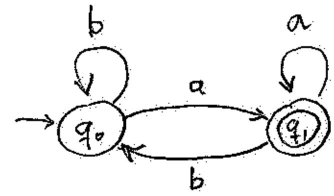


DFA LANGUAGE INCLUSION

A derministic finite automaton is a tuple

$$A = (Q, \Sigma, \delta, q_0, F)$$

states, alphabet, transition function, initial state, final/accepting states

The automaton accepts the input string $x \in \Sigma^*$ if x drives the automaton from q_0 to some accepting state i.e. $\hat{\delta}(q_0, x) \in F$. The language $L(A) \subseteq \Sigma^*$ of A is the set of strings that A accepts.

Let A_M be an automaton modelling a system, and let A_S be an automaton specifying the system.

Correctness can then be stated as language incusion $L(A_M) \subseteq L(A_S)$. How can this be (model) checked?

$$\begin{aligned} L(A_M) \subseteq L(A_S) &\Leftrightarrow L(A_M) \cap \overline{L(A_S)} = \emptyset \\ &\Leftrightarrow L(A_M) \cap L(\overline{A}_S) = \emptyset \\ &\Leftrightarrow L(A_M \times \overline{A}_S) = \emptyset \end{aligned}$$

Procedure:

1. Construct complement automaton $\overline{A}_S$
2. Construct product automaton $A_M \times \overline{A}_S$
3. Check reachability of an accepting state

The same idea can be used for LTL model checking, that is, for deciding $\mathcal{M}, s \models \phi$. But...

In our case we have infinite valuation sequences $\rho = v_0 v_1 v_2 \ldots$ where $v \subseteq Atoms$ (i.e. alphabet is 2^{Atoms}).

We already discussed satisfaction $\rho \models \phi$.

If we can construct an automaton $B_{\mathcal{M},s}$ which accepts exactly the valuation sequences induced by the paths of $\mathcal{M}$ starting at s, and if we can construct an automaton B_ϕ which accepts exactly the valuation sequences satisfying ϕ, then $\mathcal{M}, s \models \phi$ if and only if $L(B_{\mathcal{M},s}) \subseteq L(B_\phi)$.

First we need automata over infinite words.

BÜCHI AUTOMATA

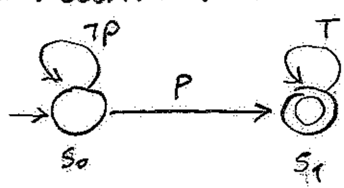
A Büchi automaton is a tuple $B = (Q, \Sigma, \delta, q_0, F)$ defined as before, but with a modified acceptance condition: the automaton accepts the infinite input string $x \in \Sigma^\omega$ if x drives the automaton to visit some accepting state infinite often, i.e. $inf(x) \cap F \neq \emptyset$.

The alphabet we consider here is $\Sigma = 2^{Atoms}$.

A nice generalization is to use propositional formulas over Atoms as symbols in the alphabet.

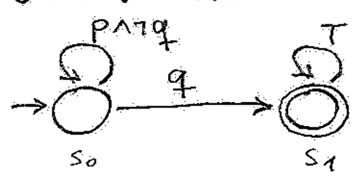
Büchi automata have the same expressive power as LTL formulas.

Here is a Büchi automaton for the formula Fp:

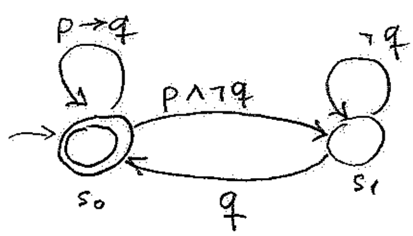


SPIN generates a corresponding automaton with spin -f "<> p".

And here is a Büchi automaton for $p \, U \, q$:



Finally a Büchi automaton for $G(p \to Fq)$:



Büchi automata can be used as a specification language on its own.

Notice that $L(B) \neq \emptyset$ if and only if there is a reachable loop involving an accepting state.

