

Algorithms for Model Checking (2IW55)

Lecture 3

Symbolic Model Checking for CTL

Chapter 2, 6.1, 6.2. Also read Chapter 5

Tim Willemse

(timw@win.tue.nl)

<http://www.win.tue.nl/~timw>

HG 6.76

Specification of Kripke Structures

Fixed Points

Symbolic Model Checking

Implementing Symbolic Model Checking

Example (GCD)

Consider the following program:

```
repeat
  if  $x > y$   $\rightarrow$   $x := x - y$ ;
  []  $x < y$   $\rightarrow$   $y := y - x$ ;
fi
until false
```

This program uses:

- ▶ **variables**: $\{x, y\}$, with an (implicit) **domain** of variables: \mathbb{N}
- ▶ **States** of this program are **functions** of type: $\{x, y\} \rightarrow \mathbb{N}$
- ▶ An example state could be: $\{x \mapsto 5, y \mapsto 15\}$
- ▶ An **execution** is a sequence of transitions: e.g.

$$\{x \mapsto 5, y \mapsto 15\} \rightarrow \{x \mapsto 5, y \mapsto 10\} \rightarrow \{x \mapsto 5, y \mapsto 5\} \rightarrow \{x \mapsto 5, y \mapsto 5\} \rightarrow \dots$$

Example (SWAP)

Consider the following program fragment:

```
z := x;    % l1
x := y;    % l2
y := z;    % l3
```

- ▶ Besides variables $x, y, z : \mathbb{N}$, this program has a **program counter**, whose values are **labels** (line numbers)
- ▶ Let $pc : \{l_1, l_2, l_3\}$. Now, a state is a function that gives a value to $\{x, y, z, pc\}$
- ▶ A possible **execution** is the following sequence:

$$\begin{aligned} & \{x \mapsto 5, y \mapsto 15, z \mapsto 500, pc \mapsto l_1\} \\ \rightarrow & \{x \mapsto 5, y \mapsto 15, z \mapsto 5, pc \mapsto l_2\} \\ \rightarrow & \{x \mapsto 15, y \mapsto 15, z \mapsto 5, pc \mapsto l_3\} \\ \rightarrow & \{x \mapsto 15, y \mapsto 5, z \mapsto 5, pc \mapsto l_4\} \end{aligned}$$

Symbolic Representation

- ▶ Note: in general, there are **infinitely many states and transitions**. Even after restricting to MAXINT, the number often still is overwhelming.
- ▶ However, many of the states behave very similar (e.g. the start value of z did not matter)
- ▶ Idea: the set of states can be represented very **concisely** by a number of formulae
- ▶ for GCD:
 - initial set of states: $x < 100 \wedge y < 100$
 - next state predicate:

$$(x > y \wedge x' = x - y \wedge y' = y) \vee (x < y \wedge y' = y - x \wedge x' = x)$$

- ▶ for SWAP:
 - initial states: $x = 5 \wedge y = 15$
 - next state predicate:
- $$(pc = l_1 \wedge pc' = l_2 \wedge z' = x \wedge \dots) \vee \dots$$

The system specification is represented by **first-order formulae** (later: propositional logic only)

- ▶ Let V be a set of **variables** v_0, v_1, \dots, v_n
- ▶ Let D be the **domain** of these variables
- ▶ The **states** of the Kripke Structure will be functions $v : V \rightarrow D$
- ▶ A formula $S_0(V)$ represents the initial states
- ▶ Let V' be a copy of the variables in V : v'_0, v'_1, \dots, v'_n
- ▶ A formula $\mathcal{R}(V, V')$ represents the transition relation.
 - V denotes the value of the variables **before** the transition
 - V' denotes the value of the variables **after** the transition.

Example

- ▶ $V = \{E(mma), L(inus)\}$,
- ▶ $D = \{p(laying), q(uestioning), a(nswered)\}$
- ▶ $S_0(E, L) := E = p \wedge L = p$
- ▶ $\mathcal{R}(E, L, E', L') := R_1 \vee R_2 \vee R_3 \vee R_4 \vee R_5 \vee R_6$, where:
 - $R_1 := E = p \wedge E' = q \wedge L' = L$
 - $R_2 := E = q \wedge E' = a \wedge L' = L \wedge L \neq a$
 - $R_3 := E = a \wedge E' = p \wedge L' = L$
 - $R_4 := L = p \wedge L' = q \wedge E' = E$
 - $R_5 := L = q \wedge L' = a \wedge E' = E \wedge E \neq a$
 - $R_6 := L = a \wedge L' = p \wedge E' = E$

Notes:

- ▶ this corresponds to the demanding children Kripke Structure in previous lectures
- ▶ a specification for n children gives $O(3^n)$ states \Rightarrow State space explosion

Specification of Kripke Structures

Fixed Points

Symbolic Model Checking

Implementing Symbolic Model Checking

Consider a Kripke Structure $M = \langle S, R, L \rangle$

- ▶ Identify **sets of states** and **predicates on states**
- ▶ So, two notations are often mixed:
 - subsets: $X \subseteq S$ or $X \in \mathcal{P}(S)$
 - predicates: $X \in 2^S$ or $X : S \rightarrow \{0, 1\}$
 $s \in X \Leftrightarrow X(s) = 1$ and $s \notin X \Leftrightarrow X(s) = 0$
- ▶ Also: CTL formulae are identified with the set of states where they hold: f versus $\{s \mid s \models f\}$
- ▶ As a consequence, \forall, \wedge and \cup, \cap are mixed: compare $\emptyset \cup E G f$ and $\text{false} \vee E G f$

Predicate Transformers and Monotonicity

Consider a Kripke Structure $M = \langle S, R, L \rangle$

- ▶ The set $(\mathcal{P}(S), \subseteq)$ is a partial order (AKA the **complete lattice of state predicates**)
- ▶ A **predicate transformer** is a function on predicates. For example, the relations *Pre* and *Post* that lift the transition relation *R* to **sets** of states:

$$\begin{aligned} \text{Pre}_R(X) &= \{s \in S \mid \exists t \in X. s R t\} \\ \text{Post}_R(X) &= \{t \in S \mid \exists s \in X. s R t\} \end{aligned}$$

- ▶ Let $\tau : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ be an arbitrary predicate transformer.
- ▶ τ is **monotonic** iff $P \subseteq Q$ implies $\tau(P) \subseteq \tau(Q)$.
- ▶ We write $\tau^i(X)$ for applying τ *i* times to *X*:

$$\begin{cases} \tau^0(X) &= X \\ \tau^{i+1}(X) &= \tau(\tau^i(X)) \end{cases}$$

Let $\tau : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$.

- ▶ A **fixed point** of τ is a set Z such that $\tau(Z) = Z$
- ▶ The **least fixed point** of τ , denoted $\mu X. \tau(X)$ is a set Z such that:
 - $Z = \tau(Z)$ (i.e. Z is a fixed point)
 - for all X , if $\tau(X) = X$, then $Z \subseteq X$
- ▶ The **greatest fixed point** of τ , denoted $\nu X. \tau(X)$ is a set Z such that:
 - $Z = \tau(Z)$ (i.e. Z is a fixed point)
 - for all X , if $\tau(X) = X$, then $X \subseteq Z$

A theorem by Tarski: a **monotonic** operator on $\mathcal{P}(S)$ always has least and greatest fixed points:

- ▶ $\mu Z. \tau(Z) = \bigcap \{X \mid \tau(X) \subseteq X\}$
- ▶ $\nu Z. \tau(Z) = \bigcup \{X \mid X \subseteq \tau(X)\}$

Assume now that:

- ▶ S (hence also $\mathcal{P}(S)$) is finite, and
- ▶ $\tau : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ is monotonic

Then:

1. $\forall i. \tau^i(\emptyset) \subseteq \tau^{i+1}(\emptyset)$ (induction on i and monotonicity)
2. There exists an i such that $\tau^i(\emptyset) = \tau^{i+1}(\emptyset)$ (sets become bigger and S is finite)
3. If $\tau^i(\emptyset) = \tau^{i+1}(\emptyset)$, then $\tau^i(\emptyset)$ is a fixed point of τ (by definition)
4. If X is a fixed point of τ , then $\forall i. \tau^i(\emptyset) \subseteq X$ (induction on i and monotonicity)

So an approximant τ^i can be found such that $\tau^i(\emptyset) = \tau^{i+1}(\emptyset)$, and this set is the least fixed point of τ .

Similarly, the smallest i such that $\tau^i(S) = \tau^{i+1}(S)$ yields the greatest fixed point.

Algorithms for computing the least fixed point and the greatest fixed point based on the observations on the previous slide.

```
function lfp( $\tau:\mathcal{P}(S)\rightarrow\mathcal{P}(S)$ ) :  $\mathcal{P}(S)$   
   $Q := \emptyset$ ;  
   $Q' := \tau(Q)$ ;  
  while  $Q \neq Q'$  do  
     $Q := Q'$ ;  
     $Q' := \tau(Q')$ ;  
  end while  
  return  $Q$ ;  
end function
```

```
function Gfp( $\tau:\mathcal{P}(S)\rightarrow\mathcal{P}(S)$ ) :  $\mathcal{P}(S)$   
   $Q := S$ ;  
   $Q' := \tau(Q)$ ;  
  while  $Q \neq Q'$  do  
     $Q := Q'$ ;  
     $Q' := \tau(Q')$ ;  
  end while  
  return  $Q$ ;  
end function
```

Specification of Kripke Structures

Fixed Points

Symbolic Model Checking

Implementing Symbolic Model Checking

CTL operators can be seen as fixed point operators. Fix a Kripke Structure $M = \langle S, R, L \rangle$. Identify a CTL formula f with predicate $\{s \mid s \models f\}$.

- ▶ $A F f = \mu Z. f \cup A X Z$ and $E F f = \mu Z. f \cup E X Z$
- ▶ $A G f = \nu Z. f \cap A X Z$ and $E G f = \nu Z. f \cap E X Z$
- ▶ $E [f U g] = \mu Z. g \cup (f \cap E X Z)$

Intuition:

- ▶ least and greatest fixed points deal differently with **loops**:
 - Greatest fixed point: recursion includes loops, so possibly infinitely many “steps”
 - Least fixed point: finite recursion through loops, so only finitely many “steps”
- ▶ Eventualities least fixed points
(a **witness** of the eventuality is needed in finitely many steps)
- ▶ Globally greatest fixed points
(an infinite path without error is OK)

Proof obligations for $E G$:

1. The transformer $Z \mapsto f \wedge E X Z$ is monotonic, so its fixed point can be computed by iteration, see lfp and gfp
(If $Z_1 \subseteq Z_2$ then $f \wedge E X Z_1 \subseteq f \wedge E X Z_2$).
2. $E G f$ is a fixed point of $Z \mapsto f \wedge E X Z$
($E G f = f \wedge E X E G f$)
3. $E G f$ is the largest such fixed point
(for all Z : if $Z = f \wedge E X Z$, then $Z \subseteq E G f$)
 - ▶ For 1,2,3: prove $X \subseteq Y$ by $\forall s. s \in X \Rightarrow s \in Y$.
 - ▶ For 2: prove \subseteq and \supseteq .
 - ▶ For 2,3: use the semantics of CTL-formulae

Proof obligations for $E [U]$ are similar (see for yourself)

CTL model checking with Fixed Points

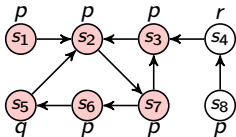
Function $\text{check}(f)$ takes a formula f and returns the set of states where f holds: $\{s \mid s \models f\}$ (given a fixed Kripke Structure $M = \langle S, R, L \rangle$).

$\text{check}(p)$	$\{s \mid p \in L(s)\}$
$\text{check}(\neg f)$	$S \setminus \text{check}(f)$
$\text{check}(f \vee g)$	$\text{check}(f) \cup \text{check}(g)$
$\text{check}(E X f)$	$\text{Pre}_R(\text{check}(f))$
$\text{check}(E [f U g])$	$\text{lfp}(Z \mapsto \text{check}(g) \cup (\text{check}(f) \cap \text{Pre}_R(Z)))$
$\text{check}(E G f)$	$\text{gfp}(Z \mapsto \text{check}(f) \cap \text{Pre}_R(Z))$

Recall: $\text{Pre}_R(Z) = \{s \in S \mid \exists t \in Z. s R t\}$

Example

- ▶ To check: $E [p \text{ U } q]$
- ▶ Compute: $\mu Z. q \vee (p \wedge E X Z)$ (with lfp)



$$Z_0 = \text{false} = \emptyset$$

$$Z_1 = q \vee (p \wedge E X Z_0) = \{s_5\}$$

$$Z_2 = q \vee (p \wedge E X Z_1) = \{s_5, s_6\}$$

$$Z_3 = q \vee (p \wedge E X Z_2) = \{s_5, s_6, s_7\}$$

$$Z_4 = q \vee (p \wedge E X Z_3) = \{s_2, s_5, s_6, s_7\}$$

$$Z_5 = q \vee (p \wedge E X Z_4) = \{s_1, s_2, s_3, s_5, s_6, s_7\}$$

$$Z_6 = q \vee (p \wedge E X Z_5) = \{s_1, s_2, s_3, s_5, s_6, s_7\}$$

$Z_5 = Z_6$, so this is the least fixed point.

