MightyL: A Compositional Translation from MITL to Timed Automata*

Thomas Brihaye[1], Gilles Geeraerts[2], Hsi-Ming Ho[1], Benjamin Monmege[3]

[1] Université de Mons, Belgium, thomas.brihaye,hsi-ming.ho@umons.ac.be
[2] Université libre de Bruxelles, Belgium, gigeerae@ulb.ac.be
[3] Aix Marseille Univ, CNRS, LIF, France, benjamin.monmege@lif.univ-mrs.fr

1 Introduction

The design of critical software that respect real-time specifications is a notoriously difficult problem. In this context, verification of programs against formal specifications is crucial, in order to handle the thin timing behaviours. In the untimed setting, a logic widely used both in academia and industry is *Linear Temporal Logic* (LTL) [16]. A crucial ingredient of its success is the possibility to translate LTL formulae into (Büchi) automata. In the real-time counterpart, *Metric Interval Temporal Logic* (MITL) [1] has been introduced twenty years ago where it was established that it can be translated into (Büchi) *timed automata* (TA). Beyond verification of real-time software, there are numerous interests in MITL from other domains, e.g. automated planning and scheduling [18], control engineering [8] and systems biology [3]. The translation from MITL to TAs is complicated and has led to some simplified constructions, e.g. [7, 13]. However, despite these efforts, the tool support for MITL is still lacking to this day. To the best of our knowledge, the only implementation of an automata-based construction is described in [5, 6], but is not publicly available. Since existing verification tools based on timed automata have been around for quite some time and have been successful (e.g. Uppaal [12] first appeared in 1995), it would be preferable if such translation can be used with these tools.

In the present paper, we attempt to amend the situation by proposing a more practical construction from MITL to (Büchi) timed automata. Compared to [5, 6], our construction has the following advantages:

1. While we also use *one-clock alternating timed automata* (OCATA) [14] as an intermediate formalism, our construction exploits the 'very-weakness' of the structure of OCATAs obtained from MITL formulae to reduce state space. In particular, our construction subsumes LTL2BA [9] in the case of LTL.
2. The number of clocks in the resulting TA is reduced by a factor of up to two. This is achieved via a more fine-grained analysis of the possible clock values.

* This work has been supported by the FRS/F.N.R.S. PDR grant SyVeRLo, and (partially) funded by the DeLTA project (ANR-16-CE40-0007) and the SensAS project (INS2I JCJC'17).

3. The construction is *compositional*: for each location of the OCATA $\mathcal{A}$ obtained from the input MITL formula, we construct a 'component' TA and establish a connection between the runs of $\mathcal{A}$ and the runs of the synchronous product of these components. Thanks to this connection, we can give the output TA in terms of components; this greatly simplifies the implementation, and speeds up its execution.
4. The construction is compatible with off-the-shelf model-checkers: our tool MightyL generates output automata in the Uppaal xml format which, besides Uppaal [12] itself, can also be analysed by LTSmin [10] with opaal front-end, TiAMo [4], ITS-tools [17], DiVinE [2], etc.

2 Implementation

We have implemented our translation from MITL formulae to generalised Büchi timed automata in a tool called MightyL, written in OCaml. When the input formula is in $\mathsf{MITL}_{0,\infty}$, the translation can be done in polynomial time. For the general case, it runs in exponential time (assuming a succinct encoding of constants, as is the case here). We can then use Uppaal [12] to check the satisfiability of φ over finite timed words, or LTSmin [10] with opaal front-end to check satisfiability over infinite timed words. Our tool is publicly available, and can even be executed directly on the website

`http://www.ulb.ac.be/di/verif/mightyl`

We check the satisfiability of MITL formulae on examples, inspired by the benchmarks of [6,9]. For $k \in \mathbb{N}$ and an interval I, we consider the satisfiable formulae: $F(k, I) = \bigwedge_{i=1}^{k} \mathbf{F}_I p_i$, $G(k, I) = \bigwedge_{i=1}^{k} \mathbf{G}_I p_i$, $U(k, I) = (\cdots (p_1 \, \mathbf{U}_I \, p_2) \, \mathbf{U}_I \cdots) \, \mathbf{U}_I \, p_k$, $R(k, I) = (\cdots (p_1 \, \mathbf{R}_I \, p_2) \, \mathbf{R}_I \cdots) \, \mathbf{R}_I \, p_k$, and $\theta(k, I) = \neg((\bigwedge_{i=1}^{k} \mathbf{G}\mathbf{F} p_i) \Rightarrow \mathbf{G}(q \Rightarrow \mathbf{F}_I r))$. We also consider an example inspired by motion planning problems [11,15]. In this benchmark, a robot must visit target points $t_1, t_2, t_3, \ldots, t_k$ within given time frames (in our case, t_i must be in $[3(i-1), 3i]$), while enforcing a safety condition $\mathbf{G}\neg p$. This specification is modelled by the satisfiable MITL formula $\mu(k) = \bigwedge_{i=1}^{k} \mathbf{F}_{[3(i-1),3i]} t_i \wedge \mathbf{G}\neg p$. In Table 1, we report on the time taken by the execution of MightyL; LTSmin (split into the time taken by opaal front-end to translate the model into C++ code, the compilation time of the resulting C++ code, and the time taken by LTSmin for the actual model-checking); and Uppaal, on all these examples (for the motion planning, only finite words are relevant, hence we report only on the Uppaal running time).

