

Model Checking

We next sketch an algorithm for determining in which states of a model \mathcal{M} a given CTL formula ϕ is satisfied.

The *input* to the algorithm consists of a CTL model $\mathcal{M} = (S, \rightarrow, L)$ and a CTL formula ϕ .

The *output* includes the set of all states $s \in S$, such that

$$\mathcal{M}, s \models \phi.$$

For simplicity we assume that the formula ϕ contains no connectives other than $\perp, \neg, \wedge, EX, AF$, and EU .

This restriction is not essential as this set of connectives is adequate for CTL. Arbitrary CTL formulas (with other connectives) need to be preprocessed and translated into equivalent form in terms of the selected connectives.

A key observation for the algorithm is that to determine whether a formula ψ is satisfied (in a state s) one only needs to know whether its immediate subformulas are satisfied (possibly in states other than s).

The model checking algorithm systematically labels the states of \mathcal{M} by all (immediate or nonimmediate) subformulas of ϕ that are satisfied there.

The Labelling Procedure

The labelling procedure repeats the following steps for all subformulas ψ of ϕ , beginning with the smallest formulas and proceeding to increasingly larger formulas up to ϕ :

- Label no state with \perp .
- If ψ is an atomic description p , then label a state s with p if $p \in L(s)$.
- If $\psi = \neg\psi_1$ then label s with ψ if it is not already labelled with ψ_1 .
- If $\psi = \psi_1 \wedge \psi_2$ then label s with ψ if it is already labelled with ψ_1 and ψ_2 .
- If $\psi = EX\psi_1$ then label s with ψ if one of its successor states is labelled with ψ_1 .
- If $\psi = AF\psi_1$ then first label each state s already labelled with ψ_1 by ψ , and then repeatedly label any state with ψ for which all successor states are labelled with ψ , until there is no change.
- If $\psi = E[\psi_1 U \psi_2]$ then first label each state s already labelled with ψ_2 by ψ , and then repeatedly label any state with ψ that is already labelled with ψ_1 and for which at least one successor state is labelled with ψ , until there is no change.

Model Checking Rules

A different view of the model checking procedure is that it computes, for all subformulas ψ of a given formula ϕ , the set $\llbracket \psi \rrbracket^{\mathcal{M}}$ (or $\llbracket \psi \rrbracket$ for simplicity) of all states that satisfy ψ , based on the following recursive rules (which assume that the set of states S has n elements):

$$\begin{aligned}\llbracket \perp \rrbracket &= \emptyset \\ \llbracket p \rrbracket &= \{s \in S : p \in L(s)\} \\ \llbracket \neg \psi \rrbracket &= S \setminus \llbracket \psi \rrbracket \\ \llbracket \psi_1 \wedge \psi_2 \rrbracket &= \llbracket \psi_1 \rrbracket \cap \llbracket \psi_2 \rrbracket \\ \llbracket EX\psi_1 \rrbracket &= J(\llbracket \psi_1 \rrbracket) \\ \llbracket AF\psi_1 \rrbracket &= F_{\psi_1}^n(\emptyset) \\ \llbracket E[\psi_1 U \psi_2] \rrbracket &= H_{\psi_1, \psi_2}^n(\emptyset)\end{aligned}$$

where

$$\begin{aligned}F_{\psi_1}(X) &= \llbracket \psi_1 \rrbracket \cup K(X) \\ H_{\psi_1, \psi_2}(X) &= \llbracket \psi_2 \rrbracket \cup (\llbracket \psi_1 \rrbracket \cap J(X)) \\ J(X) &= \{s \in S : s \rightarrow s' \text{ for some } s' \text{ in } X\} \\ K(X) &= \{s \in S : s' \in X \text{ whenever } s \rightarrow s'\}\end{aligned}$$

and, for any function $F : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$, functions F^i , for $i \geq 0$, are defined inductively by:

$$F^i(X) = \begin{cases} X & \text{if } i = 0 \\ F(F^{i-1}(X)) & \text{if } i > 0 \end{cases}$$

Monotone Functions

Let S be a set of states and $F : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ be a function on the power set of S .

Definition

1. We say that F is *monotone* if $X \subseteq Y$ implies $F(X) \subseteq F(Y)$ for all subsets X and Y of S .
2. We say that a subset X of S is a *fixed point* of F if $F(X) = X$.

