Behavior models and verification

Lecture 7

Jan Kofroň, František Plášil
Symbolic CTL model checking

Markov chains
Timed automata
Labelled transition system
Kripke structure

Model

Property specification

CTL
LTL
TCTL
PCTL

AG(start → AF heat)

Model checker

Property satisfied
Property violated
Error report

Jan Kofroň, František Plášil, Lecture 7
Today

- **Lattices**
- **Fixpoints**
- **Symbolic model checking CTL using OBDD**
  - and lattices
  - and fixpoints
Let $S$ be a set

The set $P(S)$ of all subsets of $S$ forms a **complete lattice**

Each element $S'$ of the lattice can also be thought as a **predicate on $S$**

The least element is $\emptyset$ (false), the greatest element is $S$ (true)

A function that maps $P(S)$ to $P(S)$ is called a **predicate transformer**
Let $\tau: P(S) \to P(S)$ be a predicate transformer

$\tau$ is **monotonic** provided that

$Q \subseteq R$ implies $\tau(Q) \subseteq \tau(R)$

$Q$ is a fixpoint of $\tau$ iff $\tau(Q) = Q$
Theorem (Knaster-Tarski): A monotonic predicate transformer $\tau$ on $P(S)$ always has the least fixpoint, $\mu Z. \tau(Z)$, and the greatest fixpoint, $\nu Z. \tau(Z)$

- $\mu Z. \tau(Z) = \cap \{Z | \tau(Z) \subseteq Z\}$
- $\nu Z. \tau(Z) = \cup \{Z | \tau(Z) \supseteq Z\}$
Fixpoint representations

- We write \( \tau^i(Z) \) to denote \( i \) applications of \( \tau \) to \( Z \)
  - \( \tau^0(Z) = Z, \tau^{i+1}(Z) = \tau(\tau^i(Z)) \)

- **Lemma:** If \( \tau \) is monotonic, then for every \( i \):
  - \( \tau^i(\text{false}) \subseteq \tau^{i+1}(\text{false}) \)
  - \( \tau^i(\text{true}) \supseteq \tau^{i+1}(\text{true}) \)

- **Lemma:** If \( \tau \) is monotonic and \( S \) finite, then:
  - there is an integer \( i_0 \) s.t. for every \( i \geq i_0 \),
    \( \tau^i(\text{false}) = \tau^{i_0}(\text{false}) \)
  - there is an integer \( j_0 \) s.t. for every \( j \geq j_0 \):
    \( \tau^j(\text{true}) = \tau^{j_0}(\text{true}) \)

- **Lemma:** If \( \tau \) is monotonic and \( S \) finite, then:
  - there is an integer \( i_0 \) s.t. \( \mu Z. \tau(Z) = \tau^{i_0}(\text{false}) \)
  - there is an integer \( j_0 \) s.t. \( \nu Z. \tau(Z) = \tau^{j_0}(\text{true}) \)
Fixpoint representations

- We are interested only in **finite** Kripke structures → finite $S$

- The least and greatest fixpoints of a monotonic predicate transformer can be computed using the following algorithms
function Lfp(tau : PredicateTransformer): Predicate
    Q := false;
    Q' = tau(Q);
    while (Q <> Q') do
        Q := Q';
        Q' := tau(Q);
    end while;
    return Q;
end function

function Gfp(tau : PredicateTransformer): Predicate
    Q := true;
    Q' = tau(Q);
    while (Q <> Q') do
        Q := Q';
        Q' := tau(Q);
    end while;
    return Q;
end function
CTL operators as fixpoints

- We identify a CTL formula $f$ with the set/predicate
  \[ \{ s \mid M, s \models f \} \] in $P(S)$
- EG, EU may be characterized as least or greatest fixpoints of an appropriate predicate transformer:
  - $EG \, q = \nu Z (q \land EX \, Z)$
  - $E[p \, U \, q] = \mu Z (q \lor (p \land EX \, Z))$

The same holds for EF, AG, AF, AU, however those operators can be expressed using EG, EU

Intuitively:
- least fixpoints correspond to eventualities
- greatest fixpoints correspond to properties that should hold forever
EG as fixpoint

Kripke structure $M$

$\tau^0(\text{true})$

$\tau^1(\text{true})$

$M, s_0 \models EG \ q$

$EG \ q = \forall Z. (q \land EX \ Z)$

$\tau(Z) = \{s: s \models q \land (\exists t: s \rightarrow t \land t \in Z)\}$
EU as fixpoint

