Computational methods for Dynamic Answer Set Programming

Susana Hahn

University of Potsdam, Germany

`hahnmartinlu@uni-potsdam.de`

1 Introduction

In our daily lives, we commonly encounter problems that require reasoning with time. For instance, planning our day, determining our route to work, or scheduling our tasks. We refer to these problems as 'dynamic' because they involve movement and change over time, which sometimes includes metric information to express deadlines and durations. For example, getting to the office within the next hour while ensuring that you have had breakfast beforehand. In industrial settings, the complexity of these problems increases significantly. We see this complexity in scenarios such as train scheduling, production sequencing, and many other operations. Therefore, modeling these large-scale problems requires addressing both dynamic aspects and complex combinatorial optimizations, which is a significant challenge.

Semantic formalisms for expressing dynamic knowledge have been around for many years. Dynamic logics provide the means to describe ordered events, making them powerful tools for domains that need to capture actions and changes. These formalisms are typically approached from a theoretical perspective, and the systems built around them tend to be single-purpose, lacking the flexibility to fully model complex problems. This creates a need for systems that offer comprehensive modeling capabilities for dynamic domains, efficient solving techniques, and tools for industrial integration. Answer Set Programming (ASP) is a prime candidate for solving knowledge-intensive search and optimization problems. This declarative approach offers a rich modeling language and effective solvers. However, ASP is primarily suited for static knowledge and lacks built-in solutions for managing dynamic knowledge.

The overall goal of this research project is to extend ASP into a general-purpose technology for dynamic domains. The first step is to develop the logical foundations for enhancing ASP's base logic with concepts from dynamic, temporal, and metric logic. Significant progress in this area has already been made by previous efforts of our research group, providing a solid foundation for further development. We need to identify fragments of these languages that offer the necessary modeling power for our target dynamic problems while maintaining properties that allow for formalization and translation using various approaches. These approaches include using different structures, such as automata and other transition systems, and compiling durations into other formalisms, such as linear constraints. Implementing these approaches will leverage existing technology in the ASP system *clingo* and its surrounding tools. This project will employ advanced programming techniques in ASP to create effective systems for modeling complex dynamic problems. Additionally, we aim to add interactive capabilities to these systems to benefit both modelers and end users. We anticipate that incorporating these features into ASP will enhance users' ability to model dynamic problems and perform various reasoning tasks.

2 Background

Dynamic logics

One of the most commonly used temporal logics [21] is Linear-time Temporal Logic (LTL) [35]. LTL provides modal operators to express temporal properties such as ○ (next), □ (always), and ◇ (eventually). LTL can also be defined in terms of Dynamic Logic (DL) [27], which allows for writing regular expressions over (infinite) traces and mixing declarative with procedural specifications. These types of specifications are also targeted by action languages such as GOLOG [40], which is based on situation calculus. Another interesting approach is Metric Temporal Logic (MTL) [38], which allows measuring time differences between events. This measurement is done by assigning a time value to states. Metric Logic can be used in different applications such as scheduling [41], routing [37], and more [31]. Originally, these temporal formalisms were investigated for infinite traces. However, in the past decade, the case of finite traces $\cdot_f$ has gained interest, as it aligns with a large range of AI applications and constitutes a computationally more feasible variant. The introduction of LTL_f and Linear Dynamic Logic over finite traces (LDL_f) [19] served as a stepping stone to define the syntax and classical semantics under this restriction.

There are several tasks addressed by these formalisms and other action theories. The most commonly known are satisfiability checking, model checking, and synthesis [47, 48, 20, 49]. Furthermore, other more elaborate tasks closer to real-world scenarios include reactive control [8], diagnostics [33], planning [4, 25, 5], and verification.

