The Transformation Logics

Alessandro Ronca
University of Oxford
alessandro.ronca@cs.ox.ac.uk

Abstract

We introduce a new family of temporal logics designed to finely balance the trade-off between expressivity and complexity. Their key feature is the possibility of defining operators of a new kind that we call *transformation operators*. Some of them subsume existing temporal operators, while others are entirely novel. Of particular interest are transformation operators *based on semigroups*. They enable logics to harness the richness of semigroup theory, and we show them to yield logics capable of creating *hierarchies of increasing expressivity and complexity* which are non-trivial to characterise in existing logics. The result is a genuinely novel and yet unexplored *landscape of temporal logics*, each of them with the potential of matching the trade-off between expressivity and complexity required by specific applications.

1 Introduction

We introduce the *Transformation Logics*, a new family of temporal logics designed to finely balance the trade-off between expressivity and complexity. Their key feature is the possibility of defining operators of a new kind that we call *transformation operators*. They capture patterns over sequences, and they can be thought of as a generalisation of temporal operators. The subclass of transformation operators based on *finite semigroups* is of particular interest. Such *semigroup-like* operators suffice to capture all regular languages, and remarkably they allow for creating *hierarchies of increasing expressivity and complexity* which are non-trivial to define in existing logics. The base level of the hierarchies is obtained using the operator defined by the *flip-flop monoid*. The other levels are obtained introducing operators based on *simple groups*—the building blocks of all groups. Simple groups have been systematically classified into a finite number of families, cf. (Gorenstein, Lyons, and Solomon 2018). The classification provides a compass in the landscape of groups, and a roadmap in the exploration of temporal logics, as it is made clear by our results.

Our motivation arises from the usage of temporal logics in AI. They are used in *reinforcement learning* to specify reward and dynamics functions (Icarte et al. 2018; De Giacomo et al. 2020); in *planning* for describing temporally-extended goals (Camacho et al. 2017; Bonassi et al. 2023); in *stream reasoning* to express programs with the ability of referring to different points of a stream of data (Beck, Dao-Tran, and Eiter 2018; Ronca et al. 2022; Walega et al. 2023).

In the above applications, the required trade-off between expressivity and complexity depends on the specific case at hand. When the basic expressivity of the star-free regular languages suffices, one can employ logics such as Past LTL (Manna and Pnueli 1991) and LTLf (De Giacomo and Vardi 2013). In all the other cases, one needs to resort to more expressive logics. The existing extensions of the above logics have the expressivity of all regular languages, cf. ETL (Wolper 1983) and LDLf (De Giacomo and Vardi 2013). This is a big leap in expressivity, which may incur an unnecessarily high computational complexity.

We show next two examples where the required expressivity lies in fragments between the star-free regular languages and all regular languages. These intermediate fragments can be precisely characterised in the Transformation Logics.

Example 1. *An agent is assigned a task that can be completed multiple times. We receive an update every minute telling us whether the agent has completed the task in the minute that has just elapsed. We need to detect whether the agent has completed the task at least once on every past day.*

The example describes a periodic pattern, which is beyond the star-free regular languages. It requires to count minutes modulo $24 * 60 = 1440$ in order to establish the end of every day. This can be expressed in the Transformation Logics using a transformation operator defined by the cyclic group C_{1440}; or alternatively using three transformation operators defined the cyclic groups C_2, C_3, and C_5, respectively. Cyclic group operators yield an ability to capture many useful periodic patterns. At the same time, they belong to the special class of *solvable group operators*, which enjoys good properties such as a more favourable computational complexity compared to larger classes of operators.

The next one is an example where solvable group operators do not suffice, and we need to resort to *symmetric group operators*, incurring a higher computational complexity.

Example 2. *A cycling race with n participants takes place, and we need to keep track of the live ranking. At each step an overtake can happen, in which case it is communicated to us in the form $(i, i+1)$ meaning that the cyclist in position i has overtaken the one in position $i+1$. We know the initial*

ranking, and we need to keep track of the live ranking.

The ranking in the example corresponds to the symmetric group S_n, which is not solvable for $n \geq 5$. The example can be specified in any Transformation Logic featuring a transformation operator defined by the group S_n.

2 Main Contributions

We introduce the Transformation Logics, prove a series of expressivity and complexity results, and analyse their relationship with Past LTL.