Kripke structure $M$

$\tau^1(\text{false})$

$\tau^2(\text{false})$

$\tau^3(\text{false})$

$M, s_0 \models E[p \ U \ q]$

$E[p \ U \ q] = \mu Z. (q \lor (p \land EX Z))$

$\tau(Z) = \{ s : s \models q \} \lor \{ s : s \models p \land (\exists t : s \rightarrow t \land t \in Z) \}.$
Symbolic model checking for CTL

- Explicit state model checking (presented earlier) is linear in size of Kripke structure and length of formula
  - State explosion problem

- Symbolic model checking algorithm operates on Kripke structures represented using OBDDs
Symbolic model checking for CTL

- Quantified Boolean formulae
  - Instead of common Boolean operators, we have (for a variable \( x \) and a formula \( f \))
    - \( \exists x \ f \)
    - \( \forall x \ f \)
  - The same expressive power as ordinary propositional formulae

- Using OBDDs, the quantification operators can be implemented as
  - \( \exists x : f = f \mid_{x \leftarrow 0} \lor f \mid_{x \leftarrow 1} \)
  - \( \forall x : f = f \mid_{x \leftarrow 0} \land f \mid_{x \leftarrow 1} \)
Symbolic model checking for CTL

• On top level the same approach as in explicit model checking algorithm
  - Decomposing formula into sub-formulae and checking them in bottom-up manner

• Different handling of particular sub-formulae
  - Based on Check*() procedures
Symbolic model checking for CTL

- **Check(CTLFormula f)**
  - $f$ is an atomic proposition $p \rightarrow$ return the OBDD for $p$
  - $f = \neg f_1$ or $f = f_1 \land f_2$ or $f = f_1 \lor f_2 \rightarrow$ use the function Apply (*) and return the resulting OBDD

- Formulae of the form $\text{EX } f \rightarrow$ return $\text{CheckEX}(\text{Check}(f))$
  - $\text{CheckEX(OBDD } o)$
  - $o$ represents the formula $f$
  - $\text{CheckEX}(o(<v>)) = \exists<v'>[o(<v'>) \land R(<v>, <v'>)]$
  - $R$ is the OBDD representing the transition relation

- Formulae of the form $E[f U g] \rightarrow$ $\text{CheckEU}(\text{Check}(f), \text{Check}(g))$
  - Based on the fixpoint characterization of EU
    $E[f U g] = \mu Z.(g \lor (f \land \text{EX } Z))$
  - Uses the Lfp procedure

- Formulae of the form $\text{EG } f \rightarrow$ $\text{CheckEG}(\text{Check}(f))$
  - Based on the fixpoint characterization of EG
    $\text{EG } f = \nu Z.(f \land \text{EX } Z)$
  - Uses the Gfp procedure
Example of symbolic CTL model checking

$AF \ x = \neg EG (\neg x)$

Jan Kofroň, František Plášil, Lecture 7
Example of symbolic CTL model checking

- $EG \neg x = \forall Z. (\neg x \land EX Z)$

- $\tau(Z) = \{ s : s \models \neg x \land (\exists t : s \rightarrow t \land t \in Z) \}$

- We start with $Z$ as the set of all states \textit{(true)}:

- In each iteration, we conjunct predecessors of $Z$ with the set of states satisfying $\neg x$
Example of symbolic CTL model checking

\[
\neg x \land (\exists x_0', x_1': Z' \land TR)
\]
Example of symbolic CTL model checking

\[ \neg x \land (\exists x_0', x_1': Z' \land TR) \]
Example of symbolic CTL model checking

\( \neg x \land Z \land x_0 \Rightarrow 0 \rightarrow 1 \)
Example of symbolic CTL model checking

Fixpoint reached $\rightarrow$ proceed upwards to: $\neg EG(\neg x)$:

```
x0
\[
\begin{array}{c}
0 \\
1
\end{array}
\]

\[
\begin{array}{c}
0 \\
1
\end{array}
\]
```

\[
\{ \\
\begin{array}{c}
a,x \\
b \\
c,x \\
d
\end{array}
\}
\]
A step aside: Fairness constraints

Producer \hspace{1cm} \text{Reliable Channel} \hspace{1cm} \text{Consumer}

\begin{align*}
\text{AG}(&\text{Sent} \rightarrow \text{AF(OK)}) \ \\
M_1 &= (S_1, I_1, R_1, L_1)
\end{align*}
A step aside: Fairness constraints

- **Fairness constraint**
  - An arbitrary set of states (a subset of S), usually described as a CTL formula

- **Fair path**
  - **Each** fairness constraint is true **infinitely often** along the path
  - The path quantifiers in the logic are then restricted to fair paths
A step aside: Fairness constraints

