Model Checking with CTL

Presented by Jason Simas
Model Checking with CTL

Based Upon:


*Model Checking*. Clarke, Grumberg and Peled. 1999. (1-26)
Content

- Context
  - Model Checking
  - Models

- CTL
  - Syntax
  - Semantics
  - Checking Algorithm
Model Checking

- $M \models \varphi$
  - $M$ is the model
    - Requires a description language
  - $\varphi$ is the property to check
    - Requires a specification language
  - $\models$ is the “satisfaction relation”
    - Algorithm to check whether $(M, \varphi) \in \models$
    - Outputs either “yes” or “no” (+ trace)
Models

- Fundamentals
- Language Definition
- Example Model
Fundamentals

- Want to prove properties
- Model all relevant sub properties
- Model abstraction level $\leq$ properties
- $\Rightarrow$ Model how properties change
  - Over time? (sort of)
  - Over property change? (yes)
    - Abstract out time
Modeling Property Change

- Model = States + Transitions + Labels
  - States
    - Possibilities of which properties can be true together +
    - Possibilities of which properties can follow each other +
  - Transitions
    - Possibilities of which states can follow each other
  - Labels
    - Possibilities of which properties are true for each state
- States need not be unique wrt labels
- Use a directed graph
Definition: Model for CTL

\[ M = (S, \rightarrow, L) \]

- \( S \) is a finite set of states \( \{s_0, s_1, \ldots, s_n\} \)
- \( \rightarrow \) is a set of transitions
  - \( \rightarrow \subseteq S \times S \) and
  - for every \( s \in S \) there is some \( s' \in S \) such that \( s \rightarrow s' \)
- \( L \) is a labeling function \( L: S \rightarrow \mathcal{P}(\text{Atoms}) \)
  - \( S \) is the set of states of \( M \)
  - \( \mathcal{P}(\text{Atoms}) \) is the power set of \( \text{Atoms} \)
    - \( \text{Atoms} \) is the set of all propositions
Mutual Exclusion (Interleaved)
Mutual Exclusion (Interleaved)

\[ M = (S, \rightarrow, L) \text{ where} \]

- \[ S = \{ s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_9, \} \]
- \[ \rightarrow = \]
  - \{ (s_0, s_1), (s_1, s_2), (s_1, s_3), (s_2, s_4), (s_3, s_4), (s_2, s_0), (s_4, s_5), (s_0, s_5), (s_5, s_6), (s_5, s_9), (s_6, s_7), (s_9, s_7), (s_6, s_0), (s_7, s_1) \}
- \[ L = \]
  - \{ (s_0, \{n_1, n_2\}), (s_1, \{t_1, n_2\}), (s_2, \{c_1, n_2\}), (s_3, \{t_1, t_2\}), (s_4, \{c_1, t_2\}), (s_5, \{n_1, t_2\}), (s_6, \{n_1, c_2\}), (s_7, \{t_1, c_2\}), (s_9, \{t_1, t_2\}) \}

- Note: \( s_3, s_9 \) are distinct for “turns”
Properties

- Remember \( M \models \varphi \)
  - \( \varphi \) specifies properties of states/transitions
  - Need a specification language for \( \varphi \), CTL

- CTL: Computation Tree Logic
  - Specifying properties of “computation trees”
  - “Logic” = Language + Inference Rules
    - Inference Rules = Algorithm for check
Computation Trees

- A tree such that starting at some state \( s \),
  - There exists edges to each of its children \( (s \rightarrow s') \)
  - Same is true for each child, ad infinitum
Example: “Efficiency”

- For each “cycle” (n_i → n_i) some process enters its critical section
- CTL: $AG ((s_1 \lor s_5) \rightarrow AX (A\neg(s_1 \lor s_5) \lor (c_1 \lor c_2)))$
CTL Syntax

\[ \varphi := \]
\[ \bot | \top | p | \]
\[ (\neg \varphi) | (\varphi \land \varphi) | (\varphi \lor \varphi) | (\varphi \rightarrow \varphi) \]
\[ AX \varphi | EX \varphi | A[\varphi U \varphi] | E[\varphi U \varphi] | AG \varphi | EG \varphi | AF \varphi | EF \varphi \]

Atoms: \( \bot, \top, p \)
- p is an arbitrary atomic property either true or false
  - Example: \( c_1 \): “process 1 is in its critical section”

Propositional Connectives: \( \land, \lor, \neg, \rightarrow \)

Temporal Connectives: \( EG, AG, EX, AX, EF, AF, EU, AU \)
- Note: \( EU \varphi_1 \varphi_2 \) same as \( E[\varphi_1 U \varphi_2] \)

Binding Precedence:
- Unary Connectives: \( \neg, AX, EX, AG, EG, AF, EF \)
- Binary Connectives: \( \rightarrow, AU, EU \)

\( \top, c_1, c_1 \land c_2, AX (c_1 \land c_2), A[c_1 U c_1], E[\top U (AX (c_1 \land c_2))] \)
CTL Semantics

