LAMAS&SR Proceedings

Julian Gutierrez and Vadim Malvone Editors

Logics and strategic reasoning play a central role in multi-agent systems. Logic can be used, for instance, to express the agents' abilities, knowledge, and objectives. Strategic reasoning refers to algorithmic methods that allow for developing good behaviour for the agents of the system. At the intersection, we find logics that can express the existence of strategies or equilibria, and can be used to reason about them. The LAMAS&SR workshop merges two international workshops: LAMAS (Logical Aspects of Multi-Agent Systems), which focuses on all kinds of logical aspects of multi-agent systems from the perspectives of AI, computer science, and game theory, and SR (Strategic Reasoning), devoted to all aspects of strategic reasoning in formal methods and AI.

LAMAS: The LAMAS workshop provides a meeting forum for the research community working on various logical aspects of multi-agent systems from the perspectives of artificial intelligence, computer science, and game theory. It addresses the whole range of issues that arise in the context of using logic in multi-agent sytems, from theoretical foundations to algorithmic methods and implemented tools. The workshop LAMAS has been regularly organised since 2002 and became the main annual event of the LAMAS research network.

SR: Strategic reasoning is a key topic in the multi-agent systems research area. The extensive literature in this field includes a number of logics used for reasoning about the strategic abilities of the agents in the system, but spans also game theory, decision theory or epistemic logics to name a few. The aim is to provide sound theoretical foundations and tools to tackle a variety of strategic problems in formal methods and artificial intelligence involving agents in adversarial settings. The SR workshop has been organised annually since 2013, often in co-location with the most important conferences in formal methods and AI.

LAMAS&SR: Over the years the communities and research themes of both workshops got closer and closer, with a significant overlap in the participants and organisers of both events. For this reason, the next editions of LAMAS and SR will be unified under the same flag, formally joining the two communities.

Julian Gutierrez
Monash University, Melbourne, Australia, e-mail: julian.gutierrez@monash.edu

Vadim Malvone
Télécom Paris, Paris, France e-mail: vadim.malvone@telecom-paris.fr

Different Strokes in Randomised Strategies: Revisiting Kuhn's Theorem under Finite-memory Assumptions (extended abstract)*

James C. A. Main and Mickael Randour

F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Games on graphs. Games on (possibly stochastic) graphs have been studied for decades, both for their own interest (e.g., [17, 14, 20]) and for their value as a framework for *reactive synthesis* (e.g., [21, 25, 7, 3]). The core problem is almost always to find *optimal strategies* for the players: strategies that guarantee winning for Boolean winning conditions (e.g., [18, 29, 10, 8]), or strategies that achieve the best possible payoff in quantitative contexts (e.g., [17, 5, 11]). In multi-objective settings, one is interested in *Pareto-optimal* strategies (e.g., [13, 28, 26, 16]), but the bottom line is the same: players are looking for strategies that guarantee the best possible results.

In reactive synthesis, we model the interaction between a system and its uncontrollable environment as a two-player antagonistic game, and we represent the specification to ensure as a winning objective. An optimal strategy for the system in this game then constitutes a formal blueprint for a *controller* to implement in the real world [3].

Randomness in strategies. In essence, a *pure strategy* is simply a function mapping histories (i.e., the past and present of a play) to an action.

Optimal strategies may require *randomisation* when dealing with inherently probabilistic goals, balancing multiple objectives, or in contexts of partial information: see, e.g., [12, 26, 16]. There are different ways of randomising strategies. For instance, a *mixed* strategy is essentially a probability distribution over a set of pure strategies. That is, the player randomly selects a pure strategy at the beginning of the game and then follows it for the entirety of the play without resorting to randomness ever again. By contrast, a *behavioural* strategy randomly selects an action at each step: it thus maps histories to probability distributions over actions.

Kuhn's theorem. In full generality, these two definitions yield different classes of strategies (e.g., [15] or [24, Chapter 11]). Nonetheless, Kuhn's theorem [2] proves their equivalence under a mild hypothesis: in games of *perfect recall*, for any mixed strategy there is an equivalent behavioural strategy and vice versa. A game is said to be of perfect recall for a given player if said player never forgets

* Mickael Randour is a Research Associate of the Fonds de la Recherche Scientifique - FNRS and James C. A. Main is a Research Fellow of the Fonds de la Recherche Scientifique - FNRS. Both authors are members of the TRAIL Institute. This work has been supported by the Fonds de la Recherche Scientifique - FNRS under Grant n° T.0188.23 (PDR ControlleRS).

their previous knowledge and the actions they have played (i.e., they can observe their own actions). Let us note that perfect recall and *perfect information* are two different notions: perfect information is not required to have perfect recall.

Let us highlight that Kuhn's theorem crucially relies on two elements. First, mixed strategies can be distributions over an *infinite* set of pure strategies. Second, strategies can use *infinite memory*, i.e., they are able to remember the past completely, however long it might be. Indeed, consider a game in which a player can choose one of two actions in each round. One could define a (memoryless) behavioural strategy that selects one of the two actions by flipping a coin each round. This strategy generates infinitely many sequences of actions, therefore any equivalent mixed strategy needs the ability to randomise between infinitely many different sequences, and thus, infinitely many pure strategies. Moreover, infinitely many of these sequences require infinite memory to be generated (due to their non-regularity).

Finite-memory strategies. From the point of view of reactive synthesis, infinite-memory strategies, along with randomised ones relying on infinite supports, are undesirable for implementation. This is why a plethora of recent advances has focused on *finite-memory* strategies, usually represented as (a variation on) Mealy machines, i.e., finite automata with outputs. See, e.g., [20, 13, 9, 16, 4, 6]. Randomisation can be implemented in these finite-memory strategies in different ways: the *initialisation*, *outputs* or *transitions* can be randomised or deterministic respectively.

Depending on which aspects are randomised, the expressiveness of the corresponding class of finite-memory strategies differs: in a nutshell, *Kuhn's theorem crumbles when restricting ourselves to finite memory*. For instance, we show that some finite-memory strategies with only randomised outputs (i.e., the natural equivalent of behavioural strategies) cannot be emulated by finite-memory strategies with only randomised initialisation (i.e., the natural equivalent of mixed strategies). Similarly, it is known that some finite-memory strategies that are encoded by Mealy machines using randomisation in all three components admit no equivalent using randomisation only in outputs [1, 15].

Our contributions. The results mentioned in the following are presented in [22]. We consider *two-player zero-sum stochastic games* (e.g., [27, 14, 23, 6]), encompassing two-player (deterministic) games and Markov decision processes as particular subcases. We establish a *Kuhn-like taxonomy* of the classes of finite-memory strategies obtained by varying which of the three aforementioned components are randomised: we illustrate it in Figure 1.

Let us highlight a few elements. Naturally, the least expressive model corresponds to pure strategies. In contrast to what happens with infinite memory, and as noted in the previous paragraph, we see that mixed strategies are strictly less expressive than behavioural ones. We also observe that allowing randomness both in initialisation and in outputs (RRD strategies) yields an even more expressive class — and incomparable to what is obtained by allowing randomness in updates only. Finally, the most expressive class is obviously obtained when

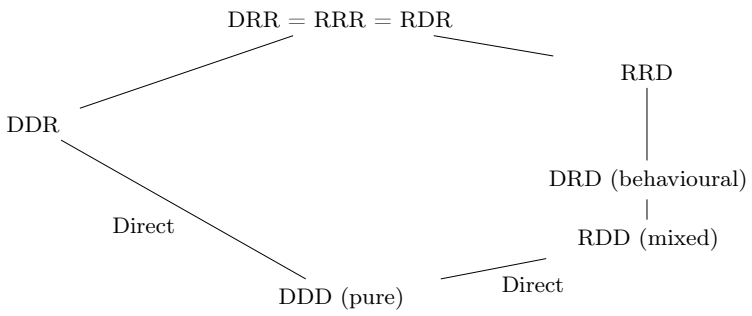

Fig. 1. Lattice of strategy classes in terms of expressible probability distributions over plays against all strategies of the other player. In the three-letter acronyms, the letters, in order, refer to the initialisation, outputs and updates of the Mealy machines: D and R respectively denote deterministic and randomised components.

allowing randomness in all components; yet it may be dropped in initialisation or in outputs without reducing the expressiveness — but not in both simultaneously.

