Introduction to Software Verification

Orna Grumberg

Lectures Material
winter 2015-16
Lecture 8
Explicit Model Checking for Fair CTL
Fair CTL ($\text{CTL}^F$)

- Same syntax as CTL
- Different semantics

$\text{CTL}^F$ formulas are interpreted over fair Kripke structures
Fair Kripke Structures

Fair Kripke structure \( M = (S, S_0, R, L, F) \)

- \( S, S_0, R, L \) - as before
- \( F \subseteq 2^S \) is a set of fairness constraints

- \( F = \{P_1, \ldots, P_k\} \) where
  - \( P_i \subseteq S \)
  - or
  - \( P_i \) is a CTL formula
Fairness

Fair paths:

• $\pi = s_0, s_1, s_2, ...$
• $\text{inf} (\pi) = \{ s \mid s = s_i \text{ for infinitely many } i \}$

$\pi$ is fair if for every $P \in F$, $\text{inf}(\pi) \cap P \neq \emptyset$
Semantics of Fair CTL

- $M, s \models_F \text{EX } \psi \iff$ there exists a fair path $\pi = s_0, s_1, \ldots$ from $s$ such that $M, s_1 \models_F \psi$

- $M, s \models_F \text{AX } \psi \iff$ for every fair path $\pi = s_0, s_1, \ldots$ from $s$, $M, s_1 \models_F \psi$

- Similarly for $\text{EG, AG, EU, AU,} \ldots$
Model checking Fair CTL

• Needs to consider only fair paths

• \( g \in \text{label}(s) \iff M,s \models F g \)
Reminder: Model Checking $g = EG f_1$
without fairness

Observation:
• $s \models EG f_1$
  iff
• $s$ is the start of a path where all states satisfy $f_1$
  iff
• $s$ has a finite path to a Strongly Connected Component (SCC), where all states satisfy $f_1$
Model Checking $g = \text{EG } f_1$

with fairness

Observation:

- $s \models F \text{EG } f_1$ iff
- $s$ is the start of a fair path where all states satisfy $f_1$
  iff
- $s$ has a finite path to a Fair Strongly Connected Component (FSCC), where all states satisfy $f_1$
$M,s \models_F \text{EG } f_1$

Strongly connected component $C$ is fair iff for every $P \in F$, $C \cap P \neq \emptyset$

Reduced structure:
Remove from $M$ all states s.t. $f_1 \not\in \text{label}(s)$.

Resulting model: $M' = (S', R', L', F')$
- $S' = \{ s \mid M, s \models_F f_1 \}$
- $R'$, $L'$ defined as before
- $F' = \{ P_i \cap S' \mid P_i \in F \}$
\( M, s \models_{F} EG f_1 \)

**Theorem:** \( M, s \models_{F} EG f_1 \) iff

1. \( s \in S' \) and
2. There is a path in \( M' \) from \( s \) to some state \( t \) in a nontrivial maximal fair strongly connected component of \( M' \)

**Proof:** similar to theorem for EG without fairness
$M, s \models_{F} EG f_1$

procedure **CheckFairEG** ($f_1$)

$S' := \{ s \mid f_1 \in \text{label}(s) \}$

$SCC := \{ C \mid C \text{ is a nontrivial fair SCC of } M' \}$

$T := \bigcup_{C \in SCC} \{ s \mid s \in C \}$

For all $s \in T$ do $\text{label}(s) := \text{label}(s) \cup \{ EG f_1 \}$

while $T \neq \emptyset$ do

choose $s \in T$; $T := T \setminus \{ s \}$

for all $t \in S'$ s.t. $R(t,s)$ do

if $EG f_1 \notin \text{label}(t)$ then

$\text{label}(t) := \text{label}(t) \cup \{ EG f_1 \}$;

$T := T \cup \{ t \}$

end for all

end while

**Complexity increases to** $O((|S| + |R|) \cdot |F|)$
$M, s \models_F EGa$ with $F = \{ \{1, 2\}, \{1, 5\} \}$

$F' = \{ \{1\} \}$

$MSCC = \{ \{1\}, \{3, 4\}, \{6\} \}$

$FMSCC = \emptyset$

Set of states satisfying $E_F Ga$ is empty
Model Checking other Fair CTL Formulas

• Add a new atomic proposition fair to all states which are a start of a fair path
• \(\text{fair} \in \text{label(s)} \iff M,s \models F \text{EG true}\)

\[
M,s \models F \text{ EX } f_1 \iff M,s \models \text{EX } (f_1 \land \text{fair})
\]

\[
M,s \models F \text{ E } [f_1 \text{ U } f_2] \iff M,s \models \text{E } [f_1 \text{ U } (f_2 \land \text{fair})]
\]

Overall complexity: \(O(|f| \cdot (|S| + |R|) \cdot |F|)\)
Computing $S_{\text{fair}}$ with $F=\{\{1,2\}, \{1,5\}\}$

MSCC = $\{\{0\}, \{1\}, \{2,3,4,5\}, \{6\}\}$

FMSCC = $\{\{2,3,4,5\}\}$

$S_{\text{fair}} = \{2,3,4,5\} \cup \{0,1\}$
$M, s \models_{F} E(aUc) \text{ with } F=\{ \{1,2\}, \{1,5\} \}$

