Reverse Engineering of Temporal Queries with and without LTL Ontologies: First Steps

Extended Abstract

Marie Fortin[1], Boris Konev[1], Vladislav Ryzhikov[2], Yury Savateev[2], Frank Wolter[1] and Michael Zakharyaschev[2,3]

[1]Department of Computer Science, University of Liverpool, UK
[2]Department of Computer Science and Information Systems, Birkbeck, University of London, UK
[3]School of Data Analysis and Artificial Intelligence, HSE University, Moscow, Russia

Abstract

In reverse engineering of database queries, one aims to construct a query from a set of positively and negatively labelled answers and non-answers. The query can then be used to explore the data further or as an explanation of the answers and non-answers. We consider this reverse engineering problem for queries formulated in various fragments of positive linear temporal logic *LTL* over data instances given by timestamped atomic concepts. We focus on the design of suitable query languages and the complexity of the separability problem: 'does there exist a query in the given query language that separates the given answers from the non-answers?'. We deal with both plain *LTL* queries and those that are mediated by ontologies providing background knowledge and formulated in fragments of clausal *LTL*.

Keywords

Reverse engineering of queries, query-by-example, explanation, linear temporal logic, ontology-mediated query, computational complexity.

1. Introduction

Supporting users of databases by constructing a query from examples of answers and non-answers to the query has been a major research area for many years [1]. In the database community, research has focussed on standard query languages such as (fragments of) SQL, graph query languages, and SPARQL [2, 3, 4, 5, 6, 7, 8, 9]. The KR community has been concerned with constructing queries from examples under the open world semantics and with background knowledge given by an ontology [10, 11, 12, 13, 14]. In both cases, the focus has been on general multi-purpose query languages. A fundamental problem that has been investigated by both communities is known as *separability* or *query-by-example*: given sets E^+ and E^- of pairs $(\mathcal{D}, \boldsymbol{d})$ with a database $\mathcal{D}$ and a tuple $\boldsymbol{d}$ in $\mathcal{D}$, and a query language $\mathcal{Q}$, does there exist a query $\boldsymbol{q} \in \mathcal{Q}$ that separates (E^+, E^-) in the sense that $\mathcal{D} \models \boldsymbol{q}(\boldsymbol{d})$ for all $(\mathcal{D}, \boldsymbol{d}) \in E^+$ and $\mathcal{D} \not\models \boldsymbol{q}(\boldsymbol{d})$ for all $(\mathcal{D}, \boldsymbol{d}) \in E^-$ (or $\mathcal{O}, \mathcal{D} \models \boldsymbol{q}(\boldsymbol{d})$ for all $(\mathcal{D}, \boldsymbol{d}) \in E^+$ and $\mathcal{O}, \mathcal{D} \not\models \boldsymbol{q}(\boldsymbol{d})$ for all

DL 2022: 35th International Workshop on Description Logics, August 7–10, 2022, Haifa, Israel
marie.fortin@liverpool.ac.uk (M. Fortin); boris.konev@liverpool.ac.uk (B. Konev); vlad@dcs.bbk.ac.uk (V. Ryzhikov); yury@dcs.bbk.ac.uk (Y. Savateev); wolter@liverpool.ac.uk (F. Wolter); michael@dcs.bbk.ac.uk (M. Zakharyaschev)

$(\mathcal{D}, \boldsymbol{d}) \in E^-$ if an ontology $\mathcal{O}$ is present)?[1] There are various strategies to ensure that the query $\boldsymbol{q}$ is a generalisation of the positive examples and does not overfit. For instance, one can ask for the existence of a small separating query in $\mathcal{Q}$ or one can choose a query language that enforces generalisation by not admitting disjunction. In the latter case, query-by-example is often very hard computationally: it is coNExpTime-complete for conjunctive queries (CQs) over standard relational databases [15, 16] and even undecidable for CQs under $\mathcal{ELI}$ or $\mathcal{ALC}$ ontologies [17].