Many of these tasks are solved by translating complex constructs into simpler ones, for instance, by reducing MTL into LTL [31]. Another very common strategy for addressing these problems is mapping them into automata. This automata-theoretic approach involves constructing an automaton that accepts exactly the models of a dynamic formula. This relationship has been extensively researched in areas such as satisfiability checking, model checking, planning [4, 25, 5], and synthesis [47, 48, 20, 49]. Non-deterministic finite automata (NFA) [30] and Deterministic Finite Automata (DFA) have been used for finite traces, though they are of exponential size relative to the input formula. To tackle this issue, [19] proposed a translation from LTL_f and LDL_f to a more elaborate but succinct automaton: Alternating Automaton on Finite Words (AFA) [18, 48, 19]. These automata, an adaptation of Alternating Büchi Automata to finite traces, extend NFAs with universal transitions. This translation, however, led to circular definitions for some dynamic formulas and did not include past operators. These issues were addressed in [45], where the authors introduced Automata Linear Dynamic Logic over finite traces ($ALDL_f$) and presented a translation into even more sophisticated automata: Two-Way Alternating Finite Automata (2AFA) [39, 36]. In addition to alternation, this type of automaton allows multiple head movements: stationary, left, and right. More evolved translations from metric logic into automata have also been developed, such as translating MTL into Timed Automata [42].

Answer Set Programming

Answer Set Programming (ASP) [10] is a well-established approach to declarative problem-solving where problems are encoded as logic programs. The combination of its rich modeling language and highly effective solving engines makes ASP a very attractive choice. ASP semantics can be formalized using equilibrium models [44] of the logic of here-and-there (HT) [29]. This logic has also been extended to here-and-there with constraints (HT_c) [17], which introduces difference constraints, a simplified form of linear constraints, into HT.

The ASP system *clingo* [24] is known for its high-performing engines. The system provides various

tools for extending the language and customizing the solving process. *clingo*'s theory language capabilities allow for defining custom syntactic expressions. Additionally, *clingo* offers two methods for capturing new functionalities [34]: meta-programming, which uses a reification feature enabling the expression of new functionalities using ASP, and a sophisticated Python API for manipulating and customizing the system's internal workflow. This customization includes techniques such as multi-shot solving, which allows precise control of the solving process by modularizing the problem. These features have led to the creation of several hybrid ASP systems. In particular, *clingcon* [7] and *clingo*[DL] [32] extend the language of *clingo* with linear constraints using the semantics for HT_c.

Temporal Logic Programming

In the 1980s, Temporal Logic Programming (TLP) emerged [22, 23, 1, 43]. TLP was revised after the appearance of ASP, resulting in what we know as Temporal ASP. The idea is to extend the equilibrium models of HT, to deal with dynamic scenarios. Research began with infinite traces, giving rise to (Linear) Temporal Here-and-There (THT) [3] and (Linear) Dynamic Logic of Here-and-There (DHT) [9], along with their non-monotonic counterparts for temporal stable models, namely Temporal Equilibrium Logic (TEL) [3] and Dynamic Equilibrium Logic (DEL) [14]. The strategy behind these temporal formalisms is to capture time as sequences of HT-interpretations, resulting in an expressive non-monotonic modal logic. This approach allowed the definition of Temporal Logic Programs found in [2].

The temporal operators and semantics of the finite version TEL_f were introduced into the ASP system *clingo* enriching its modeling power and yielding the first temporal ASP solver *telingo* [16]. This system uses the *clingo* capabilities for theory extensions as well as multi-shot solving in an incremental manner. Subsequent work incorporated dynamic operators from DEL_f [14, 13] by unfolding their definitions into TEL_f relying on the introduction of auxiliary atoms (in a Tseitin-style [46]). This technique, however, is dependent on a fixed trace length, and the type of translation makes the final logic program hard to interpret. In [15], TEL was further extended with metric temporal operators constrained by intervals over natural numbers, resulting in Metric Equilibrium Logic (MEL).

3   Contributions and future work

Automata techniques

To this moment, I have pursued different translations of dynamic and temporal logic with finite traces into automata, and implemented them using ASP. In the current status of the project, I have not yet explored the non-monotonic side of temporal reasoning with automata. Even though the semantics I have used so far for the temporal formalisms have been monotonic, I was able to incorporate them in ASP by restricting the dynamic formulas to integrity constraints where their behavior is classical. With this in mind, at the moment, one can only use these formulas to filter plans via a translation into automata, which is one of our primary goals.

