`ltl3tela`: LTL to Small Deterministic or Nondeterministic Emerson-Lei Automata*

Juraj Major[1], František Blahoudek[2], Jan Strejček[1], Miriama Sasaráková[1], and Tatiana Zbončáková[1]

[1] Masaryk University, Brno, Czech Republic
[2] University of Mons, Belgium
{major, xblahoud, strejcek}@fi.muni.cz

Abstract. The paper presents a new tool `ltl3tela` translating LTL to deterministic or nondeterministic transition-based Emerson-Lei automata (TELA). Emerson-Lei automata use generic acceptance formulae with basic terms corresponding to Büchi and co-Büchi acceptance. The tool combines algorithms of Spot library, a new translation of LTL to TELA via alternating automata, a pattern-based automata reduction method, and few other heuristics. Experimental evaluation shows that `ltl3tela` can produce deterministic automata that are, on average, noticeably smaller than deterministic TELA produced by state-of-the-art translators Delag, Rabinizer 4, and Spot. For nondeterministic automata, the improvement over Spot is smaller, but still measurable.

1 Introduction

Translation of LTL formulae into equivalent automata over infinite words is an important part of many methods for model checking, control synthesis, monitoring, vacuity checking etc. Different applications require different types of automata, which are specified by restrictions on automata structure or acceptance condition. Two most popular structures are deterministic and nondeterministic. While nondeterministic automata have been traditionally considered with Büchi, deterministic automata have typically used Rabin, Streett, or parity acceptance as deterministic Büchi automata are strictly less expressive.

With rising number of practical automata applications, we can see a clear shift towards more complex acceptance conditions which allow to construct and manipulate automata with less states and often lead to performance improvements. This trend started slowly with generalized Büchi and generalized Rabin acceptance and accelerates after introduction of the Hanoi Omega-Automata Format [3]. The format reinvented a generic acceptance condition originally considered by Emerson and Lei in 1980s [9], which can uniformly present all the mentioned conditions.

Recently, first tools that aim to take advantage of Emerson-Lei (EL) acceptance in order to produce smaller automata appeared. In particular, the tool

* J. Major and J. Strejček have been supported by Czech Science Foundation, grant GA19-24397S. F. Blahoudek has been supported by F.R.S.-FNRS under Grant n° F.4520.18 (ManySynth).

Delag [13] translates LTL to deterministic EL automata and Spot [8] can produce deterministic or nondeterministic EL automata for LTL since version 2.6. The development of tools producing small EL automata creates a demand for efficient algorithms that directly process these automata (without previous transformation to any simpler automata type). Fresh results of this kind are algorithms for checking emptiness of EL automata and for probabilistic model checking under properties specified by deterministic EL automata [5].

We present a tool `ltl3tela` that combines algorithms of Spot with a novel LTL to EL-automata translation, pattern-based automata reduction method, and some other techniques in order to translate LTL to deterministic or nondeterministic *transition-based Emerson-Lei automata (TELA)* with low number of states. The overall translation algorithm is explained in Section 2 together with precise definition of TELA and brief description of individual translation components. Section 3 discusses implementation and basic usage of the tool, and Section 4 compares `ltl3tela` with state-of-the-art tools translating LTL to deterministic or nondeterministic automata using random formulae and formulae collected from the literature.

2 Translation Algorithm

Transition-based Emerson-Lei Automata (TELA)

A *nondeterministic TELA* $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, M, \delta, q_I, \varphi)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, M is a finite set of *acceptance marks*, $\delta \subseteq Q \times 2^\Sigma \times 2^M \times Q$ is a set of *edges*, $q_I \in Q$ is the *initial state*, and φ is the acceptance formula built by the following grammar with m ranging over M.

$$\varphi \ ::= \ \mathsf{t} \mid \mathsf{f} \mid \mathsf{Inf}(m) \mid \mathsf{Fin}(m) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi)$$