Specification of Kripke Structures

Fixed Points

Symbolic Model Checking

Implementing Symbolic Model Checking

We wish to avoid representing the state space and its subsets explicitly. To efficiently implement symbolic model checking, we need:

- ▶ A concise representation of sets of states
- ▶ Quick operations for:
 - Boolean operators \wedge, \vee, \neg
 - Existential quantification (for the relational composition)
 - Equivalence test

Solution: *Ordered Binary Decision Diagrams (OBDD)*

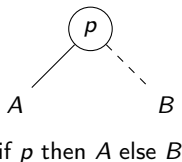
- ▶ Symbolic model checking is restricted to finite Kripke Structures
- ▶ All finite data can be encoded in “bits”
- ▶ Boolean functions can be represented **concisely** as (Ordered) Binary Decision Diagrams
- ▶ Binary Decision Diagrams are **directed acyclic graphs**, with the following ingredients:

1

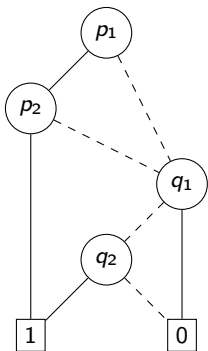
True

0

False



BDD representation of $(p_1 \wedge p_2) \vee (\neg q_1 \wedge q_2)$:



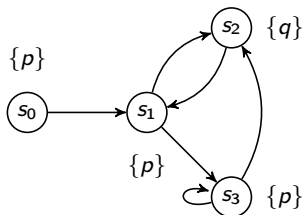
- ▶ In **ordered** BDDs, tests along a path occur in a **fixed** order (e.g. $p_1 < p_2 < q_1 < q_2$).
- ▶ Theorem[Bryant'86]: OBDDs are a **unique** representation for Boolean Functions.
- ▶ Claim: many practical formulae have a **concise OBDD representation** due to maximal sharing
- ▶ Disclaimer 1: some small formulae have only exponentially large BDDs. (multiplier)
- ▶ Disclaimer 2: the size of an OBDD can **crucially** depend on the ordering of the variables

More on OBDDs:

- ▶ OBDDs are implemented as maximally shared **pointer structures** in memory.
- ▶ The order of variables is fixed (some implementations feature **dynamic reordering**)
- ▶ **Equivalence test** can be performed in **constant time**, in particular, also checking for **satisfiability** and **tautology**.
- ▶ Boolean operations can be performed efficiently. Let B_1 and B_2 be OBDDs with m and n nodes, respectively, then:
 - OBDDs for $B_1 \wedge B_2$ and $B_1 \vee B_2$ can be computed in $\mathcal{O}(m \cdot n)$ time.
 - OBDDs for $\neg B_1$ can be computed in $\mathcal{O}(m)$ time.
 - the OBDD of $\exists x. B_1$ can be computed in $\mathcal{O}(m^2)$ time.
- ▶ Note: still a formula of size $\mathcal{O}(n)$ may have a BDD of size $\mathcal{O}(2^n)$.

- ▶ The implementation of a **symbolic model checking** relies on a representation of all sets in check , lfp and gfp by OBDDs.
- ▶ Hence, in summary, symbolic model checking:
 - **Recursively** processes subformulae
 - Represent the set of states satisfying a subformula by **OBDDs**
 - Treats temporal operators by **fixed point computations**
 - Relies on **efficient implementation** of equivalence test, and \wedge, \vee, \neg and \exists connectives on OBDDs.

Consider the following Kripke Structure:



Consider the following formulae, where p and q are atomic propositions:

- (A) $\mathbf{A(F}(q))$
 (B) $\mathbf{A}[q \mathbf{R} p]$

- Determine the set of states where (A) and (B) hold using the standard CTL model checking algorithm, based on graph algorithms .
- Determine the set of states where (A) and (B) hold using the symbolic model checking algorithm for CTL . Use explicit set notation to represent states.