The first approach, found in [11], proposes an adaptation of the translation of LDL_f to AFA from [19], which is incorporated into ASP using meta programming in *clingo*. This implementation is solely based on ASP, relying on the theory extension to define the language for LDL_f formulas, and the reified output of *clingo*. This reification corresponds to a linearized representation of the dynamic formula as facts based on the grammar defined for the theory. Then, using an ASP encoding, these facts are translated into a declarative representation of the corresponding alternating automaton. In the full version of this work [12], we explore other existing tools for computing an automaton from a dynamic formula. In order

to employ them, we developed two different translations from LDL$_f$ to Monadic Second Order Logic (MSO). For the implementation, we parsed the formula with *clingo*'s API and called the state-of-the-art system MONA [28] to obtain a DFA, which is then transformed into facts. By having a unified declarative representation to capture the different automata (AFA and DFA), we were able to craft a single encoding for checking the acceptance of the automata.

Following (soon to be published) work presents a novel translation from LDL$_f$ into 2AFA. Leveraging the transitions without head movement provided by these automata, we were able to remove the recursive nature of our old translation, thus eliminating the circular issue carried from [19]. Furthermore, the left head movement allows us to readily refer to time points in the past. These new target automata, nonetheless, represented a formalization challenge since there is a lack of literature available in contrast to simpler automata. This translation was implemented as part of the `adlingo`[1] system. Just like the previous work, the implementation was done using meta programming and theory extensions. Additionally, it integrates the use of *clingraph* [26] to visualize the automata and its runs using an ASP encoding.

Linear constraints to encode durations

My latest work has focused on constructing the foundations of Metric Logic Programs (MLP). With these programs, we plan to abstract the basic modalities and forms needed to model metric problems in ASP. The semantics of these programs are those of MEL, where, by restricting the syntax, we aim to simplify the computation. The first approach for this work has been submitted to ICLP24. As in the automata approach, we are restricting the research to the finite setting for a given (fixed) horizon. This work defines the basic syntax for a MLP in which all rules are universal, meaning that they must hold in every step. This contrasts with the approach used for temporal logic programs in [2], where the rules were separated into initial, dynamic, and final, which facilitates the implementation using incremental solving. For our approach, the removal of this division simplifies the use of meta-programming techniques to define the translation, as well as the overall semantics. The use of meta-programming allows us to define the translations in ASP, making the implementation transparent and modular. A big advantage of this approach is the clear mapping between the translation and the implementation in ASP. As a consequence, it simplifies the proves of correctness and completeness of the translation.

In this first exploration, we restricted the rules of metric logic programs to only use the metric next modality. For instance, with the rule $\circ_{[20..40)} school \leftarrow drive$, we express that *"If I start driving, I must be at the school in the next step, which should happen in 20 to 40 minutes"*. We suspect that the core of our target dynamic problems can be modeled with this restricted fragment. In a nutshell, the first part of this translation represents the state changes and follows the same semantics as in TEL$_f$. The second part accounts for the timed aspect of metric logic. For this part, we explore two approaches: one where the translation is done to HT, and a second one where the target logic is HT$_c$. As a result, we were able to see what we expected: a succinct and performant translation of time into linear constraints. We also observed that our restricted language did allow us to model the transitions and time restrictions, but was not enough to represent the goal conditions of the problems. These conditions usually require more complex metric operators to talk about states that are further away in time, for instance, *"At some point in the next 2 hours, I will be back home"*.

---

[1] https://github.com/potassco/adlingo

3.1 Future work

The quest to conceptualize metric logic programming is far from over. In view of the results from the last project, we have started to craft a translation that handles more metric operators. The translation is planned to follow a Tseitin-style translation like the one in [2]. We want to further investigate HT_c for encoding time, and examine the integration of non-monotonic reasoning and optimization in the timed aspect of MLP. Additionally, we plan to investigate how far ASP can address reactive-dynamic tasks where the user and environment play a role by interacting with the system.

References

[1] M. Abadi & Z. Manna (1989): *Temporal Logic Programming*. Journal of Symbolic Computation 8, pp. 277–295, doi:10.1016/S0747-7171(89)80070-7.

[2] F. Aguado, P. Cabalar, M. Diéguez, G. Pérez, T. Schaub, A. Schuhmann & C. Vidal (2023): *Linear-Time Temporal Answer Set Programming*. Theory and Practice of Logic Programming 23(1), pp. 2–56, doi:10.1017/S1471068421000557.

[3] F. Aguado, P. Cabalar, M. Diéguez, G. Pérez & C. Vidal (2013): *Temporal equilibrium logic: a survey*. Journal of Applied Non-Classical Logics 23(1-2), pp. 2–24, doi:10.1080/11663081.2013.798985.