In many applications, the input data is timestamped and queries are naturally formulated in languages with temporal operators. Taking into account the prohibitive complexity of many query-by-example problems already in the static case, it does not seem wise to start an investigation of the temporal case by considering temporal extensions of standard query languages (which can only lead to computationally even harder problems). Instead, we investigate the simpler but still very useful case in which data, $\mathcal{D}$, is a set of timestamped atomic concepts. Our query languages are positive fragments of linear temporal logic *LTL* with the temporal operators $\Diamond$ (eventually), $\bigcirc$ (next), and U (until) interpreted under the strict semantics [18]. To avoid overfitting, we only consider such fragments without $\vee$. The most expressive query language we deal with, $\mathcal{Q}[\mathsf{U}]$, is thus defined as the set of formulas constructed from atoms using $\wedge$ and U (via which $\bigcirc$ and $\Diamond$ can be defined). The fragments $\mathcal{Q}[\Diamond]$, $\mathcal{Q}[\bigcirc]$, and $\mathcal{Q}[\bigcirc, \Diamond]$ are defined analogously.

Within this temporal setting, we take a broad view of the potential applications of the reverse engineering of queries and the query-by-example problem. On the one hand, there are non-expert end-users who would like to explore data via queries but are not familiar with temporal logic. They usually are, however, capable of providing data examples illustrating the queries they are after. Query-by-example supports such users in the construction of those queries. On the other hand, the positive and negative data examples might come from an application, and the user is interested in possible explanations of the examples. Such an explanation is then provided by a temporal query separating the positive examples from the negative ones. In this case, our goal is similar to recent work on learning linear temporal logic formulas and, more generally, explainable AI [19, 20, 21, 22, 23]. The following example illustrates this point.

Example 1. Imagine an engineer whose task is to explain the behaviour of the monitored equipment (say, why an engine stops) in terms of qualitative sensor data such as 'low temperature', which can be represented by the atomic concept T, 'strong vibration', V, etc. Suppose the engine stopped after the runs $\mathcal{D}_1^+$ and $\mathcal{D}_2^+$ shown below but did not stop after the runs $\mathcal{D}_1^-$, $\mathcal{D}_2^-$, $\mathcal{D}_3^-$, where we assume the runs to start at 0 and measurements to be recorded at $0, 1, 2, \dots$:

$$\mathcal{D}_1^+ = \{T(2), V(4)\}, \mathcal{D}_2^+ = \{T(1), V(3)\}, \mathcal{D}_1^- = \{T(1)\}, \mathcal{D}_2^- = \{V(4)\}, \mathcal{D}_3^- = \{V(1), T(2)\}.$$

The $\Diamond$-query $\boldsymbol{q} = \Diamond(T \wedge \Diamond\Diamond V)$ is true at 0 in the positive data instances $\mathcal{D}_i^+$, false in the negative ones $\mathcal{D}_i^-$, and so provides a possible natural explanation of what could cause the engine failure. The example set $(\{\mathcal{D}_3^+, \mathcal{D}_4^+\}, \{\mathcal{D}_4^-\})$ with

$$\mathcal{D}_3^+ = \{T(1), V(2)\}, \;\; \mathcal{D}_4^+ = \{T(1), T(2), V(3)\}, \;\; \mathcal{D}_4^- = \{T(1), V(3)\}$$

can be explained by the U-query $T \,\mathsf{U}\, V$. Using background knowledge of the domain, we can

[1] If such a $\boldsymbol{q}$ exists, then (E^+, E^-) is often called *satisfiable* w.r.t. $\mathcal{Q}$ and the construction of $\boldsymbol{q}$ is called *learning*.

compensate for sensor failures, which result in incomplete data. To illustrate, suppose that $\bar{\mathcal{D}}_1^+ = \{H(3), V(4)\}$, where H stands for 'heater is on'. If a background ontology $\mathcal{O}$ contains the axiom $\bigcirc H \to T$ saying that a heater can only be triggered by the low temperature at the previous moment, then the same $\boldsymbol{q}$ will separate $\{\bar{\mathcal{D}}_1^+, \mathcal{D}_2^+\}$ from $\{\mathcal{D}_1^-, \mathcal{D}_2^-, \mathcal{D}_3^-\}$ under $\mathcal{O}$. ⊣

