Lydia: A Tool for Compositional LTL$_f$/LDL$_f$ Synthesis

Marco Favorito, Giuseppe De Giacomo
Department of Computer, Control and Management Engineering,
Sapienza University of Rome,
Via Ariosto, 25, 00185 Rome RM, Italy
{favorito,degiacomo}@diag.uniroma1.it

Abstract

This demonstration describes `Lydia`, a tool for the translation of Linear Temporal Logic on Finite Traces (LTL$_f$) and Linear Dynamic Logic on Finite Traces (LDL$_f$) into Deterministic Finite Automata (DFA), and for performing LTL$_f$/LDL$_f$ synthesis. `Lydia` implements a novel *compositional* technique to handle such transformation, and the contribution has been accepted as a publication to the main track. This demo is a companion of the accepted publication. Notably, it is the first tool that provides the transformation from LDL$_f$ to DFA. The tool can be used in different ways: as a command-line tool, as a C++ library, through an exposed web service, with a web app, and using the Python programming language.[1]

Introduction

`Lydia` is a software library that implements the transformation formot Linear Temporal Logic on Finite Traces (LTL$_f$) and Linear Dynamic Logic on Finite Traces (LDL$_f$) into Deterministic Finite Automata. As a side-effect, it can be used to perform LTL$_f$/LDL$_f$ synthesis by feeding the DFA in input to `Syft+` (Zhu et al. 2017; Bansal et al. 2020). The technique, that we call *compositional*, has been studied in (De Giacomo and Favorito 2021) and has been accepted as a publication at ICAPS'21 in the main track[2]. The experimental result show competitiveness and sometime advantages with respect to competitive tools for the LTL$_f$ fragment, and it is the first scalable tool for the LDL$_f$-to-DFA transformation.

The library

The main library has been developed in C++, and it is published as open-source software on GitHub: https://github.com/whitemech/lydia. It requires the following dependencies:

- Flex & Bison[3], for parsing a formula written in a specific format (more later);
- CUDD 3.0 (Somenzi 2004), a decision diagram package;
- MONA DFA library (Henriksen et al. 1995). MONA is a tool for translation of Monadic Second-Order logic to DFAs; `Lydia` only uses its DFA library.
- `Syft+` (Zhu et al. 2017), for doing synthesis once the DFA of the LTL$_f$/LDL$_f$ formula has been computed.
- Graphviz, for automata rendering purposes.

How to use `Lydia`

The tool can be accessed in several ways: *(i)* as a CLI tool, *(ii)* as a C++ library, *(iii)* as a REST Web service, *(iv)* as a web app, *(v)* using the Python library `Logaut`.

`Lydia` as a CLI tool

`Lydia` supports a command-line interface. The CLI tool, currently at version v0.1.1, can be downloaded from GitHub[4]. Please refer to the README for its usage[5]. Note, however, that it requires the installation of the dependencies, as those are linked dynamically.

The logic formulae, either LTL$_f$ or LDL$_f$, are expected to be written in a textual format following a specific grammar. One of the authors wrote a tool-agnostic standard specification for such grammar (Favorito 2020), available at this link[6], and `Lydia` expects the LTL$_f$ formula or the LDL$_f$ formula to be written in that format.

We have also prepared a Docker image[7] with all the dependencies installed, and ready to use. The Docker image tag is `whitemech/lydia`, and can be pulled as usual using the `docker` CLI tool.

`Lydia` as C++ library

In the same GitHub repository there is the C++ source code of the project. It is possible to use the functionalities as a C++ library, using the available APIs. In the future we would like to make this usage pattern more easily accessible, and to fully document the public APIs, whose documentation is currently lacking.

[1] Accepted as Demo at ICAPS 2021: https://icaps21.icaps-conference.org/demos/

[2] https://ojs.aaai.org/index.php/ICAPS/article/view/15954

[3] https://www.oreilly.com/library/view/flex-bison/9780596805418/ch01.html

[4] https://github.com/whitemech/lydia/releases/tag/v0.1.1

[5] https://github.com/whitemech/lydia/tree/v0.1.1#readme

[6] https://marcofavorito.me/tl-grammars

[7] https://hub.docker.com/repository/docker/whitemech/lydia/general

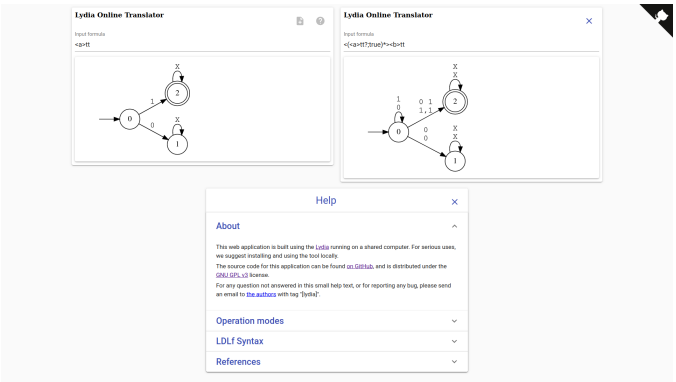

Figure 1: A screenshot of the web application.

