Chasing the Best Büchi Automata for Nested Depth-first Search Based Model Checking

František Blahoudek*
Faculty of Informatics
Masaryk University
Brno, Czech Republic
xblahoud@fi.muni.cz

Abstract

The automata-based model checking uses a translation of formulae in Linear Temporal Logic (LTL) into Büchi automata (BA). The performance of the model checkers can be heavily influenced by the BA used. In this paper we discuss several heuristics commonly used to decide which BA should be used for given verification task, suggest a novel heuristic for this problem and finally evaluate the heuristics using common LTL-to-BA translators, model checker Spin and benchmark of real verification tasks. Our evaluation shows that heuristics based only on number of states of BA or the degree of determinism often give wrong answer or are unable to answer. On a concrete example we further demonstrate our suggestion to exploit some partial knowledge about systems to improve our heuristic.

1 Introduction

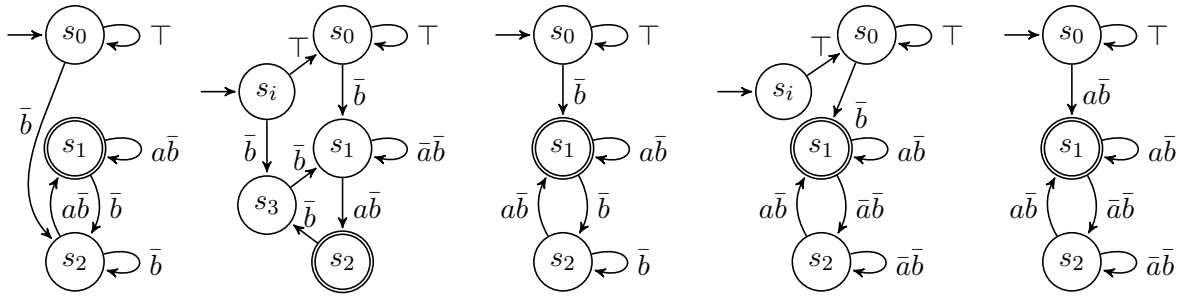
In the automata-theoretic approach to explicit model checking of *Linear-time Temporal Logic* (LTL) [14] the specification given as LTL formula φ is first negated into $\neg\varphi$, which is then translated into a *Büchi automaton* (BA) $\mathcal{A}_{\neg\varphi}$ that accepts all the executions of the system violating φ; the model checker then constructs the synchronous product $\mathcal{S} \otimes \mathcal{A}_{\neg\varphi}$, where $\mathcal{S}$ is the state space (automaton) for the system to be verified. The product is then checked for emptiness. If the language of $\mathcal{S} \otimes \mathcal{A}_{\neg\varphi}$ is non-empty, the witness of non-emptiness is the execution of $\mathcal{S}$ that invalidates φ, so called the *counterexample*. If the language of $\mathcal{S} \otimes \mathcal{A}_{\neg\varphi}$ is empty, the system satisfies the specification φ.

There are many LTL-to-BA translators and they often produce different automata for the same formula. For example, Figure 1 shows five automata for the LTL formula $\neg(\mathsf{GF}a \rightarrow \mathsf{GF}b)$ produced by different translators. In this paper we discuss different methods to decide which one of them should be used for the model checking to be efficient.

Usually both the state space $\mathcal{S}$ and the product $\mathcal{S} \otimes \mathcal{A}_{\neg\varphi}$ are built *on-the-fly* and only the part needed by the emptiness check is built. There are several emptiness check procedures used in different model checkers. We decided to study the performance of widely used, open-source model checker Spin which uses a sequential emptiness check based on *Nested Depth-First Search* (NDFS) [9].

The choice of automaton used as $\mathcal{A}_{\neg\varphi}$ can influence both the size of $\mathcal{S} \otimes \mathcal{A}_{\neg\varphi}$ and the emptiness check procedure. Among other aspects the choice of BA influences which part of $\mathcal{S} \otimes \mathcal{A}_{\neg\varphi}$ is

*The author has been supported by The Czech Science Foundation, grant GBP202/12/G061.



