My summaries about automaton used in model checking

Zhang Yedi

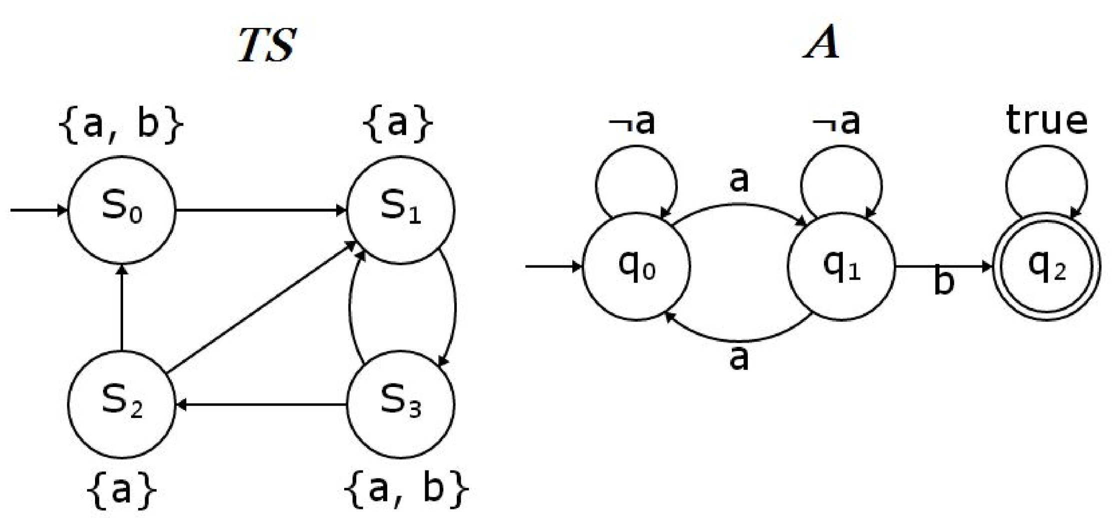
1. First Section

1. One scenario where ***'automaton is closed under complementation'*** is necessary:

If an automaton $\mathcal{A}$ is closed under complementation, and an AMA $\mathcal{M}^{\langle\langle A\rangle\rangle\varphi}$ can recognize the set of configurations of $\mathcal{P}$ satisfying $\langle\langle A\rangle\rangle\varphi$ in a bottom-up approach. We assume that $\mathcal{M}^{\neg\langle\langle A\rangle\rangle\varphi}$ has also been computed to recognize $\mathcal{C}_{\mathcal{P}}$ (T Chen et al. 2016).

2. Automaton's production (reference) :

The question is like:



And the answer should be like:

In Section 4.2.2 of the book "Principles of Model Checking", there is a definition (Definition 4.16; Page 165) of "Product of Transition System and NFA". You are right about the states (i.e., $S \times Q$) of the product but make mistakes about its transition relation. Below I focus on the transition relation.

Definition 4.16 Product of Transition System $TS = (S, Act, \rightarrow, I, AP, L)$ and NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$. The transition relation $\rightarrow'$ of their product is the smallest relation defined by the rule

$$\frac{s \rightarrow^{\alpha} t \ \wedge \ q \rightarrow^{L(t)} p}{(s, q) \rightarrow'^{\alpha} (t, p)}$$

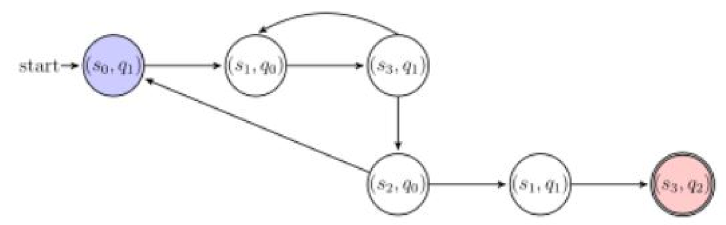
Intuitively, the transition system TS generates atomic propositions and feeds them into the automaton $\mathcal{A}$, driving the automata running. This semantics can be used to verify if the TS satisfies some property expressed by an automaton.

Additionally, the start states of the product is

$$I' = \{(s_0, q) \mid s_0 \in I \wedge \exists q_0 \in Q_0 . q_0 \rightarrow^{L(s_0)} q\}.$$

That is, q_0 in automaton has moved one step forward, driven by the atomic propositions in s_0.

Based on the definition above, I calculate the product as follows (please check it):



3. Why PDA $\rightarrow$ Multi-Automaton is necessary?

PDA has an infinite sets of states (because it has an infinite stack space, even though its control states and alphabet are all finite). So for a PDA, a state should be a configuration like $\langle p_i, \omega\rangle$. In the case of finite states systems, the sets X_i are all finite, and the sequence $\{X_i\}_{i\geq 0}$ is guaranteed to reach a fix point, which immediately provides an algorithm to compute $pre^*(S)$. Unfortunately, these properties no longer hold for any non-trivial class of infinite states systems.

Introduction to Alternating Multi-Automaton: We can regard that an AMA has one initial state for each control location in Pushdown automaton, so for AMA, it doesn't just has only one initial state, instead, it has a initial states set. And the automaton recognizes the configuration $\langle p, \omega\rangle$ if it accepts the word ω from the initial state corresponding to p. (AMA is just a tool to represent a set of configurations, and not to confuse its "behaviour"with that of the pushdown system.)[BEM97].

Multi-Automaton: Given a MA $\mathcal{A}$, it returns a ***regular*** set of configurations C of PDA, what's more, we can get another MA $\mathcal{A}_{pre^*}$ recognizing $pre^*(C)$ (a closure set of $pre(C)$).

```
def main():
    return Null
```

Literatur

[BEM97] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings*, pages 135–150, 1997.