The queries used in Example 1 are of a particular 'linear' form and suggest a restriction to *path queries* in which the order of the atoms is fixed and not left open as in $\Diamond A \land \Diamond B$. More precisely, *path* $\bigcirc\Diamond$*-queries* in the class $\mathcal{Q}_p[\bigcirc, \Diamond]$ take the form

$$\boldsymbol{q} = \rho_0 \land \boldsymbol{o}_1(\rho_1 \land \boldsymbol{o}_2(\rho_2 \land \cdots \land \boldsymbol{o}_n\rho_n)), \tag{1}$$

where $\boldsymbol{o}_i \in \{\bigcirc, \Diamond\}$ and ρ_i is a conjunction of atoms; $\mathcal{Q}_p[\Diamond]$ and $\mathcal{Q}_p[\bigcirc]$ restrict $\boldsymbol{o}_i$ to $\{\Diamond\}$ and $\{\bigcirc\}$, respectively; and *path* U*-queries* in the class $\mathcal{Q}_p[\mathsf{U}]$ take the form

$$\boldsymbol{q} = \rho_0 \land (\lambda_1 \,\mathsf{U}\, (\rho_1 \land (\lambda_2 \,\mathsf{U}\, (\dots (\lambda_n \,\mathsf{U}\, \rho_n) \dots)))), \tag{2}$$

where λ_i is a conjunction of atoms or $\bot$. Path queries are motivated by two observations. First, if a query language $\mathcal{Q}$ allows conjunctions of queries, then, dually to the overfitting problem for disjunction, the admission of multiple negative examples becomes trivialised: if queries $\boldsymbol{q}_\mathcal{D}$ separate $(E^+, \{\mathcal{D}\})$ for $\mathcal{D} \in E^-$, then the conjunction $\bigwedge_{\mathcal{D}\in E^-} \boldsymbol{q}_\mathcal{D}$ separates (E^+, E^-). In particular, the query-by-example problem becomes polynomially reducible to its version with a single negative example. This is clearly not the case for path queries.

Example 2. Let $\mathcal{D}_1 = \{A(1), B(2)\}$, $\mathcal{D}_2 = \{B(1), A(2)\}$, $\mathcal{D}_3 = \{A(1)\}$ and $\mathcal{D}_4 = \{B(1)\}$. Then $(\{\mathcal{D}_1, \mathcal{D}_2\}, \mathcal{D}_3)$ and $(\{\mathcal{D}_1, \mathcal{D}_2\}, \mathcal{D}_4)$ are separated in $\mathcal{Q}_p[\Diamond]$ by $\Diamond B$ and $\Diamond A$, respectively; $(\{\mathcal{D}_1, \mathcal{D}_2\}, \{\mathcal{D}_3, \mathcal{D}_4\})$ is separated in $\mathcal{Q}[\Diamond]$ by $\Diamond B \land \Diamond A$, but it is not $\mathcal{Q}_p[\Diamond]$-separable. ⊣

Second, numerous natural types of query classes from applications are represented by path queries. For example, the existence of a *common subsequence* of the positive examples that is not a subsequence of any negative example corresponds to the existence of a separating query in $\mathcal{Q}_p[\Diamond]$ with $\rho_0 = \top$ and $\rho_i \neq \top$ for $i > 0$, and the existence of a *common subword* of the positive examples that is not a *subword* of any negative example corresponds to the existence of a separating query of the form $\Diamond(\rho_1 \land \bigcirc(\rho_2 \land \cdots \land \bigcirc\rho_n))$. The unique characterisability and learnability of path queries is investigated in [24].

Except for $\mathcal{Q}_p[\bigcirc] = \mathcal{Q}[\bigcirc]$ (modulo logical equivalence), no nontrivial inclusion relations hold between the separation capabilities of the query languages introduced above, as illustrated by the following example.

Example 3. (1) Let $\mathcal{D}_1 = \{A(1)\}$, $\mathcal{D}_2 = \{A(2)\}$ and $E = (\{\mathcal{D}_1\}, \{\mathcal{D}_2\})$. Then $\bigcirc A$ separates E but no query in $\mathcal{Q}[\Diamond]$ does. On the other hand, E is not $\mathcal{Q}$-separable under $\mathcal{O} = \{\bigcirc A \to A\}$, for any class $\mathcal{Q}$ defined above, as $\mathcal{O}, \mathcal{D}_1 \models \boldsymbol{q}(0)$ implies $\mathcal{O}, \mathcal{D}_2 \models \boldsymbol{q}(0)$ for all $\boldsymbol{q} \in \mathcal{Q}$.