[4] J. Baier, C. Fritz, M. Bienvenu & S. McIlraith (2008): *Beyond Classical Planning: Procedural Control Knowledge and Preferences in State-of-the-Art Planners*. In D. Fox & C. Gomes, editors: *Proceedings of the Twenty-third National Conference on Artificial Intelligence (AAAI'08)*, AAAI Press, pp. 1509–1512. Available at `https://auld.aaai.org/Library/AAAI/2008/aaai08-251.php`.

[5] J. Baier & S. McIlraith (2006): *Planning with First-Order Temporally Extended Goals using Heuristic Search*. In Y. Gil & R. Mooney, editors: *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI'06)*, AAAI Press, pp. 788–795. Available at `https://www.aaai.org/Library/AAAI/2006/aaai06-125.php`.

[6] M. Balduccini, Y. Lierler & S. Woltran, editors (2019): *Proceedings of the Fifteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'19)*. Lecture Notes in Artificial Intelligence 11481, Springer-Verlag, doi:10.1007/978-3-030-20528-7.

[7] M. Banbara, B. Kaufmann, M. Ostrowski & T. Schaub (2017): *Clingcon: The Next Generation*. Theory and Practice of Logic Programming 17(4), pp. 408–461, doi:10.1017/S1471068417000138.

[8] C. Baral, S. Tran Cao & L. Tuan (2002): *A transition function based characterization of actions with delayed and continuous effects*. In: *KR*, Citeseer, pp. 291–302. Available at `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=d4ae7ddfcdc012e519b473d03dd3c7caffaa09e1`.

[9] A. Bosser, P. Cabalar, M. Diéguez & T. Schaub (2018): *Introducing Temporal Stable Models for Linear Dynamic Logic*. In M. Thielscher, F. Toni & F. Wolter, editors: *Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'18)*, AAAI Press, pp. 12–21. Available at `https://aaai.org/ocs/index.php/KR/KR18/paper/view/18047`.

[10] G. Brewka, T. Eiter & M. Truszczyński (2011): *Answer set programming at a glance*. Communications of the ACM 54(12), pp. 92–103, doi:10.1145/2043174.2043195.

[11] P. Cabalar, M. Diéguez, S. Hahn & T. Schaub (2021): *Automata for Dynamic Answer Set Solving: Preliminary Report*. In: *Proceedings of the Fourteenth Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP'21)*. Available at `https://ceur-ws.org/Vol-2970/aspocpinvited1.pdf`.

[12] P. Cabalar, M. Dieguez, S. Hahn & T. Schaub (2021): *Automata for dynamic answer set solving: Preliminary report*. CoRR abs/2109.01782, doi:10.48550/arXiv.2109.01782 .

[13] P. Cabalar, M. Diéguez, F. Laferriere & T. Schaub (2020): *Implementing Dynamic Answer Set Programming over finite traces*. In G. De Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín & J. Lang,

editors: *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20)*, IOS Press, pp. 656–663, doi:10.3233/FAIA200151.

[14] P. Cabalar, M. Diéguez & T. Schaub (2019): *Towards Dynamic Answer Set Programming over finite traces*. In Balduccini et al. [6], pp. 148–162, doi:10.1007/978-3-030-20528-7_12.

[15] P. Cabalar, M. Diéguez, T. Schaub & A. Schuhmann (2022): *Metric Temporal Answer Set Programming over Timed Traces*. In G. Gottlob, D. Inclezan & M. Maratea, editors: *Proceedings of the Sixteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'22)*, Lecture Notes in Artificial Intelligence 13416, Springer-Verlag, pp. 117–130, doi:10.1007/978-3-031-15707-3_10.

[16] P. Cabalar, R. Kaminski, P. Morkisch & T. Schaub (2019): *telingo = ASP + Time*. In Balduccini et al. [6], pp. 256–269, doi:10.1007/978-3-030-20528-7_19.

[17] P. Cabalar, R. Kaminski, M. Ostrowski & T. Schaub (2016): *An ASP Semantics for Default Reasoning with Constraints*. In R. Kambhampati, editor: *Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*, IJCAI/AAAI Press, pp. 1015–1021, doi:10.5555/3060621.3060762.