Constructions used to establish inclusions between classes of randomised finite-memory strategies are effective. To show that any mixed strategy can be emulated by a behavioural one, we derive an appropriate Mealy machine via an adapted subset construction. To show that randomisation in the initialisation can be dropped from the most expressive model without losing expressiveness, we add a new initial state to the Mealy machine that emulates the initial distribution and the choice of the first action. Finally, to show that removing randomisation in the outputs in the most expressive model is not restrictive, we incorporate the randomisation over actions in the randomised initialisation and updates of the Mealy machine.

To compare the expressiveness of strategy classes, we consider *outcome-equivalence*. Intuitively, two strategies are outcome-equivalent if, against any strategy of the opponent, they yield identical probability distributions (i.e., they induce identical Markov chains). Hence we are agnostic with regard to the objective, winning condition, payoff function, or preference relation of the game, and with regard to how they are defined (e.g., colours on actions, states, transitions, etc).

Finally, let us note that in our setting of two-player stochastic games, the perfect recall hypothesis holds. Most importantly, we assume that actions are visible. Lifting this hypothesis drastically changes the relationships between the different models. We note that *our results hold in games of imperfect information* too, assuming visible actions, and that *our results hold in games with more than two players.*

References

1. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. Theor. Comput. Sci. **386**(3), 188–217 (2007)
2. Aumann, R.J..: 28. Mixed and Behavior Strategies in Infinite Extensive Games, pp. 627–650. Princeton University Press (2016)
3. Bloem, R., Chatterjee, K., Jobstmann, B.: Graph games and reactive synthesis. In: Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.) Handbook of Model Checking, pp. 921–962. Springer (2018)
4. Bouyer, P., Le Roux, S., Oualhadj, Y., Randour, M., Vandenhove, P.: Games where you can play optimally with arena-independent finite memory. Log. Methods Comput. Sci. **18**(1) (2022)
5. Bouyer, P., Markey, N., Randour, M., Larsen, K.G., Laursen, S.: Average-energy games. Acta Inf. **55**(2), 91–127 (2018)
6. Bouyer, P., Oualhadj, Y., Randour, M., Vandenhove, P.: Arena-independent finite-memory determinacy in stochastic games. In: Haddad, S., Varacca, D. (eds.) 32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference. LIPIcs, vol. 203, pp. 26:1–26:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
7. Brenguier, R., Clemente, L., Hunter, P., Pérez, G.A., Randour, M., Raskin, J., Sankur, O., Sassolas, M.: Non-zero sum games for reactive synthesis. In: Dediu, A., Janousek, J., Martín-Vide, C., Truthe, B. (eds.) Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9618, pp. 3–23. Springer (2016)
8. Brihaye, T., Delgrange, F., Oualhadj, Y., Randour, M.: Life is random, time is not: Markov decision processes with window objectives. In: Fokkink and van Glabbeek [19], pp. 8:1–8:18
9. Bruyère, V., Filiot, E., Randour, M., Raskin, J.: Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. Inf. Comput. **254**, 259–295 (2017)
10. Bruyère, V., Hautem, Q., Randour, M.: Window parity games: an alternative approach toward parity games with time bounds. In: Cantone, D., Delzanno, G. (eds.) Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016. EPTCS, vol. 226, pp. 135–148 (2016)
11. Bruyère, V., Hautem, Q., Randour, M., Raskin, J.: Energy mean-payoff games. In: Fokkink and van Glabbeek [19], pp. 21:1–21:17
12. Chatterjee, K., Doyen, L.: Partial-observation stochastic games: How to win when belief fails. In: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012. pp. 175–184. IEEE Computer Society (2012)
13. Chatterjee, K., Randour, M., Raskin, J.: Strategy synthesis for multi-dimensional quantitative objectives. Acta Inf. **51**(3-4), 129–163 (2014)
14. Condon, A.: The complexity of stochastic games. Inf. Comput. **96**(2), 203–224 (1992)
15. Cristau, J., David, C., Horn, F.: How do we remember the past in randomised strategies? In: Montanari, A., Napoli, M., Parente, M. (eds.) Proceedings First Symposium on Games, Automata, Logic, and Formal Verification, GANDALF 2010, Minori (Amalfi Coast), Italy, 17-18th June 2010. EPTCS, vol. 25, pp. 30–39 (2010)

16. Delgrange, F., Katoen, J., Quatmann, T., Randour, M.: Simple strategies in multi-objective MDPs. In: Biere, A., Parker, D. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12078, pp. 346–364. Springer (2020)
17. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. Int. Journal of Game Theory **8**(2), 109–113 (1979)
18. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs. SIAM J. Comput. **29**(1), 132–158 (1999)
19. Fokkink, W., van Glabbeek, R. (eds.): 30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands, LIPIcs, vol. 140. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
20. Gimbert, H., Zielonka, W.: Games where you can play optimally without any memory. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3653, pp. 428–442. Springer (2005)
21. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], Lecture Notes in Computer Science, vol. 2500. Springer (2002)
22. Main, J.C.A., Randour, M.: Different strokes in randomised strategies: Revisiting kuhn's theorem under finite-memory assumptions. In: Klin, B., Lasota, S., Muscholl, A. (eds.) 33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland. LIPIcs, vol. 243, pp. 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
23. Maitra, A., Sudderth, W.: Stochastic games with Borel payoffs. In: Neyman, A., Sorin, S. (eds.) Stochastic Games and Applications. pp. 367–373. Springer Netherlands, Dordrecht (2003)
24. Osborne, M.J., Rubinstein, A.: A course in game theory. The MIT Press, Cambridge, USA (1994), electronic edition
25. Randour, M.: Automated synthesis of reliable and efficient systems through game theory: A case study. In: Proc. of ECCS 2012, pp. 731–738. Springer Proceedings in Complexity XVII, Springer (2013)
26. Randour, M., Raskin, J., Sankur, O.: Percentile queries in multi-dimensional Markov decision processes. Formal Methods Syst. Des. **50**(2-3), 207–248 (2017)
27. Shapley, L.S.: Stochastic games. Proceedings of the National Academy of Sciences **39**(10), 1095–1100 (1953)
28. Velner, Y., Chatterjee, K., Doyen, L., Henzinger, T.A., Rabinovich, A.M., Raskin, J.: The complexity of multi-mean-payoff and multi-energy games. Inf. Comput. **241**, 177–196 (2015)
29. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. **200**(1-2), 135–183 (1998)

Reactive Synthesis of Linear Temporal Logic on Finite Traces: An Evolving Journey

Shufang Zhu

University of Oxford
shufang.zhu@cs.ox.ac.uk

Abstract. Reactive synthesis holds the promise of automatically generating a verifiably correct program from a high-level specification. In this work, we focus on reactive synthesis problems of Linear Temporal Logic on finite traces (LTL$_f$) and present an evolving journey. We first present the advances in LTL$_f$ synthesis. Then we show that when concerning environment specifications expressed in LTL, we can practically diminish the difficulty of the challenging problem of LTL synthesis and keep the simplicity of LTL$_f$ synthesis in interesting cases.

1 Introduction

Reactive synthesis promises to automatically generate a verifiably correct program from a high-level specification [18]. A popular such specification language is Linear Temporal Logic (LTL) [17]. Unfortunately, synthesizing programs from *general* LTL formulas, which rely on first constructing a game arena and then solving the game, remains challenging [13, 16]. Nevertheless, the synthesis problem of a finite trace variant of LTL, which is LTL$_f$ [14], has shown to be much simpler than LTL synthesis [12]. The key idea is that synthesizing LTL$_f$ formulas only involves games on finite traces instead of infinite traces as for LTL, though both problems share the same worst-case complexity of 2EXPTIME-complete.

In this paper, we will review an evolving journey motivated by this idea. We start from an attempt to devise a symbolic LTL$_f$ synthesis framework [22], which consists of a backward reachability game on the constructed Deterministic Finite Automaton (DFA) of the corresponding LTL$_f$ formula and has demonstrated its significant efficiency in various application scenarios. Then, the journey evolves into a forward LTL$_f$ synthesis technique that synthesizes a strategy while constructing the DFA, thus being possible to avoid the 2EXPTIME worst-case complexity [19, 11]. Next, we study LTL$_f$ synthesis under environment specifications, which are constraints on the environment that rule out certain environment behaviours [1, 6]. A key observation is that even if we consider an agent with LTL$_f$ tasks on finite traces, environment specifications need to be expressed over infinite traces since accomplishing the agent tasks may require an unbounded number of environment actions [1, 6]. While a naive solution to LTL$_f$ synthesis under environment specifications expressed in LTL would be reducing the problem to LTL synthesis, which remains challenging [1, 6], we show in this paper

that we can avoid the detour to LTL synthesis and keep the simplicity of LTL_f synthesis in interesting cases. More specifically, we consider the following certain environment specifications: safety [7], simple fairness and stability [20], and Generalized-Reactivity(1) (GR(1)) [8]. Furthermore, we show that even when the environment specifications are expressed in general LTL, we can still partially avoid the full detour to LTL synthesis [9].