$S_{\text{fair}} = \{0,1,2,3,4,5\}$

$E_{F}(aUc) \equiv E(aU(c \wedge \text{fair}) = \varphi$

$S_{a} = \{1,3,4,6\}$  $S_{c} = \{0,3,6\}$

$S_{\text{fair} \wedge c} = \{0,3\}$

$S_{\varphi} = \{0,3,4\}$
Microwave Example

Fairness:
\[ \text{Start} \land \text{Close} \land \neg \text{Error} \]
Property

- $AG \ (\text{start} \rightarrow \text{AF Heat})$
- $\neg EF \ (\text{start} \land E\text{G} \neg \text{Heat})$
- $\neg E \ (\text{true U (start} \land E\text{G} \neg \text{Heat}))$

Now we check it with respect to a fair Kripke structure
$\neg (true \ U (\text{start} \land EG \ \neg \text{Heat}))$
\neg E (\text{true} \ U (\text{start} \land EG \neg \text{Heat}))

All states belong to the same nontrivial fair MSCC
\Rightarrow All states labeled fair
\[ E \ (\text{true} \ U \ (\text{start} \ \land \ EG \ \neg \text{Heat})) \]

\[ S(\text{Start}) : \{2,5,6,7\} \]
\[ S(\neg \text{Heat}) : \{1,2,3,5,6\} \]
\[ S(\neg \text{EG Heat}) : \emptyset \]

Not fair
$\neg E \left( \text{true} \cup (\text{start} \land \text{EG} \neg \text{Heat}) \right)$

$S(\text{Start}) : \{2,5,6,7\}$  
$S(\neg \text{Heat}) : \{1,2,3,5,6\}$  
$S(\text{EG} \neg \text{Heat}) : \emptyset$

$S(\text{Start} \land \text{EG} \neg \text{Heat}) : \emptyset$  
$S(\text{EU}) : \emptyset$  
$S(\text{f}) : \{1,2,3,4,5,6,7\}$
Model Checking

- Emerging as an industrial standard tool for verification of hardware designs: Intel, IBM, Cadence (Jasper), Synopsys,…

- Recently applied successfully also for software verification: SLAM (Microsoft), Java PathFinder and SPIN (NASA), BLAST (EPFL), CBMC (Oxford), ExpliSAT (IBM, Haifa)…
Clarke, Emerson, and Sifakis received the 2007 Turing award for their contribution to Model Checking
Main Limitation of Model Checking:

The state explosion problem:

Model checking is efficient in time but suffers from high space requirements:

The number of states in the system model grows exponentially with
- the number of variables
- the number of components in the system
Solutions to the state-explosion problem

Symbolic model checking:
The model is represented symbolically

- BDD-based model checking
- SAT-based Bounded Model Checking (BMC)
- SAT-based Unbounded Model Checking
Other solutions to the state-explosion problem

Small models replace the full, concrete model:

- Abstraction
- Compositional verification
- Partial order reduction
- Symmetry
Symbolic (BDD-based) Model Checking for CTL
BDD-based Symbolic Model Checking

A solution to the state explosion problem: BDD-based model checking

- Binary Decision Diagrams (BDDs) are used to represent the model and sets of states.

- It can handle systems with hundreds of Boolean variables.
Binary Decision Diagrams (BDDs)

- Data structure for representing Boolean functions

- Boolean function:

\[
f: \{0,1\}^k \rightarrow \{0,1\}
\]

\[
f(x_1, ..., x_k) = x_{k+1}
\]

where \( x_1, ..., x_k, x_{k+1} \in \{0,1\} \)
Binary Decision Diagrams (BDDs)

Advantages of BDDs:

- Often (but not always) concise in size
- Canonical representation
- Most Boolean operations can be performed on BDDs in polynomial time in the BDD size
BDD for \( f(a, b, c) = (a \land b) \lor c \)

Decision tree

BDD
BDDs in Model Checking

- Every set $A \subseteq U$ can be represented by its characteristic function
  
  $$f_A(u) = \begin{cases} 
  1 & \text{if } u \in A \\
  0 & \text{if } u \notin A 
  \end{cases}$$

- If the elements of $U$ are encoded by sequences over $\{0,1\}^n$ then $f_A$ is a Boolean function and can be represented by a BDD
• A Boolean function represents the set of all elements for which the function is 1
Representing a Model with BDDs

• Assume that states in model $M$ are encoded by $\{0, 1\}^n$ and described by Boolean variables $v_1 \ldots v_n$

• $S_f$ can be represented by a Boolean function (BDD) over $v_1 \ldots v_n$

• $R$ (a set of pairs of states $(s, s')$) can be represented by a BDD over $v_1 \ldots v_n$ $v_1' \ldots v_n'$
Example: Representing a Model with BDDs

\[ S = \{ s_1, s_2, s_3 \} \]
\[ R = \{ (s_1,s_2), (s_2,s_2), (s_3,s_1) \} \]

**State encoding:**

\[ s_1: \quad v_1v_2=00 \]
\[ s_2: \quad v_1v_2=01 \]
\[ s_3: \quad v_1v_2=11 \]

For \( A = \{s_1, s_2\} \) the Boolean formula representing \( A \):

\[ f_A(v_1,v_2) = (\neg v_1 \land \neg v_2) \lor (\neg v_1 \land v_2) = \neg v_1 \]
\[ f_R(v_1, v_2, v'_1, v'_2) = \]
\[ (\neg v_1 \land \neg v_2 \land \neg v'_1 \land v'_2) \lor \]
\[ (\neg v_1 \land v_2 \land \neg v'_1 \land v'_2) \lor \]
\[ (v_1 \land v_2 \land \neg v'_1 \land \neg v'_2) \]

\[ f_A \text{ and } f_R \text{ can be represented by BDDs.} \]