For example, the above functions J and K are monotone. If $X \subseteq Y$, then

$$\begin{aligned} J(X) &= \{s \in S : s \rightarrow s' \text{ for some } s' \text{ in } X\} \\ &\subseteq \{s \in S : s \rightarrow s' \text{ for some } s' \text{ in } Y\} \\ &= J(Y) \end{aligned}$$

and

$$\begin{aligned} K(X) &= \{s \in S : s' \in X \text{ whenever } s \rightarrow s'\} \\ &\subseteq \{s \in S : s' \in Y \text{ whenever } s \rightarrow s'\} \\ &= K(Y) \end{aligned}$$

Least and Greatest Fixed Points

Theorem (Knaster-Tarski).

Let S be a set with n elements. If $F : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ is a monotone function, then

1. $F^n(\emptyset)$ is the least fixed point of F , and
2. $F^n(S)$ is the greatest fixed point of F .

The theorem asserts the existence of least and greatest fixed points, respectively, and in addition provides a way of computing them.

Sketch of proof.

First use mathematical induction to prove that

$$\emptyset = F^0(\emptyset) \subseteq F^1(\emptyset) \subseteq \dots \subseteq F^n(\emptyset) \subseteq F^{n+1}(\emptyset).$$

Observe that this sequence (i) is nondecreasing with respect to the subset relation, (ii) contains $n + 2$ sets, and (iii) contains only subsets of S (i.e., sets of at most n elements). This implies that some set in the sequence must be equal to the next set.

In other words, $F^i(\emptyset) = F^{i+1}(\emptyset)$, and hence $F^i(\emptyset)$ is a fixed point of F , for some $i \leq n$. We have

$$F^i(\emptyset) = F^{i+1}(\emptyset) = \dots = F^n(\emptyset).$$

and thus $F^n(\emptyset)$ is a fixed point of F .

Let X be another fixed point of F . Use mathematical induction again to prove that $F^i(\emptyset) \subseteq X$, for all i , which implies that $F^n(\emptyset)$ is a least fixed point of F .

The second part of the theorem can be proved in a similar way, but using S instead of \emptyset and the superset relation instead of the subset relation.

Correctness of Model Checking

Consider the functions F_ϕ , G_ϕ , and $H_{\phi,\psi}$ defined by:

$$\begin{aligned}F_\phi(X) &= \llbracket \phi \rrbracket \cup K(X) \\G_\phi(X) &= \llbracket \phi \rrbracket \cap J(X) \\H_{\phi,\psi}(X) &= \llbracket \psi \rrbracket \cup (\llbracket \phi \rrbracket \cap J(X))\end{aligned}$$

where ϕ and ψ are CTL formulas and J and K are as defined previously, that is,

$$\begin{aligned}J(X) &= \{s \in S : s \rightarrow s' \text{ for some } s' \text{ in } X\} \text{ and} \\K(X) &= \{s \in S : s' \in X \text{ whenever } s \rightarrow s'\}.\end{aligned}$$

Theorem

1. The functions F_ϕ , G_ϕ , and $H_{\phi,\psi}$ are monotone.
2. $\llbracket AF\phi \rrbracket$ is a least fixed point of F_ϕ .
3. $\llbracket EG\phi \rrbracket$ is a greatest fixed point of G_ϕ .
4. $\llbracket E(\phi U \psi) \rrbracket$ is a least fixed point of $H_{\phi,\psi}$.

Sketch of proof.

1. We already know that J and K are monotone. Thus, if $X \subseteq Y$, then $J(X) \subseteq J(Y)$ and $K(X) \subseteq K(Y)$. Suppose $X \subseteq Y$. Then

$$\begin{aligned} \llbracket \phi \rrbracket \cup K(X) &\subseteq \llbracket \phi \rrbracket \cup K(Y), \\ \llbracket \phi \rrbracket \cap J(X) &\subseteq \llbracket \phi \rrbracket \cap J(Y), \text{ and} \\ \llbracket \psi \rrbracket \cup (\llbracket \phi \rrbracket \cap J(X)) &\subseteq \llbracket \psi \rrbracket \cup (\llbracket \phi \rrbracket \cap J(Y)). \end{aligned}$$

In other words, the functions F_ϕ , G_ϕ , and $H_{\phi,\psi}$ are monotone.

2. We first show that $\llbracket AF\phi \rrbracket$ is a fixed point of F_ϕ and for that purpose use the equivalence,

$$AF\phi \equiv \phi \vee AX AF\phi.$$

Specifically, we have

$$\begin{aligned} s \in \llbracket AF\phi \rrbracket & \\ \text{iff } s \models AF\phi & \\ \text{iff } s \models \phi \text{ or } s \models AX AF\phi & \\ \text{iff } s \in \llbracket \phi \rrbracket \text{ or } s \in K(\llbracket AF\phi \rrbracket) & \\ \text{iff } s \in \llbracket \phi \rrbracket \cup K(\llbracket AF\phi \rrbracket) & \\ \text{iff } s \in F_\phi(\llbracket AF\phi \rrbracket) & \end{aligned}$$

In other words, $F_\phi(\llbracket AF\phi \rrbracket) = \llbracket AF\phi \rrbracket$.