2 LTL_f Synthesis

Reactive synthesis can be viewed as a game between the *environment* and the *agent*, contrasting each other by controlling two disjoint sets of variables $\mathcal{X}$ and $\mathcal{Y}$, respectively. Reactive synthesis aims to synthesize an agent strategy such that no matter how the environment behaves, the combined trace from two players satisfies desired properties [18]. An environment strategy is a function $\gamma : (2^{\mathcal{Y}})^+ \rightarrow 2^{\mathcal{X}}$, and an agent strategy is a function $\sigma : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$. A trace $\pi = (X_0 \cup Y_0)(X_1 \cup Y_1) \cdots \in (2^{\mathcal{X} \cup \mathcal{Y}})^\omega$, is *compatible* with an environment strategy γ if $\gamma(Y_0 Y_1 \dots Y_i) = X_i$ for every $i \geq 0$. A trace π being compatible with an agent strategy σ is defined analogously. Sometimes, we write $\sigma(\pi^k)$ instead of $\sigma(X_0 X_1 \cdots X_k)$ for simplicity. We denote the unique infinite sequence that is compatible with γ and σ as $\pi(\sigma, \gamma)$. The *synthesis problem* for an agent task specified as an LTL_f formula φ is to find an agent strategy $\sigma : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$ such that for every environment strategy $\gamma : (2^{\mathcal{Y}})^+ \rightarrow 2^{\mathcal{X}}$, there exists $k \geq 0$, chosen by the agent, such that the finite trace $\pi(\sigma, \gamma)^k \models \varphi$ i.e., φ is agent realizable. One can solve LTL_f synthesis by translating φ to an LTL formula ψ and then solving LTL synthesis on ψ [22].

Backward LTL_f Synthesis. The first solution to LTL_f synthesis problem was based on a reduction to reachability game [12], which proceeds as follows: build the corresponding DFA of the agent task φ, solve the reachability game over it, and hence return the winning strategy for the agent. However, the size of the constructed DFA can be, in the worst case, doubly-exponential in the size of the formula. To combat this difficulty, we proposed a symbolic LTL_f synthesis framework representing the DFA as Boolean formulas [22] using Binary Decision Diagrams (BDDs) [5]. This synthesis framework has shown outperformance compared to the explicit approach described in [12] and the solution of reducing to LTL synthesis. Furthermore, it also has been integrated into state-of-the-art LTL_f synthesizers, e.g., Lisa [3] and Lydia [10]. The main difficulty of this synthesis framework is that it requires computing the entire DFA of the LTL_f specification, hence cannot avoid the worst-case 2EXPTIME blowup.

Forward LTL_f Synthesis. To combat the worst-case 2EXPTIME blowup, we investigated LTL_f forward synthesis adopting an AND-OR graph search that can create on-the-fly the DFA corresponding to the LTL_f specification [19, 11]. This technique exploits formula progression to build directly deterministic transitions from a current state. Crucially, in [11] we exploit the structure that formula progression provides to branch on propositional formulas (representing several

evaluations) instead of individual evaluations as in [19]. This drastically reduces the branching factor of the AND-OR graph to be searched.

3 LTL$_f$ Synthesis Under Environment Specifications

In standard synthesis, the agent assumes the environment to be free to choose an arbitrary move at each step, but in reality, often the agent has some knowledge of how the environment works, which it can exploit to enforce the goal, specified as an LTL$_f$ formula φ. Here, we specify the environment behavior by an LTL formula *env* and call it *environment specification* [1]. Given an LTL formula *env*, we say that an agent strategy (resp. environment strategy) *enforces* φ, written $\sigma \rhd env$ (resp., $\gamma \rhd env$), if for every environment strategy γ (resp. agent strategy σ), we have $\pi(\sigma, \gamma) \models \varphi$. This LTL formula *env*, in particular, specifies the set of environment strategies that enforces *env*. As usual, we require that *env* must be *environment realizable*, i.e., the set of environment strategies that enforce *env* is nonempty. The problem of *synthesis under environment specifications* is to find an agent strategy σ such that $\forall \gamma \ \rhd env, trace(\sigma, \gamma)^k \models \varphi$ for some $k \in \mathcal{N}$.

In the following, we present the solutions of LTL$_f$ synthesis under LTL environment specifications for certain types of environment specifications.

Safety Environment Specifications. Intuitively, a *safety* property excludes traces whose "badness" follows from a finite prefix. One can write a safety environment specification either with Safety LTL, a syntactic fragment of LTL or LTL$_f$ in all prefix semantics, for more details of which, we refer to [21, 2] and [8], respectively. To solve the problem, we can first translate the safety environment specification *env* into a Deterministic Safety Automaton (DSA) S [21] and solve a safety game for the environment on S. By restricting S to the environment winning region, we can obtain all the environment strategies that can enforce *env*. Finally, we need to solve the reachability game over the product of the corresponding DFA of LTL$_f$ formula φ and the restricted part of the DSA S [7], thus obtaining an agent winning strategy if there exists one.

Fairness and Stability Environment Specifications. We consider two different basic forms of environment specifications: a basic form of fairness $\Box\Diamond\alpha$ (always eventually α) and a basic form of stability $\Diamond\Box\alpha$. The key idea of solving such problems is integrating the environment specification as the winning condition of the reduced game between the environment and the agent [20]. We can solve the problem as follows. First, translate the LTL$_f$ formula φ into a DFA D. Then, in case of fairness environment specifications, solve the fair DFA game on D, in which the environment (resp. the agent) winning condition is to remain in a region (resp., to avoid the region), where α holds infinitely often. Meanwhile, the accepting states are forever avoidable by applying a nested fixed-point computation on D. Analogous solution techniques apply to the case of stability environment specifications.

Generalized-Reactivity(1) Specifications. There have been great successes with LTL synthesis on the GR(1) [4] approach: focusing on a significant syntactic fragment of LTL that uses safety conditions to determine the possible transitions

in a game between the environment and the agent, plus one powerful notion of fairness. We brought it together with the successes on LTL_f synthesis, devising an approach to solve LTL_f synthesis of agent task φ under GR(1) environment specification env_{gr1} [8]. In more detail, we first observe that the agent's goal is to satisfy $\neg env_{gr1} \vee \varphi$, while the environment's goal is to satisfy $env_{gr1} \wedge \neg\varphi$. Then, focusing on the environment point of view, we show that the problem of LTL_f synthesis under GR(1) environment specification can be reduced into a GR(1) game, in which the game arena is the complement of the DFA for φ, i.e., a DSA expressing safety conditions, and env_{gr1} is the GR(1) winning condition. Since we want a winning strategy for the agent, we need to deal with the complement of the GR(1) game to obtain a winning strategy for the antagonist.

General LTL Environment Specifications. Regarding LTL_f synthesis with general LTL environment specifications, dealing with LTL synthesis seems to be unavoidable. Nevertheless, we developed a two-stage technique that maximizes the simplicity of LTL_f synthesis and mitigates the difficulty of LTL synthesis [9]. Intuitively, the two-stage technique first solely deals with the agent task, LTL_f formula φ, and thus confines the difficulty of handling the LTL environment specification env to the bare minimum in the second stage. In detail, the two-stage techniques proceed as follows: (*i*) Build the DFA D of φ and solve the reachability game for the agent over D. If the agent has a winning strategy σ in D then the algorithm returns σ. Otherwise, continue to Stage 2. (*ii*) Perform the following steps: (*ii.a*) Remove from D the agent winning region, obtaining D'; (*ii.b*) Build the Deterministic Parity Automaton (DPA) P of env and get the product of D' and P, obtaining a new DPA $A = D' \times P$, and solve the parity game for the environment over A; (*ii.c*) If the agent has a winning strategy in A, then the synthesis problem is realizable and hence returns the agent winning strategy as a combination of the agent winning strategies in the two stages.

