

# Chapter 15

## Model checking

### Contents

---

<b>15.1 Symbolic Representation of Kripke Structures</b> . . . . .	<b>224</b>
<b>15.2 Model-checking reachability properties</b> . . . . .	<b>224</b>
15.2.1 Computing a symbolic representation of the set of reachable states . . . . .	225
15.2.2 Verifying safety properties . . . . .	227
15.2.3 Backward reachability . . . . .	227
<b>15.3 Bounded model checking</b> . . . . .	<b>227</b>
<b>15.4 CTL model checking</b> . . . . .	<b>227</b>
<b>Exercises</b> . . . . .	<b>227</b>

---

There are many reasons for wishing to verify temporal properties of transition systems in a completely automatic way. This can be done by first building a Kripke structure which represents the transition system, then expressing the property by a temporal formula and using *model checking* to check whether the Kripke structure is a model of this formula.

**DEFINITION 15.1 (Model Checking)** *Model checking* is the following decision problem: given a Kripke structure  $K$  and a temporal formula  $A$ , is  $A$  valid in  $K$ ? When  $K$  is given symbolically, one speaks of *symbolic model checking*.  $\square$

There are several algorithms for finite-state symbolic model checking. In this chapter we consider only one of them. The algorithm is based on the observation that one can represent symbolically sets of states which satisfy a given temporal formula using monotonicity properties of temporal operators. Since temporal operators are monotone, one can find a symbolic representation of sets of states using a fixpoint construction.

We will begin in Section 15.2 with a special case of model-checking reachability properties. Though these properties are simpler than properties expressed by arbitrary temporal formulas, they can be used to capture the main idea of using monotonicity. In addition, it

has been observed that reachability properties are the most common properties arising in verification. Then in Section 15.3 we introduce a concept of *bounded model checking* which can be used to approximate model checking when the search space is too large. Section 15.4 gives an algorithm for full CTL model checking.

## 15.1 Symbolic Representation of Kripke Structures

For transition systems, we will also be interested in their symbolic representation. These notions can be defined in almost the same way as for transition systems. First, we introduce a set  $\mathcal{X}' = \{v' \mid v \in \mathcal{X}\}$  of *next state variables*. Then, we show how states and pairs of states of a Kripke structure can be interpreted as interpretations.

**DEFINITION 15.2 (States as Interpretations)** Let  $K = (S, In, R, L)$  be a Kripke structure,  $s \in S$  be a state of  $K$ ,  $x \in \mathcal{X}$  be a variable, and  $v \in \text{dom}(x)$ . We define

$$s \models x = v \stackrel{\text{def}}{=} L(s)(x) = v.$$

Likewise, for every pair of variables  $x \in \mathcal{X}$ ,  $x' \in \mathcal{X}'$  and pair of states  $(s, s')$ , we define

$$\begin{aligned} (s, s') \models x = v &\stackrel{\text{def}}{=} s \models x = v; \\ (s, s') \models x' = v &\stackrel{\text{def}}{=} s' \models x = v. \end{aligned} \quad \square$$

In other words, we identify the state  $s$  with the interpretation  $L(s)$ . Indeed,  $L(s)(x) = v$  if and only if  $L(s) \models x = v$ , so we have  $L(s) \models x = v$  if and only if  $s \models x = v$ .

Since we consider states and pairs of states as interpretations, we can carry over the definition of truth of arbitrary formulas of PLFD to states and pairs of states.

**DEFINITION 15.3 (Symbolic Representation of Kripke Structure)** Let  $K$  be a Kripke structure.  $S_0$  be a set of states and  $A$  be a formula with variables in  $\mathcal{X}$ . We say that  $A$  *symbolically represents*, or simply *represents*  $S_0$ , if  $S_0 = \{s \mid s \models A\}$ . Let  $B$  be a formula with variables in  $\mathcal{X} \cup \mathcal{X}'$  and  $t$  be a set of pairs of states. We say that  $B$  *symbolically represents*, or simply *represents*  $t$ , if  $t = \{(s, s') \mid (s, s') \models B\}$ . In particular, we can speak about the symbolic representation of the transition relation of  $K$ .