($\mathcal{B}_1$) LTL2BA ($\mathcal{B}_2$) MoDeLLa ($\mathcal{B}_3$) LTL3BA ($\mathcal{B}_4$) LTL3BA (det) ($\mathcal{B}_5$) Spot

Figure 1: Automata for the formula $\neg(\mathsf{GF}a \to \mathsf{GF}b)$. Note that edges in the automata are labelled by Boolean formulae over atomic propositions, where $\bar{a}$ means $\neg a$, $\top$ stands for *true*, and $a\bar{b}$ used later means $a \wedge \neg b$. Formally, an edge labelled with a formula ρ represents all the transitions that are labelled with a subset M of atomic propositions such that $M \models \rho$.

built in case the product contains some accepting cycle (counterexample). Namely, small change of order of transitions used in the file describing $\mathcal{A}_{\neg\varphi}$ can have tremendous impact on running time of Spin. Similarly to [6] we did not find any conclusion about "good order". Therefore we address here only cases without counterexample, where the whole product has to be built and explored by the NDFS.

We assume the reader is familiar with LTL and Büchi automata.

2 Benchmarking Büchi Automata

The authors of LTL-to-BA translators try to minimize the size [2, 5, 12, 7, 13] and/or the degree of determinism of the automata produced [1, 11, 3]. We can use these values as *heuristics* to answer the following question: *Which of two given automata $\mathcal{A}_1, \mathcal{A}_2$ should be used as $\mathcal{A}_{\neg\varphi}$ for the model checking to be faster?* A benchmark of LTL-to-BA translators belongs usually to one of the following categories:

Size: The size of an automaton is usually understood as the number of *states* (states($\mathcal{A}$)), or the number of *transitions* (trans($\mathcal{A}$)).

Determinism: The degree of determinism is measured by the number of *non-deterministic states* (nd-states($\mathcal{A}$)). A drawback of this heuristic is a fact that it does not quantify how non-deterministic these states are.

Products with random state spaces: Popular LTL-to-BA benchmarking tools like `lbtt` and `ltlcross` (from Spot library) perform following steps for each automaton: (1) build products with a fixed set of random state spaces, (2) sum their sizes, and (3) compare the results. This approach gives results depending on the random state spaces used.

2.1 Proposed heuristic

Figure 1 shows five Büchi automata generated by some of available LTL-to-BA translators for the formula $\neg(\mathsf{GF}a \to \mathsf{GF}b)$. The automata $\mathcal{B}_1$ and $\mathcal{B}_3$ have the same number of states, transitions and non-deterministic states, but as Table 1 indicates, the amount of work performed by Spin

Table 1: Statistics about generated automata and Spin's run on the empty product between model `peterson.4.pm` and formula $\neg(\mathsf{GF}a \to \mathsf{GF}b)$. The corresponding automata are shown in Fig. 1.

	automaton size				statistics from Spin's execution		
	states	ndst	edges	trans	stored states	visited trans	time
$\mathcal{B}_1$ LTL2BA	3	3	6	12	1577440	7684k	5.95s
$\mathcal{B}_2$ MoDeLLa	5	2	8	18	1580893	7670k	6.13s
$\mathcal{B}_3$ LTL3BA	3	3	6	12	2299250	15583k	12.10s
$\mathcal{B}_4$ LTL3BA (det)	4	1	7	14	2297625	15561k	12.00s
$\mathcal{B}_5$ Spot	3	1	6	9	848641	2853k	2.26s

measured in the number of visited transitions (and also the running time) differs substantially for $\mathcal{B}_1$ and $\mathcal{B}_3$. This illustrates that the size and determinism of automata are not always a relevant factor in the performance of the model checking process of Spin. For this reason we have designed heuristics based on a *product with universal model*.

