Examples & board definitions for part 4 of the "Model Checking and Games" lecture series

Ruediger Ehlers

September 6, 2019

1 Büchi automaton examples

1.1 Example 1

So let us consider an ultimately period word instead. Let $w = aba(b)^\omega$. We obtain the run $\pi = q_0q_0q_1q_0(q_1q_0)^\omega$. As q_1 is an accepting state and it occurs infinitely often along the run, π is an accepting run. Hence, $\mathcal{A}$ accepts the word w.

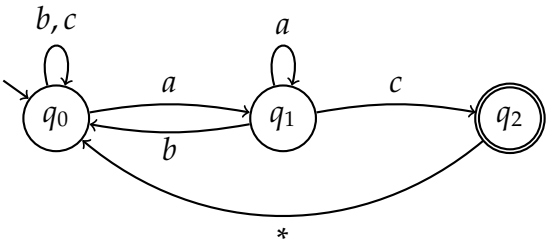
1.2 Example 2

Let us consider the word $w = abbaaabbaabbb\ldots$. It induces several words such as $\pi = q_0q_0q_0q_0q_0q_0q_0q_0q_0q_0q_0q_0q_0\ldots$, which is not accepting. Then there are finite runs such as $\pi^2 = q_0q_1q_1$ and $\pi^3 = q_0q_0q_0q_0q_0q_0q_1q_1$, which are finite and also not accepting.

The only way for w to be accepted is if a word ends with b^ω, as only in this way, state q_1 can be visited infinitely often.

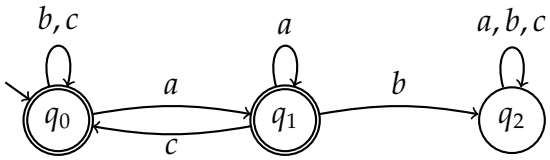
2 Let's build some Büchi automata!

2.1 Example 1



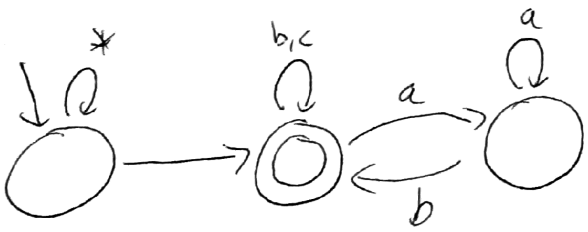
This is "similar" to the LTL property $\mathsf{GF}(a \wedge \mathsf{X}c)$

2.2 Example 2



This is "similar" to the LTL property $\mathsf{G}(a \rightarrow \neg \mathsf{X}b)$

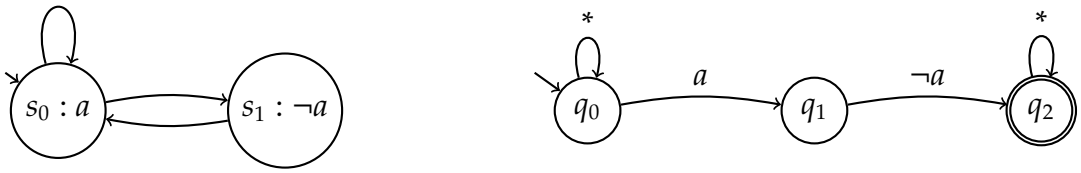
2.3 Example 3



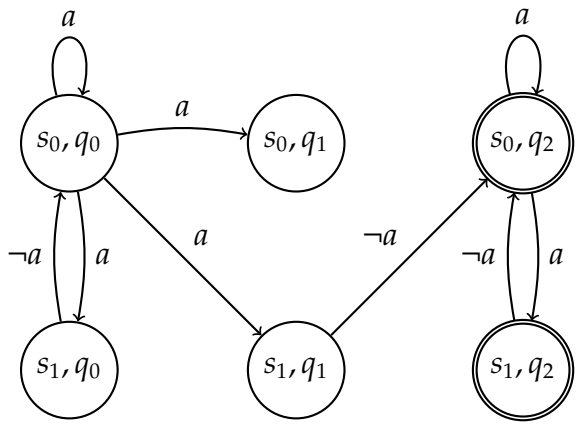
This is "similar" to the LTL property $\mathsf{F}(a \rightarrow (a \,\mathcal{U}\, b))$

3 Product automata – Example

Example Kripke structure & Example Büchi automaton accepting all words on which a is followed by $\neg a$ (over the alphabet $2^{\{a,b\}}$):

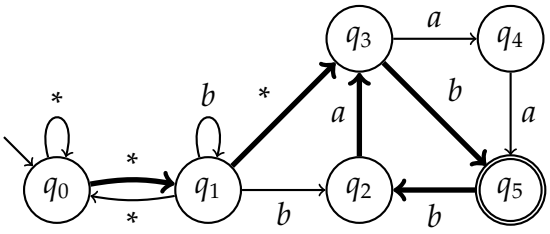


Product:



4 Lassos

Example automaton with an example accepting lasso:



5 The Double-DFS algorithm

5.1 Depth-first search

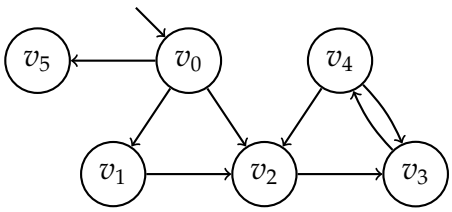
We start by recalling at how depth-first search works. The basic algorithm traverses a graph and backtracking whenever it visits a node that it has already seen. Given a graph $G = (V, E)$ and an initial vertex $v_0 \in V$, the basic operation of the algorithm is as follows:

```
visited = set([])
stack = []

def dfs(node):
    stack.push(node)
    if node in visited:
        stack.pop()
        return
    visited.add(node)
    for all (v,v') in E with v==node:
        dfs(v')
    stack.pop()
    return

dfs(v0)
```

Let us look at the operation of the DFS algorithm on an example graph:

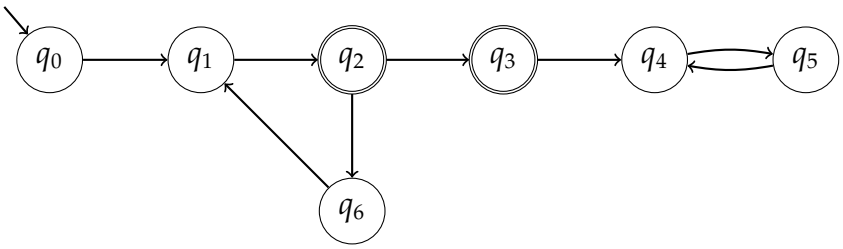


The algorithm proceeds on the following `nodes` with the following stack and visited contents:

node	stack	visited
v_0	$[v_0]$	$[v_0]$
v_1	$[v_0, v_1]$	$[v_0, v_1]$
v_2	$[v_0, v_1, v_2]$	$[v_0, v_1, v_2]$
v_3	$[v_0, v_1, v_2, v_3]$	$[v_0, v_1, v_2, v_3]$
v_4	$[v_0, v_1, v_2, v_3, v_4]$	$[v_0, v_1, v_2, v_3, v_4]$
v_3	$[v_0, v_1, v_2, v_3, v_4, v_3]$	no change
v_4	$[v_0, v_1, v_2, v_3, v_4]$	no change
v_2	$[v_0, v_1, v_2, v_3, v_4, v_2]$	no change
v_4	$[v_0, v_1, v_2, v_3, v_4]$	no change
v_3	$[v_0, v_1, v_2, v_3]$	no change
v_2	$[v_0, v_1, v_2]$	no change
v_1	$[v_0, v_1]$	no change
v_0	$[v_0]$	no change
v_5	$[v_0, v_5]$	$[v_0, v_1, v_2, v_3, v_4, v_5]$
v_0	$[v_0]$	no change
-	-	-

Note that since our set of edges is unordered, there are multiple possible executions of the algorithm.

5.2 Example for the double-DFS algorithm



(This is a modification of an example by Ofer Strichman)

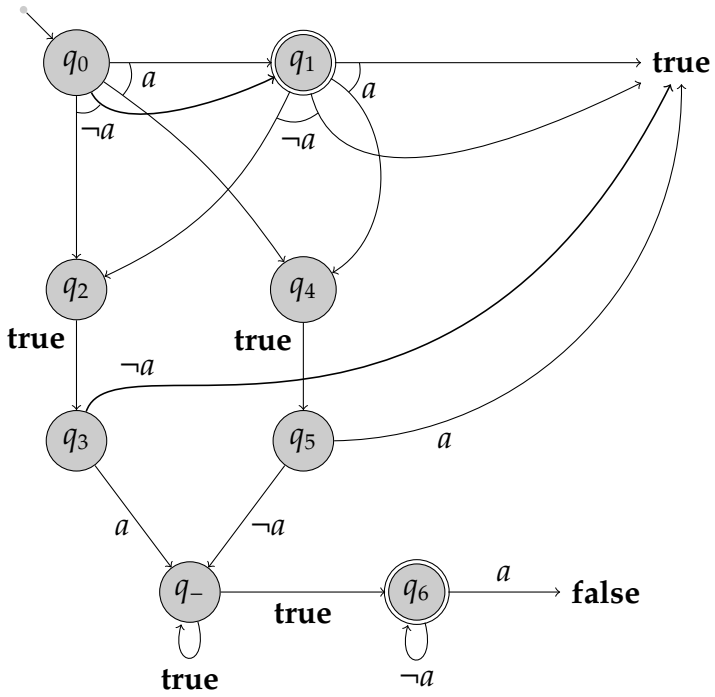
6 Alternating automaton & Run trees

Assume an automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ with:

