Behavior models and verification

Lecture 7

http://d3s.mff.cuni.cz

Jan Kofroň, František Plášil
Symbolic CTL model checking

Property specification

\[ AG(\text{start} \rightarrow AF \text{heat}) \]

Model

Markov chains
Timed automata
Labelled transition system
Kripke structure

Model checker

Property satisfied

Error report

Property violated
Today

- Lattices
- Fixpoints
- Symbolic model checking CTL using OBDD
  - and lattices
  - and fixpoints
Lattice

- **Lattice** is a structure consisting of a partially ordered set $S$ of elements where every two elements have a unique supremum (least upper bound or join) and a unique infimum (greatest lower bound or meet).
- 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 $\tau: P(S) \rightarrow P(S)$ 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 $i$:
  - $\tau^i(false) \subseteq \tau^{i+1}(false)$
  - $\tau^i(true) \supseteq \tau^{i+1}(true)$

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

- **Lemma:** If $\tau$ is monotonic and $S$ finite, then:
  - $\exists i_0: \mu Z. \tau(Z) = \tau^{i_0}(false)$
  - $\exists j_0: \nu Z. \tau(Z) = \tau^{j_0}(true)$
Fixpoint representations

- We are interested only in finite Kripke structures \( \rightarrow \) finite \( S \)

- The least and greatest fixpoints of a monotonic predicate transformer can be computed using the following algorithms
Fixpoint representations

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

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

Intuitively:
\[ \begin{align*}
\text{least fixpoints correspond to eventualities} \\
\text{greatest fixpoints correspond to properties that should hold forever}
\end{align*} \]
EG as fixpoint

Kripke structure $M$

$M, s_0 \models EG\ q$

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

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

Kripke structure $M$

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

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

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

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

$\tau(Z) = \{s: s \models q\} \lor \{s: s \models p \land (\exists t: s \rightarrow t \land t \in Z)\}$
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
  - $\exists x : f = f|_{x \leftarrow 0} \lor f|_{x \leftarrow 1}$
  - $\forall x : f = f|_{x \leftarrow 0} \land f|_{x \leftarrow 1}$
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 \( p \) → return the OBDD for \( p \)
  - \( f = \neg f_1 \) or \( f = f_1 \land f_2 \) or \( f = f_1 \lor f_2 \) → use the function Apply (*) and return the resulting OBDD

- Formulae of the form \( \text{EX } f \) → return CheckEX(Check(f))
  - CheckEX(OBDD o)
  - o represents the formula f (set of states satisfying 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 \( \text{E}[f \ U \ g] \) → CheckEU(Check(f), 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 \( \text{EG } f \) → CheckEG(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

TR:  

\[ \neg x : \]

\[ \neg x : \]

\[ AF \ x = \neg EG(\neg x) \]

\[ \neg x \]
Example of symbolic CTL model checking

- \( \mathcal{E}G \neg x = \forall Z. (\neg x \land E X Z) \)

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

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

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

\[ \neg x \land (\exists x_0', x_1' : Z' \land TR) \]

\[ \bigwedge \]

\[ (a, x \rightarrow b, c, x \rightarrow d) \]

\[ (x_0 \rightarrow 0 \rightarrow x_0', x_0' \rightarrow 1 \rightarrow x_1', x_1' \rightarrow 1 \rightarrow x_1, x_1 \rightarrow 0 \rightarrow x_0, x_0 \rightarrow 1 \rightarrow x_1) \]

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

\( \neg x \land (\exists x_0', x_1': Z' \land TR) \)

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

\[ \neg x \land Z \land (x0 \rightarrow 0 \lor 1) \rightarrow (x0 \rightarrow 0 \lor 1) \]
Fixpoint reached → proceed upwards to:
\( \neg E \neg G (\neg x) \):
A step aside: Fairness constraints

Producer

Reliable Channel

Consumer

`AG(Sent → AF(OK))` ???

\[ M_1 = (S_1, I_1, R_1, L_1) \]
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

\[ \neg \text{Idle}, \text{i.e.} \{ s_1, s_2, s_4, s_5 \} \]

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

- \[ M = (S, I, R, L, F) \]
- \[ F = \{ P_1, ..., P_k \} \]
- 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: \(\varphi_1 \in \text{label}(s) \iff M, S \models F \varphi_1\)
  - 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

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

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

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

- \( F = \{P_1, \ldots, P_n\} \)
- CheckFair
- Formulae of the form \( EG \ f \rightarrow \text{CheckFair}EG \)
  - uses the fixpoint characterization of \( EG \)
    - without fairness ... \( EG \ f = \nu Z. (f \land EX Z) \)
    - with fairness ... \( EG \ f = \nu Z. (f \land (\land_{k=1}^{n} EX E [f U (Z \land P_k)])) \)
      - 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

- fair = CheckFairEG(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 $E[f U g] \rightarrow \text{CheckFairEU}$
  - $\text{CheckFairEU}(f(<v>), g(<v>)) = \text{CheckEU}(f(<v>), g(<v>) \land \text{fair})$