4 Conclusion

Reactive synthesis on LTL_f has been an exciting research problem. Our work on this problem spans from standard LTL_f synthesis to synthesis concerning environment specifications with efficient solution techniques. In the future, we would like to consider environment specifications expressed in different languages, e.g., PDDL [15], a popular specification language in planning.

Acknowledgments

We thank the contributions of all the co-authors (in the order of publications): Jianwen Li, Geguang Pu, Lucas M. Tabajara, Moshe Y. Vardi, Giuseppe De Giacomo, Antonio Di Stasio, Giuseppe Perelli, Shengping Xiao, Yingying Shi, Marco Favorito, Suguman Bansal and Yong Li. This work is partially supported by the ERC Advanced Grant WhiteMech (No. 834228).

References

1. Aminof, B., De Giacomo, G., Murano, A., Rubin, S.: Planning under LTL environment specifications. In: ICAPS. pp. 31–39 (2019)
2. Bansal, S., De Giacomo, G., Di Stasio, A., Li, Y., Y. Vardi, M., Zhu, S.: Compositional safety LTL synthesis. In: VSTTE. pp. 1–19
3. Bansal, S., Li, Y., Tabajara, L.M., Vardi, M.Y.: Hybrid Compositional Reasoning for Reactive Synthesis from Finite-Horizon Specifications. In: AAAI. pp. 9766–9774 (2020)
4. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of Reactive(1) designs. vol. 78, pp. 911–938 (2012)
5. Bryant, R.E.: Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. ACM Comput. Surv. **24**(3), 293–318 (1992)
6. Camacho, A., Bienvenu, M., McIlraith, S.A.: Finite LTL synthesis with environment assumptions and quality measures. In: KR. pp. 454–463 (2018)
7. De Giacomo, G., Di Stasio, A., Perelli, G., Zhu, S.: Synthesis with mandatory stop actions. In: KR. pp. 237–246 (2021)
8. De Giacomo, G., Di Stasio, A., Tabajara, L.M., Vardi, M.Y., Zhu, S.: Finite-trace and generalized-reactivity specifications in temporal synthesis. In: IJCAI (2021)
9. De Giacomo, G., Di Stasio, A., Vardi, M.Y., Zhu, S.: Two-stage technique for LTL_f synthesis under LTL assumptions. In: KR (2020)
10. De Giacomo, G., Favorito, M.: Compositional approach to translate ltlf/ldlf into deterministic finite automata. In: ICAPS (to appear). vol. 14 (2021)
11. De Giacomo, G., Favorito, M., Li, J., Y. Vardi, M., Xiao, S., Zhu, S.: Ltl_f synthesis as AND-OR graph search: Knowledge compilation at work. In: IJCAI. pp. 2591–2598 (2022)
12. De Giacomo, G., Vardi, M.Y.: Synthesis for LTL and LDL on Finite Traces. In: IJCAI (2015)
13. Finkbeiner, B.: Synthesis of reactive systems. In: Dependable Software Systems Engineering, pp. 72–98 (2016)
14. Giacomo, G.D., Vardi, M.Y.: Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In: IJCAI (2013)
15. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: Pddl – the planning domain definition language – version 1.2. Tech. rep., TR-98-003, Yale Center for Computational Vision and Contro (1998)
16. Meyer, P.J., Sickert, S., Luttenberger, M.: Strix: Explicit reactive synthesis strikes back! In: Chockler, H., Weissenbacher, G. (eds.) CAV. pp. 578–586 (2018)
17. Pnueli, A.: The temporal logic of programs. In: FOCS. pp. 46–57 (1977)
18. Pnueli, A., Rosner, R.: On the Synthesis of a Reactive Module. In: POPL (1989)
19. Xiao, S., Li, J., Zhu, S., Shi, Y., Pu, G., Vardi, M.Y.: On-the-fly synthesis for LTL over finite traces. In: AAAI. pp. 6530–6537 (2021)
20. Zhu, S., De Giacomo, G., Pu, G., Vardi, M.Y.: LTL_f synthesis with fairness and stability assumptions. In: AAAI. pp. 3088–3095 (2020)
21. Zhu, S., Tabajara, L.M., Li, J., Pu, G., Vardi, M.Y.: A Symbolic Approach to Safety LTL Synthesis. In: HVC. pp. 147–162 (2017)
22. Zhu, S., Tabajara, L.M., Li, J., Pu, G., Vardi, M.Y.: Symbolic LTL_f Synthesis. In: IJCAI. pp. 1362–1369 (2017)

Probabilistic Judgment Aggregation with Conditional Independence Constraints

Magdalena Ivanovska[1][0000−0002−3916−3486] and Marija Slavkovik[2][0000−0003−2548−8623]

[1] BI Norwegian Business School, Oslo, Norway
magdalena.ivanovska@bi.no
[2] University of Bergen, Bergen, Norway
marija.slavkovik@uib.no

Abstract. Probabilistic judgment aggregation is concerned with aggregating judgments about probabilities of logically related issues. It takes as input imprecise probabilistic judgments over the issues given by a group of agents and defines rules of aggregating the individual judgments into a collective opinion representative for the group. The process of aggregation can be subject to constraints, i.e., aggregation rules can be required to satisfy certain properties. We explore how probabilistic independence constraints can be incorporated into the aggregation process.

Keywords: probabilistic logic · judgment aggregation · conditional independence.

1 Introduction

Judgment aggregation (JA) is concerned with aggregating judgments about the truth of logically related statements [10, 5]. Consider the following example of a *discursive dilemma*: A committee of three academics is deciding on whether to award tenure to a candidate based on excellence in research (r), teaching (t), and service (s). As we can see in the table below, using the majority aggregation rule gives an inconsistent outcome. Suppose that, instead of judging categorically

	r	t	s	$r \wedge t \wedge s$
A_1	true	false	true	false
A_2	false	true	true	false
A_3	true	true	false	false
maj	true	true	true	→true ↓false

on the candidate performing well in research, teaching, service, and tenure, the committee members express their *beliefs* over $\{r, t, s, r \wedge t \wedge s\}$ as numbers from the $[0, 1]$ interval, and the aggregation is done by a threshold majority rule. This

not only offers a more flexible representation framework, but also increases the possibilities for consistent aggregation. The probabilistic judgment aggregation we introduce in [8] takes this a step further and allows for imprecise probabilities over the issues. In [8] we define several classes of aggregators and properties that they might satisfy.

The case of mutually exclusive and exhaustive set of issues and precise probabilities, reduces to *opinion pooling* [3]. An impossibility result in opinion pooling states that the only aggregation rules that satisfy some basic desirable properties are the linear pools. However, linear pools do not in general preserve probabilistic independence of events. It is not hard to imagine that the committee members' assumptions about a candidate's abilities in research influence those in teaching and vice versa. A committee member might think that the candidate that performs well in research can not possibly do well in teaching due to lack of time and focus; or that bad teaching performance indicates poor research abilities as well; or that these two abilities are independent on each other. We are interested in representing the opinions about conditional independencies among the issues and incorporating them into the aggregation process.

2 Probabilistic judgement aggregation framework

We use the framework introduced in [8] and modified in [9]. Let $\mathcal{L}$ be a set of propositional logic formulas. An *agenda* is a finite set of *issues* $\Phi \subset \mathcal{L}$,

$$\Phi = \{\varphi_1, \ldots, \varphi_m\} , \tag{1}$$

where φ_i, $i = 1, \ldots, m$, is neither a tautology nor a contradiction. We call $\Phi^{\cup} = \Phi \cup \{\neg\varphi \mid \varphi \in \Phi\}$ the *extended agenda* of Φ. A *likelihood judgement* on the issue $\varphi \in \Phi^{\cup}$ is a simple likelihood formula of the type $\ell(\varphi) \geq a$, where $a \in [0, 1]$, expressing that the *likelihood* (interpreted as probability in this paper) of the statement φ being true is at least a. This formula is an instance of the logic of likelihood (see [4] and [6]), the language of which consists of Boolean combinations of linear likelihood formulas of the type

$$a_1\ell(\varphi_1) + \ldots + a_m\ell(\varphi_m) \geq b , \tag{2}$$

where a_i, b are real numbers, and φ_i are pure propositional formulas.[3] The semantics is provided by probability spaces, where $\ell(\varphi)$ is interpreted as the probability of the event (set of worlds) where φ is true. In [4], the authors provide a sound and complete axiomatic system which is a combination of axioms for propositional reasoning, reasoning about inequalities, and probability axioms.

