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 6.81

Outline

Specification of Kripke Structures

Example (GCD)

Consider the following program:

$$\begin{array}{l} \textbf{repeat} \\ \quad \textbf{if } x > y -> x := x - y; \\ \quad [] x < y -> y := y - x; \\ \quad \textbf{fi} \\ \textbf{until } \text{false} \end{array}$$

This program uses:

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

$$\{x \mapsto 5, y \mapsto 15\} \to \{x \mapsto 5, y \mapsto 10\} \to \{x \mapsto 5, y \mapsto 5\} \to \{x \mapsto 5, y \mapsto 5\} \to \ldots$$

Specification of Kripke Structures

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:

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

Specification of Kripke Structures

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:

$$(pc = l_1 \wedge pc' = l_2 \wedge z' = x \wedge \ldots) \vee \ldots$$

Specification of Kripke Structures

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

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

Specification of Kripke Structures

Example

- $\mathcal{V} = \{\mathcal{E}(lla), \mathcal{J}(ohn)\}$,
- $\mathcal{D} = \{p(laying), q(uestioning), a(nswered)\}$
- $\mathcal{S}_0(\mathcal{E}, \mathcal{J}) := \mathcal{E} = p \wedge \mathcal{J} = p$
- $\mathcal{R}(\mathcal{E}, \mathcal{J}, \mathcal{E}', \mathcal{J}') := \mathcal{R}_1 \vee \mathcal{R}_2 \vee \mathcal{R}_3 \vee \mathcal{R}_4 \vee \mathcal{R}_5 \vee \mathcal{R}_6$, where:
  - $\mathcal{R}_1 := \mathcal{E} = p \wedge \mathcal{E}' = q \wedge \mathcal{J}' = \mathcal{J}$
  - $\mathcal{R}_2 := \mathcal{E} = q \wedge \mathcal{E}' = a \wedge \mathcal{J}' = \mathcal{J} \wedge \mathcal{J} \neq a$
  - $\mathcal{R}_3 := \mathcal{E} = a \wedge \mathcal{E}' = p \wedge \mathcal{J}' = \mathcal{J}$
  - $\mathcal{R}_4 := \mathcal{J} = p \wedge \mathcal{J}' = q \wedge \mathcal{E}' = \mathcal{E}$
  - $\mathcal{R}_5 := \mathcal{J} = q \wedge \mathcal{J}' = a \wedge \mathcal{E}' = \mathcal{E} \wedge \mathcal{E} \neq a$
  - $\mathcal{R}_6 := \mathcal{J} = a \wedge \mathcal{J}' = p \wedge \mathcal{E}' = \mathcal{E}$

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

Outline

Fixed Points

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

- Identify sets of states and predicates on states
- So, two notations are often mixed:
  - subsets: $X \subseteq \mathcal{S}$ or $X \in \mathcal{P}(\mathcal{S})$
  - predicates: $X \in 2^{\mathcal{S}}$ or $X : \mathcal{S} \rightarrow \{0, 1\}$
    $s \in X \Leftrightarrow X(s) = 1$ 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, $\vee, \wedge$ and $\cup, \cap$ are mixed: compare $\emptyset \cup \mathsf{E\ G}\ f$ and $\mathsf{false} \vee \mathsf{E\ G}\ f$

Fixed Points

Predicate Transformers and Monotonicity

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

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

$$\begin{aligned} \mathcal{P}re_{\mathcal{R}}(\mathcal{X}) &= \{s \in \mathcal{S} \mid \exists t \in \mathcal{X}.\ s\ \mathcal{R}\ t\} \\ \mathcal{P}ost_{\mathcal{R}}(\mathcal{X}) &= \{t \in \mathcal{S} \mid \exists s \in \mathcal{X}.\ s\ \mathcal{R}\ t\} \end{aligned}$$

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

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

Fixed Points

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

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

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

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

Fixed Points

Assume now that:

- $\mathcal{S}$ (hence also $\mathcal{P}(\mathcal{S})$) is finite, and
- $\tau : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ 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 $\mathcal{S}$ 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 $\mathcal{X}$ is a fixed point of τ, then $\forall i. \tau^i(\emptyset) \subseteq \mathcal{X}$ . . . . . . (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 $\tau^i(\mathcal{S}) = \tau^{i+1}(\mathcal{S})$ yields the greatest fixed point.

