Roadmap

Problem
Given a model $\mathcal{M}$ (usually a Kripke structure) that represents the behaviour of a system, and a temporal logic formula $f$ that represents a desired property of the system, determine whether the model satisfies the formula:

$$\mathcal{M}, s \models f$$

Model Checking Algorithms that we’ll study:
- Explicit CTL Model Checking: labelling a graph
- Symbolic CTL Model Checking: representing sets of states using propositional logic

Symbolic CTL Model Checking

In explicit state model checking, we represent the Kripke structure as a graph and implement the model checking algorithm as graph traversal.

This method has also been called enumerative graph search [AH98] because one state is processed at a time.

Representing the graph and walking over the graph take time. Can we make this process more efficient in practice?

- Use a more efficient means of representing states, and means of processing states. In particular, we can represent sets of states.

Symbolic Model Checking

Symbolic model checking means model checking by describing sets of states as propositional logic formulae.

These logical formulae (Boolean functions) can be represented using efficient means of manipulating Boolean functions (such as binary decision diagrams – BDDs).

Symbolic model checking (using BDDs) was invented by Ken McMillan in the early 1990’s.
**Agenda**

Symbolic CTL model checking:

1. Describe the model checking algorithm as computations over sets of states; abstract the operation of finding the set of states that can reach another set of states as a pre-image computation
2. Characteristic functions to represent sets of states
3. Use Boolean functions (logical formulae) as characteristic functions to represent sets of states; describe the Kripke structure as a Boolean characteristic function (symbolic)
4. (next class) Binary Decision Diagrams – a data structure for manipulating Boolean functions
5. (next class) CTL model checking as fixed point operations in the $\mu$-calculus

---

**Explicit CTL Model Checking**

The model checking algorithm was given as five procedures, each of which labelled the graph:

- Check $\neg f_1$
- Check $f_1 \lor f_2$
- CheckEX($f_1$)
- CheckEG($f_1$)
- CheckEU($f_1$, $f_2$)

At the end we check that the initial state is labelled with the formula we are checking.

---

**Sets of States**

Let's abstract this algorithm and describe it in terms of operations on sets of states.

The abstracted algorithm will return the set of states satisfying the formula. This set of states is the set of states “labelled” with the formula in the explicit CTL m/c algorithm.

For formula $f_1$, the algorithm returns SAT($f_1$).

We recursively call the algorithm on its subformulae.

*Src: Huth and Ryan [HR04]*

---

**CTL Model Checking**

If the formula is an atomic proposition $f_1$:

**Previous version:**
start with the states labelled with $L$, so nothing to do.

**New version:** return \{ $s \in S$ | $f_1 \in L(s)$ \}

If the formula is $\neg f_1$:

**Previous version:** Add $\neg f_1$ to all $\text{label}(s)$ if $f_1 \notin \text{label}(s)$

**New version:** return $S - \text{SAT}(f_1)$

If the formula is $f_1 \lor f_2$:

**Previous version:**

Add $f_1 \lor f_2$ to $\text{label}(s)$ if either $f_1$ or $f_2$ are in $\text{label}(s)$

**New version:** SAT($f_1$) $\cup$ SAT($f_2$)
**CTL Model Checking: EX**

Previous version:
CheckEX($f_1$)

\[ K = \{ s \mid f_1 \in \text{label}(s) \} \]

while $K \neq \emptyset$
do
choose $s \in K$;
$K := K \setminus \{s\}$;for all $(t, s) \in R$ do
  label$(t)$ := label$(t) \cup \{\text{EX} f_1\}$;

New version:

function SAT_EX($f_1$)
\[ K := \text{SAT}(f_1) \]
\[ Y = \{ t \in S \mid \exists s \cdot s \in K \land (t, s) \in R \} \]
return $Y$;

** CTL Model Checking: EU**

Previous version:
CheckEU($f_1, f_2$)

\[ K := \{ s \mid f_2 \in \text{label}(s) \} \]

for all $s \in K$ do

label$(s)$ := label$(s) \cup \{\text{EG} f_1 \}$;while $K \neq \emptyset$
do
choose $s \in K$;
$K := K \setminus \{s\}$;for all $(t, s) \in R$ do
  if $\text{EG} f_1 \in \text{label}(t)$ and $f_1 \in \text{label}(t)$ then
    label$(t)$ := label$(t) \cup \{\text{EG} f_1\}$;
  end if
until $\text{oldK} = K$;return $K$;