We model the information sources as sets of likelihood judgements on $\Phi^{\cup}$. Given a set of n agents $N = \{1, \ldots, n\}$, $n \geq 2$, a *likelihood profile* is the tuple $\hat{P} = (\hat{J}_1, \ldots, \hat{J}_n)$, where $\hat{J}_k$ is the set of likelihood judgements of the agent $k \in N$:

$$\hat{J}_k = \{\ell(\varphi) \geq a_k(\varphi) \mid \varphi \in \Phi^{\cup}\} , \tag{3}$$

[3] Expressions with other inequalities or equality can be defined as abbreviations.

where $a_k(\varphi) \in [0,1]$ are called the *judgement coefficients* of the k-th agent. Each agent provides judgment coefficients (probability lower bounds) for both an issue φ and its negation which, in accordance with the probability axioms, provides a likelihood interval for the issue. In the cases where $a_k(\varphi)+a_k(\neg\varphi) = 1$, these intervals collapse into a point, i.e., we obtain precise likelihood judgments. Abstention on φ can be modelled by taking $a_k(\varphi) \geq 0$ and $a_k(\neg\varphi) \geq 0$.

A probabilistic judgement set $\hat{J}$ is *rational* if it is consistent (as a set of formulas in the logic of likelihood) and *final* (it does not imply stronger judgments, i.e., narrower likelihood intervals, than the ones it contains. The full details about the probabilistic JA framework can be found in [9].

3 Conditional independence constraints

Conditional probability can be expressed in the logic of likelihood: $\ell(\varphi|\psi) \geq a$ as an abbreviation of $\ell(\varphi \wedge \psi) - a\ell(\psi) \geq 0$. However, as noticed in [6], probabilistic independence is not expressible in this logic since it requires multiplication of likelihood terms. One can extend the logic by defining polynomial likelihood formulas, or by adding conditional independence statements directly in the language. We choose to do the latter, as a more intuitive way of expression that is also compatible with probabilistic graphical models like Bayesian networks.

Definition 1. *Given an alphabet $\mathcal{L}$, a conditional independence (CI) statement is a formula of the form*

$$I(\alpha, \beta|\gamma), \tag{4}$$

where α, β, and γ are pure propositional formulas. We read it as: "The likelihood of α is independent of the truth of β if γ is true." If $\gamma = \top$, the independence is unconditional and we denote it by $I(\alpha, \beta)$.

In a given probability space, the statement (4) is interpreted through conditional independence of events. The following theorem is a direct consequence of this interpretation.

Theorem 1. *Let α, β, β_1, β_2, and γ be propositions over a language $\mathcal{L}$. Then:*

(CI1) *If $I(\alpha, \beta|\gamma)$, then $I(\beta, \alpha|\gamma)$,*
(CI2) *$I(\alpha, \top|\gamma)$,*
(CI3) *If $I(\alpha, \beta|\gamma)$, then $I(\alpha, \neg\beta|\gamma)$,*
(CI4) *If $\beta_1 \wedge \beta_2 = \bot$, $I(\alpha, \beta_1|\gamma)$, and $I(\alpha, \beta_2|\gamma)$, then $I(\alpha, \beta_1 \vee \beta_2|\gamma)$,*
(CI5) *$I(\alpha, \beta|\gamma)$ iff $I(\alpha, \beta \wedge \gamma|\gamma)$.*

Using the theorem, we can prove that independence of logically related issues is only possible in the special case when the issues are either true or false.

Proposition 1. *Let α and β be two logically related issues. (There are truth value assignments of α and β that cannot co-exist.) Then $I(\alpha, \beta|\gamma)$ iff $\ell(\alpha|\gamma) = a$ or $\ell(\beta|\gamma) = b$, where $a, b \in \{0, 1\}$.*

Given an alphabet $\mathcal{L}$ and an agenda Φ, we will call the elements of $\mathcal{L} \cap \Phi$ *basic issues*, and the rest of the elements of Φ we call *composite issues*. As noticed in [2], it is more intuitive to assume that the agents will have an opinion about probabilistic independence of the basic issues. This is partly supported by the above theorem and proposition. Hence, we can limit ourselves to only representing probabilistic independence judgments over the basic issues in the agenda. An exception can be made in the conditioning proposition γ in (4) if we, for example, want to represent *context-specific independence*, i.e., situations where only one of $I(\alpha, \beta|\gamma)$ and $I(\alpha, \beta|\neg\gamma)$ holds. For example, we might think that, given that the candidate is good in research, the probability of good teaching performance is independent of good community service, $I(t, s|r)$. However, if we know that the research is not good, then knowing the status of teaching might change the probability of good service, i.e., $I(t, s|\neg r)$ is not the case.

We suggest including judgments about probabilistic independence in the form of (conditional) independence statements either as: 1. part of the judgment profile, i.e., independence statements given by each agent in addition to its likelihood judgments; or as 2. aggregation constraints imposed by an *agenda setter*. In the first case, the CI statements of each agent should be consistent with its likelihood statements (as an additional rationality requirement of the profile) and they should be prioritized in the aggregation process: The collective CI statements are derived first, then the likelihood judgments are updated accordingly, after which they are aggregated. In the second case, the aggregated CI statements are applied on the aggregated likelihoods only.

In order to update the likelihoods, similarly as in [7], we introduce the following axiom that connects independence and likelihood formulas:

(LCI) From $I(\alpha, \beta|\gamma)$ and $\ell(\alpha|\gamma) \geq (\leq) a$, it follows $\ell(\alpha|\beta \wedge \gamma) \geq (\leq) a$.

The above axiom together with the other axioms of the logic of likelihood is used for updating either the individual or the collective likelihood judgements.

4 Conclusions and related work

We introduce a way of representing opinions about probabilistic independence of events in the probabilistic JA profiles. We suggest methods of aggregation that prioritize the qualitative information about conditional independence over the quantitative information about likelihoods of statements, as a more reliable one.

Bradley et al. [1] consider aggregating probabilistic judgments expressed as causal Bayesian networks. The aggregation is done by first aggregating the individual graphs into a single one, then refactoring the individual probability distributions according to the resulting graph, and at last aggregating the respective conditional probability distributions. It is possible to follow their ideas in order to aggregate the CI information. The axiomatic approach, however, provides the possibility for representing context-specific independences as well.

References

1. Bradley, R., Dietrich, F., List, C.: Aggregating causal judgments. Philosophy of Science **81**(4), 491–515 (2014), http://www.jstor.org/stable/10.1086/678044
2. Dietrich, F., List, C.: Probabilistic opinion pooling generalized. part two: the premise-based approach. Social Choice and Welfare **39**, 787–814 (2017)
3. Dietrich, F., List, C.: Probabilistic opinion pooling. In: Hajek, A., Hitchcock, C. (eds.) Oxford Handbook of Philosophy and Probability. Oxford: Oxford University Press (2016)
4. Fagin, R., Halpern, J.Y., Megiddo, N.: A Logic for Reasoning about Probabilities. Information and Computation **87**, 78–128 (1990). https://doi.org/10.1016/0890-5401(90)90060-U
5. Grossi, D., Pigozzi, G.: Judgment Aggregation: A Primer. Morgan and Claypool Publishers, San Rafael, CA, USA (2014). https://doi.org/10.2200/S00559ED1V01Y201312AIM027
6. Halpern, J.Y.: Reasoning about uncertainty. MIT Press (2005), https://mitpress.mit.edu/books/reasoning-about-uncertainty-second-edition
7. Ivanovska, M., Giese, M.: Probabilistic logic with conditional independence formulae. In: Proceedings of ECAI 2010 - 19th European Conference on Artificial Intelligence. pp. 983–984 (2010). https://doi.org/10.3233/978-1-60750-606-5-983
8. Ivanovska, M., Slavkovik, M.: Aggregating probabilistic judgments. In: Moss, L.S. (ed.) Proceedings Seventeenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2019. EPTCS, vol. 297, pp. 273–292 (2019). https://doi.org/10.4204/EPTCS.297.18, https://doi.org/10.4204/EPTCS.297.18
9. Ivanovska, M., Slavkovik, M.: Probabilistic judgement aggregation by opinion update. In: Torra, V., Narukawa, Y. (eds.) Modeling Decisions for Artificial Intelligence - 19th International Conference, MDAI 2022, Sant Cugat, Spain, August 30 - September 2, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13408, pp. 26–37. Springer (2022). https://doi.org/10.1007/978-3-031-13448-7_3, https://doi.org/10.1007/978-3-031-13448-7_3
10. List, C., Puppe, C.: Judgment aggregation: A survey. In: Anand, P., Puppe, C., Pattanaik, P. (eds.) The Handbook of Rational and Social Choice. Oxford University Press, UK (2009). https://doi.org/10.1093/acprof:oso/9780199290420.003.0020