Fixed Points

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

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

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

Outline

1. Specification of Kripke Structures
2. Fixed Points
3. Symbolic Model Checking
4. Implementing Symbolic Model Checking

Symbolic Model Checking

CTL operators can be seen as fixed point operators. Fix a Kripke Structure $\mathcal{M} = \langle \mathcal{S}, \mathcal{R}, \mathcal{L} \rangle$. 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)

Symbolic Model Checking

Proof obligations for E G :

1. The transformer $Z \mapsto f \wedge \text{E X } Z$ is monotonic, so its fixed point can be computed by iteration, see lfp and gfp
(If $Z_1 \subseteq Z_2$ then $f \wedge \text{E X } Z_1 \subseteq f \wedge \text{E X } Z_2$).
2. E G f is a fixed point of $Z \mapsto f \wedge \text{E X } Z$
($\text{E G } f = f \wedge \text{E X E G } f$)
3. E G f is the largest such fixed point
(for all Z: if $Z = f \wedge \text{E X } Z$, then $Z \subseteq \text{E G } f$)

- 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)

Symbolic Model Checking

CTL model checking with Fixed Points

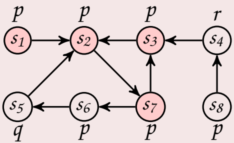
Function check(f) takes a formula f and returns the set of states where f holds: $\{s \mid s \models f\}$ (given a fixed Kripke Structure $\mathcal{M} = \langle \mathcal{S}, \mathcal{R}, \mathcal{L} \rangle$).

$\mathsf{check}(p)$	$\{s \mid p \in \mathcal{L}(s)\}$
$\mathsf{check}(\neg f)$	$\mathcal{S} \setminus \mathsf{check}(f)$
$\mathsf{check}(f \vee g)$	$\mathsf{check}(f) \cup \mathsf{check}(g)$
$\mathsf{check}(\mathsf{E\ X}\ f)$	$\mathcal{P}re_{\mathcal{R}}(\mathsf{check}(f))$
$\mathsf{check}(\mathsf{E}\ [f\ \mathsf{U}\ g])$	$\mathsf{lfp}\big(\mathcal{Z} \mapsto \mathsf{check}(g) \cup (\mathsf{check}(f) \cap \mathcal{P}re_{\mathcal{R}}(\mathcal{Z}))\big)$
$\mathsf{check}(\mathsf{E\ G}\ f)$	$\mathsf{gfp}\big(\mathcal{Z} \mapsto \mathsf{check}(f) \cap \mathcal{P}re_{\mathcal{R}}(\mathcal{Z})\big)$

Recall: $\mathcal{P}re_{\mathcal{R}}(\mathcal{Z}) = \{s \in \mathcal{S} \mid \exists t \in \mathcal{Z}. s\ \mathcal{R}\ t\}$

Symbolic Model Checking

Example



- To check: E G p
- Compute: $\nu Z . p \wedge \mathsf{E\ X}\ Z$ (with gfp)

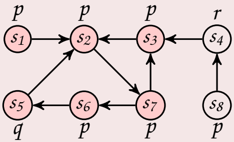
$$\begin{aligned} Z_0 &= \text{true} = \{s_i \mid 1 \leq i \leq 8\} \\ Z_1 &= p \wedge \mathsf{E\ X}\ Z_0 = \{s_1, s_2, s_3, s_6, s_7, s_8\} \\ Z_2 &= p \wedge \mathsf{E\ X}\ Z_1 = \{s_1, s_2, s_3, s_7\} \\ Z_3 &= p \wedge \mathsf{E\ X}\ Z_2 = \{s_1, s_2, s_3, s_7\} \end{aligned}$$

$Z_2 = Z_3$, so this is the greatest fixed point.

