Performance Investigation of a Subset-Tuple Büchi Complementation Construction

Daniel Weibel

November 2, 2014

Contents

Chapter 1

The Fribourg Construction

The Fribourg construction draws from several ideas: the subset construction, run analysis based on reduced split trees, and Kurshan's construction [8] for complementing DBW. Following the classification we used in Section **??**, it is a slice-based construction. Some of its formalisations are similar to the slice-based construction by Vardi and Wilke [25], however, the Fribourg construction has been developed independently. Furthermore, as we will see in Chapter **??**, the empirical performance of Vardi and Wilke's construction and the Fribourg construction differ considerably, in favour of the latter.

Basically, the Fribourg construction proceeds in two stages. First it constructs the so-called upper part of the complement automaton, and then adds to it its so-called lower part. These terms stem from the fact that it is often convenient to draw the lower part below the previously drawn upper part. The partitioning in these two parts is inspired by Kurshan's complementation construction for DBW. The upper part of the Fribourg construction contains no accepting states and is intended to model the finite "start phase" of a run. At every state of the upper part, a run has the non-deterministic choice to either stay in the upper part or to move to the lower part. Once in the lower part, a run must stay there forever (or until it ends if it is discontinued). That is, the lower part models the infinite "after-start phase" of a run. The lower part now includes accepting states in a sophisticated way so that at least one run on word w will be accepted if and only if all the runs of the input NBW on w are rejected.

As it may be apparent from this short summary, the construction of the lower part is much more involved than the construction of the upper part.

1.1 First Stage: Constructing the Upper Part

The first stage of the subset-tuple construction takes as input an NBW A and outputs a deterministic automaton B'. This $B'$is the upper part of the final complement automaton B of A. The construction of B'can be seen as a modified subset construction. The difference to the normal subset construction lies in the inner structure of the constructed states. While in the subset

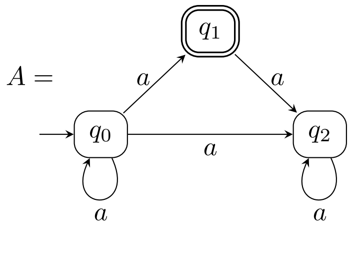


Figure 1.1: Example automaton A

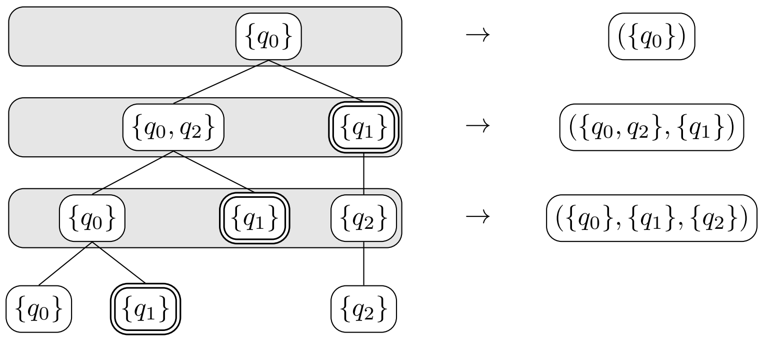


Figure 1.2: Mapping from levels of a reduced split tree, to states of the subset-tuple construction.

construction a state consists of a subset of the states of the input automaton, a B'-state in the subset-tuple construction consists of a *tuple of subsets* of A-states. The subsets in a tuple are pairwise disjoint, that is, every A-state occurs at most once in a B'-state. The A-states occurring in a B'-state are the same that would result from the classic subset construction. As an example, if applying the subset construction to a state $\{q_0\}$ results in the state $\{q_0, q_1, q_2\}$, the subset-tuple construction might yield the state $(\{q_0, q_2\}, \{q_1\})$ instead.

The structure of B'-states is determined by levels of corresponding reduced split trees. Vardi, Kähler, and Wilke refer to these levels as *slices* in their constructions [25, 5]. Hence the name slice-based approach. In the following, we will use the terms levels and slices interchangebly. A slice-based construction can work with either left-to-right or right-to-left reduced split trees. Vardi, Kähler, and Wilke use the left-to-right version in their above cited publications. In this thesis, in contrast, we will use right-to-left reduced split trees, which were also used from the beginning by the authors of the subset-tuple construction.

Figure 1.2 shows how levels of a right-to-left reduced split tree map to states of the subset-tuple construction. In essence, each node of a level is represented as a set in the state, and the order of the nodes determines the order of the sets in the tuple. [INFORMATION ABOUT ACC AND NON-ACC IS NEEDED IN THE LOWER PART BUT IMPLICIT IN THE STATES OF A]. To determine the successor of a state, say $(\{q_0, q_2\}, \{q_1\})$, one can regard this state as level of a reduced split tree, determine the next level and map this new level to a state. In the example of Figure 1.2, the successor of $(\{q_0, q_2\}, \{q_1\})$ is determined in this way to $(\{q_0\}, \{q_1\}, \{q_2\})$.

Apart from this special way of determining successor states, the construction of B'proceeds similarly as the subset construction. One small further difference is that if at the end of determining a successor for every state in B', the automaton is not complete, it must be made complete with an *accepting* sink state. The steps for constructing $B'$from A can be summarised as follows.

- Start with the state $(\{q_0\})$ if q_0 is the initial state of A
- Determine for each state in B'a successor for every input symbol
- It at the end $B'$is not complete, make it complete with an accepting sink state

For the example automaton A in Figure 1.1, we would start with $(\{q_0\})$, determine $(\{q_0, q_2\}, \{q_1\})$ as its a-successor, whose a-successor in turn we determine a $(\{q_0\}, \{q_1\}, \{q_2\})$. The a-successor of $(\{q_0\}, \{q_1\}, \{q_2\})$ is $(\{q_0\}, \{q_1\}, \{q_2\})$ again what results in a loop. Figure 1.3 shows the final upper part $B'$of A.

1.2 Second Stage: Adding the Lower Part

The second stage of the subset-tuple construction adds the lower part to the upper part B'. The two parts together form the final complement automaton B. The lower part is constructed by

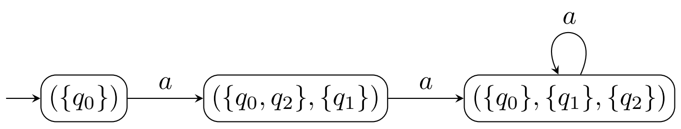


Figure 1.3: Upper part $B'$of example automaton A.

again applying a modified subset construction to the states of the upper part B'. This modified subset construction is an extension of the construction for the upper part. The addition is that each set gets decorated with a colour. These colours later determine which states of the lower part are accepting states.

We divide our discussion of the lower part in two sections. In the following one (1.2.1), we explain the "mechanical" construction of the lower part, the steps that have to be done to arrive at the final complement automaton B. In the next section (1.3) we give the idea and intuition behind the construction and explain why it works.

1.2.1 Construction

As mentioned, every set of the states of the lower part gets a colour. There are three colours and we call them 0, 1, and 2. In the end we have to be able to disinguish the states of the upper part from the states of the lower part. This can be achieved by preliminarily assigning the special colour -1 to every set of the states of the upper part. After that the extended modified subset construction is applied, taking the states of the upper part (except a possible sink state) as the pre-existing states.

At first, the extended modified subset construction determines the successor tuple (without the colours) of an existing state in the same way as the construction of the upper part. We will refer to the state being created as p and to the existing state as p_{pred}. Then, one of the colours 0, 1, or 2 is determined for each set s of p. We denote the colour of s as $c(s)$. The choice of $c(s)$ depends on three factors.

- Whether p_{pred} has a set with colour 2 or not
- The colour of the predecessor set s_{pred} of s
- Whether s is an accepting or non-accepting set

The predecessor set s_{pred} is the set of p_{pred} that in the corresponding reduced split tree is the parent node of the node corresponding to s. Figure 1.4 shows the values of $c(s)$ for all possible situations as two matrices. There is one matrix for the two cases of factor 1 above (p_{pred} has colour 2 or not) and the other two factors are laid out along the rows and columns of either matrix. Note that $c(s_{pred}) = -1$ is only present in the upper matrix, because in this case p_{pred} is a state of the upper part and cannot contain colour 2.