[18] A. Chandra, D. Kozen & L. Stockmeyer (1981): *Alternation*. Journal of the ACM 28(1), pp. 114–133, doi:10.1145/322234.322243.

[19] G. De Giacomo & M. Vardi (2013): *Linear Temporal Logic and Linear Dynamic Logic on Finite Traces*. In F. Rossi, editor: *Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI'13)*, IJCAI/AAAI Press, pp. 854–860. Available at `https://www.ijcai.org/Abstract/13/132`.

[20] G. De Giacomo & M. Vardi (2015): *Synthesis for LTL and LDL on Finite Traces*. In Q. Yang & M. Wooldridge, editors: *Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence (IJCAI'15)*, AAAI Press, pp. 1558–1564. Available at `https://ijcai.org/Abstract/15/223`.

[21] S. Demri, V. Goranko & M. Lange (2016): *Temporal Logics in Computer Science: Finite-State Systems*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, doi:10.1017/CBO9781139236119.

[22] M. Fujita, S. Kono, H. Tanaka & T. Moto-Oka (1986): *Tokio: Logic Programming Language Based on Temporal Logic and its Compilation to Prolog*. In E. Shapiro, editor: *Proceedings of the Third International Conference on Logic Programming (ICLP'86)*, Lecture Notes in Computer Science 225, Springer, pp. 695–709, doi:10.1007/3-540-16492-8_119.

[23] D. Gabbay (1987): *Modal and Temporal Logic Programming*. In A. Galton, editor: *Temporal Logics and their Applications*, chapter 6, Academic Press, pp. 197–237.

[24] M. Gebser, R. Kaminski, B. Kaufmann & T. Schaub (2019): *Multi-shot ASP solving with clingo*. Theory and Practice of Logic Programming 19(1), pp. 27–82, doi:10.1017/S1471068418000054.

[25] G. De Giacomo & S. Rubin (2018): *Automata-Theoretic Foundations of FOND Planning for LTLf and LDLf Goals*. In J. Lang, editor: *Proceedings of the Twenty-seventh International Joint Conference on Artificial Intelligence (IJCAI'18)*, ijcai.org, pp. 4729–4735, doi:10.24963/ijcai.2018/657.

[26] S. Hahn, O. Sabuncu, T. Schaub & T. Stolzmann (2024): *Clingraph: A System for ASP-based Visualization*. Theory and Practice of Logic Programming, doi:10.1017/S147106842400005X.

[27] D. Harel, J. Tiuryn & D. Kozen (2000): *Dynamic Logic*. MIT Press, doi:10.1145/568438.568456.

[28] J. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, R. Paige, T. Rauhe & A. Sandholm (1995): *Mona: Monadic Second-Order Logic in Practice*. In E. Brinksma, R. Cleaveland, K. Larsen, T. Margaria & B. Steffen, editors: *Proceedings of the First International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS'95)*, Lecture Notes in Computer Science 1019, Springer-Verlag, pp. 89–110, doi:10.1007/3-540-60630-0_5.

[29] A. Heyting (1930): *Die formalen Regeln der intuitionistischen Logik*. In: *Sitzungsberichte der Preussischen Akademie der Wissenschaften*, Deutsche Akademie der Wissenschaften zu Berlin, pp. 42–56.

[30] J. Hopcroft & J Ullman (1979): *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

[31] U. Hustadt, A. Ozaki & C. Dixon (2020): *Theorem Proving for Pointwise Metric Temporal Logic Over the Naturals via Translations*. Journal of Automated Reasoning 64(8), pp. 1553–1610, doi:10.1007/s10817-020-09541-4.

[32] T. Janhunen, R. Kaminski, M. Ostrowski, T. Schaub, S. Schellhorn & P. Wanko (2017): *Clingo goes Linear Constraints over Reals and Integers*. Theory and Practice of Logic Programming 17(5-6), pp. 872–888, doi:10.1017/S1471068417000242.

[33] S. Jiang & R. Kumar (2004): *Failure diagnosis of discrete-event systems with linear-time temporal logic specifications*. IEEE Transactions on Automatic Control 49(6), pp. 934–945, doi:10.1109/TAC.2004.829616.

