

Master's Thesis Notes

Daniel Weibel, 16.04.2014

1 Theoretical Part

1.1 Introduction

| | | |
|---|--|--|
| <p>Subset construction on Büchi automaton</p> <p>Hand-drawn diagram showing the subset construction of a Büchi automaton. It starts with a single state 0. Transitions on 'a' and 'b' lead to new states. State 13 is marked as an accepting state (double circle).</p> | <p>Run trees of word ababab... on nondet. and det. automaton</p> <p>Hand-drawn diagrams showing run trees for the word "ababab..." on a nondeterministic and a deterministic automaton. The nondet. tree branches out, while the det. tree is a single path.</p> | <p>Impact of subset construction on run tree</p> <p>Hand-drawn diagrams illustrating the impact of subset construction on a run tree. It shows how multiple paths in the original run tree are merged into a single path in the subset-constructed tree.</p> |
|---|--|--|

The subset construction merges several nondeterministic runs to a single run. All the information about specific nondeterministic runs get lost: their state sequences, if and where they die, the number of nondeterministic runs until a given point, etc.

For automata on finite words this is not a problem, because all we are interested in are the states the automaton is in after reading all the symbols of the input word.

With ω -automata, however, we are interested in the behaviour of *specific runs*. In the case of Büchi automata we need to know whether a specific run visited an accepting state infinitely often.

However, after the subset construction, this information is lost in the determinized automaton. All that we can say after the subset construction is that an *accepting state* q has been visited *infinitely often*. This can have three reasons:

1. A finite number of runs visited q *infinitely often*
2. An infinite number of runs visited q *infinitely often*
3. An infinite number of runs visited q *finitely often*

Cases 1 and 2 satisfy the Büchi acceptance condition, because there must be at least one single run that visited q infinitely often. In Case 3 however, there is no specific run that visited the q infinitely often, and thus, the Büchi acceptance condition is not met. This is the case in the above example.

This means, we cannot distinguish the invalid Case 3 from the valid Cases 1 and 2 anymore. Hence, a "subset-constructed" automaton additionally recognises the words that lead to instances of Case 3, which are not accepted by the nondeterministic automaton.

What if in the subset construction accepting and non-accepting states are never mixed?

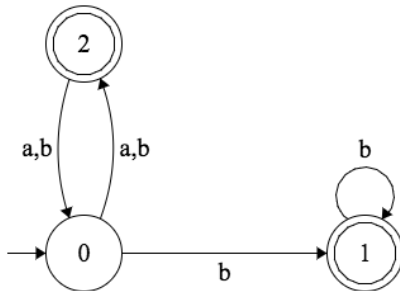
Lemma (informal): if accepting and non-accepting states are not mixed in subsets, then the above problem with the subset construction does not occur.

Proof (informal): subset state $q = \{q_1, q_2, \dots, q_n\}$, with q_1, \dots, q_n accepting, visited infinitely often \rightarrow there exists a specific run that visited at least one of q_1, \dots, q_n infinitely often.

There exists an infinite path (run) in the run tree (König's Lemma). This run necessarily visits subset state q infinitely often (one component of q at a time). Since the number of component states of q (q_1, \dots, q_n) is finite, the run must visit at least one of them infinitely often.

1.2 Questions

1.2.1 Mixing of accepting and non-accepting states



Example: mix states 0 and 2 into state $q = \{0,2\}$. In the end it must hold that if q is visited infinitely often (implying that both 0 and 2 are visited infinitely often), then there is a single run that visited 2 infinitely often. Which approach should we take for formulating the condition(s) that must hold in order to mix states?

1.2.2 Complementation

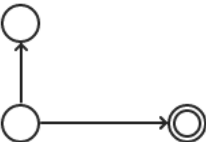
Standard subset construction on Büchi automaton A:

| | Nondeterministic | | Deterministic |
|---------------------|------------------|-------------|---------------|
| Language | $L(A)$ | \subseteq | $L(D)$ |
| | $ $ | | $ $ |
| Complement language | $L(A')$ | \supseteq | $L(D')$ |

From the information between A and D, how do we finally arrive at A' ?

2 Practical Part

- Existing Ruby program
- Web-based application
- More functionality
 - For NFA, drawing run tree of a given input word

| Non-deterministic | Deterministic |
|--|---|
| <div style="border: 1px solid black; padding: 5px; min-height: 150px;"><pre>states: - "0" - "1" - "2" alphabet: - a - b start_state: "0" accept_states: - "1" - "2" transitions: "0": a: - "2" b: - "1" - "2" "1": b: - "1" "2": </pre></div> <div style="text-align: center; margin: 10px 0;"><div style="border: 1px solid black; border-radius: 10px; padding: 2px 10px;">Draw</div><div style="width: 10px; height: 10px; margin: 0 auto;"></div></div> <div style="border: 1px solid black; padding: 10px; min-height: 100px; text-align: center;"></div> <div style="margin-top: 10px;"><div style="display: flex; align-items: center;"><div style="flex: 1;">Input Testing <div style="border: 1px solid black; padding: 2px; width: 100%;">aabbababababaa</div></div><div style="margin-left: 10px;"><div style="border: 1px solid black; border-radius: 10px; padding: 2px 10px;">Run</div></div></div></div> | <div style="border: 1px solid black; padding: 5px; min-height: 150px;"></div> <div style="text-align: center; margin: 10px 0;"><div style="display: flex; justify-content: space-around; width: 100%;"><div style="border: 1px solid black; border-radius: 10px; padding: 2px 10px;">Save</div><div style="border: 1px solid black; border-radius: 10px; padding: 2px 10px;">Draw</div></div><div style="width: 10px; height: 10px; margin: 0 auto;"></div></div> <div style="border: 1px solid black; padding: 10px; min-height: 100px;"></div> <div style="margin-top: 10px;"><div style="display: flex; align-items: center;"><div style="flex: 1;">Input Testing <div style="border: 1px solid black; padding: 2px; width: 100%;"></div></div><div style="margin-left: 10px;"><div style="border: 1px solid black; border-radius: 10px; padding: 2px 10px;">Run</div></div></div></div> |

☐ Classic subset construction
☒ Modified subset construction
☐ Other construction

Construct