(2) Let $E = (\{\mathcal{D}_1, \mathcal{D}_2\}, \emptyset)$ with $\mathcal{D}_1$ and $\mathcal{D}_2$ as in (1). Then $\Diamond A$ separates E but no $\mathcal{Q}[\bigcirc]$-query does. Observe that at least two positive examples are needed to achieve this effect. However, $\bigcirc\bigcirc\bigcirc B$ separates E under $\mathcal{O} = \{A \to \Box B\}$.

(3) Let $E = (\{\{A(1), B(2)\}, \{A(2), B(3)\}\}, \{\{A(3), B(5)\}\})$. Then $\Diamond(A \land \bigcirc B)$ separates E but no query in $\mathcal{Q}[\bigcirc]$ or $\mathcal{Q}[\Diamond]$ does.

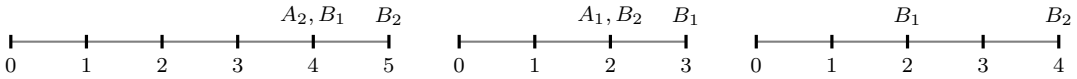
(4) $A \,\mathsf{U}\, B$ separates $(\{\{B(1)\}, \{A(1), B(2)\}\}, \{\{B(2)\}\})$ but no $\mathcal{Q}[\bigcirc, \Diamond]$-query does. ⊣

2. Our Contribution

We now briefly present our initial results on the complexity of the separability problem for *LTL* queries, both plain and mediated by an *LTL*-ontology.

Ontology-free LTL queries. Separability in $\mathcal{Q}[\bigcirc]$ is almost trivial as it corresponds to the existence of a time point where some atom holds in all positive examples but in no negative example, which is decidable in polynomial time. For the query languages ranging from $\mathcal{Q}_p[\Diamond]$ and $\mathcal{Q}[\Diamond]$ to $\mathcal{Q}_p[\bigcirc, \Diamond]$ and $\mathcal{Q}[\bigcirc, \Diamond]$ and also $\mathcal{Q}_p[\mathsf{U}]$, separability turns out to be NP-complete. The upper bound is proved by observing that, in any of these languages, every separable example set can be separated by a query of polynomial size. The matching lower bound is established by reduction of the NP-hard problem of deciding whether the words in a given set contain a common subsequence of a given length [25]. Separability by $\mathcal{Q}[\mathsf{U}]$-queries turns out to be trickier because of the interplay of $\wedge$, the left- and the right-hand sides of the U-operator.

Example 4. The example set below, where the instance on the right is negative, is separated by



the $\mathcal{Q}[\mathsf{U}]$-query $\Diamond\big((A_1 \,\mathsf{U}\, B_1) \wedge (A_2 \,\mathsf{U}\, B_2)\big)$ but is not separable in any other class of queries.

We give a separability criterion in terms of U-simulations between subsets of the disjoint union of the positive examples and points of a negative example (cf. [26]). Then, using a game-theoretic variant of U-simulations, we show that a separating $\mathcal{Q}[\mathsf{U}]$-query can be constructed in PSpace. However, at the moment, we only have an NP-lower bound for separability.

Separability under LTL Ontologies. Apart from full *LTL*, we consider its fragment $\mathit{LTL}^{\Box\Diamond}$ that only uses the operators $\Box$ and $\Diamond$, also known as the *Prior logic* [27, 28, 29, 30], and the Horn fragment $\mathit{LTL}^{\Box\bigcirc}_{\mathit{horn}}$ containing axioms of the form $C_1 \wedge \dots \wedge C_k \to C_{k+1}$, where the C_i are atoms possibly prefixed by $\Box$ and $\bigcirc$ for $i \leq k+1$, and also by $\Diamond$ for $i \leq k$. The ontology axioms are supposed to hold at all times.