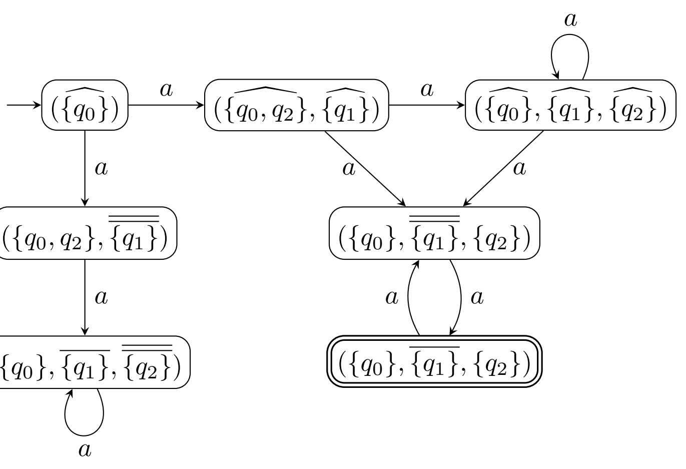
We will use the following notation to denote the colour of s: $\widehat{s}$ if $c(s) = -1$, s if $c(s) = 0$, $\overline{s}$ if $c(s) = 1$, and $\overline{\overline{s}}$ if $c(s) = 2$. Let us look now at a concrete example of this construction. We will add the lower part to the upper part B'in Figure 1.3, and thereby complete the complementation of the example automaton A in Figure 1.1.

First of all, we assign colour -1 all the sets of the states of B'. We might then start processing the state $(\widehat{\{q_0\}})$, let us call it p_{pred}. The resulting successor tuple, without the colours, of p_{pred} is, as in the upper part, $(\{q_0, q_2\}, \{q_1\})$. We now have to determine the colours of the sets $\{q_0, q_2\}$ and $\{q_1\}$. Since p_{pred} does not contain any 2-coloured sets, we need only to consult the upper matrix in Figure 1.4. For $\{q_1\}$, the predecessor set is $\widehat{\{q_1\}}$ with colour -1. Furthermore $\{q_1\}$ is accepting. So, the colour of $\{q_1\}$ is 2, because we end up in the first-row, second-column cell of the upper matrix ($M_1(1, 2)$). The other set, $\{q_0, q_2\}$, in turn is non-accepting, so its colour is 0 ($M_1(1, 1)$). The successor state of $(\widehat{\{q_0\}})$ is thus $(\{q_0, q_2\}, \overline{\overline{\{q_1\}}})$.

p_{pred} has no sets with colour 2	s non-accepting	s accepting
$c(s_{pred}) = -1$	0	2
$c(s_{pred}) = 0$	0	2
$c(s_{pred}) = 1$	2	2

p_{pred} has set(s) with colour 2	s non-accepting	s accepting
$c(s_{pred}) = 0$	0	1
$c(s_{pred}) = 1$	1	1
$c(s_{pred}) = 2$	2	2

Figure 1.4: Colour rules.



$(\widehat{\{q_0\}})$
$(\widehat{\{q_0, q_2\}}, \widehat{\{q_1\}})$
$(\widehat{\{q_0\}}, \widehat{\{q_1\}}, \widehat{\{q_2\}})$
$(\{q_0, q_2\}, \overline{\overline{\overline{\{q_1\}}}})$
$(\{q_0\}, \overline{\overline{\overline{\{q_1\}}}}, \{q_2\})$
$(\{q_0\}, \overline{\{q_1\}}, \overline{\overline{\overline{\{q_2\}}}})$
$(\{q_0\}, \overline{\{q_1\}}, \{q_2\})$
a

Figure 1.5: The final complement automaton B.