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_-\}$
- $\Sigma = 2^{\{a\}}$
- $\delta(q_0, \neg a) = q_1 \wedge q_2$
- $\delta(q_0, a) = q_1 \wedge q_4$
- $\delta(q_1, \neg a) = q_2 \wedge \mathbf{true}$
- $\delta(q_1, a) = q_4 \wedge \mathbf{true}$
- $\delta(q_2, \neg a) = q_3$
- $\delta(q_2, a) = q_3$

- $\delta(q_3, \neg a) = \mathbf{true}$
- $\delta(q_3, a) = q_-$
- $\delta(q_4, \neg a) = q_5$
- $\delta(q_4, a) = q_5$
- $\delta(q_5, \neg a) = q_-$
- $\delta(q_5, a) = \mathbf{true}$
- $\delta(q_-, \neg a) = q_- \vee q_6$
- $\delta(q_-, a) = q_- \vee q_6$
- $\delta(q_6, \neg a) = q_6$
- $\delta(q_6, a) = \mathbf{false}$
- $Q_0 = \{q_0\}$
- $F = \{q_1, q_6\}$

We can represent this notation in graphical form as follows:



The automaton induces the following run trees (for some given words):

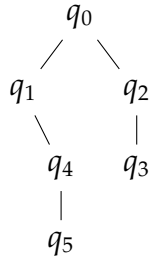


Figure 1: A run tree for the word $w = \{a \mapsto 0\}\{a \mapsto 1\}\{a \mapsto 0\}\{a \mapsto 1\}(\{a \mapsto 0\})^{\omega}$.

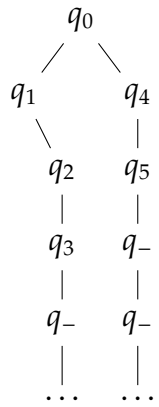


Figure 2: A run tree for the word $w = \{a \mapsto 1\}\{a \mapsto 0\}\{a \mapsto 0\}\{a \mapsto 1\}(\{a \mapsto 1\})^\omega$.

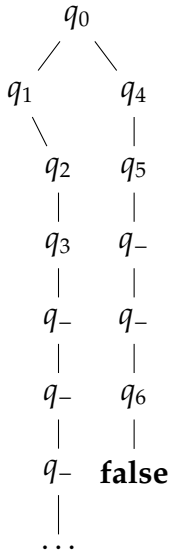


Figure 3: A **syntactically incorrect** run tree for the word $w = \{a \mapsto 1\}\{a \mapsto 0\}\{a \mapsto 0\}\{a \mapsto 1\}(\{a \mapsto 1\})^\omega$.

LTL to Alternating Automata

Consider the LTL formula $\psi = p\,\mathcal{U}\,((\mathsf{G}\,q) \wedge (\mathsf{F}\,\neg r))$ over $\mathsf{AP} = \{p, q, r\}$.

We obtain the following alternating Büchi automaton:

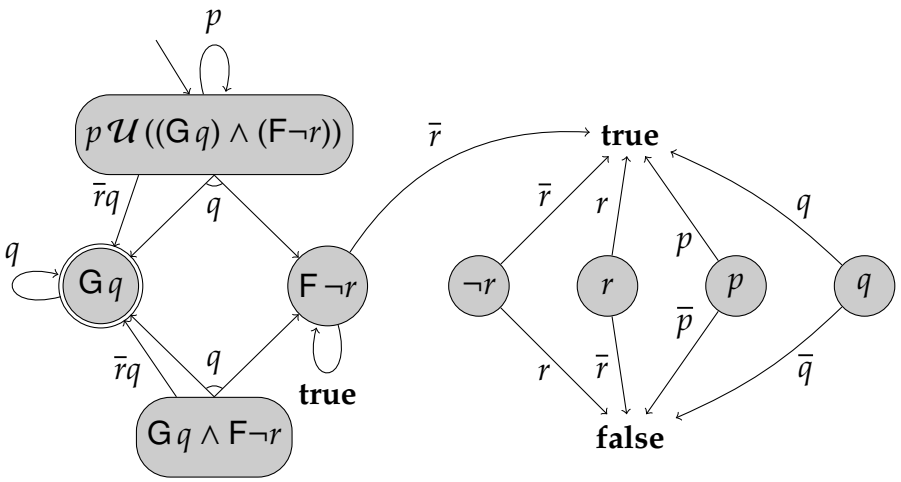


Figure 4: The alternating automaton for this specification. Note that the Boolean functions of the transition function have been simplified prior to drawing them. In particular, conjunctions with **true** and disjunctions with **false** have been collapsed.