Separability by (path) $\Diamond$-queries is Σ^p_2-complete under $\mathit{LTL}^{\Box\Diamond}$ ontologies and PSpace-complete under $\mathit{LTL}^{\Box\bigcirc}_{\mathit{horn}}$ ontologies. For *LTL* ontologies, we have a NExpTime upper bound. We conjecture that exactly the same bounds can be proved for (path) $\bigcirc\Diamond$-queries. As concerns $\mathcal{Q}_p[\mathsf{U}]$-queries, separability under $\mathit{LTL}^{\Box\bigcirc}_{\mathit{horn}}$ ontologies is shown to be between ExpSpace and NExpTime; for 'branching' $\mathcal{Q}[\mathsf{U}^-]$-queries without nesting U-operators on the left of U, it can be decided in ExpTime using U-simulations. We establish the upper bounds by constructing two exponential-size transition systems S^+ and S^- from $(\mathcal{O}, E^+)$ and $(\mathcal{O}, E^-)$ such that (i) there is a *trace-based simulation* of S^+ by S^- iff (E^+, E^-) is separated in $\mathcal{Q}_p[\mathsf{U}]$ and (ii) there is a *tree-based simulation* of S^+ by S^- iff (E^+, E^-) is separated in $\mathcal{Q}[\mathsf{U}^-]$. The existence of trace-based and tree-based simulations can be decided in PSpace- and P, respectively [31].

The obtained complexity results are summarised in the table below:

$\mathsf{QBE}(\mathcal{L}, \mathcal{Q})$	LTL	$LTL_{horn}^{\Box\bigcirc}$	$LTL^{\Box\Diamond}$	$\mathsf{QBE}(\mathcal{Q})$
$\mathcal{Q}[\mathsf{U}]$	?	?	?	$\geq$ NP, $\leq$ PSpace
$\mathcal{Q}[\mathsf{U}^-]$		$\leq$ ExpTime		
$\mathcal{Q}_p[\mathsf{U}]$		$\geq$ NExpTime, $\leq$ ExpSpace		= NP
$\mathcal{Q}[\bigcirc, \Diamond]$		$\leq$ ExpTime		
$\mathcal{Q}_p[\bigcirc, \Diamond]$		$\leq$ ExpSpace		
$\mathcal{Q}[\Diamond]$	$\leq$ NExpTime	= PSpace	$= \Sigma_2^p$	
$\mathcal{Q}_p[\Diamond]$				

Acknowledgments

This research was supported by the EPSRC UK grants EP/S032207 and EP/S032282 for the joint project 'quant$^{\text{MD}}$: Ontology-Based Management for Many-Dimensional Quantitative Data' and by the RSF grant 22-11-00323 when M. Zakharyaschev was visiting the HSE University.

References

[1] D. M. L. Martins, Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities, Inf. Syst. 83 (2019) 89–100.

[2] M. Zhang, H. Elmeleegy, C. M. Procopiuc, D. Srivastava, Reverse engineering complex join queries, in: Proc. of SIGMOD, 2013, pp. 809–820.

[3] Y. Y. Weiss, S. Cohen, Reverse engineering SPJ-queries from examples, in: Proc. of PODS, 2017, pp. 151–166.

[4] D. V. Kalashnikov, L. V. Lakshmanan, D. Srivastava, Fastqre: Fast query reverse engineering, in: Proc. of SIGMOD, 2018, pp. 337–350.

[5] D. Deutch, A. Gilad, Reverse-engineering conjunctive queries from provenance examples., in: Proc. of EDBT, 2019, pp. 277–288.

[6] S. Staworko, P. Wieczorek, Learning twig and path queries, in: Proc. of ICDT, 2012, pp. 140–154.

[7] P. Barceló, M. Romero, The complexity of reverse engineering problems for conjunctive queries, in: Proc. of ICDT, 2017, pp. 7:1–7:17.

[8] S. Cohen, Y. Y. Weiss, The complexity of learning tree patterns from example graphs, ACM Trans. Database Syst. 41 (2016) 14:1–14:44. URL: https://doi.org/10.1145/2890492. doi:10.1145/2890492.

[9] M. Arenas, G. I. Diaz, E. V. Kostylev, Reverse engineering SPARQL queries, in: Proc. of WWW, 2016, pp. 239–249.

[10] V. Gutiérrez-Basulto, J. C. Jung, L. Sabellek, Reverse engineering queries in ontology-enriched systems: The case of expressive Horn description logic ontologies, in: Proc. of IJCAI-ECAI, 2018.

[11] M. Ortiz, Ontology-mediated queries from examples: a glimpse at the DL-Lite case, in: Proc. of GCAI, 2019, pp. 1–14.

[12] J. C. Jung, C. Lutz, H. Pulcini, F. Wolter, Logical separability of incomplete data under ontologies, in: Proc. of KR, 2020.

