CS 745 / ECE 725
Computer Aided Verification

Lecture 7: Symbolic CTL Model Checking

Jo Atlee

DC 2337, jmatlee@uwaterloo.ca

Office Hours: Mon 1:00-2:00, Wed 1:00-2:00

http://www.student.cs.uwaterloo.ca/~cs745

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:

$$\mathcal{M}, s_0 \models f$$

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

Motivation

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 every state is itemized and processed, one by one.

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

In particular, we can represent and process sets of states.

Symbolic Model Checking

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

These logical formulae can be expressed as Boolean functions, and can be represented and manipulated using an efficient data structure called 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.
2. Represent sets of states as characteristic functions
3. Represent Boolean functions (logical connectives) as characteristic functions; 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 μ-calculus

Explicit CTL Model Checking

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

- Check $\neg f_1$
- Check $f_1 \vee f_2$
- 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 same set of states "labelled" with the formula in the explicit CTL M/C algorithm.

Basic idea: 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 $\{s \in S \mid f_1 \in L(s)\}$

If the formula is $\neg f_1$:
Previous version: Add $\neg f_1$ to all label(s) if $f_1 \notin$ label(s)
New version: return $S -$ SAT(f_1)

If the formula is $f_1 \vee f_2$:
Previous version:
Add $f_1 \vee f_2$ to label(s) if either f_1 or f_2 are in label(s)
New version: SAT$(f_1) \cup$ SAT(f_2)

CTL Model Checking: EX

Previous version:

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

New version:

function SAT_EX (f_1)
  K = SAT (f_1);
  Y = $\{t \in S \mid \exists s \bullet\ s \in K \wedge (t, s) \in R\}$
  return Y;

CTL Model Checking: EU

Previous version:

CheckEU(f_1, f_2)
  $K := \{s \mid f_2 \in \underline{\text{label}}(s)\}$;
  for all $s \in K$ do
    $\underline{\text{label}}(s) := \underline{\text{label}}(s) \cup \{\textbf{E}[f_1 \ \textbf{U}\ f_2]\}$;
  while $K \neq \emptyset$ do
    choose $s \in K$;
    $K := K \setminus \{s\}$;
    for all $(t, s) \in R$ do
      if $\textbf{E}[f_1 \ \textbf{U}\ f_2] \notin \underline{\text{label}}(t)$
        and $f_1 \in \underline{\text{label}}(t)$ then
        $\underline{\text{label}}(t) := \underline{\text{label}}(t) \cup \{\textbf{E}[f_1 \ \textbf{U}\ f_2]\}$;
        $K := K \cup \{t\}$;

New version:

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

CTL Model Checking: EG

Previous version:

CheckEG(f_1)
  $K = \{s \mid f_1 \in \underline{\text{label}}(s)\}$;
  for all $s \in K$ do
    $\underline{\text{label}}(s) := \underline{\text{label}}(s) \cup \{\textbf{EG} f_1\}$;
  do
    $K := \{s \mid \textbf{EG} f_1 \in \underline{\text{label}}(s)\}$;
    for all $t \in K$ do
      $\underline{\text{label}}(t) := \underline{\text{label}}(t) - \{\textbf{EG} f_1\}$;
      for all $(t, u) \in R$ do
        if $u \in K$ then
          $\underline{\text{label}}(t) := \underline{\text{label}}(t) \cup \{\textbf{EG} f_1\}$;
  until $K = \{s \mid \textbf{EG} f_1 \in \underline{\text{label}}(s)\}$;

New version:

function SAT_EG(f_1)
  K := SAT(f_1);
  do
    $\underline{\text{oldK}}$:= K;
    K := $\underline{\text{oldK}} \cap$
      $\{t \in S \mid \exists u \bullet u \in \underline{\text{oldK}} \wedge (t, u) \in R\}$;
  until $\underline{\text{oldK}} = K$;
  return K;

Model Checking: SAT

function SAT (f)
  case f {
    true : return S;
    false : return $\emptyset$;
    atomic f_1 : return $\{s \in S \mid f_1 \in L(s)\}$
    $\neg f_1$: return $S -$ SAT(f_1)
    $f_1 \vee f_2$: return SAT(f_1) $\cup$ SAT(f_2)
    ...
    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($\neg$(**EX**($\neg f_1$)))
    ...
  }

Using Pre-image

New versions of the SAT subroutines rely on a function that computes a pre-image of a set of states.

pre-image: $\text{pre}_\exists(X) = \{t \in S \mid \exists s \bullet\ s \in X \wedge (t, s) \in R\}$

```
function SAT_EX (f1)
  return pre∃ (SAT (f1));

function SAT_EU (f1,f2)
  K = SAT(f2);
  W = SAT(f1);
  do
    oldK := K;
     K := oldK ∪ (W ∩ pre∃ (oldK))
  until oldK = K
  return K;
```

```
function SAT_EG(f1)
  K := SAT(f1);
  do
    oldK := K;
     K := oldK ∩ pre∃ (oldK);
  until oldK = K;
  return K;
```


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: $\text{pre}_\exists(X) = \{t \in S \mid \exists s \bullet\ s \in X \wedge (t, s) \in R\}$

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 μ-calculus

Characteristic Function

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

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

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 $\{\mathsf{F}, \mathsf{T}\}^n$ to $\{\mathsf{F}, \mathsf{T}\}$.

