Algorithmic Verification
Comp4151
Lecture 5-A
Ansgar Fehnker

Overview
Explicit state CTL
- Bottom-up recursive labelling algorithm
- Linear in the size of Kripke structure and formula

However
- Suffers from state explosion problem

Fixpoint characterization of CTL
- Leads to set based algorithm

Semantic of CTL with fixpoints
Given Kripke structure $M = (S, s_0, \rightarrow, \mu)$, with $n$ states, and $\phi$ in ENF over atomic propositions AP.

- $[[\text{true}]] = S$
- $[[\text{false}]] = \emptyset$
- $[[\phi \land \psi]] = S \setminus [[\phi]] \cap [[\psi]]$
- $[[\exists x \phi]] = \{ s \mid \exists (s, x) \in \rightarrow \}$
- $[[\forall x \phi]] = f^*_\phi(S)$

- Translates to an algorithm based on sets
- Sets of states as unordered list lists are inefficient
- We need
  - Compact set representation
  - Efficient operations on sets
Boolean Functions

Definition

A boolean variable is variable over values \(\{0,1\}\).

A boolean function over \(n\) arguments is a function \(f: \{0,1\}^n \rightarrow \{0,1\}\).

Two common representations of boolean functions:
- propositional formula
- truth table

Boolean Function

Example:

\[ f(x_1, x_2, x_3) = (x_1 \lor x_2) \land x_3 \]

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(f(x_1, x_2, x_3))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

compact, but satisfiability and comparison hard

Binary Decision Tree

- Represents boolean function as a decision tree
- Non-terminal node \(i\) labeled with
  - variable \(\text{var}(i)\)
  - successors \(\text{lo}(i)\) and \(\text{hi}(i)\)
- Terminal nodes are labeled 0 or 1
- Fast lookup of \(f(x_1, \ldots, x_n)\) given \(x_1, \ldots, x_n\)

exponential size: \(2^n\) lines for \(n\) variables
Boolean Functions

**Binary Decision Tree**

\[ f(x_0, x_1, x_2) = (x_0 \lor x_1) \land x_2 \]

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f(x_0, x_1, x_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Binary Decision Diagrams**

Represents boolean functions as directed acyclic graph with a single initial node.

More compact representation than decision trees.

Omit duplicated nodes and unnecessary tests.
Binary Decision Diagrams

Reduction

Rule 1: Share identical terminal nodes.

Rule 2: Remove redundant tests

Rule 3: Share identical non-terminal nodes.

Rules applied iteratively
Binary Decision Diagrams

Definitions

A BDD is reduced if none of reduction rules can be applied.

Given a variable order $x_1 < \ldots < x_n$, a BDD is ordered if for each node $i$ holds

$$j \in \{ l(i), h(i) \} \implies x_i < x_j$$

OBDDs

Theorem

Two reduced ordered BDDs with respect to the same order $x_1 < \ldots < x_n$ represent the same boolean function iff they have the same structure.

Furthermore

- A function is a tautology if its ROBDD $u$ is equal to $1$.
- A function is a satisfiable if its ROBDD $u$ is not equal to $0$.

ROBDDs provide a canonical for boolean functions

OBDDs

Variable ordering

- Size of an (reduced) OBDD depends on the order
OBDDs

Variable ordering

Example: \((a \land b) \lor (a \land b) \lor (a \land b)\)

Intermediate Summary

OBDDs provide a compact representation for boolean functions (most of the times)

Satisfiability and comparison are easy to check

However:
- What about other operations on OBDDs?
- What is the relation with model checking?

Operations on OBDDs

Reduce

Bottom-up labelling based on the reduction rules

Label nodes with integer numbers as follows
- two terminal nodes with the same value get the same label
  - if \(\text{lo}(i) = \text{hi}(i)\) replace \(\text{label}(i)\) to \(\text{label}(\text{lo}(i))\)
  - if \(\text{var}(i) = \text{var}(j)\) and \(\text{lo}(i) = \text{lo}(j)\) and \(\text{hi}(i) = \text{hi}(j)\) for some \(j\) set \(\text{label}(i)\) to \(\text{label}(j)\)
  - otherwise label(i) with next unused integer

After labelling redirect edges bottom-up
Operations on OBDDs

Reduce

Restrict variable $x$ of function $f$ to constant value $b$, i.e. replace all occurrences of $x$ by $b$.