We can then continue the construction right with this new state $(\{q_0, q_2\}, \overline{\overline{\overline{\{q_1\}}}})$, and call it p_{pred} in turn. The successing tuple without the colours of p_{pred} is $(\{q_0\}, \{q_1\}, \{q_2\})$. Since p_{pred} contains a set with colour 2, we have to consult the lower matrix of Figure 1.4 to determine the colours of $\{q_0\}$, $\{q_1\}$, and $\{q_2\}$. For $\{q_2\}$, we end up with colour 2 ($M_2(3, 1)$), because its predecessor set, which is $\overline{\overline{\overline{\{q_1\}}}}$, has colour 2. $\{q_1\}$ gets colour 1 as it is accepting and its predecessor set, $\{q_0, q_2\}$, has colour 0 ($M_2(1, 2)$). $\{q_0\}$, which has the same predecessor set, gets colour 0, because it is non-accepting ($M_2(1, 1)$). The successor state of $(\{q_0, q_2\}, \overline{\overline{\overline{\{q_1\}}}})$ is thus $(\{q_0\}, \overline{\{q_1\}}, \overline{\overline{\overline{\{q_2\}}}})$.

The construction continues in this way until every state has been processed. The resulting automaton is shown in Figure 1.5. The last thing that has to be done is to make every state of the lower part that does not contain colour 2 accepting. In our example, this is only one state. The NBW B in Figure 1.5 is the complement of the NBW A in Figure 1.1, such that $L(B) = \overline{L(A)}$. This can be easily verified, since A is empty and B is universal (with regard to the single ω-word a^ω).

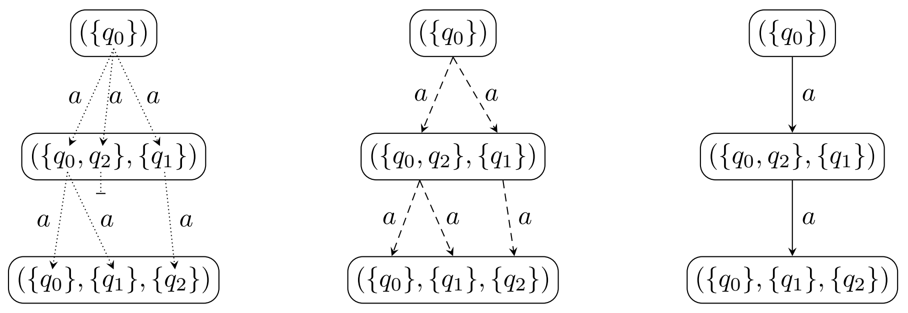


Figure 1.6: Different notions of runs.

1.2.2 Meaning and Function of the Colours

1.3 Intuition for Correctness

The general relation between a non-deterministic automaton A and its complement B is that a word w is accepted by B, if and only if all the runs of A on w are rejecting. Of course for the subset-tuple construction, as we have just described it above, this is also true. A formal proof can be found in [**?**]. In this section, in contrast, we try to give an intuitive way to understand this correctness. One one hand, there is the question, if there is an accepting run of B on w, how can we conclude that all the runs of A on w are rejected?

- If there is an accepting run of B on w, how can we conclude that all the runs of A on w are rejected?
- If all the runs of A on w are rejected, how can we conclude that there must be an accepting run of B on w?

Since this condition is on *all* runs of A, the construction somehow has to keep track of them.

1.4 Optimisations

1.4.1 Removal of Non-Accepting States (R2C)

1.4.2 Merging of Adjacent Sets (M)

1.4.3 Reduction of 2-Coloured Sets (M2)

Bibliography

[1] S. Breuers, C. Löding, J. Olschewski. Improved Ramsey-Based Büchi Complementation. In L. Birkedal, ed., *Foundations of Software Science and Computational Structures*. vol. 7213 of *Lecture Notes in Computer Science*. pp. 150–164. Springer Berlin Heidelberg. 2012.

[2] J. R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Proc. International Congress on Logic, Method, and Philosophy of Science, 1960*. Stanford University Press. 1962.

[3] S. J. Fogarty, O. Kupferman, T. Wilke, et al. Unifying Büchi Complementation Constructions. *Logical Methods in Computer Science*. 9(1). 2013.

[4] J. E. Hopcroft, R. Motwani, J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley. 2nd edition ed.. 2001.

[5] D. Kähler, T. Wilke. Complementation, Disambiguation, and Determinization of Büchi Automata Unified. In L. Aceto, I. Damgård, L. Goldberg, et al, eds., *Automata, Languages and Programming*. vol. 5125 of *Lecture Notes in Computer Science*. pp. 724–735. Springer Berlin Heidelberg. 2008.