New version:

function SAT_EU($f_1, f_2$)
\[ K := \text{SAT}(f_2) \]
\[ W := \text{SAT}(f_1) \]
do
oldK := K;
$K := \text{oldK} \cup (W \cap \{t \in S \mid \exists s \cdot s \in \text{oldK} \land (t, s) \in R\})$;until $\text{oldK} = K$;return $K$;

**CTL Model Checking: EG**

Previous version:
CheckEG($f_1$)

\[ K = \{ s \mid f_1 \in \text{label}(s) \} \]

for all $s \in K$ do

label$(s)$ := label$(s) \cup \{\text{EG} f_1\}$;for all $t \in S$ do
  if $\text{EG} f_1 \in \text{label}(t)$ then
    for all $(t, s) \in R$ do
      label$(t)$ := label$(t) \cup \{\text{EG} f_1\}$;
    end for all $(t, s) \in R$ do
      if $s \in K$ then
        label$(t)$ := label$(t) \cup \{\text{EG} f_1\}$;
      end if
    end until $K = \{ s \mid \text{EG} f_1 \in \text{label}(s) \}$;

New version:

function SAT_EG($f_1$)
\[ K := \text{SAT}(f_1) \]
do
oldK := K;
$K := \text{oldK} \cup \{t \in S \mid \exists s \cdot s \in K \land (t, s) \in R\}$;until $\text{oldK} = K$;return $K$;

**Model Checking: SAT**

function SAT($f$)
case $f$

true : return $S$;
false : return $\emptyset$;
atomic $f_1$ : return $\{ s \in S \mid f_1 \in \text{L}(s) \}$
$\neg f_1$ : return $S \setminus \text{SAT}(f_1)$
$f_1 \lor f_2$ : return $\text{SAT}(f_1) \cup \text{SAT}(f_2)$
$\ldots$

EX $f_1$ : return SAT_EX($f_1$)
EU $f_1, f_2$ : return SAT_EU($f_1, f_2$)
EG $f_1$ : return SAT_EG($f_1$)
AX $f_1$ : return SAT($\neg (\text{EX} \neg f_1)$)
$\ldots$

}
Operations on Sets

What are the operations on sets of states that we need to implement this algorithm?

- set difference
- set union
- set intersection
- set equality
- pre-image: \( \text{pre}_3(X) = \{ t \in S \mid \exists s \in X \land (t, s) \in R \} \)

Using Pre-image

pre-image: \( \text{pre}_3(X) = \{ t \in S \mid \exists s \in X \land (t, s) \in R \} \)

New version:
function SAT_EX \((f_1)\)
\[
K = \text{SAT}(f_1);
Y = \{ t \in S \mid \exists s \in K \land (t, s) \in R \};
\text{return } Y;
\]

Even better version:
function SAT_EX \((f_1)\)
\[
\text{return } \text{pre}_3(\text{SAT}(f_1));
\]

Using Pre-image

New version:
function SAT_EX \((f_1, f_2)\)
\[
K = \text{SAT}(f_2); \ W = \text{SAT}(f_1);
do\oldK := K;\ K := \oldK \cup \{t \in S \mid \exists s \in \oldK \land (t, s) \in R\};\until \oldK = K;
\text{return } K;
\]

Even better version:
function SAT_EX \((f_1, f_2)\)
\[
K := \text{SAT}(f_1);\do\oldK := K; \ K := \oldK \cup \text{pre}_3(\oldK);\until \oldK = K;
\text{return } K;
\]

Using Pre-image

New version:
function SAT_EX \((f_1, f_2)\)
\[
K := \text{SAT}(f_1);\do\oldK := K; \ K := \oldK \cap \{t \in S \mid \exists s \in \oldK \land (t, s) \in R\};\until \oldK = K;
\text{return } K;
\]

Even better version:
function SAT_EX \((f_1)\)
\[
K := \text{SAT}(f_1);\do\oldK := K; \ K := \oldK \cap \text{pre}_3(\oldK);\until \oldK = K;
\text{return } K;
\]
Summary

The CTL model checking algorithm is recursive in the structure of the formula. Each function returns the set of states that satisfy the formula.

