FL-AT: A Formal Language–Automaton Transmogrifier

Jaime Middleton[*†], **Toryn Q. Klassen**[‡], **Jorge Baier**[†§], **Sheila A. McIlraith**[‡]
[†]Pontificia Universidad Católica de Chile, Santiago, Chile
[§]Millennium Institute for Foundational Research on Data, Chile
[‡]Department of Computer Science, University of Toronto and Vector Institute, Toronto, Canada
[†]{jamiddleton@uc.cl, jabaier@ing.puc.cl} [‡]{toryn,sheila}@cs.toronto.edu

Abstract

Many sequential decision making tasks aim to realize objectives that are temporally extended in nature, involving patterns of properties or behaviours that are realized over time. These so-called non-Markovian objectives are often easily specified using formal languages, such as dialects of linear temporal logic or other regular language specifications. Nevertheless, many state-of-the-art planning and reinforcement learning systems require such goals, preferences, or reward functions to be represented in automata-like structures. FL-AT is a formal language–automaton transmogrifier, a tool designed to accept as input temporally extended objectives specified in a diversity of formal languages and translate them to semantically equivalent automata-like structures appropriate for planning or reinforcement learning. FL-AT is offered as a web service, a local program, and as a RESTful API. It is intended as a community tool that will house translators developed by different researchers.

1 Motivation

Temporally extended qualities including goals, preferences, assumptions, and rewards have been utilized in planning, reactive synthesis, and reinforcement learning to capture sequential decision making objectives involving patterns of behaviour that are realized over time. In automated planning such objectives include safety and/or liveness properties; e.g., *"Always go to the charging station when the battery is low"* or *"Go to the bank and then go home"*.

Within the ICAPS planning community many such objectives can be specified in PDDL3.0 (Gerevini et al. 2009), which incorporates a restricted subset of Linear Temporal Logic (LTL) (Pnueli 1977) interpreted over finite traces. In reactive synthesis, controller objectives, and agent and environment behaviours, are also commonly specified in LTL (Pnueli and Rosner 1989; De Giacomo and Vardi 2013). In reinforcement learning (RL), non-Markovian reward functions have been encoded as regular languages including with LTL and/or directly as automata-like structures (e.g., Toro Icarte et al. 2018a; 2018b; Brafman, De Giacomo, and Patrizi 2018; Camacho et al. 2019) .

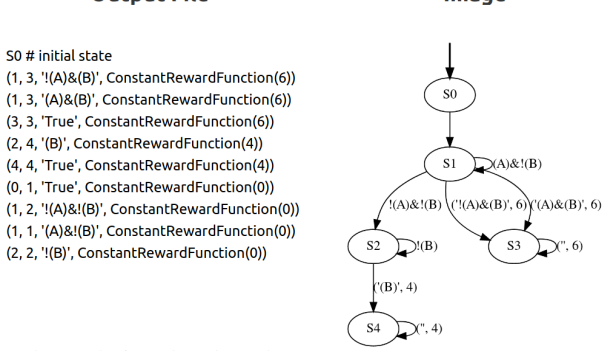
Early deterministic planning systems that planned with temporally extended properties (e.g., Bacchus and Kabanza 2000) and preferences (e.g., Bienvenu, Fritz, and McIlraith 2006; 2011) used LTL or other regular language specifications directly. However deterministic planners that represented temporally extended objectives as automata combined with heuristic search turned out to be orders of magnitude faster (e.g., Baier and McIlraith 2006b; 2006a; Baier et al. 2008; Baier, Bacchus, and McIlraith 2009). Since then, a number of planners have followed this trend, planning with temporally extended objectives specified using automata for deterministic goals (e.g., Patrizi et al. 2011; Torres and Baier 2015) and preference-based planning (Baier, Bacchus, and McIlraith 2009; Coles and Coles 2011), Fully Observable Non-Deterministic (FOND) planning (e.g., Patrizi, Lipovetzky, and Geffner 2013; Camacho et al. 2017b), reactive synthesis (e.g., Zhu et al. 2017; Camacho et al. 2018a; Zhu, Pu, and Vardi 2019), and Markov Decision Processes (e.g., Camacho et al. 2017a; 2018b; Brafman, De Giacomo, and Patrizi 2018).

Most recently, so called *reward machines* (RM) (Toro Icarte et al. 2018b; Camacho et al. 2019; Toro Icarte et al. 2020) have emerged as an innovation for reward specification in RL. A reward machine is a Mealy (or Moore) machine where the input alphabet consists of the set of possible propositional truth assignments (over a given vocabulary), and the output alphabet is reward functions. Unlike an automaton, which just accepts or rejects a trace, an RM outputs rewards for each step. Camacho et al. (2019) described how RMs can be constructed by specifying temporally extended reward specifications as formulae in a diversity of formal languages, associating each formula with a reward (as in (Bacchus, Boutilier, and Grove 1996)), and then translating them to an RM.