- $M,s \models \varphi$ where $\varphi$ is a CTL formula
  - "is $\varphi$ true for the model $M$ at state $s$?"
  - when $s$ is the initial state: $M \models \varphi$
  - Irrelevant whether $\varphi$ is true/false at other states

- Temporal Connectives:
  - $A,E$: range over paths from $s$
  - $G,X,F,U$: range over states on a path from $s$
\[ \top, \bot, p \]

- \( M,s \models \top \) and not \( M,s \models \bot \) for all \( s \in S \)

- \( M,s \models p \) iff \( p \in L(s) \)
\[ M,s \models \neg \varphi \text{ iff not } M,s \models \varphi \]
\( \land, \lor \)

- \( M,s \models \varphi_1 \land \varphi_2 \iff M,s \models \varphi_1 \) and \( M,s \models \varphi_2 \)
- \( M,s \models \varphi_1 \lor \varphi_2 \iff M,s \models \varphi_1 \) or \( M,s \models \varphi_2 \)
\[ M,s \models \varphi_1 \rightarrow \varphi_2 \iff \text{not } M,s \models \varphi_1 \text{ or } M,s \models \varphi_2 \]
AX, EX

- AX
  - $M,s \models AX \varphi$ iff for all $s'$ such that $s \rightarrow s'$ we have $M,s' \models \varphi$
  - “For all paths, for the next state, $\varphi$ is true”

- EX
  - $M,s \models EX \varphi$ iff for some $s'$ such that $s \rightarrow s'$ we have $M,s' \models \varphi$
  - “For some path, for the next state, $\varphi$ is true”
M, s |= AG φ iff for all paths s_1 → s_2 → s_3 → ... where s_1 equals s and for all s_i along the path, we have M, s_i |= φ

“For all paths, for all states along each path, φ is true”
\[ \text{EG} \]

- \( M, s \models \text{EG} \varphi \) iff for some path \( s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ... \) where \( s_1 \) equals \( s \) and for all \( s_i \) along the path, we have \( M, s_i \models \varphi \)
  - “For some path, for all states along the path, \( \varphi \) is true”
M,s \models AF \phi \text{ iff for all paths } s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots \text{ where } s_1 \text{ is } s \text{ and for some } s_i \text{ along each path, we have } M,s_i \models \phi

- “For all paths, for some state along each path, } \phi \text{ is true“}
EF

- $M, s \models EF \varphi$ iff for some path $s_1 \to s_2 \to s_3 \to \ldots$ where $s_1$ equals $s$ and for some $s_i$ along the path, we have $M, s_i \models \varphi$
  - “For some path, for some state along each path, $\varphi$ is true”
M, s \models A [\varphi_1 \text{ U } \varphi_2] \text{ iff for all paths } s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots \text{ where } s_1 \text{ equals } s, \text{ each path satisfies } \varphi_1 \text{ U } \varphi_2 \text{ (i.e. there is some } s_i \text{ along the path such that } M, s_i \models \varphi_2 \text{ and for each } j < i \text{ we have } M, s_j \models \varphi_1)\n
“\text{For all paths, for every state in each path } \varphi_1 \text{ until } \varphi_2”
EU

- \( M, s \models E [\varphi_1 U \varphi_2] \) iff for some path \( s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow... \)
  where \( s_1 \) equals \( s \), the path satisfies \( \varphi_1 U \varphi_2 \)
  (i.e. there is some \( s_i \) along the path such that \( M, s_i \models \varphi_2 \) and for each \( j < i \) we have \( M, s_j \models \varphi_1 \))
- “For some path \( \varphi_1 \) until \( \varphi_2 \)”
Inclusion of “s” in Condition

- “s” is the first state checked
  - For G, F, U
  - But not for X

Examples:
- \( M \models AF (n_1 \land n_2) \)
- \( M \models EG \neg(n_1 \land n_2) \)
- \( M \models A [\perp U (n_1 \land n_2)] \)

To exclude “s”, use X \( \varphi \)
Mutual Exclusion Properties

- **Safety:**
  - Only one process shall be in its critical section at any time
  - $AG \neg(c_1 \land c_2)$
- **Liveness:**
  - Whenever any process wants to enter its critical section, it will eventually be permitted to do so
  - $AG (t_1 \rightarrow AF c_1) \land AG (t_2 \rightarrow AF c_2)$
- **Non-blocking**
  - A process can always request to enter its critical section
  - $AG (n_1 \rightarrow EX t_1) \land AG (n_2 \rightarrow EX t_2)$
- **No strict sequencing:**
  - Processes need not enter their critical section in strict sequence
  - $EF (c_1 \land E[c_1 U (\neg c_1 \land E[\neg c_2 U c_1])]) \lor$
    $EF (c_2 \land E[c_2 U (\neg c_2 \land E[\neg c_1 U c_2])])$
Checking Algorithm

- Minimal Set of Connectives
- Algorithm
- Correctness
- Complexity
- Implementation
Minimal Set of Connectives