The universal model for a set of atomic propositions AP is an automaton $\mathcal{U}_{AP} = (2^{AP}, \delta, \emptyset)$, where states are formed by combinations of atomic propositions and $(q, \alpha, \alpha) \in \delta$ for each $q, \alpha \in 2^{AP}$. See Figure 2 for the universal model with two atomic propositions.

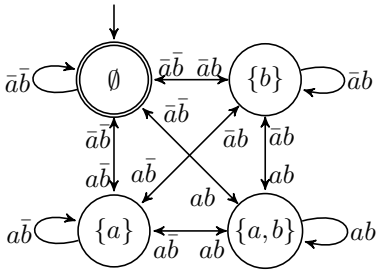


Figure 2: The universal model $\mathcal{U}_{\{a,b\}}$.

We improved the heuristic based on observation that the automata $\mathcal{B}_2$ and $\mathcal{B}_4$ contain redundant initial states s_i which can appear only once in any product. These states will not make any significant difference to the product size. In particular, only states that are reachable from some cycle can make a significant increase in the number of states of $\mathcal{S} \otimes \mathcal{A}_{\neg\varphi}$. Based on this observation we trim the product with $\mathcal{U}_{AP}$ to filter away redundant states. Henceforth we refer to the trimmed product as $\mathcal{T}$.

The number of states of $\mathcal{T}$ for the automata $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$, and $\mathcal{B}_5$ of Figure 1 are 7, 9, 8, 7, and 6 respectively. So this information helps us to distinguish between $\mathcal{B}_1$ and $\mathcal{B}_3$, moreover, it gives preference to $\mathcal{B}_4$ over $\mathcal{B}_3$ thanks to this trimming.

3 Heuristics Benchmark

In this section, we present a comparison of the heuristic proposed in the previous section, with the heuristics based on the size or determinism of automata. We examine how well they can estimate which of two given automata yields less work for Spin when no counterexample is found. We use the following software, hardware, benchmark and heuristics.

Software. We use the five LTL-to-BA translators listed in Table 2. Apart from the popular translators, Spin and LTL2BA, we use MoDeLLa which was the first tool focusing on determinism, as well as LTL3BA and Spot which represent state-of-the-art translators. The last two translators are used in several settings: the settings denoted by *LTL3BA (det)* aim to produce more deterministic automata, while the setting called *Spot (no jump)* is an experimental setting

Table 2: Reference of considered LTL-to-BA translators.

tool	version	translation command
Spin [4, 8]	6.2.5	`spin -f`
LTL2BA [5]	1.1	`ltl2ba -f`
MoDeLLa [11]	1.5.9	`modella -s -g -r12 ltlfile`
LTL3BA [1]	1.0.2	`ltl3ba -S -f`
LTL3BA (det)		`ltl3ba -S -M -f`
Spot [3]	1.2.4	`ltl2tgba -s`
Spot (no jump)		`ltl2tgba -s -x degen-lskip=0`

which produces automata with longer cycles than usual. For model checking of the tasks we used Spin with its default settings and the maximum depth of search set to 100 000 000.

Hardware. All computations were performed on an HP DL980 G7 server with 8 eight-core 64-bit processors Intel Xeon X7560 2.26GHz and 448 GiB DDR3 RAM. Each execution of Spin was restricted to a 30 minute time-out and a memory limit of 20GiB.

Benchmark. The benchmark set of models and formulae is based on real model checking tasks BEEM [10]. In addition to the original 735 pairs of a model in Promela and an LTL formula, we added two formulae to each instance of mutual exclusion protocols `anderson`, `bakery`, and `peterson` (altogether 22 instances):

1. $\mathsf{GF}(P_0@\mathrm{CS}) \to \mathsf{GF}(P_0@\mathrm{NCS})$ meaning that if a process P_0 spends infinite number of steps in a critical section, then it also spends infinite number of steps in a non-critical section,
2. $\mathsf{FG}\neg\big((P_0@\mathrm{CS} \wedge P_1@\mathrm{CS}) \vee (P_0@\mathrm{CS} \wedge P_2@\mathrm{CS}) \vee (P_1@\mathrm{CS} \wedge P_2@\mathrm{CS})\big)$ meaning that after finite number of steps, it never happens that two of the processes P_0, P_1, and P_2 are in a critical section at the same time.