Agenda

Symbolic CTL model checking:
1. Describe the model checking algorithm as computations over sets of states; abstract the operation of finding the set of states that can reach another set of states as a pre-image computation
2. Characteristic functions to represent sets of states
3. Use Boolean functions (logical formulae) as characteristic functions to represent sets of states; describe the Kripke structure as a Boolean characteristic function (symbolic)
4. (next class) Binary Decision Diagrams – a data structure for manipulating Boolean functions
5. (next class) CTL model checking as fixed point operations in the $\mu$-calculus

Characteristic Function

If $\mathcal{U}$ is a set and $A \subseteq \mathcal{U}$, the characteristic function (predicate) of $A$ is $\chi_A : \mathcal{U} \rightarrow \{T, F\}$, defined as:

$$\chi_A(x) = \begin{cases} T, & x \in A \\ F, & x \notin A \end{cases}$$

Thus, a characteristic function represents a set.

Src: Grimaldi [Gri85], p. 98.

A Boolean function is a kind of characteristic function. We’ll use Boolean characteristic functions to describe a set of states as a logical formula.

Boolean Functions

A Boolean variable is a variable ranging over the values F and T.

A Boolean function of $n$ arguments is a function from $\{F, T\}^n$ to $\{F, T\}$.

(You will also see 1 and 0 used for T and F.)

We have the usual primitive Boolean functions of negation ($\neg$), disjunction ($g + h$), and conjunction ($g \cdot h$).

For example $f(x, y) = x \cdot (y \lor \overline{x})$.

Every expression in propositional logic corresponds to a Boolean function. ($+ \equiv \lor, \cdot \equiv \land, \overline{x} \equiv \neg x$)
Representing Elements and Subsets

We can “encode” the set $S$ in Boolean values. Every element of a finite set $S$ is assigned a unique tuple (vector) of Boolean values to each element.

There are $2^n$ vectors of length $n$. Choose $n$ such that $2^{n-1} < |S| \leq 2^n$. Some vectors may not be needed and are ignored if the $|S|$ is not a power of 2.

A characteristic function where $U$ is the set of all Boolean vectors can then return a subset of $S$ by returning T only for the encodings of the elements within the set.

Src: Huth and Ryan [HR04].

Example

$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$

Choose $n$ to be 3.

Encoding:

<table>
<thead>
<tr>
<th>Element</th>
<th>Boolean vector</th>
<th>Element</th>
<th>Boolean vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>000</td>
<td>$s_4$</td>
<td>100</td>
</tr>
<tr>
<td>$s_1$</td>
<td>001</td>
<td>$s_5$</td>
<td>101</td>
</tr>
<tr>
<td>$s_2$</td>
<td>010</td>
<td>$s_6$</td>
<td>110</td>
</tr>
<tr>
<td>$s_3$</td>
<td>011</td>
<td>not used</td>
<td>111</td>
</tr>
</tbody>
</table>

Representing a Set of States

For a state $s_j$ in a Kripke structure with atomic propositions ordered as $p_0, \ldots, p_n$, $l_{ji} = \begin{cases} v_i, & p_i \in L(s_j) \\ \overline{v_i}, & p_i \notin L(s_j) \end{cases}$

A set of states $S = \{s_0, s_1, \ldots, s_m\}$ is represented by the Boolean function:

$\chi_S(v_0, \ldots, v_n) = (l_{00} \cdot l_{01} \cdot \ldots \cdot l_{0n}) + (l_{10} \cdot l_{11} \cdot \ldots \cdot l_{1n}) + \ldots + (l_{m0} \cdot l_{m1} \cdot \ldots \cdot l_{mn})$

Each conjunct represents one state in the set $S$. 

Encodings of Kripke Structures

What’s the most natural encoding for sets of states in a Kripke model $M = (S, S_0, R, L)$?