- Two CTL formulas $\varphi$ and $\psi$ are semantically equivalent iff any state in any model which satisfies one of them also satisfies the other
  - De Morgan’s Law
    - $\neg AF \varphi = EG \neg \varphi$
    - $\neg EF \varphi = AG \neg \varphi$
  - Minimal Set of Connectives: $\land, \neg, \bot, AF, EX, EU$
    - Translate: $AG, EG, EF, AX, AU$
    - For $AG$: $AG \varphi = \neg EF \neg \varphi$
    - For $EG$: $EG \varphi = \neg AF \neg \varphi$
    - For $EF$: $EF \varphi = E [\bot U \varphi]$
    - For $AX$: $AX \varphi = \neg EX \neg \varphi$
    - For $AU$: $A [\bot U \varphi] = AF \varphi$
Algorithm

- **Input:** The model $M$ and the CTL formula $\varphi$
- **Output:** The set of states of $M$ that satisfy $\varphi$
- **Steps:**
  - Translate $\varphi$ to $\varphi'$ where $\varphi'$ only has connectives in the minimal set
  - Label the states of $M$ with the sub formulas of $\varphi$ that are satisfied there, starting with the smallest sub formulas and working outwards towards $\varphi$
  - If $s_0$ is an element of the output, then “yes”
\( \bot \): then no states are labeled with \( \bot \)

\( \top \): then all states are labeled with \( \top \)
P, ¬

- $p$: then label $s$ with $p$ if $p \in L(s)$
- $\neg \psi_1$: label $s$ with $\neg \psi_1$ if $s$ is not already labeled with $\psi_1$
\( \land, \lor \)

- \(\psi_1 \land \psi_2\): label \(s\) with \(\psi_1 \land \psi_2\) if \(s\) is already labeled with both \(\psi_1\) and \(\psi_2\)
- \(\psi_1 \lor \psi_2\): label \(s\) with \(\psi_1 \lor \psi_2\) if \(s\) is already labeled with \(\psi_1\) or \(\psi_2\)
EX

- EX $\psi_1$: label any state with EX $\psi_1$ if one of its successors is labeled with $\psi_1$
AF

- AF $\psi_1$:
  - If any state $s$ is labeled with $\psi_1$, label it with AF $\psi_1$
  - Repeat: label any state with AF $\psi_1$ if all successor states
EU

- $E[\psi_1 \cup \psi_2]$:
  - If any state $s$ is labeled with $\psi_2$, label it with $E[\psi_1 \cup \psi_2]$
  - Repeat: label any state with $E[\psi_1 \cup \psi_2]$ if it is labeled with $\psi_1$ and at least one of its successors is labeled with $E[\psi_1 \cup \psi_2]$, until there is no change
Correctness: Termination

- Repeat until no change of AF and EU
  - Required since algorithm may add states and existence of states is part of condition
- Problem: “repeat” may not terminate
- Show that the functions for AF and UE terminate
  - Show that \( F_0 (F_1 (\ldots F_n (S))) = F_0 (F_1 (\ldots F_{n+1} (S))) \) for some \( n \)
Fixpoints

- **Given:** $F$ is a function $F: P(S) \rightarrow P(S)$

- **Fixpoint Sets**
  - A subset $X$ of $S$ is called a fixpoint of $F$ if $F(X) = X$
  - If we prove “repeat” has a fixpoint then we’ve proved “repeat” terminates

- **Known Theorem:**
  Every monotone function has a fixpoint
  - Is “repeat” monotone?
Monotone Functions

- Monotone Functions:
  - $F$ is monotone
    iff $X \subseteq Y$ implies $F(X) \subseteq F(Y)$ for all subsets $X$ and $Y$ of $S$

- $F_{AF}$ is monotone
  - $X, Y$ are the set of states with a label $AF_{\varphi}$
  - $F_{AF}$ only adds states, that is $F_{AF}(Z) = Z \cup \{\ldots\}$
  - Condition for what is in $\{\ldots\}$ is dependent on $Z$
  - “More states in $Z$, then more potential for adding states”
  - Since $X$ is “contained” in $Y$, then $Y$ has all the potential of $X$ (i.e. $F_{AF}(X) = F_{AF}(Y)$)
  - And if $X$ is smaller than $Y$, then $Y$ has more potential than $X$ (i.e. $F_{AF}(X) \subseteq F_{AF}(Y)$)
  - So if $X \subseteq Y$ then $F_{AF}(X) \subseteq F_{AF}(Y)$

- $F_{EU}$ is similarly monotone
Complexity

- This Algorithm: $O(f \times V \times (V + E))$
  - $f$ is the number of connectives in the formula
  - $V$ is the number of states
  - $E$ is the number of transitions
  - “linear in the size of the formula and quadratic in the size of the model”

- Better Algorithms: $O(f \times (V + E))$
Complexity: State Explosion

- Problem is size of model, not algorithm
  - Size of model \((V + E)\) is exponential in the number of variables (or properties on them)
  - Size of model \((V+E)\) is exponential in the number of components that can execute in parallel
Implementations

- SMV
  - Model Checker
  - Available from CMU
  - Created by K. McMillan
- NuSMV
  - Reimplementation
- Cadence SMV
  - Reimplementation + Compositional Focus