Let  $A_1$  be a formula with variables in  $\mathcal{X}$  and  $A_2$  be a formula with variables in  $\mathcal{X} \cup \mathcal{X}'$ . We say that the pair  $(A_1, A_2)$  *symbolically represents* a Kripke structure  $K = (S, In, R, L)$  if  $A_1$  symbolically represents  $In$  and  $A_2$  symbolically represents  $R$ .  $\square$

## 15.2 Model-checking reachability properties

In this section we give an algorithm for symbolic model-checking a special kind of reachability properties expressed by a temporal formula

$$\mathbf{EF} \textit{unsafe}, \quad (15.1)$$

where *unsafe* is a propositional formula, i.e., a formula, involving no temporal operators. Such model-checking problems often arise in connection with checking *safety properties* of systems, so that *unsafe* expresses unsafety of the system. Our algorithm will be based three observations:

- (1) If a state  $s$  is reachable from an initial state, then it can be reached in a finite number  $k$  of steps.
- (2) A symbolic representation of the set of states reachable in  $\leq k$  steps can be computed from a symbolic representation of  $K$ .
- (3) The set of states reachable in  $\leq k$  steps is a subset of the set of states reachable in  $\leq k+1$  steps, and hence for some  $k$  the set of states reachable in  $\leq k$  states coincides with the set of states reachable in  $\leq k+1$  state.

In this chapter we assume a fixed finite Kripke structure  $K = (S, I, R, L)$  and give all definitions with respect to this Kripke structure. For example, if we say that there is a transition from a state  $s$  to a state  $s'$ , we mean that  $(s, s') \in R$ . Likewise, if we write  $s \models p$ , where  $p$  is a boolean variable, we mean  $p \in L(s)$ . Later we will make more assumptions about a symbolic representation of  $K$ .

### 15.2.1 Computing a symbolic representation of the set of reachable states

**DEFINITION 15.4 (Reachability)** If there exists a sequence of states  $s_0, \dots, s_k$  such that for every  $i$  there exists a transition from  $s_i$  to  $s_{i+1}$ , then we say that  $s_k$  is *reachable from  $s_0$  in  $k$  steps*. We say that a state  $s''$  is reachable from a state  $s$  in  $\leq k$  steps if for some  $m \leq k$  the state  $s''$  is reachable from  $s$  in  $m$  steps. Finally,  $s''$  is said to be *reachable from  $s$*  if it is reachable from  $s$  in some number of steps.  $\square$

The following properties are not hard to verify.

- (1) The reachability relation is the reflexive and transitive closure of the transition relation of  $K$ .
- (2)  $s''$  is reachable from  $s$  if and only if  $s''$  is a node in the computation tree for  $K$  starting at  $s$ .
- (3)  $s''$  is reachable from  $s$  in  $k$  steps if and only if there is a path of length  $k$  from  $s$  to  $s''$  in the transition graph of  $s$ .

Given a symbolic representation of the Kripke structure  $K$ , we will now show how to express reachability in  $k$  or  $\leq k$  steps. To this end, we will first make an assumption

about a symbolic representation of  $K$ . Let  $V$  be a set of variables of  $K$ . Let  $init(V)$  be a propositional formula which represents the set  $I$  of initial states of  $K$ , i.e., for every state  $s$

$$s \models init(V) \text{ if and only if } s \in I.$$

Similarly, let  $trans(V, V')$  be a propositional formula which represents the transition relation  $R$  of  $K$ , i.e.,

$$(s, s') \models trans(V, V') \text{ if and only if } (s, s') \in R.$$

Let us define formulas  $reach_{\leq n}(V)$  which represent reachability in  $\leq k$  steps from an initial state. Though  $init$  and  $trans$  are propositional formulas, the formulas  $reach_{\leq k}$  will be quantified boolean formulas. Let

$$\begin{aligned} reach_{\leq 0}(V) &\stackrel{\text{def}}{=} init(V); \\ reach_{\leq k+1}(V) &\stackrel{\text{def}}{=} reach_{\leq k}(V) \vee \exists X (reach_{\leq k}(X) \wedge trans(X, V)). \end{aligned}$$

We claim the following result.

**LEMMA 15.5** *The formula  $reach_{\leq k}(V)$  represents reachability in  $\leq k$  steps in the following sense. For every state  $s$ , we have*

$$s \models reach_{\leq k}(V)$$

*if and only if  $s$  is reachable from an initial state in  $\leq k$  steps.* □

The proof is left as Exercise 15.1.