LTL$_f$ Synthesis Under Environment Specifications for Reachability and Safety Properties

Benjamin Aminof[1], Giuseppe De Giacomo[1,2], Antonio Di Stasio[2], Hugo Francon[3], Sasha Rubin[4], and Shufang Zhu[2]

[1] Sapienza University of Rome, Italy
benj@forsyte.at
[2] University of Oxford, UK
{giuseppe.degiacomo, antonio.distasio, shufang.zhu}@cs.ox.ac.uk
[3] ENS Rennes, France
hugo.francon@ens-rennes.fr
[4] The University of Sydney, Australia
sasha.rubin@sydney.edu.au

Abstract. In this paper, we study LTL$_f$ synthesis under environment specifications for arbitrary reachability and safety properties. We consider both kinds of properties for both agent tasks and environment specifications, providing a complete landscape of synthesis algorithms. For each case, we devise a specific algorithm (optimal wrt complexity of the problem) and prove its correctness. All these algorithms adopt some common building blocks, though combining them in different ways. While some cases are already studied in literature others are studied here for the first time.

1 Introduction

Synthesis under environment specifications consists of synthesizing an agent strategy (aka plan or program) that realizes a given task against all possible environment responses (i.e., environment strategies). The agent has some indirect knowledge of the possible environment strategies through an environment specification, and it will use such knowledge to its advantage when synthesizing its strategy [1,2,14]. This problem is tightly related to planning in adversarial nondeterministic domains [11], as discussed, e.g., in [3,8].

In this paper, we study synthesis under environment specifications, considering both *agent task specifications* and *environment specifications* expressed in Linear Temporal Logic on finite traces (LTL$_f$). These are logics that look at finite traces or finite prefixes of infinite traces. For concreteness, we focus on LTL$_f$ [9,10], but the techniques presented here extend immediately to other temporal logics on finite traces, such as Linear Dynamic Logics on finite traces, which is more expressive than LTL$_f$ [9], and Pure-Past LTL, which has the same expressiveness as LTL but evaluates a trace backward from the current instant towards the initial instant [4].

Linear temporal logics on finite traces provide a nice embodiment of the notable triangle among Logics, Automata, and Games [12]. These logics are full-fledged logics with high expressiveness over finite traces, and they can be translated into classical regular finite state automata; moreover, they can be further converted into deterministic finite state automata (DFAs). This transformation yields a game represented on a graph. In this game, one can analyze scenarios where the objective is to reach certain final states. Finally, despite the fact that producing a DFA corresponding to an LTL_f formula can require double-exponential time, the algorithms involved — generating alternating automata (linear), getting the nondeterministic one (exponential), determinizing it (exponential), playing games (poly) — are particularly well-behaved from the practical computational point of view [15,16,20].

In this paper, however, we consider LTL_f specifications in two contexts which we denote as

$$\exists\varphi \text{ and } \forall\varphi \text{ with } \varphi \text{ an arbitrary } \text{LTL}_f \text{ formula}$$

The first one specifies a *reachability* property: there exists a finite prefix $\pi_{<k}$ of an infinite trace π such that $\pi_{<k} \models \varphi$. This is the classical use of LTL_f to specify synthesis tasks [10]. The second one specifies a *safety* property: every finite prefix $\pi_{<k}$ of an infinite trace π is such that $\pi_{<k} \models \varphi$. This is the classical use of LTL_f to specify environment behaviours [1,6]. The formulas $\forall\varphi$ and $\exists\varphi$ with φ in LTL_f capture exactly two well-known classes of LTL properties in Manna and Pnueli's Temporal Hierarchy [13]. Specifically, $\exists\varphi$ captures the *co-safety properties* and $\forall\varphi$ captures the *safety properties* (in [13], expressed respectively as $\Diamond\psi$ and $\Box\psi$ with ψ an arbitrary Pure-Past LTL formulas, which consider only past operators.)

We let Env and Task denote an environment specification and a task specification, respectively, consisting of a safety ($\forall\varphi$) and/or reachability property ($\exists\varphi$). This gives rise to 12 possible cases: 3 without any environment specifications, 3 with safety environment specifications ($\forall\varphi$), 3 with reachability environment specifications ($\exists\varphi$), and 3 with both safety and reachability environment specifications ($\exists\varphi \wedge \forall\varphi$). For each of these, we provide an algorithm, which is optimal wrt the complexity of the problem, and prove its correctness. When the problem was already solved in literature, we give appropriate references (e.g., $\mathsf{Task} = \exists\varphi$ and $\mathsf{Env} = \mathit{true}$ is classical LTL_f synthesis, solved in [10]). In fact, we handle all the cases involving reachability in the environment specifications by providing a novel algorithm that solves the most general case of $\mathsf{Env} = \exists\varphi_1 \wedge \forall\varphi_2$ and $\mathsf{Task} = \exists\varphi_3 \wedge \forall\varphi_4$.[5]

All these algorithms use the same common building blocks, though combining them in different ways: the construction of the DFAs of the LTL_f formulas, Cartesian products of such DFAs, consider these DFAs as the game arena and play games for reachability/safety objectives. Also, all these problems have a 2EXPTIME-complete complexity. The hardness comes from LTL_f synthesis [10], and the membership comes from the LTL_f-to-DFA construction, which domi-

[5] In fact, this algorithm can solve all cases, but it's much more involved compared to the direct algorithms we provide for each case.

nates the complexity since computing the Cartesian products and solving reachability/safety games is polynomial.[6] Towards the actual application of our algorithms, we observe that although the DFAs of LTL$_f$ formulas are worst-case double-exponential, there is empirical evidence showing that the determinization of NFA, which causes one of the two exponential blow-ups, is often polynomial in the NFA [16,20]. Moreover, in several notable cases, e.g., in all DECLARE patterns [17], the DFAs are polynomial in the LTL$_f$ formulas, and so are our algorithms.

It is worth noting that all the cases studied here are specific Boolean combinations of $\exists\varphi$ and $\forall\phi$. It is of interest to indeed devise algorithms to handle arbitrary Boolean combinations. Indeed, considering that LTL$_f$ is expressively equivalent to pure-past LTL, an arbitrary Boolean combination of $\exists\varphi$ and $\forall\phi$ would correspond to a precise class of LTL properties in Manna & Pnueli's Temporal Hierarchy [13]: the so-called *obligation* properties. We leave this interesting research direction for future work.

2 Problem Description

Reactive Synthesis. Reactive Synthesis (aka Church's Synthesis) is the problem of turning a specification of an agent's task and of its environment into a strategy (aka policy). This strategy can be employed by the agent to achieve its task, regardless of how the environment behaves. In this framework, the agent and the environment are considered players in a turn-based game, in which players move by picking an evaluation of the propositions they control. Thus, we partition the set *Prop* of propositions into two disjoint sets of propositions $\mathcal{X}$ and $\mathcal{Y}$, and with a little abuse of notation, we denote such a partition as $Prop = \mathcal{Y} \cup \mathcal{X}$. Intuitively, the propositions in $\mathcal{X}$ are controlled by the environment, and those in $\mathcal{Y}$ are controlled by the agent. In this work (in contrast to the usual setting of reactive synthesis), the agent moves first. The agent moves by selecting an element of $2^{\mathcal{Y}}$, and the environment responds by selecting an element of $2^{\mathcal{X}}$. This is repeated forever, and results in an infinite trace (aka play).

