Introduction to Software Verification

Orna Grumberg

Lectures Material
winter 2017-18

Lecture 6

Explicit Model Checking for CTL

Model Checking [CE81,QS82]

An efficient procedure that receives:

- A finite-state model describing a system
- A temporal logic formula describing a property

It returns

yes, if the system has the property

no + Counterexample, otherwise

CTL Model Checking M |= f

- Goal: For each s, computes **label(s)**, which is the set of subformulas of f, true in s
- The Model Checking algorithm works **iteratively** on subformulas of **f**, from **simpler** subformulas to more **complex** ones
- For checking **AG(request ⇒ AF grant)**
  - Check **grant**, **request**
  - Then check **AF grant**
  - Next check **request ⇒ AF grant**
  - Finally check **AG(request ⇒ AF grant)**

Model Checking M |= f (cont.)

- We check subformula g of f only after all subformulas of g have already been checked
- For subformula g, the algorithm adds g to label(s) for every state s that satisfies g
- When we finish checking g, the following holds:
  - $g \in \text{label}(s) \Leftrightarrow M,s \vDash g$

Model Checking M |= f (cont.)

Alternative description

Denote $S_g = \{ s \mid M, s \models g \}$

- The goal of model checking is to compute S_g for each subformula g of f
  - In particular, S_f

Model Checking M |= f (cont.)

- $M \models f$ if and only if $f \in labels(s)$ for **all initial states** s of M

- $M \models f$ if and only if $S_0 \subseteq S_f$
- The algorithm has time complexity: $O(|M| \times |f|)$

Model Checking Atomic Propositions

- For atomic propsition **p∈AP**:

  $p \in label(s) \Leftrightarrow p \in L(s)$

  Held by alg — Defined by M

How do we handle **more complex** formulas?

Observation:

- Sufficient to handle $\neg$, $\vee$, EX, EU, EG

Model Checking $\neg$, $\vee$ formulas

$\neg f_1$: add to label(s) if and only if $f_1 \notin$ labels(s)

$f_1 \vee f_2$: add to label(s) if and only if
$f_1 \in$ labels(s) or $f_2 \in$ labels(s)

Model Checking $g = EX\ f_1$

add g to label(s) if and only if s has a successor t such that $f_1 \in$ labels(t)

```
procedure CheckEX (f1)
    T := { t | f1 ∈ label(t) }
    while T ≠ ∅ do
      choose t ∈T ;  T := T \ {t} ;
      for all s  s.t.  R(s,t) do
            if EX f1 ∉ label(s) then
                label(s) : = label(s) ∪ { EX f1};
      end for all
    end while
```

Model Checking $g = E(f_1\ U\ f_2)$

procedure CheckEU (f_1, f_2)

T := { s | $f_2 \in$ label(s) }

For all s$\in$T do label(s) := label(s)$\cup$ { $E(f_1\ U\ f_2)$ }

while T $\neq \emptyset$ do

choose s $\in$T ; T := T \ {s} ;

for all t s.t. R(t,s) do

if $E(f_1\ U\ f_2) \notin$ label(t) and $f_1 \in$ label(t) then

label(t) : = label(t) $\cup$ { $E(f_1\ U\ f_2)$ };

T : = T $\cup$ {t}

end for all

end while

Do not add a state to T more than once

Example $g = E(f_1 \; U \; f_2)$

- How shall we handle $g = EF\ f_1$?

Remarks:

We transform a logical question of $M,s \models f$ to a graph traversal algorithm

The algorithm is guaranteed to terminate

Model Checking $g = EG\ f_1$

$s \models EG\ f_1$

iff

There is a path π, starting at s, such that $\pi \models G\ f_1$

iff

There is a path from s to a strongly connected component, where all states satisfy f_1

Model Checking $g = EG\ f_1$

- A Strongly Connected Component (SCC) in a graph is a subgraph **C** s.t. **every node** in C is reachable from **any other node** in C **via nodes** in C

- An SCC C is maximal (MSCC) if it is not contained in any other SCC in the graph
- C is nontrivial if it contains at least one edge. Otherwise, it is trivial

Tarjan has a linear algorithm in O(|S|+|R|) for finding all MSCCs in a graph, including the trivial SCCs.

Model Checking $g = EG\ f_1$

Why using maximal SCCs?

Complexity concerns:

There are up to $2^{|S|}$ non-maximal SCCs in M

Number of maximal SCCs is at most $|S|$

- Disjoint
- Overall number of states is |S|

Model Checking $g = EG\ f_1$

Reduced structure for M and f_1:

Remove from M all states s.t. $f_1 \notin$ label(s)

R' might no longer be total

Resulting model: $M' = (S', R', L')$

- $S' = \{ s \mid M, s \models f_1 \}$
- $R' = (S' \times S') \cap R$
- $L'(s') = L(s')$ for every $s' \in S'$

Theorem: $M,s \models EG\ f_1$ iff

1. $s \in S'$ and
2. There is a path in M' from s to some state in a nontrivial maximal strongly connected component of M'

Model Checking $g = EG\ f_1$

procedure CheckEG (f_1)

$S' := \{s \mid f_1 \in label(s)\}$

$MSCC := \{ C \mid C \text{ is a nontrivial MSCC of } M' \}$

$T := \cup_{C \in MSCC} \{ s \mid s \in C \}$

For all $s \in T$ do $label(s) := label(s) \cup \{ EG\ f_1 \}$

while $T \neq \emptyset$ do

  choose $s \in T$; $T := T \setminus \{s\}$;

  for all $t \in S'$ s.t. $R(t,s)$ do

    if $EG\ f_1 \notin label(t)$ then

      $label(t) := label(t) \cup \{ EG\ f_1 \}$;

      $T := T \cup \{t\}$

  end for all

end while

Complexity for EG f_1

- Computing M': $O(|S| + |R|)$
- Computing MSCCs using Tarjan's algorithm: $O(|S'| + |R'|)$
- Labeling all states in MSCCs: $O(|S'|)$
- Backward traversal: $O(|S'| + |R'|)$

Overall: $O(|S| + |R|) = O(M)$

Theorem: $M,s \models EG\ f_1$ iff

1. $s \in S'$ and
2. There is a path in M' from s to some state in a nontrivial maximal strongly connected component of M'

Proof: