Solving Reachability Problems with SAT

April 13, 2023

Outline

Kripke Structure

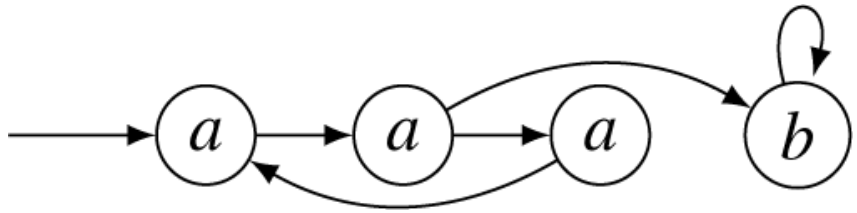
In model checking we typically model our target entity with Kripke structures.

Definition (Kripke Structure)

A Kripke structure M is a five-tuple $M = (S, S_0, R, AP, L)$

1. S is a set of states
2. $S_0 \subseteq S$ is the set of initial states
3. $R \subseteq$ is a transition relation
4. AP is the set of atomic propositions
5. $L \rightarrow 2^{AP}$ is a function that labels each state with the set of those atomic propositions that are true in that state

Kripke Structure (continued)



For example, this Kripke structure has:

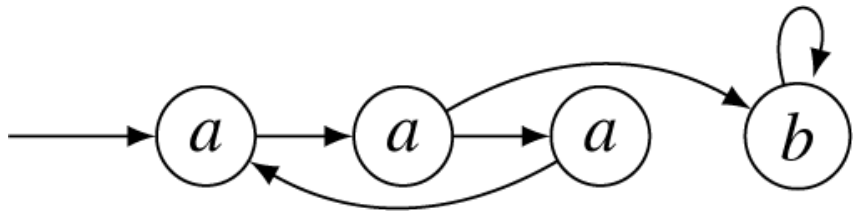
1. $S = \{s_1, s_2, s_3, s_4\}$, counting from left to right
2. $S_0 = s_1$
3. $R = \{(s_1, s_2), (s_2, s_3), (s_2, s_4), (s_3, s_1), (s_4, s_4)\}$
4. $AP = \{a, b\}$
5. $L(x) = a$, for $x \in \{s_1, s_2, s_3\}$, $L(x) = b$ for $x \in \{s_4\}$

Definition of Reachability Problem

- Given a Kripke Structure M, can we reach some state s_i in some $n \in \mathbb{N}$ steps of transition from a initial state $s_0 \in S_0$?

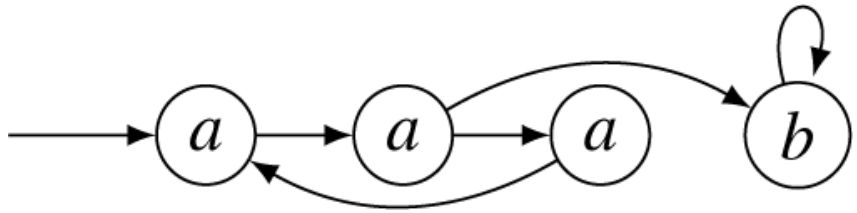
Boolean (Propositional) Encoding of States

Our objective: Encode a reachability problem on Kripke structure to a boolean formula whose satisfiability can be tested by a SAT solver.



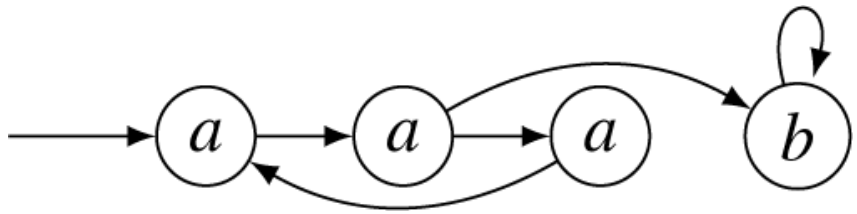
- For each state, assign a boolean word to them
- For example, $s_1 = 00$, $s_2 = 01$ and so on
- For each bit in the word, create a variable v_i
- Now we can represent the states with bits, e.g. $s_1 := \neg v_0 \wedge \neg v_1$ and $s_2 := \neg v_0 \wedge v_1$