An *agent strategy* is a function $\sigma_{\mathsf{ag}} : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$. An environment strategy is a function $\sigma_{\mathsf{env}} : (2^{\mathcal{Y}})^+ \to 2^{\mathcal{X}}$. A strategy σ is *finite-state* (aka *finite-memory*) if it can be represented as a finite-state input/output automaton that, on reading an element h of the domain of σ, outputs the action $\sigma(h)$. A trace $\pi = (Y_0 \cup X_0)(Y_1 \cup X_1)\ldots \in (2^{\mathcal{Y}\cup\mathcal{X}})^\omega$ *follows an agent strategy* $\sigma_{\mathsf{ag}} : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ if $Y_0 = \sigma_{\mathsf{ag}}(\epsilon)$ and $Y_{i+1} = \sigma_{\mathsf{ag}}(X_0 X_1 \ldots X_i)$ for every $i \geq 0$, and it *follows an environment strategy* σ_{env} if $X_i = \sigma_{\mathsf{env}}(Y_0 Y_1 \ldots Y_i)$ for all $i \geq 0$. We denote the unique infinite sequence (play) that follows σ_{ag} and σ_{env} as $\mathsf{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$. Let P be a property over the alphabet $\Sigma = 2^{Prop}$, specified by formula or DA. An agent strategy σ_{ag} (resp., environment strategy σ_{env}) *enforces* P if for every environment strategy σ_{env} (resp., agent strategy σ_{ag}), we have that $\mathsf{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is in P. In this case, we write $\sigma_{\mathsf{ag}} \rhd P$ (resp. $\sigma_{\mathsf{env}} \rhd P$). We say that P is *agent*

[6] For pure-past LTL, obtaining the DFA from a pure-past LTL formula is single exponential [4], and indeed the problems and all our algorithms become EXPTIME-complete.

(resp., environment) realizable if there is an agent (resp. environment) strategy that enforces P.

Synthesis under Environment Specifications. Typically, an agent has some knowledge of how the environment works, represented as a fully observable model of the environment, which it can exploit to enforce its task [2]. Formally, let Env and Task be properties over alphabet $\Sigma = 2^{Prop}$, denoting the environment specification and the agent task, respectively.

Note that while the agent task Task denotes the set of desirable traces from the agent's perspective, the environment specification Env denotes the set of environment strategies that describe how the environment reacts to the agent's actions (no matter what the agent does) in order to enforce Env. Specifically, Env is treated as a set of traces when we reduce the problem of synthesis under environment specification to standard reactive synthesis. We require a consistency condition of Env, i.e., there must exist at least one environment strategy that enforces Env.

An agent strategy σ_{ag} enforces Task *under the environment specification* Env, written $\sigma_{\mathsf{ag}} \rhd_{\mathsf{Env}} \mathsf{Task}$, if for all $\sigma_{\mathsf{env}} \rhd \mathsf{Env}$ we have that $\mathsf{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \mathsf{Task}$. Note that if $\mathsf{Env} = \mathit{true}$ then this just says that σ_{ag} enforces Task.

In [2] is shown that for any linear-time property, synthesis under environment specifications can be reduced to synthesis without environment specifications. Thus, in order to show that Task is realizable under Env it is sufficient to show that $\mathsf{Env} \rightarrow \mathsf{Task}$ is realizable. Moreover, to solve the synthesis problem for Task under Env, it is enough to return a strategy that enforces $\mathsf{Env} \rightarrow \mathsf{Task}$.

In this work, we provide a landscape of algorithms for LTL_f synthesis considering reachability and safety properties for both agent tasks and environment specifications. However, these synthesis problems are complex and challenging due to the combination of reachability and safety properties. To tackle this issue, one possible approach is to reduce LTL_f synthesis problems to LTL synthesis problems through suitable translations, e.g., [5,7,18,19]. However, there is currently no methodology for performing such translations when considering combinations of reachability and safety properties.

Additionally, synthesis algorithms for LTL specifications are generally more challenging than those for LTL_f specifications, both theoretically and practically [6,7,18,19]. In this work, we show that for certain combinations, we can avoid the detour to LTL synthesis and keep the simplicity of LTL_f synthesis. Specifically, we consider that Task and Env take the following forms:

$$\exists\varphi_1, \forall\varphi_1, \exists\varphi_1 \wedge \forall\varphi_2 \text{ where the } \varphi_i \text{ are } \text{LTL}_f \text{ formulas,}$$

and in addition we consider the case of no environment specification (formally, $\mathsf{Env} = \mathit{true}$). This results in 12 combinations. The algorithms developed in this work optimally solve all the combinations.

References

1. Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin. Planning and synthesis under assumptions. *CoRR*, 2018.
2. Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin. Planning under LTL environment specifications. In *ICAPS*, pages 31–39, 2019.
3. Alberto Camacho, Meghyn Bienvenu, and Sheila A. McIlraith. Towards a unified view of AI planning and reactive synthesis. In *ICAPS*, pages 58–67, 2019.
4. Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-past linear temporal and dynamic logic on finite traces. In *IJCAI*, pages 4959–4965, 2020.
5. Giuseppe De Giacomo, Antonio Di Stasio, Giuseppe Perelli, and Shufang Zhu. Synthesis with mandatory stop actions. In *KR*, pages 237–246, 2021.
6. Giuseppe De Giacomo, Antonio Di Stasio, Luca M. Tabajara, Moshe Y. Vardi, and Shufang Zhu. Finite-trace and generalized-reactivity specifications in temporal synthesis. In *IJCAI*, pages 1852–1858, 2021.
7. Giuseppe De Giacomo, Antonio Di Stasio, Moshe Y. Vardi, and Shufang Zhu. Two-stage technique for LTL$_f$ synthesis under LTL assumptions. In *KR*, pages 304–314, 2020.
8. Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTL$_f$ and LDL$_f$ goals. In *IJCAI*, pages 4729–4735, 2018.
9. Giuseppe De Giacomo and Moshe Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*, pages 854–860, 2013.
10. Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on Finite Traces. In *IJCAI*, pages 1558–1564, 2015.
11. Hector Geffner and Blai Bonet. *A Coincise Introduction to Models and Methods for Automated Planning*. Morgan&Claypool, 2013.
12. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
13. Zohar Manna and Amir Pnueli. A hierarchy of temporal properties. In *PODC*, pages 377–410, 1990.
14. Amir Pnueli and Roni Rosner. On the Synthesis of a Reactive Module. In *POPL*, pages 179–190, 1989.
15. Lucas M. Tabajara and Moshe Y. Vardi. Ltlf synthesis under partial observability: From theory to practice. In *GandALF*, volume 326 of *EPTCS*, pages 1–17, 2020.
16. Deian Tabakov and Moshe Y. Vardi. Experimental evaluation of classical automata constructions. In *LPAR*, volume 3835 of *Lecture Notes in Computer Science*, pages 396–411, 2005.
17. Michael Westergaard. Better algorithms for analyzing and enacting declarative workflow languages using LTL. In *BPM*, volume 6896 of *Lecture Notes in Computer Science*, pages 83–98, 2011.
18. Shufang Zhu, Giuseppe De Giacomo, Geguang Pu, and Moshe Y. Vardi. LTL$_f$ synthesis with fairness and stability assumptions. In *AAAI*, pages 3088–3095, 2020.
19. Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi. Symbolic LTL$_f$ Synthesis. In *IJCAI*, pages 1362–1369, 2017.
20. Shufang Zhu, Lucas M. Tabajara, Geguang Pu, and Moshe Y. Vardi. On the power of automata minimization in temporal synthesis. In *GandALF*, volume 346 of *EPTCS*, pages 117–134, 2021.

Reasoning about Exceeding Risk Threshold*

Maksim Gladyshev, Natasha Alechina, Mehdi Dastani, and Dragan Doder

Utrecht University, Utrecht, The Netherlands
{m.gladyshev, n.a.alechina, m.m.dastani, d.doder}@uu.nl

Abstract. The problem of tracing the responsibility for unsafe outcomes to decision-making actors in multi-agent systems is urgent. While all existing approaches focus on deterministic outcomes, assuming that (a group of) agents can be held responsible for φ only if φ actually happens and agents could act differently to prevent φ, we find this notion of responsibility insufficient in many scenarios. In this work we combine coalition ability operator $[G]$ from [12] with a probabilistic operator L_α from [5] that allow us to reason about probabilities and their changes. This approach allows us to claim that a group of agents can be held responsible for the unsafe outcome even if this outcome does not actually happen, but the group has caused its probability to be increased to an (unacceptably) high level. The proposed logic could be useful for analysing and assigning responsibility to groups of agents for their risky and unsafe behaviors. Finally, we establish (weak) completeness and decidability results for the proposed logic.

Keywords: Risk · Probabilistic Logic · Coalition Logic.

