CTL Model Checking

David L. Dill
CTL syntax:

$AP$ -- atomic propositions
$p \in AP$ is a formula
$f \land g$ is a formula, if $f$ and $g$ are
$\neg f$ is a formula
$AX \ f$ is a formula
$EX \ f$ is a formula
$A[f U g]$ ...
$E[f U g]$ ...

CTL model checking

CTL formulas (properties)

State graph from implementation (Kripke structure)

Model checker

OK or counter-example
State graph

• Many representations can be translated to state graphs
  – circuits
  – concurrent programs
  – various description languages
  – SCR, statecharts
• Translation is usually to a BDD representation of graph, not explicit graph.
Abbreviations:

\[ AF \ f = A[true \cup f] \ "future" \]
\[ EF \ f = E[true \cup f] \]

\[ AG \ f = \neg EF \neg f \ "globally" \]
\[ EG \ f = \neg AF \neg f \]
CTL intuition
AGp: $p$ is invariant

AFp: $p$ is inevitable

EFp: $p$ potentially holds
CTL examples:

Mutual exclusion: $AG \neg (c_1 \land c_2)$

Fairness: $AG (t_1 \rightarrow AF c_1)$

Resetability: $AG (EF "init")$ (there is always a path back to $init$)
Request/Acknowledge example

\[ A[\text{req U ack}] \land A[\text{ack U } \neg \text{req}] \land A[\neg \text{req U } \neg \text{ack}] \land A[\neg \text{ack U } \neg \text{req}] \]

(Weak Until? \( A[\neg \text{ack u } \neg \text{req}] \))
Algorithm for model checking

Idea: progressively label states with nonatomic properties.
Subformulas are treated like atomic formulas after they have been checked.

$labels(s)$ -- set of labels of state $s$ in $M$.
Initially, $labels(s) = P(s)$. 
Fixed point algorithm for model checking

We then traverse the formula to be checked bottom-up, checking subformulas of \( f \) before checking \( f \).

- Add \( \neg f \) to \( labels(s) \) if \( f \) not in \( labels(s) \)
- Add \( f \land g \) to \( labels(s) \) if \( f \) in \( labels(s) \) and \( g \) in \( labels(s) \)
- Add AX \( f \) to \( labels(s) \) if \( f \) in \( labels(s') \) for all \( s':s R s' \)
- Add EX \( f \) to \( labels(s) \) if \( f \) in \( labels(s') \) for some \( s':s R s' \)
Fixed point algorithm for model checking

Until formulas require a fixed-point iteration:

Use fact that $A[f \cup g] = g \lor AX A[f \cup g]$

Algorithm:

1. Whenever state $g$ in $\text{labels}(s)$, add $A[f \cup g]$ to $\text{labels}(s)$.

2. Repeat: Whenever all next states of $s$ have $A[f \cup g]$ in $\text{labels}(s)$, add $A[f \cup g]$ to $\text{labels}(s)$.

Similarly for $E[f \cup g]$.
Z ("zed") specifications
Z

• Logical specification language
• Probably most widely known such
• Very general: can describe
  – data structures
  – relations and tables
  – functions
• Few tools
  – syntax checkers
  – recently, theorem provers
Z has really been used

- IBM CICS specification (1992)
- Cyclotron specification (Jacky)
- Used routinely by Praxis Critical Systems, Ltd (UK)