[13] G. Cima, F. Croce, M. Lenzerini, Query definability and its approximations in ontology-based data management, in: G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, H. Tong (Eds.), CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021, ACM, 2021, pp. 271–280. URL: https://doi.org/10.1145/3459637.3482466. doi:10.1145/3459637.3482466.

[14] J. C. Jung, C. Lutz, H. Pulcini, F. Wolter, Separating data examples by description logic concepts with restricted signatures, in: Proc. of KR, 2021.

[15] R. Willard, Testing expressibility is hard, in: D. Cohen (Ed.), Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings, volume 6308 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 9–23. URL: https://doi.org/10.1007/978-3-642-15396-9_4. doi:10.1007/978-3-642-15396-9_4.

[16] B. ten Cate, V. Dalmau, The product homomorphism problem and applications, in: Proc. of ICDT, 2015, pp. 161–176.

[17] M. Funk, J. C. Jung, C. Lutz, H. Pulcini, F. Wolter, Learning description logic concepts: When can positive and negative examples be separated?, in: Proc. of IJCAI, 2019, pp. 1682–1688.

[18] S. Demri, V. Goranko, M. Lange, Temporal Logics in Computer Science, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2016.

[19] C. Lemieux, D. Park, I. Beschastnikh, General LTL specification mining (T), in: Proc. of ASE, IEEE, 2015, pp. 81–92.

[20] D. Neider, I. Gavran, Learning linear temporal properties, in: Proc. of FMCAD 2018, IEEE, 2018, pp. 1–10. URL: https://doi.org/10.23919/FMCAD.2018.8603016. doi:10.23919/FMCAD.2018.8603016.

[21] A. Camacho, S. A. McIlraith, Learning interpretable models expressed in linear temporal logic, in: Proc. of ICAPS 2018, AAAI Press, 2019, pp. 621–630. URL: https://aaai.org/ojs/index.php/ICAPS/article/view/3529.

[22] N. Fijalkow, G. Lagarde, The complexity of learning linear temporal formulas from examples, in: J. Chandlee, R. Eyraud, J. Heinz, A. Jardine, M. Zaanen (Eds.), Proceedings of the 15th International Conference on Grammatical Inference, 23-27 August 2021, Virtual Event, volume 153 of *Proceedings of Machine Learning Research*, PMLR, 2021, pp. 237–250. URL: https://proceedings.mlr.press/v153/fijalkow21a.html.

[23] R. Raha, R. Roy, N. Fijalkow, D. Neider, Scalable anytime algorithms for learning fragments of linear temporal logic, in: D. Fisman, G. Rosu (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I, volume 13243 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 263–280. URL: https://doi.org/10.1007/978-3-030-99524-9_14. doi:10.1007/978-3-030-99524-9_14.

[24] M. Fortin, B. Konev, V. Ryzhikov, Y. Savateev, F. Wolter, M. Zakharyaschev, Unique characterisability and learnability of temporal instance queries, in: Proc. of KR, 2022.

[25] D. Maier, The complexity of some problems on subsequences and supersequences, J. ACM 25 (1978) 322–336. URL: https://doi.org/10.1145/322063.322075. doi:10.1145/322063.

322075.
[26] N. Kurtonina, M. de Rijke, Bisimulations for temporal logic, J. Log. Lang. Inf. 6 (1997) 403–425. URL: https://doi.org/10.1023/A:1008223921944. doi:10.1023/A:1008223921944.
[27] A. Prior, Time and Modality, Oxford University Press, 1956.
[28] H. Ono, A. Nakamura, On the size of refutation Kripke models for some linear modal and tense logics, Studia Logica (1980) 325–333.
[29] J. P. Burgess, Basic tense logic, in: Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic, Reidel, Dordrecht, 1984, pp. 89–133.
[30] M. Y. Vardi, From Church and Prior to PSL, in: 25 Years of Model Checking - History, Achievements, Perspectives, volume 5000 of *LNCS*, Springer, 2008, pp. 150–171.
[31] O. Kupferman, M. Y. Vardi, Verification of fair transisiton systems, in: R. Alur, T. A. Henzinger (Eds.), Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings, volume 1102 of *Lecture Notes in Computer Science*, Springer, 1996, pp. 372–382. URL: https://doi.org/10.1007/3-540-61474-5_84. doi:10.1007/3-540-61474-5_84.