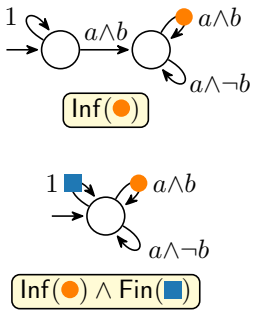
A *run* ρ is an infinite sequence of consecutive edges starting in the state q_I. Every run satisfies t (true) and does not satisfy f (false). A run satisfies $\mathsf{Inf}(m)$ if m is present on infinitely many edges of ρ, and only satisfies $\mathsf{Fin}(m)$ otherwise. A run is *accepting* if it satisfies φ. Automaton $\mathcal{A}$ is *complete* if for each $q \in Q$ and $a \in \Sigma$ there is at least one edge $(q, A, M', q') \in \delta$ such that $a \in A$. Finally, $\mathcal{A}$ is *deterministic* if for each $q \in Q$ and $a \in \Sigma$ there is at most one edge $(q, A, M', q') \in \delta$ such that $a \in A$. We emphasize that the term *nondeterministic* is not the opposite of *deterministic*, but a deterministic automaton is a special case of a nondeterministic one.

Translation Components

The most important novel component of `ltl3tela` is an LTL to nondeterministic TELA translation via *self-loop alternating automata (SLAA)* [12]. It builds upon the ideas of `ltl2ba` [10] and `ltl3ba` [1], which translate LTL to *transition-based generalized Büchi automaton (TGBA)* via very weak alternating co-Büchi automata. Our new translation uses alternating automata with similar structure,

but with more complex acceptance conditions and acceptance marks on transitions (in particular, SLAA are not weak any more). Such alternating automata are often smaller and allow us to produce smaller nondeterministic automata. We refer to this translation as `ltl3slaa3tela`.

Another new concept is a pattern-based reduction applicable to TGBA [16]. Intuitively, the procedure looks for automata subgraphs corresponding to certain patterns. Each pattern includes states that can be merged into one state for the price of a more complicated acceptance condition containing a subformula $\mathsf{Fin}(m)$, where m is a fresh acceptance mark. Hence, this pattern-based reduction method transforms TGBA into TELA. On the right, you can see a TGBA for the formula $\mathsf{FG}a \wedge \mathsf{GF}b$ (top) and the corresponding automaton after pattern-based reduction (bottom).



Further, we use a lot of functionality offered by the Spot library, in particular:

- LTL formulae parsing and preprocessing
- LTL to nondeterministic TELA translation performed by `ltl2tgba -G`
- LTL to deterministic TELA translation performed by `ltl2tgba -DG`
- automata determinization based on the algorithm of Redziejowski [14],
- automata reduction procedure based on SCC pruning [15], minimization of WDBA [6], and simulation-based reductions [15],
- functions for the synchronous product of two automata and specialized functions creating potentially smaller product if one of the automata is *suspendable* [2].

We also define the functions *min* and *trans* used later in the algorithm. The function *min* selects the minimal automaton out of a given sequence $\mathcal{A}_1, \ldots, \mathcal{A}_n$. More precisely, to each automaton $\mathcal{A}_i = (Q, \Sigma, M, \delta, q_I, \varphi)$ it assigns the tuple $(|Q|, d, |M|, |\delta|, i)$ with d being 1 for deterministic automata and 2 otherwise, and returns the automaton with the lexicographically minimal tuple.

The function $\mathit{trans}(\psi, t)$ aims to get the best automaton for ψ using the translator t. More precisely, it translates both ψ and $\neg\psi$ using t, simplifies both automata using Spot's reduction procedure and the pattern-based reduction (if applicable), resulting in $\mathcal{A}_\psi$ and $\mathcal{A}_{\neg\psi}$. If $\mathcal{A}_{\neg\psi}$ is deterministic, the function complements it (makes it complete and dualizes the acceptance condition by swapping Inf for Fin, $\mathtt{t}$ for $\mathtt{f}$, and $\wedge$ for $\vee$) into $\mathcal{A}'_\psi$, and returns $\mathit{min}(\mathcal{A}_\psi, \mathcal{A}'_\psi)$. If $\mathcal{A}_{\neg\psi}$ is not deterministic, $\mathit{trans}(\psi, t)$ simply returns $\mathcal{A}_\psi$.

