Model Checking II

How CTL model checking works

CTL

\( A \ E \quad X \ F \ G \ U \)

Model checking problem
Determine \( M, s_0 \models f \)

Or find all \( s \) s.t. \( M, s \models f \)
Need only the boolean connectives ($\neg$, $\lor$) and $E$ define others

e.g.

\[
AG\ p \iff \neg EF \neg p
\]
\[
A(p \lor q) \iff \neg E(\neg q \lor \neg p \land \neg q) \land \neg EG(\neg q)
\]

(Could skip $EF$ and define in terms of $EU$, but hopefully seeing how $EF$ is done aids understanding)

**Explicit state model checking**

Option 1 CES (original paper)

Represent state transition graph explicitly
Walk around marking states

Graph algorithms involving strongly connected components etc.
Not covered in this course (cf. SPIN)
Used particularly in software model checking
Symbolic MC

Option 2  McMillan et al

because of
STATE EXPLOSION problem
  State graph exponential in program/circuit size
  Graph algorithms linear in state graph size

INSTEAD
  Use symbolic representation of both sets of states
  and of state transition graph

Symbolic MC

Sets of states formulas
  relations between states (BDDs)

Fixed point characterisations of CTL ops

NO explicit state graph
A state

Vector of boolean variables

\[(v_1, v_2, v_3, \ldots, v_n) \in \{0, 1\}^n\]

Boolean formulas

\[(x \oplus y) \oplus z \quad (\oplus \text{ is exclusive or})\]

\[(1 \oplus 0) \oplus 0 = 1\]

Assignment [\(x=1, y=0, z=0\)] gives answer 1

is a model or satisfying assignment

Write as 100

Exercise: Find another model
Boolean formulas

\[(x \oplus y) \oplus z\]
\[(1 \oplus 1) \oplus 0 = 0\]

assignment \([x=1,y=1,z=0]\) is not a model

Formula is a tautology if ALL assignments are models and is contradictory if NONE is.
Boolean formulas

For us, interesting formulas are somewhere in between: some assignments are models, some not.

IDEA: A formula can represent a set of states (its models)

\[
\begin{align*}
\{\} & \quad \text{false} \\
\{111\} & \quad x \land y \land z \\
\{101\} & \quad x \land \neg y \land z \\
\{111,101\} & \quad x \land z \\
& \quad \ldots \\
& \quad \ldots \\
\{000,001, \ldots , 111\} & \quad \text{true}
\end{align*}
\]
Example

\[(x \oplus y) \oplus z\] represents \{100, 010, 001, 111\}
for states of the form xyz

Exercise: Find formulas (with var. names x, y, z)
for the sets
\{
\}
\{100\}
\{110, 100, 010, 000\}

What is needed now?

A good data structure for boolean formulas

Have already seen

Binary Decision Diagrams (BDDs)

Lee (Bell Systems Tech. Journal 59)
Akers (IEEE Trans. Comp 78)
Bryant (IEEE Trans. Comp. 86, most cited CS paper!)
see also Bryant’s document about a Hitachi patent from 93
McMillan saw application to symbolic MC
Binary Decision Diagrams

Canonical form (constant time comparison)

Polynomial algorithms for ‘and’, ‘or’, ‘not’ etc.

Exponential but practically efficient algorithm for boolean quantification (even of sets of variables)

Read Hu’s excellent tutorial paper (See Course Literature)

(Presentation based on lecture notes by Ken McMillan)

Combinational equivalence checking

For two circuits with single boolean outputs, make BDDs for each circuit and see if they are the same

Of course the BDDs are built up by application of BDD construction functions and, or, not etc. NOT by making decision tree and then reducing

(In practice need lots more tricks.)
Represent a set of states

Just make the BDD for a corresponding formula!

Represent a transition relation $R$

Remember that $R$ is just
a set of pairs of states

Use two sets of variables, $v$ and $v'$ (with the primed variables representing next states)

Make a formula involving both $v$ and $v'$ and from that a BDD $\text{bdd}(R,(v,v'))$
What set of states can we reach from set P in one step?

Forward Image

\[ \text{bd}(\text{Image}(P,R),v') = \exists v \ (\text{bd}(P,v) \land \text{bd}(R,(v,v'))) \]
Backward image

\[ \{ s \mid \exists t \in Q \wedge s R t \} \]

\[ \text{bdd}(\text{Image}^{-1}(Q,R),v) = \exists v' \ \text{bdd}(Q,v') \wedge \text{bdd}(R,(v,v')) \]

So far

BDDs for
1) sets of states
2) transition relation
3) calculating forward or backward image of a set

Need one last idea: iteration to a fixed point based on recursive description of CTL ops
Before we go on with MC, note that we can now compute Reachable States (see Hu paper)

Let T be the transition relation

\[ R_0(v) = \text{BDD for reset (or initial) state} \]
\[ R_1(v) = R_0(v) \lor \text{bdd(Image}(R_0,T),v) \]

\[ \ldots \]
\[ R_{i+1}(v) = R_i(v) \lor \text{bdd(Image}(R_i,T),v) \]

Will eventually converge with \( R_{i+1}(v) = R_i(v) \).

