Formal methods - Temporal logic

Francesco Penasa

March 19, 2020

2020 03 17

1 CTL model checking

1.1 general ideas

CTL Model Checking is a formal verification technique where:

1. the system is represented as a **Finite State Machine** M
2. the property is as CTL formula ϕ
$$AG(p \to AFq)$$
in all possible initial states, in all possible paths starting from the initial state (AG = in all reachable states), if p holds then for all path sooner or later we hold q.
3. the model checking algorithm checks whether in **all initial states** of M **all the executions** of the model **satisfy** the formula ($M \models \phi$).

In general it works in two macro steps:

1. construct the set of states where the formula holds:
$$[\phi] := s \in S : M, s \models \phi$$
($[\phi]$ is called the **denotation** of ϕ)
2. then compare with the set of **initial states**:
$$I \subseteq [\phi]?$$

In order to compute $[\phi]$ we proceed bottom up, this way.

1. $[q]$
2. $[AFq]$
3. $[p]$
4. $[p \to AFq]$
5. $[AG(p \to AFq)]$

To compute each $[\phi_i]$

1. labeling function: in which state a given proposition holds.
2. use set operations
3. compute **pre-images**
4. applying **tableaux rules**, until a fixpoint is reached: recursive definition of the propostitions

1.2 A simple example

1. q We find the states where q holds : state 2
2. AFq recall the AF tableau rule $AFq \leftrightarrow (q \vee AXAFq)$

1.3 Some theoretical issues

stuff

1.4 Algorithms

Assume using only $\neg \wedge EXEUEG$

1. for every $\phi_i \in Sub(\phi)$, find $[\phi_i]$
2. Check if I $\subseteq [\phi]$

1. propositional atoms: apply labeling function
2. boolean operator: apply standard set operations
3. temporal operator: apply recursively the tableaux rules, until a **fixpoint** is reached.

```
state_set Check(CTL_formula β){
        case β of
        true: return S;
        false: return {};
        ¬β1: return S \ Check(β1);
        β1 ∧ β2: return Check(β1) ∩ Check(β2);
        EXβ1: return PreImage(Check(β1));
        EGβ1: return Check_EG(Chekck(β1));
        E(β1Uβ2): return Check_EU(Check(β1), Check(β2));
}
```

PreImage Add to the S if it is a successor and it holds

Check_EG Intersect to the S if it is a successor and it holds

Check_EU Add to the S if it Intersect is a successor and it holds

Check_EF

$$X' := X \cup PreImage(X)$$

1.5 Some examples

1.5.1 Mutual exclusion

INSERT figure

$$M \models AGAFC_1?$$

To use the algorithm we rewrite it in terms of EG

$$\Rightarrow M \models \neg EFEG\neg C_1?$$

Compute the denotations of

$$[\neg C_1]$$

easy. Now this

$$[EG\neg C_1]$$

We take our current set of states and **intersect** with its preimage. Drop all states that does not hold this EXS next step EXS, repeat until there are no changes. Fixpoint reached!

$$[EFEG\neg C_1]$$

We do the **union** of our current approximation with the preimage. EXS, repeat until we reach a fixed point.

$$[\neg EFEG\neg C_1]$$

Easy af. we can prove that the property is not verified.

1.5.2 liveness

$$M \models AG(T_1 \rightarrow AFC_1)? \Rightarrow M \models \neg EF(T_1 \wedge EG\neg C_1)?$$

1.5.3 Complexity of CTL model checking

$O(|\phi|)$ steps $O(|M|)$ states to explore. $O(|M| * |\phi|)$ Typically the number of states is huge.

1.6 Subcase: invariants

Invariant properties have the form AGp (e.g., $AG\neg bad$). To check that a bad state is not reachable ($AG\neg bad == \neg EFbad$). In this very particular case we can reason that it is possible to stop when we intersect with the initial step OR when we reach a fixed point. But we can do better.

1.6.1 Symbolic forward model checking of invariants

Forward checking

1. compute [bad]
2. compute the set of initial states I

3. compute the set of reachable states from I until it intersect [bad] or a fixed point is reached.

Basic step is the **Forward Image**

$$Image(Y) := s'|s \in Y and R(s, s') holds$$

Simplest form: compute the set of reachable states.

1.7 Exercises

CTL model checking $\phi = AG((p \wedge q) \to EGq)$ TODO ADD figure
Rewrite ϕ into an equivalent formula ϕ' expressed in terms of EX, EG, EU/ EF only.
Compute bottom-up the denotations

$$p \wedge q = s_1$$

$$q = s_1 \cup s_0$$

next step

$$EGq = s_1 \cup s_0 = s_0, s_1$$

next step

$$\neg EGq = s_2$$

next steps

$$(p \wedge q) \wedge (\neg EGq) = \emptyset$$

next step

$$EF((p \wedge q) \wedge (\neg EGq)) = \emptyset$$

next step

$$\neg EF((p \wedge q) \wedge (\neg EGq)) = S = s_1, s_2, s_3$$

finished.
As consequence of the previous point, say wheter $M \models \phi$ **Yes**

CTL model checking 2

$$AG(AFp \to q)$$

Rewrite

$$pirpippir$$

Compute time

$$\neg p = s_2, s_1$$

$$EG\neg p = s_1, s_2 \cup s_0, s_1, s_2 = s_1, s_2$$

$$\neg EG\neg p = s_0$$

$$\neg q = s_1$$

$$EG\neg q = s_1 \cup s_0, s_1, s_2 = s_1$$

$$(\neg EG\neg p) \wedge (EG\neg q) = s_1 \cup s_0, s_2 = \emptyset$$

$$EF((\neg EG\neg p) \wedge (EG\neg q)) = \emptyset$$

$$\neg EF((\neg EG\neg p) \wedge (EG\neg q)) = s_0, s_1, s_2$$

2 Homework

Apply the same process to all the CTL examples of Chapter 3 EX = pre image; EG = intersection, EF equal Union to what we have; E(a U b) = b U (a inter image(a))