The Algorithm

Our translation of LTL into nondeterministic TELA follows the idea introduced by Delag (and followed also by `ltl2tgba` with the `-G` option): we split the input formula recursively into temporal subformulae, translate each subformula independently, and merge the automata for subformulae using synchronous products. This allows us to use different translations where appropriate for the subformulae. Our take on the approach is described by Algorithm 1 where we use product

Algorithm 1: `ltl3tela` translation of LTL for nondeterministic TELA

preprocess φ to contain only Boolean operators $\wedge, \vee, \neg$ and $\neg$ not before $\wedge, \vee$
$\mathcal{A} \leftarrow min(trans(\varphi, \texttt{ltl3slaa3tela}), trans(\varphi, \texttt{ltl2tgba -G}), transRec(\varphi))$

function *transRec*(φ)
  if φ *is a conjunction* **then**
    let $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n \wedge \psi_1 \wedge \ldots \wedge \psi_m$ where ψ_i are suspendable
    foreach $\rho \in \{\varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_m\}$ **do** $\mathcal{A}_\rho \leftarrow transRec(\rho)$
    return $\big(\ldots\big((\mathcal{A}_{\varphi_1} \hat{\otimes} \ldots \hat{\otimes} \mathcal{A}_{\varphi_n}) \hat{\otimes}_s \mathcal{A}_{\psi_1}\big) \hat{\otimes}_s \ldots\big) \hat{\otimes}_s \mathcal{A}_{\psi_m}$
  if φ *is a disjunction* **then**
    let $\varphi = \varphi_1 \vee \ldots \vee \varphi_n \vee \psi_1 \vee \ldots \vee \psi_m$ where ψ_i are suspendable
    foreach $\rho \in \{\varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_m\}$ **do** $\mathcal{A}_\rho \leftarrow transRec(\rho)$
    return $\big(\ldots\big((\mathcal{A}_{\varphi_1} \check{\otimes} \ldots \check{\otimes} \mathcal{A}_{\varphi_n}) \check{\otimes}_s \mathcal{A}_{\psi_1}\big) \check{\otimes}_s \ldots\big) \check{\otimes}_s \mathcal{A}_{\psi_m}$
  return $min(trans(\varphi, \texttt{ltl3slaa3tela}), trans(\varphi, \texttt{ltl2tgba -G}))$

operations $\check{\otimes}, \hat{\otimes}$ for automata union and intersection, respectively, and their optimized versions $\check{\otimes}_s, \hat{\otimes}_s$ when the right argument is a suspendable automaton. Each product operation applies Spot's automata reduction procedure and the pattern-based reduction (if applicable) before returning the product. At the end, we compare the resulting automaton with the automata produced directly for the input formula and return the minimal one.

The translation of LTL to deterministic TELA works in the same way with only two changes. First, instead of `ltl2tgba -G` we always call `ltl2tgba -DG`. Second, the function *trans* calls the Spot's automata reduction procedure with the request to produce a deterministic TELA. Hence, if `ltl3slaa3tela` produces an automaton that is not deterministic, it is determinized by Spot.

3 Implementation

The tool `ltl3tela` is written in C++14 and requires only the Spot library [8] version 2.6 or higher for compilation. The tool is available at `https://github.com/jurajmajor/ltl3tela` under the GNU GPL 3.0 license. The default mode takes an LTL formula φ as input, runs Algorithm 1 to produce a nondeterministic TELA for φ, and output it in the Hanoi Omega-Automata (HOA) format [4]. Deterministic automata are produced by `ltl3tela -D1`. The tool can also produce self-loop alternating automata. See the documentation for more details.

