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 objectives (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 Gefen 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), LTL2DFA (Fuggatti 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.
We currently support formulae ex-
any of the output formats we provide.
straints with PDDL3.0 syntax, Past Linear Temporal Logic on finite traces (f
ear Temporal Logic on finite traces (f
ommunally used input languages, Lin-
deficiency, and with almost all the functionality of the
local program version of FL-AT al-
ontrol 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 LTL\textsubscript{f} to automata. Given the availability and ease of use of MONA (Elgaard, Klarlund, and Møller 1998), we decided to reformulate the LTL\textsubscript{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 LTL\textsubscript{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


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.