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

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

$$f(x, y) = x \cdot (y + \overline{x})$$

Every expression in propositional logic corresponds to a Boolean function. ($+ \equiv \vee, \cdot \equiv \wedge, \overline{x} \equiv \neg x$)

Representing Elements and Subsets

We can "encode" the set S in Boolean values.

Every element s of a finite set S is encoded as a unique tuple (vector) of Boolean values (e.g., 101).

A subset T, then, can be represented by a Boolean function χ_T that maps a Boolean vector onto 1 if its corresponding element $s \in T$ and maps to 0 otherwise.

If the number of elements in S is $|S|$, then we need vectors of length n such that $2^{n-1} < |S| \leq 2^n$. Some vectors may not be needed if $|S|$ is not a power of 2.

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:

Element	Boolean vector	Element	Boolean vector
s_0	000	s_4	100
s_1	001	s_5	101
s_2	010	s_6	110
s_3	011	not used	111

Encodings of Kripke Structures

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

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

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 forall i:

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

Representing a Set of States

A state s_j in a Kripke structure with atomic propositions ordered as $p_0, \ldots p_n$ also has a boolean function representation $l_{j1} \cdot l_{j2} \cdot \ldots \cdot l_{jn}$, where

$$l_{ji} = \begin{cases} v_i, & p_i \in L(s_j) \\ \overline{v_i} & p_i \notin L(s_j) \end{cases}$$

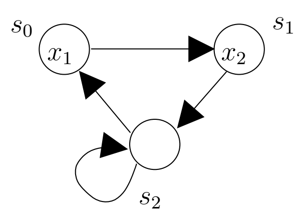
A set of states $S = \{s_0, s_1, \ldots s_m\}$ is represented by the Boolean function:

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

Each conjunct represents one state in the set S.

Example

Kripke structure:



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:

$$\begin{aligned} &\chi_R(v_0, \ldots, v_n, v'_0, \ldots, v'_n) = \\ &\quad ((l_{00} \cdot l_{01} \cdot \ldots l_{0n}) \cdot (l'_{00} \cdot l'_{01} \cdot \ldots l'_{0n})) + \\ &\quad ((l_{10} \cdot l_{11} \cdot \ldots l_{1n}) \cdot (l'_{10} \cdot l'_{11} \cdot \ldots l'_{1n})) + \\ &\quad \ldots \\ &\quad ((l_{m0} \cdot l_{m1} \cdot \ldots l_{mn}) \cdot (l'_{m0} \cdot l'_{m1} \cdot \ldots l'_{mn})) \end{aligned}$$

where l_{ji} is 1 if $p_i \in L(s_j)$ and 0 otherwise (similarly for l'_{ji}).

Boolean Function Representation

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

Operation on Set	Boolean Function Operation
set complementation from S (set difference)	$\overline{x}$
set union	$+$
set intersection	$\cdot$
set equality	=
pre-image	??

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

Pre-image

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

$$\text{pre}_\exists(Y) = \{t \in S \mid \exists s \bullet\ s \in Y \wedge (t, s) \in R\}$$

Using characteristic functions:

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

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

Replacing elements with their Boolean value representation:

$$\begin{aligned} &\chi_{\text{pre}_\exists}(v_0, \ldots, v_n) = \\ &\qquad \exists v'_0, \ldots, v'_n \bullet \chi_Y(v'_0, \ldots, v'_n) \cdot \chi_R(v_0, \ldots, v_n, v'_0, \ldots, v'_n) \end{aligned}$$

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.

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

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:

Operation on Set	Boolean Function Operation
set complementation	$\overline{x}$
set union	$+$
set intersection	$\cdot$
set equality	=
pre-image(Y)	$\chi_{\text{pre}_\exists}(s) = \exists s' \bullet \chi_Y(s') \cdot \chi_R(s, s')$.

where s and s' are vectors of Boolean variables.

Summary

- 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. Represent sets of states as characteristic functions
  3. Represent Boolean functions (logical connectives) as characteristic functions; 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 μ-calculus

References

[AH98] Rajeev Alur and Thomas A. Henzinger. Computer-Aided Verification. An Introduction to Model Building and Model Checking for Concurrent Systems. 1998. Draft: http://www-cad.eecs.berkeley.edu/ tah/CavBook/.

[CGP99] Edmund Clarke Jr., Orna Grumberg, and Doron A. Peled. Model Checking. MIT Press, 1999.

[Gri85] Ralph P. Grimaldi. Discrete and Combinatorial Mathematics. Addison-Wesley, Reading, Massachusettes, 1985.

[HR04] Michael R. A. Huth and Mark D. Ryan. Logic in Computer Science. Cambridge University Press, Cambridge, 2004. Second Edition.