4 Experimental Evaluation

We compare `ltl3tela` to state-of-the-art LTL to deterministic TELA translators `ltl2tgba -DG`, Delag, and Rabinizer 4, which produces *transition-based generalized Rabin automata (TGRA)*. For the nondeterministic case, we compare `ltl3tela` to `ltl2tgba -G` and two LTL to TGBA translators `ltl2tgba` and `ltl3ba`. For references and version numbers of all the tools see Table 1.

Table 1. References and versions of compared LTL to automata translators

tool name	version	homepage
Delag [13], Rabinizer 4 [11]	Owl 18.06	`owl.model.in.tum.de`
`ltl2tgba` [8]	Spot 2.7.4	`spot.lrde.epita.fr`
`ltl3ba` [1]	1.1.3	`sourceforge.net/projects/ltl3ba`
`ltl3tela`	2.0.0	`github.com/jurajmajor/ltl3tela`

We use two sets of LTL formulae for the comparison:

random contains 1000 random formulae generated with `randltl`[3] such that there are no duplicates and formulae equivalent to *true* or *false*,

literature contains 397 formulae from the literature provided by `genltl` (for each formula pattern, we consider instances with parameter values $1, \ldots, 5$).

Each translator has been executed on each formula using the tool `ltlcross` with 60 seconds time limit. The experiments ran on a laptop with Intel® Core™ i7-8550U, 16 GB of RAM, and Debian 9.9. Table 2 presents the cumulative numbers of states, edges, and acceptance marks for each translator and set of formulae. While all translators finished successfully on random formulae, we encountered some timeouts and *parse errors* (`ltlcross` cannot parse automata with more than 32 acceptance marks) on formulae from the literature and thus we had to remove some formulae from the cumulative numbers in Table 2. We have complete results (no timeout or parse error) on 353 formulae for deterministic automata and on 368 formulae for nondeterministic automata. The lists of excluded formulae are in Tables 3 and 4.

Table 2. Sums of states, edges, and marks of automata produced by individual translators for considered formula sets, the number of timeouts (TO) and parse errors (PE)

		random			literature				
	tool	states	edges	acc	states	edges	acc	TO	PE
det.	`ltl3tela -D1`	5934	18520	1268	2536	10641	454	39	0
	`ltl2tgba -DG`	6799	24131	1575	3905	26643	652	20	0
	Delag	7176	71672	3089	8661	2209807	1196	11	10
	Rabinizer 4	7581	31099	2786	2969	12358	1133	12	8
nondet.	`ltl3tela`	5109	12481	1135	2378	20718	544	28	0
	`ltl2tgba -G`	5391	13144	1041	2398	20555	642	12	0
	`ltl2tgba`	5413	13059	1034	2651	8721	502	11	0
	`ltl3ba -H2`	6103	15636	1616	4654	21180	822	4	0

The results show that `ltl3tela` is the slowest one, which is not surprising as it internally translates every formula several times. However, it produces significantly smaller deterministic automata (in sum) comparing the other tools. The

[3] We use the tools `randltl`, `genltl`, and `ltlcross` [7] from the Spot library 2.7.4.

Table 3. Formulae from the literature (described by the corresponding `genltl` options) that are not covered by the accumulated results in Table 2 (nondet.) due to some timeout (TO), and sizes of nondeterministic automata for these formulae