[34] R. Kaminski, T. Schaub & P. Wanko (2017): *A Tutorial on Hybrid Answer Set Solving with clingo*. In G. Ianni, D. Lembo, L. Bertossi, W. Faber, B. Glimm, G. Gottlob & S. Staab, editors: *Proceedings of the Thirteenth International Summer School of the Reasoning Web*, Lecture Notes in Computer Science 10370, Springer-Verlag, pp. 167–203, doi:10.1007/978-3-319-61033-7_6.

[35] J. Kamp (1968): *Tense Logic and the Theory of Linear Order*. Ph.D. thesis, University of California at Los Angeles.

[36] C. Kapoutsis & M. Zakzok (2021): *Alternation in two-way finite automata*. Theoretical Computer Science 870, pp. 75–102, doi:10.1016/j.tcs.2020.12.011.

[37] S. Karaman & E. Frazzoli (2008): *Vehicle routing problem with metric temporal logic specifications*. In: *2008 47th IEEE conference on decision and control*, IEEE, pp. 3953–3958, doi:10.1109/CDC.2008.4739366.

[38] R. Koymans (1990): *Specifying Real-Time Properties with Metric Temporal Logic*. Real-Time Systems 2(4), pp. 255–299, doi:10.1007/BF01995674.

[39] R. Ladner, R. Lipton & L Stockmeyer (1984): *Alternating pushdown and stack automata*. SIAM Journal on Computing 13(1), pp. 135–155, doi:10.1137/0213010.

[40] H. Levesque, R. Reiter, Y. Lespérance, F. Lin & R. Scherl (1997): *GOLOG: A Logic Programming Language for Dynamic Domains*. Journal of Logic Programming 31(1-3), pp. 59–83, doi:10.1016/S0743-1066(96)00121-5.

[41] R. Luo, R. Valenzano, Y. Li, C. Beck & S. McIlraith (2016): *Using Metric Temporal Logic to Specify Scheduling Problems*. In C. Baral, J. Delgrande & F. Wolter, editors: *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'16)*, AAAI Press, pp. 581–584. Available at `https://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12909`.

[42] D. Ničković & N. Piterman (2010): *From MTL to Deterministic Timed Automata*. In K. Chatterjee & T. Henzinger, editors: *Proceedings of the Eighth International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'10)*, Lecture Notes in Computer Science, Springer-Verlag, pp. 152–167, doi:10.1007/978-3-642-15297-9_13.

[43] M. Orgun & W. Wadge (1992): *Theory and Practice of Temporal Logic Programming*. In L. Fariñas del Cerro & M. Penttonen, editors: *Intensional Logics for Programming*, chapter 2, Oxford University Press, pp. 21–50, doi:10.1093/oso/9780198537755.003.0002.

[44] D. Pearce (2006): *Equilibrium logic*. Annals of Mathematics and Artificial Intelligence 47(1-2), pp. 3–41, doi:10.1007/s10472-006-9028-z.

[45] K. Smith & M. Vardi (2021): *Automata Linear Dynamic Logic on Finite Traces*. arXiv preprint arXiv:2108.12003.

[46] G. Tseitin (1968): *On the complexity of derivation in the propositional calculus*. Zapiski nauchnykh seminarov LOMI 8, pp. 234–259.

[47] M. Vardi (1995): *An Automata-Theoretic Approach to Linear Temporal Logic*. In F. Moller & G. Birtwistle, editors: *Logics for Concurrency: Structure versus Automata*, Lecture Notes in Computer Science 1043, Springer-Verlag, pp. 238–266, doi:10.1007/3-540-60915-6_6.

[48] M. Vardi (1997): *Alternating Automata: Unifying Truth and Validity Checking for Temporal Logics*. In W. McCune, editor: *Proceedings of the Fourteenth International Conference on Automated Deduction*

(CADE'97), *Lecture Notes in Computer Science* 1249, Springer-Verlag, pp. 191–206, doi:10.1007/3-540-63104-6_19.

[49] S. Zhu, G. Pu & M. Vardi (2019): *First-Order vs. Second-Order Encodings for LTLf-to-Automata Translation*. In T. Gopal & J. Watada, editors: *Proceedings of the Fifteenth Annual Conference on Theory and Applications of Models of Computation (TAMC'19)*, *Lecture Notes in Computer Science* 11436, Springer-Verlag, pp. 684–705, doi:10.1007/978-3-030-14812-6_43.