Use the labelling function $L : S \rightarrow 2^{AP}$.

Assumption: two different states do not have the same labels: $\forall s_1, s_2 \in S \bullet L(s_1) = L(s_2) \Rightarrow s_1 = s_2$. (If this isn’t true, we can add extra atomic propositions to make it true.)

Create an ordering on the atomic propositions $p_0, p_1, \ldots, p_n$. Represent $s \in S$ by the vector $(v_0, v_1, \ldots, v_n)$ where for all $i$:

$v_i \equiv p_i \in L(s)$
Example

Kripke structure:

\[ s_0 \xrightarrow{x_1} s_1 \quad s_0 \xrightarrow{x_2} s_2 \]

Boolean Function Representation

Recall the set of operations we needed to implement the version of the model checking algorithm that manipulated sets of states:

<table>
<thead>
<tr>
<th>Operation on Set</th>
<th>Boolean Function Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>set complementation from ( S )</td>
<td>( \overline{x} )</td>
</tr>
<tr>
<td>(set difference)</td>
<td></td>
</tr>
<tr>
<td>set union</td>
<td>+</td>
</tr>
<tr>
<td>set intersection</td>
<td>\cdot</td>
</tr>
<tr>
<td>set equality</td>
<td>\equiv</td>
</tr>
<tr>
<td>pre-image</td>
<td>??</td>
</tr>
</tbody>
</table>

pre-image takes a subset of states \( X \) and returns the set of states that can make transitions into \( X \).

Representing the Transition Relation

The transition relation is a subset of the set \( S \times S \). We can use a Boolean function to represent this subset also.

As arguments to this function, we’ll need encodings of two states (source state of transition and destination state of transition), i.e., two Boolean vectors.

As before, the binary encoding of the states is given by the labelling function.

Representing the Transition Relation

A transition from \( s \) to \( s' \) (\( (s, s') \in R \)) is represented by a pair of Boolean vectors \( ((v_1, v_2, \ldots, v_n), (v'_1, v'_2, \ldots, v'_n)) \), where \( v_i \) is T if \( p_i \in L(s) \) and F otherwise; similarly \( v'_i \) is T if \( p_i \in L(s') \) and F otherwise.

The Boolean characteristic function for a set of pairs of states:

\[ R = \{(s_1, s'_1), (s_2, s'_2), \ldots, (s_m, s'_m)\} \] is:

\[
\chi_R(v_0, \ldots, v_n, v'_0, \ldots, v'_n) = \\
( (l_{00} \cdot l_{01} \cdot \ldots \cdot l_{0n}) \cdot (l'_{00} \cdot l'_{01} \cdot \ldots \cdot l'_{0n}) ) + \\
( (l_{10} \cdot l_{11} \cdot \ldots \cdot l_{1n}) \cdot (l'_{10} \cdot l'_{11} \cdot \ldots \cdot l'_{1n}) ) + \\
\ldots \\
( (l_{m0} \cdot l_{m1} \cdot \ldots \cdot l_{mn}) \cdot (l'_{m0} \cdot l'_{m1} \cdot \ldots \cdot l'_{mn}) )
\]

where \( l_{ji} = v_i \) if \( p_i \in L(s_j) \) and \( \neg v_i \) otherwise (similarly for \( l'_{ji} \)).
**Pre-image**

The pre-image of a set of states $X$ under a transition relation $R$ is:

$$\text{pre}_R(Y) = \{ t \in S \mid \exists s \cdot s \in Y \Rightarrow (t, s) \in R \}$$

Using characteristic functions:

$$\text{pre}_R(Y) = \{ t \in S \mid \exists s \cdot \chi_Y(s) \cdot \chi_R(t, s) \}$$

$$\chi_{\text{pre}_R}(t) = \exists s \cdot \chi_Y(s) \cdot \chi_R(t, s)$$

$$\chi_{\text{pre}_R}(v_0, \ldots, v_n) =$$

$$\exists v'_0, \ldots, v'_n \cdot \text{replace } Y (v'_0 | v_0, \ldots, v'_n | v_n) \cdot \chi_R(v_0, \ldots, v_n, v'_0, \ldots, v'_n)$$

How do we handle quantifiers in Boolean functions?