Example

$$f(x_1, x_2, x_3) = (x_1 \lor x_2) \land x_3$$

$$f[0/x_2] (x_1, x_2, x_3) = (x_1 \lor 0) \land x_3 = x_1 \land x_3$$

$$f[1/x_2] (x_1, x_2, x_3) = (x_1 \lor 1) \land x_3 = x_3$$
Operations on OBDDs

Apply

Given OBDDs $B_f$ and $B_g$ for boolean formulas $f$ and $g$.
Apply($°$, $B_f$, $B_g$) computes the OBDD of $f°g$

Basic Idea

- Let $x$ be smallest variable of $B_f$ and $B_g$ (i.e., root in $B_f$ or $B_g$)
- Split into two smaller sub-problems for $x=0$ and $x=1$.
- Repeat until leaves are reached. Apply $°$ to the leaves.

Recursive application of Shannon Expansion

$$f = (\neg x_i \land f[0/x_i]) \lor (x_i \land f[1/x_i])$$

Operations on OBDDs

Apply

Given OBDDs $B_i$ and $B_j$ for boolean formulas $f$ and $g$.
Apply($°$, $B_i$, $B_j$) computes the OBDD of $f°g$

Basic Idea

- Let $x$ be smallest variable of $B_i$ and $B_j$ (i.e., root in $B_i$ or $B_j$)
- Split into two smaller sub-problems for $x=0$ and $x=1$.
- Repeat until leaves are reached. Apply $°$ to the leaves.

Recursive application of Shannon Expansion

$$f°g = (\neg x_i \land (f[0/x_i]°g[0/x_i])) \lor (x_i \land (f[1/x_i]°g[1/x_i]))$$
Operations on OBDDs

Apply

Given OBDDs $B_f$ and $B_g$ for boolean formulas $f$ and $g$.

Let $i_f$ and $i_g$ be the root nodes of $B_f$ and $B_g$.

Rule 3:
if $i_f$ is a non-terminal and $\text{var}(i_f) > \text{var}(i_g)$ or $i_g$ a terminal
$\Rightarrow$ Similar to Rule 2

Rule 4:
if $i_f$ and $i_g$ are terminal nodes labelled $b_f$ and $b_g$
$\Rightarrow$ label current node with $b_f \oplus b_g$
0 or 1

Apply

$\overline{f \circ g} = \bigvee \left( \bigwedge \left( f[0/x_1] \circ g[0/x_1] \right) \right) \bigvee \left( \bigwedge \left( f[1/x_1] \circ g[1/x_1] \right) \right)$

Apply

$\overline{f \circ g} = \bigvee \left( \bigwedge \left( f[0/x_1] \circ g[0/x_1] \right) \right) \bigvee \left( \bigwedge \left( f[1/x_1] \circ g[1/x_1] \right) \right)$
Operations on OBDDs

Apply

Reuse earlier results
Operations on OBDDs

Apply

Given boolean functions $f$ and $g$, compute $f \lor g$.

Negate

Given boolean functions $f$ and reduced OBDD $B_f$, compute $\neg f$, by swapping the leaves.

Exist

Given boolean functions $f$ variable $x$ and reduced OBDD $B_f$, compute $\exists x.f$.

We have $\exists x.f = f[0/x] \lor f[1/x]$.

$\Rightarrow$ We can use a combination of Restrict and Apply for $f[0/x]$ and $f[1/x]$.