A variety of tools exist for translating formal languages into some forms of automata, including Spot (Duret-Lutz et al. 2016), MONA (Elgaard, Klarlund, and Møller 1998), LTLf2DFA (Fuggitti 2019), Lisa (Bansal et al. 2020), and translators outlined in papers above. The growing interest in the use of automata in planning and RL motivated the development of FL-AT, which is the first tool supporting RMs as an output, and is intended as a vessel for the diversity of

[*]Work was done while the author was visiting Univ. of Toronto.

Figure 1: Snapshot of the Web Application: translating the $\mathsf{LTL_f}$ formulas aUb (a until b, with a reward of 2) and Fb (eventually b, with a reward of 4) to a Reward Machine.



translators being developed.

2 The FL-AT Transmogrifier

FL-AT is a formal language–automaton transmogrifier, a tool designed to accept as input temporally extended objectives specified in a diversity of formal languages and to translate them to semantically equivalent automata-based structures appropriate for planning and RL.

2.1 Formal Languages and Automata

FL-AT exploits the well-established correspondence between regular languages (including $\mathsf{LTL_f}$ formulas, finite Linear Dynamic Logic ($\mathsf{LDL_f}$) formulas, and regular expressions) and deterministic finite state automata (DFAs) that accept the same languages. To discuss temporally extended properties, we need the notion of a *trace*, a sequence of truth assignments. Languages like LTL can be interpreted over either infinite or finite traces, describing infinite or finite behaviours. See (De Giacomo, De Masellis, and Montali 2014) for more information on LTL and $\mathsf{LTL_f}$. Currently, FL-AT focuses on languages interpreted over finite traces. For infinite traces, any LTL formula corresponds to a *Büchi* automaton.

2.2 Functionality

FL-AT is offered as a web service, a local program, and as a RESTful API. It is intended as a community tool that will house translators developed by different researchers. It is designed to be usable by many different users from newcomer to expert. With this in mind, we opted for a clean interface with the major options available. We focused our efforts on some of the most commonly used input languages, Linear Temporal Logic on finite traces ($\mathsf{LTL_f}$), LTL-based constraints with PDDL3.0 syntax, Past Linear Temporal Logic (PLTL) and Regular Expressions (RE). Each of these has a specific method of translation, but all can be converted to any of the output formats we provide.

Input representation: We currently support formulae expressed in $\mathsf{LTL_f}$, PLTL, RE, or LTL-based constraints with PDDL3.0 syntax.

Output representation: The final translation can be either be returned as one DFA for each formula, or processed and converted into a reward machine (RM).

Output formats: There are several possible formats to return the final automaton: the DOT format; the HOA format (Babiak et al. 2015), which is standard for the synthesis community; and RMF and RMF2, which are what we call the input formats for the reward machine code by Toro Icarte et al. (2018b) and Toro Icarte et al. (2020), respectively.

We also provide the option to generate an image of the automata and the option to get a reduced expression of the automata. The latter removes all states that do not reach a final state or are unreachable.

We provide three different ways to interact with FL-AT.

Web application: The web application allows for fast and easy interactive tinkering. It is ideal for newcomers to start learning the basics and to get familiar with the functionality of the system. It also doubles as a really good platform for testing ideas due to its availability and ease of use. Figure 1 showcases the results after translating the $\mathsf{LTL_f}$ formulas aUb and Fb (which denotes "a until b" and "eventually b" respectively) into an RM.

Local program: The local program version of FL-AT allows for the most control over all the nuances of the application, and custom modifications if necessary.

RESTful API: The RESTful API provides an easy option to programmatically perform a number of translations without the necessity to download code (or unnecessary dependencies), and with almost all the functionality of the Local program option.

Technical details There are several algorithmic options for translating $\mathsf{LTL_f}$ to automata. Given the availability and ease of use of MONA (Elgaard, Klarlund, and Møller 1998), we decided to reformulate the $\mathsf{LTL_f}$ formulae as an intermediary representation that MONA can understand and translate to a Deterministic Finite Automaton (DFA). Following the same idea as LTL, PLTL is also translated to an intermediary representation and then processed through MONA. The intermediary reformulation of the $\mathsf{LTL_f}$ and PLTL formulae is based on the work of Zhu, Pu, and Vardi (2019) and Fuggitti (2019); both were reformulated as WS1S.