---

**Quantified Boolean Formulas**

Another logic is the logic of quantified Boolean formulas (QBF), which add universal and existential quantification of Boolean variables to propositional logic.

It has the same expressive power as propositional logic.

- $\exists x \cdot f = f|_{x=F} + f|_{x=T}$
- $\forall x \cdot f = f|_{x=F} \cdot f|_{x=T}$

However QBF may have a more compact representation of a formula or a more efficient means of carrying out quantification than the above substitutions.

*Src:* Clarke [CGP99]

---

**Boolean Function Representation**

Operations needed to implement the version of the model checking algorithm that manipulated sets of states:

<table>
<thead>
<tr>
<th>Operation on Set</th>
<th>Boolean Function Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>set complementation</td>
<td>$\neg x$</td>
</tr>
<tr>
<td>set union</td>
<td>$\lor$</td>
</tr>
<tr>
<td>set intersection</td>
<td>$\land$</td>
</tr>
<tr>
<td>set equality</td>
<td>$=$</td>
</tr>
<tr>
<td>pre-image(Y)</td>
<td>$\exists s \cdot \text{replace } Y' (s'</td>
</tr>
</tbody>
</table>

where $x$ and $s$ are vectors of Boolean variables.

The set of all states is represented at $T$.

---

**Symbolic CTL Model Checking**

function SAT ($f$)

```plaintext
case $f$
  
  true : return $S$;
  false : return $\emptyset$;
  atomic $f_1$ : return $\{ s \in S \mid f_1 \in L(s) \}$
  $\neg f_1$ : return $S \setminus \text{SAT}(f_1)$
  $f_1 \lor f_2$ : return $\text{SAT}(f_1) \cup \text{SAT}(f_2)$

  $\ldots$
  EX $f_1$ : return $\text{SAT}_{\text{EX}}(f_1)$
  EU $f_1$, $f_2$ : return $\text{SAT}_{\text{EU}}(f_1, f_2)$
  EG $f_1$ : return $\text{SAT}_{\text{EG}}(f_1)$
  AX $f_1$ : return $\text{SAT}((\neg (\text{EX} (\neg f_1))))$

  $\ldots$
```

---
Symbolic CTL Model Checking

\[ \text{pre}_3(x) = \exists s \cdot \chi_X(s) \cdot \chi_R(x, s) \]

function SAT_EX \( f_1 \)
\[
\text{return } \text{pre}_3(\text{SAT}(f_1)) \;
\]

function SAT_EU \( f_1, f_2 \)
\[
K = \text{SAT}(f_2); \; W = \text{SAT}(f_1);
\text{do} \;
\text{oldK} := K; \; K := \text{oldK} \cup (W \cap \text{pre}_3(\text{oldK}))
\text{until oldK} = K
\text{return K} \;
\]

function SAT_EG \( f_1 \)
\[
K := \text{SAT}(f_1);
\text{do} \;
\text{oldK} := K; \; K := \text{oldK} \cap \text{pre}_3(\text{oldK})
\text{until oldK} = K
\text{return K} \;
\]

Symbolic Model Checking

1. Represent the transition relation using a Boolean function.
2. Represent the atomic propositions of the temporal logic formula as Boolean functions.
3. Use the model checking algorithm over sets of states.
4. Implement the operations in this algorithm as operations on Boolean functions.

Usually the Boolean function for the transition relation is constructed directly from the model description in a language such as SMV, rather than turning the SMV model into a Kripke structure and then converting its transition relation into a Boolean function.

Summary

- Explicit CTL model checking (labelling a graph)
- Symbolic CTL model checking:
  1. Describe the model checking algorithm as computations over sets of states; abstract the operation of finding the set of states that can reach another set of states as a pre-image computation
  2. Characteristic functions to represent sets of states
  3. Use Boolean functions (logical formulae) as characteristic functions to represent sets of states; describe the Kripke structure as a Boolean characteristic function (symbolic)
  4. (next class) Binary Decision Diagrams – a data structure for manipulating Boolean functions
  5. (next class) CTL model checking as fixed point operations in the \( \mu \)-calculus

References