The Transformation Logics. We introduce the Transformation Logics, providing a formal syntax and semantics. Their main characteristic is the transformation operators. The operators are very general, as we demonstrate through a series of concrete examples. We develop a systematic approach in defining operators, based on semigroup theory and algebraic automata theory, cf. (Ginzburg 1968; Arbib 1969; Dömösi and Nehaniv 2005). We identify *prime operators* that can capture all finite operators. For them, we prove a series of expressivity and complexity results.

Expressivity results. We show there exists one operator, defined by the *flip-flop monoid*, which yields the expressivity of the *star-free regular languages*; as one keeps adding operators based on *cyclic groups* of prime order, the expressivity increases, up to capturing all languages that can be captured using *solvable group operators*; the expressivity of all regular languages is reached by adding the other prime operators, that can be defined by choosing groups from *the classification of finite simple groups*, cf. (Gorenstein, Lyons, and Solomon 2018).

Complexity results. We focus on the *evaluation problem*, and we show three sets of results. First, we show any Transformation Logic can be evaluated in *polynomial time*, whenever its operators can be evaluated in polynomial time. Second, for two notable families of operators, we show that polynomial-time evaluation is possible even when they are represented compactly. Third, we focus on the *data complexity* of evaluation showing that it corresponds to the three circuit complexity classes $\mathrm{AC}^0 \subsetneq \mathrm{ACC}^0 \subseteq \mathrm{NC}^1$ when we include (i) only the flip-flop operator, (ii) also cyclic operators, and (iii) all operators.

Relationship with Past LTL. We show how Past LTL formulas can be easily translated into the core Transformation Logic featuring the flip-flop operator. We also show how to extend Past LTL with transformation operators.

3 Significance of the Results

From a theoretical point of view, the Transformation Logics are a genuinely novel family of temporal logics with an explicit connection to semigroup theory. This connection allowed us to prove our expressivity and complexity results, and we are confident it will allow for deriving many more results out of the richness of semigroup theory. This line of research has the potential to provide a broader understanding of temporal logics in general, e.g., by identifying new fundamental temporal patterns defined by groups. Furthermore, the Transformation Logics allow one to easily characterise fragments of the regular languages beyond star-free, which have gone nameless so far. This ease of characterisation will favour their exploration.

From a practical point of view, the Transformation Logics allow for matching the trade-off between expressivity and complexity required by specific applications, in an innovative way compared to existing formalisms.

4 Relevance to KR

The Transformation Logics are a valuable addition to the existing set of temporal KR formalisms. The paper establishes a series of expressivity and complexity results which are of immediate interest to both KR practitioners and theoreticians. The Transformation Logics will be an interesting subject of study for the KR community.

References

Arbib, M. 1969. *Theories of Abstract Automata*. Automatic Computation. Prentice-Hall.

Beck, H.; Dao-Tran, M.; and Eiter, T. 2018. LARS: A logic-based framework for analytic reasoning over streams. *Artif. Intell.* 261.

Bonassi, L.; De Giacomo, G.; Favorito, M.; Fuggitti, F.; Gerevini, A.; and Scala, E. 2023. Planning for temporally extended goals in pure-past linear temporal logic. In *ICAPS*.

Camacho, A.; Triantafillou, E.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2017. Non-deterministic planning with temporally extended goals. In *AAAI*.

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

De Giacomo, G.; Favorito, M.; Iocchi, L.; Patrizi, F.; and Ronca, A. 2020. Temporal logic monitoring rewards via transducers. In *KR*.

Dömösi, P., and Nehaniv, C. L. 2005. *Algebraic theory of automata networks: An introduction*. SIAM.

Ginzburg, A. 1968. *Algebraic Theory of Automata*. Academic Press.

Gorenstein, D.; Lyons, R.; and Solomon, R. 2018. *The Classification of the Finite Simple Groups, Number 8*. American Mathematical Soc.

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

Manna, Z., and Pnueli, A. 1991. Completing the temporal picture. *Theor. Comput. Sci.* 83.

Ronca, A.; Kaminski, M.; Grau, B. C.; and Horrocks, I. 2022. The delay and window size problems in rule-based stream reasoning. *Artif. Intell.* 306.

Walega, P. A.; Kaminski, M.; Wang, D.; and Cuenca Grau, B. 2023. Stream reasoning with DatalogMTL. *J. Web Semant.* 76.

Wolper, P. 1983. Temporal logic can be more expressive. *Inf. Control.* 56.