The crucial for us property of  $reach_{\leq k}$  is the following.

**LEMMA 15.6**  $\{s \mid s \models reach_{\leq k}(V)\} \subseteq \{s \mid s \models reach_{\leq k+1}(V)\}$ . □

Denote by  $S_k$  the set of states reachable from an initial state in  $\leq k$  steps. By Lemma 15.6 we have  $S_0 \subseteq S_1 \subseteq S_2 \dots$ . Since the Kripke structure  $K$  is finite, its set of states is finite, so the sequence  $S_0, S_1, \dots$  cannot grow infinitely. Therefore, there should be a number  $n$  such that  $S_n = S_{n+1}$ . We claim the following.

**THEOREM 15.7** *The following conditions are equivalent.*

- (1)  $S_n = S_{n+1}$ .
- (2)  $S_n$  is the set of states reachable from initial states.
- (3) The formula  $reach_{\leq n}(V)$  is equivalent to  $reach_{\leq n+1}(V)$ .

PROOF. We prove that (1) is equivalent to (3). Indeed, two formulas are equivalent if and only if they have the same models. But by Lemma 15.5 the models of  $reach_{\leq n}(V)$  are exactly the states reachable from initial states in  $\leq n$  steps, i.e. exactly the members of  $S_n$ .

We prove that (1) implies (2). Suppose  $S_n = S_{n+1}$  and consider  $S_{n+2}$ .  $S_{n+2}$  is the set of states reachable in one step from  $S_{n+1}$ , that is the same as the set of states reachable in one step from  $S_n$ , that is  $S_{n+1}$ . So we have  $S_{n+2} = S_{n+1}$ . In the same way we can prove that  $S_m = S_n$  for all  $m > n$ . Therefore, every state reachable from an initial state in  $m > n$  steps is already reachable in  $\leq n$  steps, which implies that  $S_n$  is the set of all states reachable from initial states.

We prove that (2) implies (1). Assume that  $S_n$  is the set of states reachable from initial states. Take any state  $s \in S_{n+1}$ . Then  $s$  is reachable from an initial state, so by the assumption  $s \in S_n$ . Since  $s$  was arbitrary, we have  $S_{n+1} \subseteq S_n$ . We already established  $S_n \subseteq S_{n+1}$ , so  $S_n = S_{n+1}$ .  $\square$

This theorem gives us an algorithm for computing a symbolic representation of the set of all states reachable from an initial state. We build, one by one, the formulas

$$reach_{\leq 0}(V), reach_{\leq 1}(V), \dots$$

For each  $n$ , as soon as we build  $reach_{\leq n+1}(V)$ , we check if it is equivalent to  $reach_{\leq n}(V)$ .

Note that the formulas  $reach_{\leq n+1}(V)$  can grow to an unmanageable size for large  $n$ . It is not easy to check equivalence of huge quantified boolean formulas. There are at least two ways of getting around this problem:

- (1) use OBDDs to represent quantified boolean formulas.
- (2) ???

What do we need to implement: quantifier elimination and equivalence-checking.

### 15.2.2 Verifying safety properties

However, we are not only interested in a symbolic representation of the set of states reachable from initial states. We are interested in model-checking formulas (15.1) expressing safety properties. Let  $unsafe(V)$  be a propositional formula. We want to check whether there exists a state which is reachable from an initial state and satisfies  $unsafe(V)$ .

Algorithm and theorem of soundness and completeness.

### 15.2.3 Backward reachability

## 15.3 Bounded model checking

## 15.4 CTL model checking

### Exercises

EXERCISE 15.1 Prove Lemma 15.5 on page 226.  $\square$

EXERCISE 15.2 Define an algorithm for computing an OBDD representation of  $\exists X (reach_{\leq k}(X) \wedge trans(X, V))$  from OBDD representations of the formulas  $reach_{\leq k}(V)$  and  $trans(X, V)$ . Note that we cannot directly use the conjunction algorithm followed by a quantifier elimination, since we have an OBDD for  $reach_{\leq k}(V)$  but we have to apply the conjunction algorithm to  $reach_{\leq k}(X)$ . An efficient algorithm should combine the renaming of  $X$  into  $V$  with the conjunction algorithm.  $\square$