7 Alternating automata to non-deterministic automata

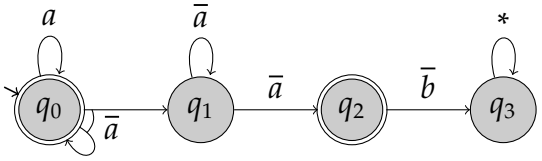


Figure 5: Example for an alternating Büchi automaton that is equivalent to the LTL property GF($a \vee$ XXb). Conjunctive branching is depicted by an arc. In order to make the run trees in Figure 6 and Figure 7 interesting, the automaton is a bit more complicated than needed.

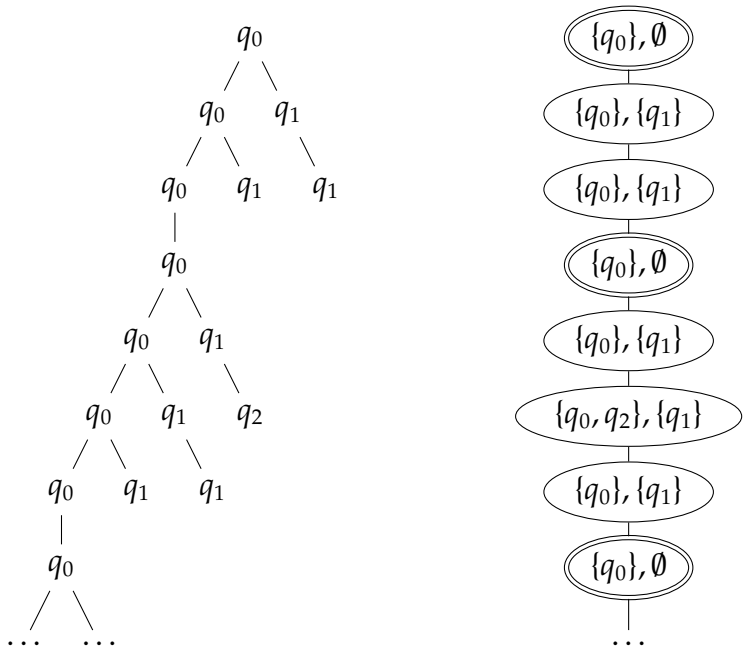


Figure 6: A run tree of the alternating automaton from Figure 5 for $w = \emptyset\emptyset\{a\}\emptyset\emptyset\{b\}\{a\}\ldots$ as the input word, and the corresponding run in the non-deterministic Büchi automaton build from the automaton in Figure 5 using the Miyano-Hayashi construction

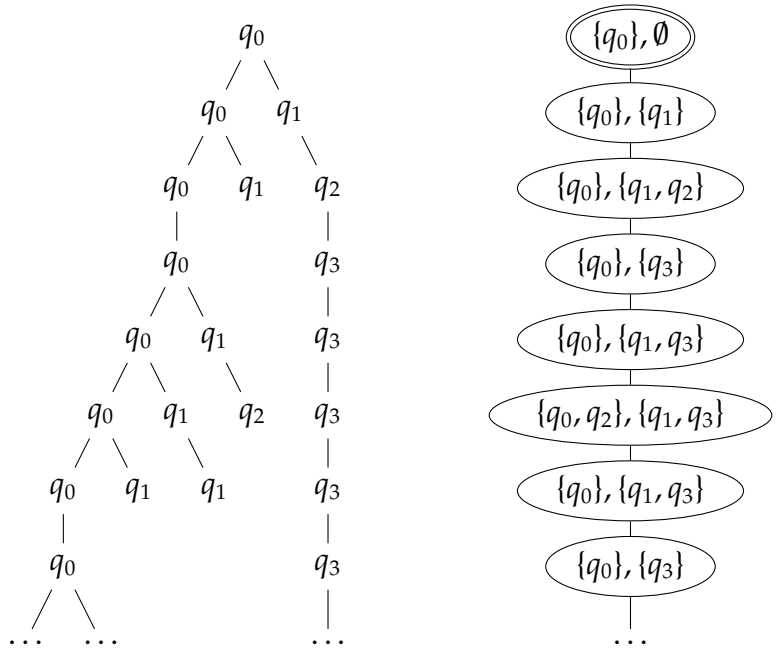


Figure 7: A (non-accepting) run tree of the alternating automaton from Figure 5 for $w = \emptyset\emptyset\{a\}\emptyset\emptyset\{b\}\{a\}\ldots$ as the input word, and the corresponding run in the non-deterministic Büchi automaton build from the automaton in Figure 5 using the Miyano-Hayashi construction