We removed pairs where the formula contains the reserved keyword of Promela `active` in a name of some atomic proposition, because Spin fails to run these tasks (12 pairs), we also removed pairs where for at least one translator Spin timeouted or ran out of memory (11 pairs). We further removed all the pairs where some counterexample was found (674 pairs). This left us with $735 + 2 \cdot 22 - 12 - 11 - 674 = 82$ verification tasks without any counterexamples.

Each of the 82 verification tasks without counterexample was verified by Spin using LTL-to-BA translators of Table 2. The triplet (*model,formula,translator*) is called *Spin instance*. We gathered the statistics about automata and product with the universal model using `ltlcross` and also information about the number of transitions visited by Spin (*Spin-trans*) for each Spin instance.

To focus on only relevant differences in Spin performance, we selected for each verification task only the pairs of Spin instances (I_1, I_2), where the difference between *Spin-trans*(I_1) and *Spin-trans*(I_2) was at least 20% of the smaller value. We call such pairs of Spin instances *matches*. We ended up with 603 matches.

Heurisitics. We used several heuristics based on automata ($\mathcal{A}$) statistics: *states($\mathcal{A}$)* answers that out of two BA the one with lower number of states should be preferred and if the numbers are equal, it returns no answer, *trans($\mathcal{A}$)* is based on the number of transitions, *nd-states($\mathcal{A}$)* is

heuristic	correct	incorrect	no answer	score
states($\mathcal{T}$); states($\mathcal{A}$)	495	57	51	438
states($\mathcal{T}$); trans($\mathcal{A}$)	511	84	8	427
states($\mathcal{A}$); states($\mathcal{T}$)	489	63	51	426
states($\mathcal{T}$)	471	48	84	423
trans($\mathcal{T}$)	495	83	25	412
states($\mathcal{T}$); trans($\mathcal{T}$)	495	91	17	404
states($\mathcal{A}$); nd-states($\mathcal{A}$)	487	90	26	397
trans($\mathcal{A}$)	459	93	51	366
states($\mathcal{A}$); trans($\mathcal{A}$)	456	96	51	360
states($\mathcal{A}$)	371	39	193	332
nd-states($\mathcal{A}$)	316	75	212	241

Table 3: Comparison of ten chosen heuristics on selected matches. The maximum possible score was 603.

based on the number of non-deterministic states. We also used heuristics based on the trimmed product with universal model ($\mathcal{T}$). The combination of two heuristics, for example states($\mathcal{A}$); trans($\mathcal{A}$), proceeds as follows: Answer as the first heuristic (states($\mathcal{A}$)) if it gives some answer; if the first heuristic gives no answer, use the result of the second heuristic (trans($\mathcal{A}$)).

3.1 Results

For each of the selected matches we compared the values of *Spin-trans* and then asked heuristics to give its opinion which automaton should be preferred. The numbers of correct, incorrect and no answers have been accumulated for each heuristic and based on these values the score was computed as *correct*$-$*incorrect*. The obtained results can be found in Table 3, sorted by score.

Table 3 shows that states($\mathcal{T}$) outperforms states($\mathcal{A}$) when compared by score, on the other hand, states($\mathcal{A}$) gives the smallest number of wrong answers. A natural combination of these two gives the best results.

4 Using Partial Knowledge about System

We can improve our proposed heuristic by adding some knowledge we have about the system to be verified. For example, from the Promela description of the system `peterson.4.pm` we can derive following information: (1) Both atomic propositions a and b mean that the process P_0 is in some location, hence a and b can never hold together; (2) There are five locations in the model, so we can assume that states without a are four times more frequent in the model than states with a (the same applies to b). We can modify the universal model accordingly to (1) by deleting state $\{a, b\}$ and accordingly to (2) by adding two copies of the state $\emptyset$.

