>
</table>
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 #17

- 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**
  - \( a \in AP \) \hspace{1cm} \text{atomic proposition}
  - \( \neg \Phi \) and \( \Phi \land \Psi \) \hspace{1cm} \text{negation and conjunction}
  - \( \exists \varphi \) \hspace{1cm} \text{there exists a path fulfilling } \varphi
  - \( \forall \varphi \) \hspace{1cm} \text{all paths fulfill } \varphi

- **Statements over paths**
  - \( \circ \Phi \) \hspace{1cm} \text{the next state fulfills } \Phi
  - \( \Phi \cup \Psi \) \hspace{1cm} \Phi \text{ holds until a } \Psi \text{-state is reached}

⇒ note that \( \circ \) and \( \cup \) *alternate* with \( \forall \) and \( \exists \)

  - \( \forall \circ \circ \circ \Phi \) and \( \forall \exists \circ \Phi \notin \text{CTL} \), but \( \forall \circ \circ \circ \circ \Phi \) and \( \forall \circ \exists \circ \circ \Phi \in \text{CTL} \)
Derived operators

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

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

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

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

weak until: $\exists (\Phi W \Psi) = \neg \forall ((\Phi \land \neg \Psi) U (\neg \Phi \land \neg \Psi))$

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

the boolean connectives are derived as usual
Visualization of semantics

\[ \exists \Diamond \text{red} \]

\[ \exists \Box \text{red} \]

\[ \exists (\text{yellow} \cup \text{red}) \]

\[ \forall \Diamond \text{red} \]

\[ \forall \Box \text{red} \]

\[ \forall (\text{yellow} \cup \text{red}) \]
Example properties in CTL
Semantics of CTL state-formulas

Defined by a relation $\models$ such that

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

\[
\begin{align*}
  s \models a & \iff a \in L(s) \\
  s \models \neg \Phi & \iff \neg (s \models \Phi) \\
  s \models \Phi \land \Psi & \iff (s \models \Phi) \land (s \models \Psi) \\
  s \models \exists \varphi & \iff \pi \models \varphi \text{ for some path } \pi \text{ that starts in } s \\
  s \models \forall \varphi & \iff \pi \models \varphi \text{ for all paths } \pi \text{ that start in } s
\end{align*}
\]
Semantics of CTL path-formulas

Define a relation $\models$ such that

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

$\pi \models \Box \Phi$ iff $\pi[1] \models \Phi$

$\pi \models \Phi \lor \Psi$ iff $(\exists j \geq 0. \pi[j] \models \Psi \land (\forall 0 \leq k < j. \pi[k] \models \Phi))$

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:

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

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

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

  – 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. $s_0 \models \exists \square \Phi$ and $s'_0 \not\models \exists \square \Phi$
A triple modular redundant system
A state $s$ satisfies $\forall \square \forall a$ if and only if for all $\pi \in \text{Paths}(s)$ an $a$-state is visited infinitely often.