[6] J. Klein. Linear Time Logic and Deterministic Omega-Automata. *Master's thesis, Universität Bonn*. 2005.

[7] J. Klein, C. Baier. Experiments with Deterministic ω-Automata for Formulas of Linear Temporal Logic. In J. Farré, I. Litovsky, S. Schmitz, eds., *Implementation and Application of Automata*. vol. 3845 of *Lecture Notes in Computer Science*. pp. 199–212. Springer Berlin Heidelberg. 2006.

[8] R. Kurshan. Complementing Deterministic Büchi Automata in Polynomial Time. *Journal of Computer and System Sciences*. 35(1):pp. 59 – 71. 1987.

[9] C. Löding. Optimal Bounds for Transformations of ω-Automata. In C. Rangan, V. Raman, R. Ramanujam, eds., *Foundations of Software Technology and Theoretical Computer Science*. vol. 1738 of *Lecture Notes in Computer Science*. pp. 97–109. Springer Berlin Heidelberg. 1999.

[10] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*. 9(5):pp. 521 – 530. 1966.

[11] M. Michel. Complementation is more difficult with automata on infinite words. *CNET, Paris*. 15. 1988.

[12] A. Mostowski. Regular expressions for infinite trees and a standard form of automata. In A. Skowron, ed., *Computation Theory*. vol. 208 of *Lecture Notes in Computer Science*. pp. 157–168. Springer Berlin Heidelberg. 1985.

[13] D. E. Muller. Infinite Sequences and Finite Machines. In *Switching Circuit Theory and Logical Design, Proceedings of the Fourth Annual Symposium on*. pp. 3–16. Oct 1963.

[14] D. E. Muller, P. E. Schupp. Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science.* 141(1–2):pp. 69 – 107. 1995.

[15] F. Nießner, U. Nitsche, P. Ochsenschläger. Deterministic Omega-Regular Liveness Properties. In S. Bozapalidis, ed., *Preproceedings of the 3rd International Conference on Developments in Language Theory, DLT'97.* pp. 237–247. Citeseer. 1997.

[16] M. Rabin, D. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development.* 3(2):pp. 114–125. April 1959.

[17] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society.* 141:pp. 1–35. July 1969.

[18] M. Roggenbach. Determinization of Büchi-Automata. In E. Grädel, W. Thomas, T. Wilke, eds., *Automata Logics, and Infinite Games.* vol. 2500 of *Lecture Notes in Computer Science.* pp. 43–60. Springer Berlin Heidelberg. 2002.

[19] S. Schewe. Büchi Complementation Made Tight. In *26th International Symposium on Theoretical Aspects of Computer Science-STACS 2009.* pp. 661–672. 2009.

[20] A. P. Sistla, M. Y. Vardi, P. Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Theoretical Computer Science.* 49(2–3):pp. 217 – 237. 1987.

[21] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control.* 54(1–2):pp. 121 – 141. 1982.

[22] W. Thomas. Automata on Infinite Objects. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science (Vol. B).* chap. Automata on Infinite Objects, pp. 133–191. MIT Press, Cambridge, MA, USA. 1990.

[23] U. Ultes-Nitsche. A Power-Set Construction for Reducing Büchi Automata to Non-Determinism Degree Two. *Information Processing Letters.* 101(3):pp. 107 – 111. 2007.

[24] M. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller, G. Birtwistle, eds., *Logics for Concurrency.* vol. 1043 of *Lecture Notes in Computer Science.* pp. 238–266. Springer Berlin Heidelberg. 1996.

[25] M. Y. Vardi, T. Wilke. Automata: From Logics to Algorithms. In J. Flum, E. Grädel, T. Wilke, eds., *Logic and Automata: History and Perspectives.* vol. 2 of *Texts in Logic and Games.* pp. 629–736. Amsterdam University Press. 2007.

[26] Q. Yan. Lower Bounds for Complementation of ω-Automata Via the Full Automata Technique. In M. Bugliesi, B. Preneel, V. Sassone, et al, eds., *Automata, Languages and Programming.* vol. 4052 of *Lecture Notes in Computer Science.* pp. 589–600. Springer Berlin Heidelberg. 2006.