formula (as `genltl` option)	`ltl3tela`	`ltl2tgba -G`	`ltl2tgba`	`ltl3ba -H2`
`--gh-r=5`	TO	1	14	244
`--hkrss-patterns=45`	TO	12	12	12
`--kr-n=2`	TO	TO	25	25
`--kr-n=3`	TO	TO	58	58
`--kr-n=4`	TO	TO	TO	131
`--kr-n=5`	TO	TO	TO	TO
`--kr-nlogn=2`	TO	TO	43	44
`--kr-nlogn=3`	TO	TO	TO	TO
`--kr-nlogn=4`	TO	TO	TO	TO
`--kr-nlogn=5`	TO	TO	TO	TO
`--kv-psi=2`	TO	19	19	29
`--kv-psi=3`	TO	39	39	73
`--kv-psi=4`	TO	TO	TO	177
`--kv-psi=5`	TO	TO	TO	417
`--ms-phi-h=5`	TO	32	64	64
`--rv-counter-carry=5`	TO	160	160	160
`--rv-counter-carry-linear=5`	TO	160	160	160
`--sejk-f=4,1`	TO	13	13	64
`--sejk-f=4,2`	TO	14	14	112
`--sejk-f=4,3`	TO	15	15	192
`--sejk-f=4,4`	TO	16	16	336
`--sejk-f=4,5`	TO	17	17	608
`--sejk-f=5,1`	TO	15	15	128
`--sejk-f=5,2`	TO	16	16	224
`--sejk-f=5,3`	TO	17	17	384
`--sejk-f=5,4`	TO	TO	TO	672
`--sejk-f=5,5`	TO	TO	TO	1216
`--sejk-k=4`	1	1	TO	82
`--sejk-k=5`	TO	1	TO	244

differences are less dramatic for nondeterministic automata. Detailed analysis of the results shows that `ltl3tela -D1` is always better (in the sense of function *min*) than `ltl2tgba -DG`. There is no such other case. A surprising result is the significantly low number of acceptance marks for `ltl3tela -D1`.

All considered formulae, measured results, scripts generating the formulae, Jupyter notebooks that run experiments and process them, and some more tables can be found at: `https://github.com/jurajmajor/ltl3tela/blob/master/ATVA19.md`

5 Conclusion

We presented the tool `ltl3tela` that combines some new ideas with functionality of Spot in order to translate LTL to small deterministic or nondeterministic

Table 4. Formulae from the literature (described by the corresponding `genltl` options) that are not covered by the accumulated results in Table 2 (det.) due to some timeout (TO) or parse error (PE), and sizes of deterministic automata for these formulae

formula (as `genltl` option)	`ltl3tela -D1`	`ltl2tgba -DG`	Delag	Rabinizer 4
`--gf-equiv=4`	1	1	1	PE
`--gf-equiv=5`	TO	1	1	PE
`--gh-r=5`	1	1	1	PE
`--hkrss-patterns=45`	TO	12	12	12
`--kr-n=2`	TO	TO	183	141
`--kr-n=3`	TO	TO	6937	TO
`--kr-n=4`	TO	TO	TO	TO
`--kr-n=5`	TO	TO	TO	TO
`--kr-nlogn=2`	TO	TO	288	TO
`--kr-nlogn=3`	TO	TO	TO	TO
`--kr-nlogn=4`	TO	TO	TO	TO
`--kr-nlogn=5`	TO	TO	TO	TO
`--kv-psi=2`	TO	106	TO	115
`--kv-psi=3`	TO	3057	TO	TO
`--kv-psi=4`	TO	TO	TO	TO
`--kv-psi=5`	TO	TO	TO	TO
`--ms-phi-h=5`	TO	32	32	420
`--ms-phi-r=4`	TO	1	1	1
`--ms-phi-r=5`	1	1	1	TO
`--ms-phi-s=4`	TO	1	1	PE
`--ms-phi-s=5`	1	1	1	TO
`--rv-counter-carry=5`	TO	160	160	160
`--rv-counter-carry-linear=5`	TO	160	163	163
`--sejk-f=3,1`	TO	2781	3	3
`--sejk-f=3,2`	TO	2782	4	4
`--sejk-f=3,3`	TO	2783	5	5
`--sejk-f=3,4`	TO	2784	6	6
`--sejk-f=3,5`	TO	2785	7	7
`--sejk-f=4,1`	TO	TO	PE	3
`--sejk-f=4,2`	TO	TO	PE	4
`--sejk-f=4,3`	TO	TO	PE	5
`--sejk-f=4,4`	TO	TO	PE	6
`--sejk-f=4,5`	TO	TO	PE	7
`--sejk-f=5,1`	TO	TO	PE	3
`--sejk-f=5,2`	TO	TO	PE	4
`--sejk-f=5,3`	TO	TO	PE	5
`--sejk-f=5,4`	TO	TO	PE	6
`--sejk-f=5,5`	TO	TO	PE	7
`--sejk-j=4`	TO	1	1	PE
`--sejk-j=5`	TO	1	1	PE
`--sejk-k=4`	TO	1	1	PE
`--sejk-k=5`	TO	1	1	PE
`--tv-uu=4`	28	51	TO	52
`--tv-uu=5`	TO	298	TO	199