The numbers of states of the trimmed product with the universal model modified as suggested for the automata $\mathcal{B}_1 - \mathcal{B}_5$ are 11, 14, 13, 12, and 9 respectively. A heuristic based on these values would be, for the particular example of Table 1, the most precise from all heuristics presented here. To be more specific, the heuristic would give answer for all matches, and the answer will be incorrect in only 3 out of 10 cases.

Algorithms for automatic extraction of useful information about model which could improve our heuristics are subject of future research, as well as new heuristics that can improve the prediction of the most suitable BA for given task and thus save time spent on verification.

References

[1] Tomáš Babiak, Mojmír Křetínský, Vojtěch Řeehák, and Jan Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *Proc. of the 18th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, volume 7214 of *LNCS*, pages 95–109. Springer, 2012.

[2] Jean-Michel Couvreur. On-the-fly verification of temporal logic. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems (FM'99)*, volume 1708 of *LNCS*, pages 253–271, Toulouse, France, September 1999. Springer-Verlag.

[3] Alexandre Duret-Lutz. LTL translation improvements in Spot 1.0. *International Journal on Critical Computer-Based Systems*, 5(1/2):31–54, March 2014.

[4] K. Etessami and G. J. Holzmann. Optimizing Büchi Automata. In *CONCUR'00*, volume 1877 of *LNCS*, pages 153–167. Springer, 2000.

[5] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 53–65, Paris, France, 2001. Springer-Verlag.

[6] Jaco Geldenhuys and Antti Valmari. More efficient on-the-fly LTL verification with Tarjan's algorithm. *Theoretical Computer Science*, 345(1):60–82, November 2005.

[7] Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: Improving translation of LTL formulæ to Büchi automata. In D.A. Peled and M.Y. Vardi, editors, *Proceedings of the 22nd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'02)*, volume 2529 of *LNCS*, pages 308–326, Houston, Texas, November 2002. Springer-Verlag.

[8] Gerard J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2003.

[9] Gerard J. Holzmann, Doron A. Peled, and Mihalis Yannakakis. On nested depth first search. In Jean-Charles Grégoire, Gerard J. Holzmann, and Doron A. Peled, editors, *Proceedings of the 2nd Spin Workshop (SPIN'96)*, volume 32 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, May 1996.

[10] Radek Pelánek. BEEM: benchmarks for explicit model checkers. In *Proceedings of the 14th international SPIN conference on Model checking software (SPIN'07)*, volume 4595 of *LNCS*, pages 263–267. Springer-Verlag, 2007.

[11] Roberto Sebastiani and Stefano Tonetta. "More deterministic" vs. "smaller" Büchi automata for efficient LTL model checking. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Proceedings of the 12th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'03)*, volume 2860 of *LNCS*, pages 126–140, L'Aquila, Italy, October 2003. Springer-Verlag.

[12] Fabio Somenzi and Roderick Bloem. Efficient Büchi automata for LTL formulæ. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'00)*, volume 1855 of *LNCS*, pages 247–263, Chicago, Illinois, USA, 2000. Springer-Verlag.

[13] Xavier Thirioux. Simple and efficient translation from LTL formulas to Büchi automata. In Rance Cleaveland and Hubert Garavel, editors, *Proceedings of the 7th International ERCIM Workshop in*

Formal Methods for Industrial Critical Systems (FMICS'02), volume 66(2) of *Electronic Notes in Theoretical Computer Science*, Málaga, Spain, July 2002. Elsevier.

[14] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In Faron Moller and Graham M. Birtwistle, editors, *Proceedings of the 8th Banff Higher Order Workshop (Banff'94)*, volume 1043 of *LNCS*, pages 238–266, Banff, Alberta, Canada, 1996. Springer-Verlag.