Operations on OBDDs

Some complexities

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce</td>
<td>$O(</td>
</tr>
<tr>
<td>Negate</td>
<td>constant</td>
</tr>
<tr>
<td>Apply</td>
<td>$O(</td>
</tr>
<tr>
<td>Exists</td>
<td>$O(</td>
</tr>
</tbody>
</table>
Model Checking and OBDDs

Another Intermediate Summary

- OBDDs provide a compact representation for boolean functions (most of the times)
- Satisfiability and comparison are easy to check
- Efficient algorithms for boolean operations
- However:
  - What is the relation with model checking?

Symbolic CTL Model Checking

Observations

- Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$ can be used to represent subsets of $\{0,1\}^n$
- Each finite set (of states) can be mapped to $\{0,1\}^n$
- Boolean function can be presented as OBDD.
- Given Kripke structure $M = (S, S_0, R, L)$
  - Set of states $S$
  - Set of initial states $S_0$
  - Transition relation $R$ is set of pairs of states
  - Labelling function $L: S \rightarrow \mathbb{2}^S$ maps labels to sets of states
- States, initial states, transition relation and labeling can be represented as OBDDs

Symbolic CTL Model Checking

State space

- Each state can be represented as vector of boolean values
- The set of initial states can be represented by a boolean function

Example:

$S_0 = \{(x_1, x_2) | \neg x_1 \land \neg x_2\}$

Symbolic CTL Model Checking

Transition relation

- Each transition can be represented pairs of states
- The transition relation can be represented by a boolean function over $x_1, \ldots, x_n, x'_1, \ldots, x'_n$

Example:

$R$ is the disjunction of boolean functions for all transitions

$((0,0), (0,1))$ satisfies $(\neg x_1 \land \neg x_2 \land \neg x'_1 \land x'_2)$
Symbolic CTL Model Checking

Labelling

- $L^{-1}$ maps labels to sets of states
- This set can be represented as boolean function

Example

$\text{Example } L^{-1}(p) = \{(0,0), (0,1), (1,0)\}$

$L^{-1}(p)$ is represented by:

$\neg x_1 \vee \neg x_2 \vee x_1 \wedge x_2 = \neg x_2 \vee \neg x_1 \vee x_2$

Symbolic CTL Model Checking

Symbolic Representation

Given $M = (S, S_0, R, L)$ over set of atomic proposition $AP$

OBDD operations

Booelan operators

- $[\text{true}] = S$
- $[\text{false}] = \emptyset$
- $[\{ p \}] = \{ s \mid p \in \mu(s) \}$
- $[\neg \phi] = S \setminus [\phi]$
- $[\phi \wedge \psi] = [\phi] \cap [\psi]$
- $[\phi \vee \psi] = \text{OBDD with leaf 1}$
- $\text{OBDD with leaf 0}$
- $\text{OBDD for } L^{-1}(p)$
- $\neg \phi$
- $\text{Apply}(\neg, B_1, B_2)$

Temporal Operators

- $[\text{EX } \phi] = \{ s \mid \exists s' \in R(s, s') \rightarrow \phi \}$
- $\exists x_1, \ldots, x_n([\phi(x_1, \ldots, x_n)])$
Symbolic CTL Model Checking

OBDD operations

Temporal Operators

- $\mathbf{\parallel E \parallel U \parallel}$ is least fixpoint of $f_{EU}(P) = [\mathbf{\parallel} \mathbf{\parallel} \cup ([\mathbf{\parallel}] \cap [\mathbf{EX P}])$
- Implemented by combination of Apply and Exists

Symbolic CTL Model Checking

OBDD operations

Temporal Operators

- $\mathbf{\parallel EG \parallel}$ is the greatest fixpoint of $f_{EG}(P) = [\mathbf{\parallel} \mathbf{\parallel} \cap [\mathbf{EX P}])$
- Implemented by combination of Apply and Exists

Symbolic CTL Model Checking

Summary

- Fixpoint characterization of CTL translates to an algorithm based on sets
- Set in general and Kripke structures in particular can be modelled with boolean function
- OBDDs provide a canonical representation of boolean functions that provides
  - Compact representation of sets
  - Efficient operations on sets
- Next lecture: SMV

14