automata with Emerson-Lei acceptance condition. Experiments indicated that our tool produces (on average) the smallest automata compared to state-of-the-art translators. In particular, we produced deterministic automata with the least number of states, edges, and acceptance marks compared to other translators.

References

1. T. Babiak, M. Křetínský, V. Řehák, and J. Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *TACAS'12*, *LNCS* 7214, pp. 95–109. Springer, 2012.
2. T. Babiak, T. Badie, A. Duret-Lutz, M. Křetínský, and J. Strejček. Compositional approach to suspension and other improvements to LTL translation. In *Model Checking Software - 20th International Symposium, SPIN 2013*, *LNCS* 7976, pp. 81–98. Springer, 2013.
3. T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. The hanoi omega-automata format. In *Computer Aided Verification*, pp. 479–486. Springer, 2015.
4. T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. The Hanoi omega-automata format. In *CAV'15*, *LNCS* 9206, pp. 479–486. Springer, 2015.
5. C. Baier, F. Blahoudek, A. Duret-Lutz, J. Klein, D. Müller, and J. Strejček. Generic emptiness check for fun and profit. In *Proc. of ATVA 2019*, 2019. To appear.
6. C. Dax, J. Eisinger, and F. Klaedtke. Mechanizing the powerset construction for restricted classes of ω-automata. In *ATVA'07*, *LNCS* 4762. Springer, 2007.
7. A. Duret-Lutz. Manipulating LTL formulas using Spot 1.0. In *ATVA'13*, *LNCS* 8172, pp. 442–445. Springer, 2013.
8. A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *ATVA'16*, *LNCS* 9938, pp. 122–129. Springer, 2016.
9. E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.
10. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *CAV'01*, *LNCS* 2102, pp. 53–65. Springer, 2001.
11. J. Křetínský, T. Meggendorfer, S. Sickert, and C. Ziegler. Rabinizer 4: From LTL to your favourite deterministic automaton. In *CAV'18*, *LNCS* 10981, pp. 567–577. Springer, 2018.
12. J. Major. Translation of LTL into nondeterministic automata with generic acceptance condition. Master's thesis, Masaryk University, Brno, 2017.
13. D. Müller and S. Sickert. LTL to deterministic Emerson-Lei automata. In *Proc. Eighth International Symposium on Games, Automata, Logics and Formal Verification (GandALF)*, vol. 256 of *EPTCS*, pp. 180–194, 2017. URL `http://arxiv.org/abs/1709.02102`.
14. R. R. Redziejowski. An improved construction of deterministic omega-automaton using derivatives. *Fundam. Inform.*, 119(3-4):393–406, 2012.
15. F. Somenzi and R. Bloem. Efficient Büchi automata for LTL formulae. In *CAV'00*, *LNCS* 1855, pp. 247–263. Springer, 2000.
16. T. Zbončáková. Redukce omega-automatů s využitím Emerson-Lei akceptační podmínky. Bachelor's thesis, Masaryk University, Brno, 2018.