References

1. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. Journal of the ACM 43(1), 116–146 (1996)
2. Barnat, J., Brim, L., Havel, V., Havlícek, J., Kriho, J., Lenco, M., Rockai, P., Still, V., Weiser, J.: DivinE 3.0 - an explicit-state model checker for multithreaded C & C++ programs. In: CAV'13. LNCS, vol. 8044, pp. 863–868. Springer (2013)

Table 1. Execution time for the satisfiability checks of benchmarks of [6, 9].

Formula	MightyL	LTSmin	Uppaal
$F(5,[0,\infty))$	9ms	3.48s/2.18s/0.12s	0.75s
$F(5,[0,2])$	7ms	3.76s/2.23s/0.15s	0.84s
$F(5,[2,\infty))$	6ms	3.76s/2.26s/0.91s	1.64s
$F(3,[1,2])$	70ms	6m5.15s/38.01s/0.22s	9.00s
$F(5,[1,2])$	70ms	>15m	2m6s
$G(5,[0,\infty))$	10ms	3.83s/2.43s/0.05s	0.75s
$G(5,[0,2])$	10ms	4.01s/2.51s/0.10s	0.82s
$G(5,[2,\infty))$	9ms	4.06s/2.47s/0.04s	0.85s
$G(5,[1,2])$	15ms	7.81s/2.99s/0.09s	1.12s
$\mu(1)$	13ms	-	0.39s
$\mu(2)$	21ms	-	2.33s
$\mu(3)$	76ms	-	15.77s
$\mu(4)$	87ms	-	2m23s

Formula	MightyL	LTSmin	Uppaal
$U(5,[0,\infty))$	16ms	1.90s/1.44s/0.05s	0.41s
$U(5,[0,2])$	8ms	2.08s/1.54s/0.06s	0.42s
$U(5,[2,\infty))$	8ms	2.08s/1.53s/0.09s	0.52s
$U(3,[1,2])$	49ms	4m0.14s/23.54s/0.09s	4.92s
$U(5,[1,2])$	97ms	>15m	21.80s
$R(5,[0,\infty))$	7ms	1.86s/1.42s/0.03s	0.40s
$R(5,[0,2])$	7ms	1.97s/1.44s/0.03s	0.40s
$R(5,[2,\infty))$	7ms	1.92s/1.42s/0.03s	0.42s
$R(5,[1,2])$	10ms	5.37s/2.16s/0.04s	0.62s
$\theta(1,[100,1000])$	9ms	1.88s/1.74s/0.04s	0.25s
$\theta(2,[100,1000])$	13ms	5.04s/3.17s/0.19s	0.86s
$\theta(3,[100,1000])$	14ms	36.57s/16.27s/3.20s	21.84s
$\theta(4,[100,1000])$	15ms	5m30s/4m18s/2m16s	18m39s

3. Bartocci, E., Bortolussi, L., Nenzi, L.: A temporal logic approach to modular design of synthetic biological circuits. In: CMSB'13. LNCS, vol. 8130, pp. 164–177. Springer (2013)
4. Bouyer, P., Colange, M., Markey, N.: Symbolic optimal reachability in weighted timed automata. In: CAV'16. LNCS, vol. 9779, pp. 513–530. Springer (2016)
5. Brihaye, T., Estiévenart, M., Geeraerts, G.: On MITL and alternating timed automata. In: FORMATS'13. LNCS, vol. 8053, pp. 47–61. Springer (2013)
6. Brihaye, T., Estiévenart, M., Geeraerts, G.: On MITL and alternating timed automata of infinite words. In: FORMATS'14. LNCS, vol. 8711. Springer (2014)
7. D'Souza, D., Matteplackel, R.: A clock-optimal hierarchical monitoring automaton construction for mitl. Research Report 2013-1, IIS (2013), `http://www.csa.iisc.ernet.in/TR/2013/1/lics2013-tr.pdf`
8. Fu, J., Topcu, U.: Computational methods for stochastic control with metric interval temporal logic specifications. In: CDC'15. pp. 7440–7447. IEEE (2015)
9. Gastin, P., Oddoux, D.: Fast LTL to Büchi automata translation. In: CAV'01. LNCS, vol. 2102, pp. 53–65. Springer (2001)
10. Kant, G., Laarman, A., Meijer, J., van de Pol, J., Blom, S., van Dijk, T.: LTSmin: High-performance language-independent model checking. In: TACAS'15. LNCS, vol. 9035, pp. 692–707. Springer (2015)
11. Karaman, S.: Optimal Planning with Temporal Logic Specifications. Master's thesis, Massachussetts Institute of Technology (2009)
12. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. International Journal on Software Tools for Technology Transfer 1(1-2), 134–152 (1997)
13. Maler, O., Nickovic, D., Pnueli, A.: From MITL to timed automata. In: FORMATS'06. LNCS, vol. 4202, pp. 274–289. Springer (2006)
14. Ouaknine, J., Worrell, J.: On the decidability and complexity of metric temporal logic over finite words. Logical Methods in Computer Science 3(1) (2007)
15. Plaku, E., Karaman, S.: Motion planning with temporal-logic specifications: Progress and challenges. AI Communications 29, 151–162 (2016)
16. Pnueli, A.: The temporal logic of programs. In: FOCS'77. pp. 46–57. IEEE (1977)
17. Thierry-Mieg, Y.: Symbolic model-checking using ITS-tools. In: TACAS'15. LNCS, vol. 9035, pp. 231–237. Springer (2015)
18. Zhou, Y., Maity, D., Baras, J.S.: Timed automata approach for motion planning using metric interval temporal logic. Research Report 1603.08246, arXiv (2016)