Boolean (Propositional) Encoding of States (continued)



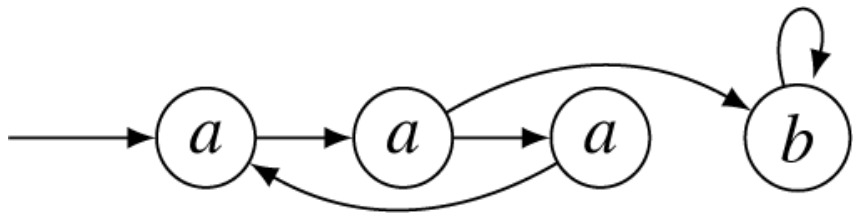
- We denote the next state by appending a apostrophe symbol to the original variable name, for example, v_0' and v_1'
- For initial state S_1, we can encode the initial state set S_0 as a boolean function $S_0(v_0, v_1) = \neg v_0 \wedge v_1$
- Every transition (arrow) can be represented by a formula in the form of $T_{s_1 \to s_2} = s_1 \wedge s_2 = \neg v_0 \wedge \neg v_1 \wedge \neg v_0' \wedge v_1'$
- Transition function can be represented by disjunction of individual transitions, e.g.
  $R(v_0, v_1, v_0', v_1') = T_{s_1 \to s_2} \vee T_{s_2 \to s_2} \vee T_{s_2 \to s_4} \vee T_{s_3 \to s_1} \vee T_{s_4 \to s_4}$

Modeling a Procedure with Multiple Steps of Transitions



- With the previously mentioned method, we can only model a one-step transition system
- But we want to know that whether a state can be reached in n steps of transition
- How do we model more steps?

Modeling a Procedure with Multiple Steps of Transitions



- With the previously mentioned method, we can only model a one-step transition system
- But we want to know that whether a state can be reached in n steps of transition
- How do we model more steps?
- Answer: We can make multiple copies of the transition function!

Bounded Model Checking

- For a two step transition system, we only have one copy of the transition function
- Recall that $S_0 \wedge R(v_0, v_1, v_0', v_1')$ can represent the states that can be reached in one step of transition
- To make our formula shorter, we rewrite $R(v_0, v_1, v_0', v_1')$ as $R(V, V')$
- $R(V, V')$ is a word-level modeling of the transition function and denotes the transition from first step (V) to the second step (V')

Bounded Model Checking (continued)

- Now we can make copies of the transition system by feeding different V values to the function R
- e.g. for three steps we can do
  $S_0 \wedge R(V, V') \wedge R(V', V'') \wedge R(V'', V''')$
- If we want to know whether we can reach some state S_f in 3 steps, we can append s_f to the end of the conjunction:
  $S_0 \wedge R(V, V') \wedge R(V', V'') \wedge R(V'', V''') \wedge S_f$
- Generalize to n steps: $S_0 \wedge R(V_0, V_1) \wedge \cdots \wedge R(V_{n-1}, V_n) \wedge S_f$
- If SAT, then the model returned by SAT solver is the complete trace of transition
- If UNSAT, then S_f can not be reached in n steps

Modeling a Maze Solving Game

We want to find a path from upper-left corner to the bottom-right corner of the maze.

```
0001101010
1000000000
1001110001
0000001100
1111101100
0010000010
0001010110
0110001110
0100110010
```

Encoding Each State of a Maze

- For a $n \times n$ maze, obviously we have n^2 states
- An easiest way to encode states is to assign each one of them a natural number
- e.g. for a state on row i and column j, assign value $i \cdot n + j$ to it
- We define the numbering function $N(i, j) = i \cdot n + j$

Encoding the Transition Function

- For each maze cell, there are at most four states that can be reached (left, right, up, down)
- We can describe two connected state m, n by adding a clause $N(i_m, j_m) \wedge N(i_n, j_n)$ where i_m, j_m and i_n, j_n are the coordinate of m and n accordingly
- The transition function will be:
  $R(i, j, i', j') = \bigwedge_{m,n \in \text{All Adjacent States}} N(i_m, j_m) \wedge N(i'_n, j'_n)$