Symbolic Model Checking

Example

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



$$
\begin{aligned}
Z_0 &= \text{false} = \emptyset \\
Z_1 &= q \vee (p \wedge \mathsf{E}\,\mathsf{X}\; Z_0) = \{s_5\} \\
Z_2 &= q \vee (p \wedge \mathsf{E}\,\mathsf{X}\; Z_1) = \{s_5, s_6\} \\
Z_3 &= q \vee (p \wedge \mathsf{E}\,\mathsf{X}\; Z_2) = \{s_5, s_6, s_7\} \\
Z_4 &= q \vee (p \wedge \mathsf{E}\,\mathsf{X}\; Z_3) = \{s_2, s_5, s_6, s_7\} \\
Z_5 &= q \vee (p \wedge \mathsf{E}\,\mathsf{X}\; Z_4) = \{s_1, s_2, s_3, s_5, s_6, s_7\} \\
Z_6 &= q \vee (p \wedge \mathsf{E}\,\mathsf{X}\; Z_5) = \{s_1, s_2, s_3, s_5, s_6, s_7\}
\end{aligned}
$$

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

Outline

Implementing Symbolic Model Checking

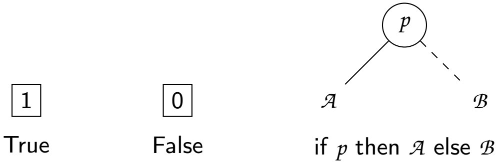
We wish to avoid representing the state space and its subsets explicitly. To efficiently implement symbolic model checking, we need:

- 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)*

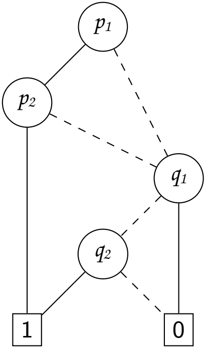
Implementing Symbolic Model Checking

- 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:



Implementing Symbolic Model Checking

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

Implementing Symbolic Model Checking

More on OBDDs:

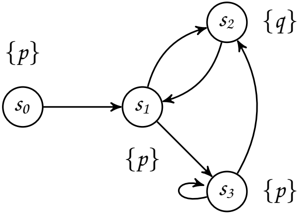
- OBDDs are implemented as maximally shared pointer structures in memory.
- The order of variables is fixed (some implementations feature dynamic reordering)
- Equivalence test can be performed in constant time, in particular, also checking for satisfiability and tautology.
- Boolean operations can be performed efficiently. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be OBDDs with m and n nodes, respectively, then:
  - OBDDs for $\mathcal{B}_1 \wedge \mathcal{B}_2$ and $\mathcal{B}_1 \vee \mathcal{B}_2$ can be computed in $\mathcal{O}(m \cdot n)$ time.
  - OBDDs for $\neg\mathcal{B}_1$ can be computed in $\mathcal{O}(m)$ time.
  - the OBDD of $\exists x.\mathcal{B}_1$ can be computed in $\mathcal{O}(m^2)$ time.
- Note: still a formula of size $\mathcal{O}(n)$ may have a BDD of size $\mathcal{O}(2^n)$.

Implementing Symbolic Model Checking

- 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.

Exercise

Consider the following Kripke Structure:



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

$$(\mathcal{A}) \quad \mathcal{A}(\mathcal{F}(q))$$
$$(\mathcal{B}) \quad \mathcal{A}[q \; \mathcal{R} \; p]$$

1. Determine the set of states where $(\mathcal{A})$ and $(\mathcal{B})$ hold using the standard CTL model checking algorithm, based on graph algorithms .
2. Determine the set of states where $(\mathcal{A})$ and $(\mathcal{B})$ hold using the symbolic model checking algorithm for CTL . Use explicit set notation to represents states.