- Fair Kripke structure
  - $M = (S, I, R, L, F)$
  - $S, I, R, L$ as usual
  - $F \subseteq 2^S$
    - generalized Büchi acceptance conditions

- Semantics of a fair Kripke structure
  - $M, s \models_F f$
    - only fair paths are considered
A step aside: Fairness constraints

We define a fairness constraint described by the CTL formula \( \neg \text{Idle} \), i.e.
\[
\{ s_1, s_2, s_4, s_5 \}
\]

\[
M_2 = (S_1, I_1, R_1, L_1, F)
\]
\[
F = \{ \{s_1, s_2, s_4, s_5\} \} 
\]
Explicit model checking of CTL with fairness constraints

- $M = (S, I, R, L, F)$
- $F = \{P_1, \ldots, P_k\}$
- Modification of the CTL explicit model checking algorithm
Explicit model checking of CTL with fairness constraints

- Handling EG $\varphi_1$
  - A strongly connected component $C$ of the graph $M$ is fair with respect to $F$ iff for each $P_i \in F$, there is a state $t_i \in (C \cap P_i)$
  - $M' = (S', I', R', L', F')$
    - $S' = \{ s \in S : M, s \models_F \varphi_1 \}$
    - $I' = I \cap S'$
    - $R' = R \mid_{S' \times S'}$
    - $L' = L \mid_{S'}$
    - $F' = \{ P_i \cap S' | P_i \in F \}$
  - **Lemma.** $M, s \models_F EG \varphi_1$ iff the following two conditions are satisfied:
    - $s \in S'$
    - There exists a path in $S'$ that leads from $s$ to some node $t$ in a nontrivial fair strongly connected component of the graph $(S', R')$
Explicit model checking of CTL with fairness constraints

- CheckFairEG($\varphi_1$)
  - Assumption: $\varphi_1 \in label(s) \iff M, S \models F \varphi_1$
  - Similar to CheckEG
    - Difference: SCC now consists of the set of nontrivial fair strongly connected components
Explicit model checking of CTL with fairness constraints

- \( \varphi_1 \) fairly holds
- \( \varphi_1 \) does not fairly hold
- = fairness constraint
Explicit model checking of CTL with fairness constraints

Construction of \((S', I', R', L', F')\)
Explicit model checking of CTL with fairness constraints

Identification of nontrivial fair strongly connected components
Explicit model checking of CTL with fairness constraints

= EG $\varphi_1$ fairly holds
Explicit model checking of CTL with fairness constraints

- Trick: a new atomic proposition $fair$
  - $fair$ is true in a state $s$ iff there is a fair path starting from $s$
  - $M, s \models_F EG \text{true}$
  - CheckFairEG(true)

- $M, s \models_F p \leftrightarrow M, s \models p \land fair$
- $M, s \models_F EX \varphi_1 \leftrightarrow M, s \models EX(\varphi_1 \land fair)$
- $M, s \models_F E[\varphi_1 U \varphi_2] \leftrightarrow M, s \models E[\varphi_1 U(\varphi_2 \land fair)]$
  - CheckEU($\varphi_1$, $\varphi_2 \land fair$)
Symbolic model checking of CTL with fairness constraints

- \( F = \{P_1, \ldots, P_n\} \)
- CheckFair
- Formulae of the form \( \text{EG } f \rightarrow \text{CheckFairEG} \)
  - uses the fixpoint characterization of \( \text{EG} \)
    - without fairness ... \( \text{EG } f = \nu Z. (f \land \text{EX } Z) \)
    - with fairness ... \( \text{EG } f = \nu Z. (f \land (\land_{k=1}^{n} \text{EX } E [f \land (Z \land P_k)])) \)
      - all of the states in \( Z \) satisfy \( f \), and
      - for all fairness constraints \( P_k \in F \) and all states \( s \in Z \), there is a sequence of states of the length one or greater from \( s \) to a state in \( Z \) satisfying \( P_k \) s.t. all states on the path satisfy \( f \)
Fairness in symbolic model checking

- $\text{fair} = \text{CheckFairEG}(\text{True})$
- Formulae of the form $\text{EX } f \rightarrow \text{CheckFairEX}$
  - $\text{CheckFairEX}(f(<v>)) = \text{CheckEX}(f(<v>) \land \text{fair})$
- Formulae of the form $E [f U g] \rightarrow \text{CheckFairEU}$
  - $\text{CheckFairEU}(f(<v>), g(<v>)) =$
    $\text{CheckEU}(f(<v>), g(<v>) \land \text{fair})$