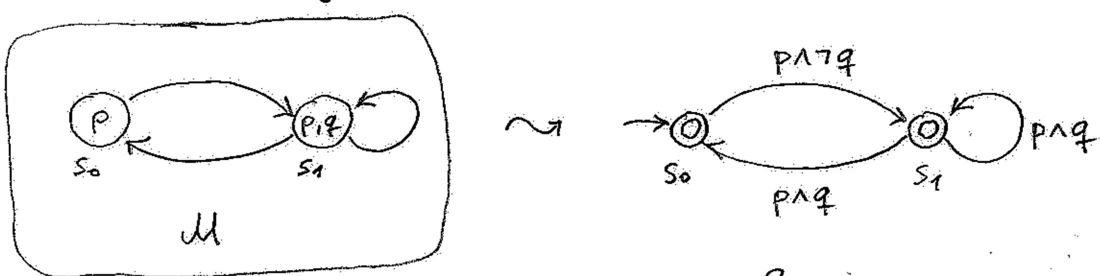
MODEL CHECKING $\mathcal{M}, s \models \phi$

$$\mathcal{M}, s \models \phi \quad \Leftrightarrow \quad L(B_{\mathcal{M},s}) \subseteq L(B_\phi)$$

$$\Leftrightarrow \quad L(B_{\mathcal{M},s} \times B_{\neg\phi}) = \emptyset$$

1) From $\mathcal{M}$ and s to $B_{\mathcal{M},s}$

The conversion just moves the valuations from the states to the outgoing edges, and makes all states accepting. The initial state is s.

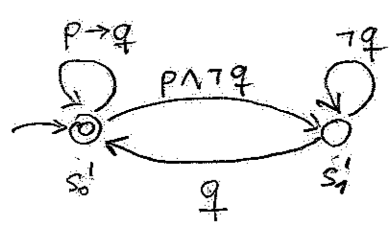


$B_{\mathcal{M},s_0}$

2) From ϕ to $B_{\neg\phi}$

This is a complicated but standard "tableau" construction.

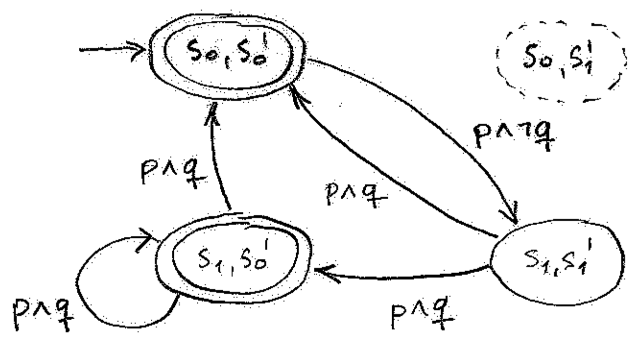
For $\boxed{\phi = F(p \wedge G\neg q)}$ we have $\neg\phi \equiv G(p \to Fq)$ for which there is a Büchi automaton:



$B_{\neg\phi}$

3) Constructing the product automaton $B_{\mathcal{M},s} \times B_{\neg\phi}$

This is a simple, standard construction: the states of the product automaton are pairs of states (s_1, s_2) such that s_1 is a state of $B_{\mathcal{M},s}$ and s_2 is a state of $B_{\neg\phi}$.
A pair (s_1, s_2) is an initial state if both s_1 and s_2 are, and similarly for accepting states.
There is a transition $(s_1, s_2) \xrightarrow{\phi_1 \wedge \phi_2} (s_1', s_2')$ exactly when $s_1 \xrightarrow{\phi_1} s_1'$ in $B_{\mathcal{M},s}$ and $s_2 \xrightarrow{\phi_2} s_2'$ in $B_{\neg\phi}$.
Note that edges labelled with unsatisfiable formulas can be safely removed, as well as states not reachable from the initial state.



"monitored execution"

$B_{\mathcal{M},s_0} \times B_{\neg\phi}$

4) Checking non-emptyness of $L(B_{\mathcal{M},s} \times B_{\neg\phi})$

a. Compute the SCCs reachable from the initial state
one component: $\{(s_0,s_0'), (s_1,s_1'), (s_1,s_0')\}$

b. Intersect their union with the accepting states
$\{(s_0,s_0'), (s_1,s_0')\}$

c. If empty, then $\mathcal{M}, s \models \phi$ holds
Else, compute counter-example by taking an accepting path and projecting onto $\mathcal{M}$: $s_0 \, s_1$