Why??
Before we go on with MC, note that we can now compute Reachable States (see Hu paper)

Let $T$ be the transition relation

$R_0(v) = \text{BDD for reset (or initial) state}$

$R_1(v) = R_0(v) \lor \text{bdd(Image(R_0,T),v)}$

$\cdots$

$R_{i+1}(v) = R_i(v) \lor \text{bdd(Image(R_i,T),v)}$

Will eventually converge with $R_{i+1}(v) = R_i(v)$

Sequential Equivalence Checking via Reachability

Just make the transition system for the combination including (big) comparator. Check that ok is true in all reachable states. In practice need more tricks, see Hu
Symbolic MC of CTL

Compute set of states satisfying a formula recursively (and use BDDs as rep.)

consider $\neg$, $\lor$, EX, EF, EG, EU

define others

<table>
<thead>
<tr>
<th>CTL formula f</th>
<th>H(f) set of states satisfying f</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$ (atomic)</td>
<td>${s \mid a \text{ in } L(s)}$ (cf.Lars)</td>
</tr>
<tr>
<td>$\neg p$</td>
<td>$S - H(p)$</td>
</tr>
<tr>
<td>$p \lor q$</td>
<td>$H(p) \cup H(q)$</td>
</tr>
<tr>
<td>EX $p$</td>
<td>familiar ?</td>
</tr>
</tbody>
</table>
CTL formula $f$  \hspace{1cm} H($f$) set of states satisfying $f$

EX $p$  \hspace{1cm} Image$^{-1}$(H($p$), $R$)

CTL formula $f$  \hspace{1cm} BDD for set of states satisfying $f$

EX $p$  \hspace{1cm} $\exists v' \ bdd(p, v') \land bdd(R, (v, v'))$
Remaining operators

Recursive characterisation

\[ EF \ p \quad \Leftrightarrow \quad p \lor \ EX \ (EF \ p) \]

CTL

\[ EF \ p \quad \Leftrightarrow \quad p \lor \ EX \ (EF \ p) \]

Start with the empty set of states, \( \emptyset \), as a first guess, and improve it by applying \( p \lor EX \ (. ) \) to it
Fixed point iteration (formulas)

\[ S_0 = \text{false} \]
\[ S_1 = p \lor \text{EX (false)} = p \]
\[ S_2 = p \lor \text{EX (p)} \]
\[ S_3 = p \lor \text{EX (p} \lor \text{EX (p)}) \]

and so on

Will eventually terminate. Why?

Now think sets

\[ S_0 = \emptyset = \text{empty set of states} \]
\[ S_1 = H(p) \cup \text{Image}^{-1}(\emptyset, R) = H(p) \]
\[ S_2 = H(p) \cup \text{Image}^{-1}(H(p), R) \]

\[ S_{i+1} = H(p) \cup \text{Image}^{-1}(S_i, R) \]

Will eventually terminate. Why?
Fixed point iteration

$p \lor \text{EX } p$

Fixed point iteration
Fixed point iteration

\[ p \lor \text{EX} (p \lor \text{EX} p) \]

Eventually stops!

Fixed point iteration

Eventually stops!
LEAST fixed point

Started with a small set (empty, indeed) and made it larger.
All ok because
\[ F(B) = H(p) \cup \text{Image}^{-1}(B, R) \text{ is monotonic} \]
(i.e. if \( B \subseteq B' \) then \( F(B) \subseteq F(B') \))
Write least \( y \) s.t. \( y = F(y) \) as \( \mu y. F(y) \)

EG

EG \( p \iff p \land \text{EX} (\text{EG} \ p) \)

This time need to work downwards
Fixed point iteration (formulas)

\[ S_0 = \text{true} \]
\[ S_1 = p \land \text{EX (true)} \]
\[ S_2 = p \land \text{EX (p \land \text{EX (true)})} \]

and so on

Will eventually terminate. Why?

Now think sets

\[ S_0 = S = \text{the entire set of states} \]
\[ S_1 = H(p) \cap \text{Image}^{-1}(S,R) \]
\[ S_2 = H(p) \cap \text{Image}^{-1}(H(p) \cap \text{Image}^{-1}(S,R),R) \]

\[ S_{i+1} = H(p) \cap \text{Image}^{-1}(S_i,R) \]

Will eventually terminate.
Greatest fixed point

$\text{EG } p \quad \iff \quad p \land \text{EX } (\text{EG } p)$

$\text{H}(\text{EG } p)$

$= \forall y. \text{H}(p) \cap \text{Image}^{-1}(y, R)$

NB: We can do all of these operations using BDDs to represent the sets
Fixed point iteration in the other direction

$\text{P}$

Fixed point iteration in the other direction

$\text{P} \wedge \text{EX } \text{P}$
Fixed point iteration in the other direction

\[ p \land \text{EX} (p \land \text{EX} p) \]

Fixed point iteration in the other direction

\[ p \land \text{EX} (p \land \text{EX} p) \]
EU

\[ E(p \cup q) \iff q \lor (p \land \text{EX}(E(p \cup q))) \]

\[ H(E(p \cup q)) \]

\[ = \mu y. H(q) \cup (H(p) \cap \text{Image}^{-1}(y,R)) \]

This is a generalisation of the EF case.
Remember that \( E(\text{True} \cup q) = \text{EF} q \)

Exercise: define \( H(A(p \cup q)) \). See earlier slide

All done with BDDS (and recursion and fixed point iteration)
Concrete example

Exercise: calculate the set of states satisfying

\[ \text{EF } p \]

\[ p = \text{dreq} \land q0 \land \text{dack} \]

(answer will be available later)
Concrete example

Also calculate for

$$EG \, p$$

$$p = (dreq \lor q0 ) \land dack$$