Next we show that $\llbracket AF\phi \rrbracket$ is a *least* fixed point of F_ϕ . Let X be a fixed point of F_ϕ , i.e., $F_\phi(X) = X$. We prove by contradiction that $\llbracket AF\phi \rrbracket \subseteq X$.

First note that if a state s is an element of $\llbracket AF\phi \rrbracket \setminus X$, then $s \notin \llbracket \phi \rrbracket$, $s \in K(\llbracket AF\phi \rrbracket)$, and $s \notin K(X)$. From this we may infer that there is a state s' such that $s \rightarrow s'$ and $s' \in \llbracket AF\phi \rrbracket \setminus X$. Thus, if $\llbracket AF\phi \rrbracket \setminus X$ contains any elements, then we can inductively define a sequence

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$$

of states in $\llbracket AF\phi \rrbracket \setminus X$. Furthermore, $s_i \not\models \phi$, for all i , which contradicts that $s_0 \models AF\phi$.

In short, $\llbracket AF\phi \rrbracket \setminus X = \emptyset$, which implies that $\llbracket AF\phi \rrbracket \subseteq X$.

3. We may use the equivalence,

$$EG\phi \equiv \phi \wedge EX EG\phi$$

to prove that $\llbracket EG\phi \rrbracket$ is a fixed point of G_ϕ . We need to show it is a *greatest* fixed point.

Let X be a fixed point of G_ϕ , i.e., $G_\phi(X) = X$. We need to show that $X \subseteq \llbracket EG\phi \rrbracket$. First note that if $s \in X$ then (i) $s \in \llbracket \phi \rrbracket$ and (ii) $s \rightarrow s'$, for some state $s' \in X$. We may use this observation to inductively define, for any state $s \in X$, an infinite sequence

$$s = s_0 \rightarrow s_1 \cdots \rightarrow s_n \rightarrow \cdots$$

such that $s_i \in X \cap \llbracket \phi \rrbracket$, for all i . The existence of this an infinite computation path implies that $s \in \llbracket EG\phi \rrbracket$. We conclude that $X \subseteq \llbracket EG\phi \rrbracket$.

4. The proof of the last part is left as an exercise.

Complexity

The computation of all sets $\llbracket \psi \rrbracket$ can be done in time $O(f \cdot V \cdot (V + E))$, where f is the number of connectives in ϕ , V the number of states, and E the number of transitions. The textbook describes a specific such algorithm that is linear in the size of the input formula and quadratic in the size of the model.

A more efficient version of the algorithm uses EX , EU , and EG as the selected temporal connectives and processes corresponding formulas in a more sophisticated way.

Specifically, the labelling procedure for EX and EU formulas searches the transition system in a “backwards breadth-first” manner to avoid having to pass over any node twice.

The computation of sets $\llbracket EG\psi \rrbracket$ is restricted to subgraphs obtained by deleting all states that do not satisfy ψ (and their transitions). For this restricted graph one computes strongly connected components and then uses a backwards breadth-first search to identify all states that can reach a strongly connected component (containing at least one transition). It is these latter states that are labelled with $EG\psi$.

This improved algorithm has complexity $O(f \cdot (V + E))$, i.e., is linear in both the size of the input formula and the size of the model. Unfortunately, the size of a model may be exponential in the number of variables and the number of components of the system that execute in parallel.

Fairness

The verification of $\mathcal{M}, s \models \phi$ might fail because the model \mathcal{M} may contain unrealistic behavior that does not reflect possible computations in the actual system being analyzed.

For example, in the mutual exclusion example we may want to add transitions that allow a process to stay in its critical section as long as it needs. Unfortunately, the extended model will not satisfy the liveness property, even though it may be reasonable to assume that a process *will eventually exit from its critical section* after a finite amount of time.

The model checking method can be refined to deal with so-called *fairness constraints* of the form

Property ϕ is true infinitely often.

For instance, in the mutual exclusion example, we may want to specify fairness constraints, $\neg c_1$ and $\neg c_2$.

In the modified model checking method one considers only *fair computation paths*, i.e., those paths along which the specified fairness constraints are satisfied infinitely often.

Formally this can also be expressed by replacing the “path quantifiers” A and E by more restrictive variants, A_C and E_C , meant to range over fair computation paths.