`Lydia` as a REST Web service

We have developed RESTful APIs that allow to access `Lydia`'s functionalities via HTTP requests. The source code of the API server is published on GitHub: https://github.com/whitemech/lydia-web-app/tree/master/server. The APIs are specified using the OpenAPI 3.0 standard, and are accessible at this link.

The URL to access the APIs is http://lydia.whitemech.it/api, and the documentation of the APIs is at http://lydia.whitemech.it/api/ui. The REST APIs are written in the OpenAPI format, and are accessible in the The most important method is `/api/automaton`, which allows to call the `Lydia` tool remotely; the input is a formula written in the standard format described earlier, and the output is the automaton, that can be obtained in different formats: JSON, DOT and SVG. Due to space constraint, we cannot provide more details on these methods and we refer to the API documentation generated from the OpenAPI specification (link above).

`Lydia` as a web application

We also developed a Web application to easily access the REST APIs described above. The web app is deployed at this link: http://lydia.whitemech.it. Figure 1 depicts a screenshot of the application. The web application has been developed in React.js, a well-known framework for front-end development.

The application has a text box in which the user can specify the LDL_f formula in the standard format specified in the aforementioned standard (Favorito 2020). After pressing `Enter`, the web app sends the request to the REST API `/api/automaton` and in case of success it returns the automaton in SVG format, and updates the UI to show the result. The user can open another window to use the app, which is useful in case of comparison between two formulae. There is also a short documentation and information on the usage of the app.

Among the possible improvements there are:

- Provide a way to switch between LTL_f and LDL_f (now only LDL_f);
- Allow to download the automaton in different formats, e.g. HOA (Babiak et al. 2015);
- Manipulate, zoom-in and zoom-out the printed automaton;
- Allow the user to submit a synthesis task, which is possible since it is supported by the `Lydia` software library.

`Lydia` using `Logaut`

One of the authors has also developed a series of Python libraries to make it easier to access the `Lydia`'s functionalities.

Pylogics[8] is a Python package that allows manipulation and parsing of several logic formalisms: propositional logic, LTL_f, LDL_f, and PLTL. For the formalisms covered by the standard (Favorito 2020), the parser accepts formulae written in that format. Pythomata[9] is a Python library to create and manipulate finite automata, in particular DFAs. `Logaut`[10] is a Python library bridges Pylogics and Pythomata implementing wrappers to back-ends to perform translation from temporal logic formalisms to DFAs.

We have implemented the back-end component for `Lydia`, which makes `Lydia` very easy to use from the Python language, e.g.:

```
from logaut import ltl2dfa
from pylogics.parsers import parse_ltl
formula = parse_ltl("F(a)")
dfa = ltl2dfa(formula, backend=''lydia")
```

Acknowledgements

This work is partially supported by the ERC Advanced Grant WhiteMech (No. 834228) and by the EU ICT-48 2020 project TAILOR (No. 952215).

References

Babiak, T.; Blahoudek, F.; Duret-Lutz, A.; Klein, J.; Křetínskỳ, J.; Müller, D.; Parker, D.; and Strejček, J. 2015. The Hanoi omega-automata format. In *CAV*.

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

De Giacomo, G.; and Favorito, M. 2021. Compositional Approach to Translate LTLf/LDLf into Deterministic Finite Automata. In *ICAPS (to appear)*, volume 14.

Favorito, M. 2020. A Standard Grammar for Temporal Logics on Finite Traces - v0.2.0. *CoRR* abs/2012.13638.

Henriksen, J. G.; Jensen, J.; Jørgensen, M.; Klarlund, N.; Paige, R.; Rauhe, T.; and Sandholm, A. 1995. *Mona: Monadic second-order logic in practice*, 89–110.

Somenzi, F. 2004. CUDD: CU decision diagram package. *Software available from http://vlsi. colorado. edu* .

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

[8] https://whitemech.github.io/pylogics/

[9] https://github.com/whitemech/pythomata

[10] https://github.com/whitemech/logaut