3 Summary and Future Work

We presented FL-AT, a formal language to automaton translation tool that compiles many of the most popular temporally extended objective specification languages, via an easy-to-use interface. This tool is available with three different interfaces, as a local program, a RESTful API and in a web-based version that supports easy prototyping and testing. FL-AT currently supports four specification languages with plans to continue to increase the number of supported languages.

References

Babiak, T.; Blahoudek, F.; Duret-Lutz, A.; Klein, J.; Křetínský, J.; Müller, D.; Parker, D.; and Strejček, J. 2015. The Hanoi Omega-Automata format. In *Computer Aided Verification - 27th International Conference, CAV 2015*, 479–486.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence* 116(1-2):123–191.

Bacchus, F.; Boutilier, C.; and Grove, A. J. 1996. Rewarding behaviors. In *AAAI*, 1160–1167.

Baier, J. A., and McIlraith, S. A. 2006a. Planning with first-order temporally extended goals using heuristic search. In *AAAI*, 788–795.

Baier, J. A., and McIlraith, S. A. 2006b. Planning with temporally extended goals using heuristic search. In *ICAPS*, 342–345.

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence* 173(5-6):593–618.

Baier, J. A.; Fritz, C.; Bienvenu, M.; and McIlraith, S. 2008. Beyond classical planning: Procedural control knowledge and preferences in state-of-the-art planners. In *AAAI, Nectar Track*, 1509–1512.

Bansal, S.; Li, Y.; Tabajara, L. M.; and Vardi, M. Y. 2020. Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications. In *AAAI*, 9766–9774.

Bienvenu, M.; Fritz, C.; and McIlraith, S. 2006. Planning with qualitative temporal preferences. In *KR*, 134–144.

Bienvenu, M.; Fritz, C.; and McIlraith, S. A. 2011. Specifying and computing preferred plans. *Artificial Intelligence* 175(7–8):1308–1345.

Brafman, R. I.; De Giacomo, G.; and Patrizi, F. 2018. LTLf/LDLf Non-Markovian Rewards. In *AAAI*, 1771–1778.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2017a. Non-Markovian rewards expressed in LTL: guiding search via reward shaping. In *SOCS*, 159–160.

Camacho, A.; Triantafillou, E.; Muise, C.; Baier, J. A.; and McIlraith, S. A. 2017b. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018a. Finite LTL synthesis as planning. In *ICAPS*.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2018b. Non-Markovian rewards expressed in LTL: Guiding search via reward shaping (extended version). In *GoalsRL, a workshop collocated with ICML/IJCAI/AAMAS*.

Camacho, A.; Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2019. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, 6065–6073.

Coles, A., and Coles, A. 2011. LPRPG-P: relaxed plan heuristics for planning with preferences. In *ICAPS*.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, 854–860.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *AAAI*, 1027–1033.

Duret-Lutz, A.; Lewkowicz, A.; Fauchille, A.; Michaud, T.; Renault, E.; and Xu, L. 2016. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *ATVA*, volume 9938 of *LNCS*, 122–129. Springer.

Elgaard, J.; Klarlund, N.; and Møller, A. 1998. MONA 1.x: new techniques for WS1S and WS2S. In *Proc. 10th International Conference on Computer-Aided Verification, CAV '98*, volume 1427 of *LNCS*, 516–520. Springer-Verlag.

Fuggitti, F. 2019. LTLf2DFA. Version 1.0.0.post0. Zenodo. https://doi.org/10.5281/zenodo.3888410.

Gerevini, A. E.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5-6):619–668.

Patrizi, F.; Lipovetzky, N.; De Giacomo, G.; and Geffner, H. 2011. Computing infinite plans for LTL goals using a classical planner. In *IJCAI*, 2003–2008.

Patrizi, F.; Lipovetzky, N.; and Geffner, H. 2013. Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *IJCAI*.

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, 179–190.

Pnueli, A. 1977. The temporal logic of programs. In *FOCS*, SFCS '77, 46–57.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2018a. Teaching multiple tasks to an RL agent using LTL. In *AAMAS*, 452–461.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2018b. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2112–2121.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2020. Reward machines: Exploiting reward function structure in reinforcement learning.

Torres, J., and Baier, J. A. 2015. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *IJCAI*, 1696–1703.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. Symbolic LTLf synthesis. In *IJCAI*, 1362–1369.

Zhu, S.; Pu, G.; and Vardi, M. Y. 2019. First-order vs. second-order encodings for LTL_f-to-automata translation. In *Theory and Applications of Models of Computation - 15th Annual Conference, TAMC 2019*, 684–705.