1 Introduction

Safety of AI systems is a well-recognised and important concern. In multi-agent settings, where autonomous agents interact in complex ways, it is important to not only be able to determine whether an unsafe outcome is possible in principle, but also, when such an outcome occurs, determine why it occurred, which actions by which agents have caused it, and whether it could have been prevented. The existing approaches [13, 10] assume that the responsibility for an (unsafe or undesirable) outcome can be assigned to a group of agents if the (unsafe or undesirable) outcome actually holds. However, unsafe outcomes are not necessarily the states of affairs where a bad event has actually happened, but also the states of affairs where the probability that the bad event happens is unacceptably high.

In order to formally investigate this broad notion of responsibility, we propose and investigate a logic combining coalition ability operator from [12] with a probabilistic operator from [5] that allow reasoning about probabilities and their changes. In this logic, we can express that the probability of an event is greater

* The extended version of this paper will appear in the proceedings of 20th International Conference on Principles of Knowledge Representation and Reasoning.

than a certain number. This allows us to analyse various aspects of AI systems that involve reasoning about risks and probabilistic uncertainty in general.

This paper belongs to a large body of work on combining modalities in order to analyse AI systems. There are well-known logics that combine temporal and probabilistic modalities as well as temporal and strategic. Recently, logics combining strategic modalities with probabilities have also been proposed, but they concentrate on probabilities of the outcomes of actions and strategies, rather than strategies to enforce a particular probability distribution [3, 11, 6, 9, 2].

2 Logic for Reasoning about Risk

2.1 Models

Definition 1 (CGS, pointed). *A concurrent game structure (CGS) is a tuple* $\Gamma = (\mathbb{AG}, S, Act, d, o)$*, comprising a nonempty finite set of all agents* $\mathbb{AG} = \{1, \dots, k\}$*, a nonempty finite set of states* S*, where* $S_0 \subseteq S$ *denotes the set of initial states, and a nonempty finite set of (atomic) actions* Act*. Function* $d : \mathbb{AG} \times S \longrightarrow \mathcal{P}(Act) \backslash \{\varnothing\}$ *defines nonempty sets of actions available to agents at each state, and* o *is a (deterministic) transition function that assigns the outcome state* $s' = o(s, (\alpha_1, \dots, \alpha_k))$ *to a state* s *and a tuple of actions* $(\alpha_1, \dots, \alpha_k)$ *with* $\alpha_i \in d(i, s)$ *and* $1 \leq i \leq k$*, that can be executed by* $\mathbb{AG}$ *in* s*. For* α_G *that is an action profile of a non-grand coalition* $G \subset \mathbb{AG}$*,* $o(s, \alpha_G)$ *is defined as the set containing all outcomes of* α_G *completed by actions of agents outside the coalition. A pointed CGS is given by* (Γ, s)*, where* Γ *is a CGS and* s *is a state in it.*

Given a CGS Γ, a positional (memoryless) *strategy* for an agent $a \in \mathbb{AG}$ or a-strategy, is a function $str_a : S \longrightarrow d(a, S)$. Given a coalition $G = \{a_1, \dots, a_m\}$, a positional strategy $str_G = \langle str_{a_1}, \dots, str_{a_m} \rangle$ maps each state from S to a tuple of actions $\langle str_{a_1}(s), \dots, str_{a_m}(s) \rangle$.

A model $\mathcal{M} = (\mathbb{AG}, S, Act, d, o, \mathsf{Past}, P, V)$ of our logic is a CGS endowed with a temporal relation Past, a probability function P and a valuation function V. For a temporal relation $\mathsf{Past} \subseteq S \times S$ we use $s' \in \mathsf{Past}(s)$ to denote $s \mathsf{Past} s'$, i.e. s' is one-step reachable from s by Past relation. We require that $\forall s, x, y \in S$: if $x \in \mathsf{Past}(s)$ and $y \in \mathsf{Past}(s)$, then $x = y$ i.e., each state has at most one temporal predecessor. By this reason we can use $s' \in \mathsf{Past}(s)$ and $s' = \mathsf{Past}(s)$ interchangeably. We use this extension to ensure that given a state $s \in S$ we can always identify the unique previous state $s' = \mathsf{Past}(s)$. This assumption is important since verifying responsibility requires evaluating strategic power of the agents on the previous step. To guarantee that this temporal relation Past and a transition function o are aligned, we impose the following constraints:

R0 $\forall s \in S, s_0 \in S_0 : s_0 \neq o(s, str_{\mathbb{AG}})$ for any strategy $str_{\mathbb{AG}}$
R1 $\forall s, s' \in S : s' \in \mathsf{Past}(s)$, then $s \notin S_0$
R2 $s \notin S_0 \Rightarrow \exists s' \in S : s' \in \mathsf{Past}(s)$
R3 $s' \in \mathsf{Past}(s) \Rightarrow \exists str_{\mathbb{AG}}, s = o(s', str_{\mathbb{AG}})$

Intuitively, R0 states that the grand coalition cannot enforce an initial state. R1 means that initial states have no past. Property R2 means that non-initial states have a past. And R3 implies that if s' is the past of s, then the grand coalition must be able to move from s' to s. As a result of this semantic choice each initial state $s_0 \in S_0$ generates a tree of transitions: each state has a unique Past predecessor and a non-empty set of o-successors.

We also require that our model is also endowed with a probability function $P : S \mapsto (2^S \mapsto [0,1])$ assigning each state with a probability measure on S. Every $P(s)$ must satisfy the following conditions for all $s \in S$:

P1 $P(s)(S) = 1$,
P2 $P(s)(\varnothing) = 0$,
P3 $P(s)$ is (finitely) additive, i.e.
$P(s)(\bigcup_{0 \le i \le m} X_i) = \sum_{0 \le i \le m} P(s)(X_i)$,
where $X_i \cap X_j = \varnothing$ for any $i \neq j$,
P4 $P(s)$ is reflexive, i.e. $P(s)(\{s\}) > 0$,
P5 $P(s)(\{s'\}) > 0$ implies $P(s) = P(s')$.

The first three conditions are standard properties of probability, and reflexivity (the actual state of affairs has a non-zero probability) is a natural property of probability measure associated with states. Condition P5 enforces the fact that given a state $s \in S$ and another state $s' \in S$, such that s assigns s' a non-negative probability (i.e. s' belongs to the support set of S), it holds that s and s' share the same counterfactual probabilistic information, so $P(s) = P(s')$. Finally, V is a standard valuation function $V : Prop \longrightarrow 2^S$.

2.2 Language and Semantics

In this section we introduce a logic for reasoning about group responsibility for taking risk denoted GRR.

Definition 2 (Language). *The language of* GRR *is defined by the following grammar*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid L_\alpha\varphi \mid [G]\varphi \mid \boxminus\varphi,$$

where p ranges over $Prop \cup \{init\}$, G ranges over $2^{\mathbb{AG}}$ and $\alpha \in \mathbb{Q} \cap [0,1]$.

We use this special proposition variable *init* to distinguish initial states. $L_\alpha\varphi$ operator means "probability of φ is at least α". Derived operators $M_\alpha\varphi$ "probability of φ is at most α" and $I_\alpha\varphi$ "probability of φ is equal (identical) to α" can be defined as $M_\alpha\varphi \equiv L_{1-\alpha}\neg\varphi$ and $I_\alpha\varphi \equiv L_\alpha\varphi \wedge M_\alpha\varphi$ respectively. It also follows that $\neg L_\alpha\varphi$ and $\neg M_\alpha\varphi$ can be read as 'probability of φ is strictly smaller than α' and 'probability of φ is strictly greater than α' respectively. Formula $[G]\varphi$ reads as "group G can enforce φ to be true" and the formula $\boxminus\varphi$ reads as "φ was true at the previous step". The Boolean connectives $\vee, \rightarrow, \leftrightarrow, \bot$ and $\top$ are defined in the usual manner using $\neg$ and $\wedge$. The dual operator for $\boxminus$ is defined in the standard way: $\diamondminus\varphi \equiv \neg\boxminus\neg\varphi$.

Axioms:		(A2)	$L_\alpha \top$
(Taut)	All propositional tautologies	(A5)	$L_\alpha\varphi \to \neg L_\beta \neg\varphi$, where $\alpha+\beta>1$
(CL1)	$\neg[G]\bot$	(A8)	$\neg L_\alpha\varphi \to M_\alpha\varphi$
(CL2)	$[G]\top$	(T)	$L_1\varphi \to \varphi$
(CL3)	$\neg[\varnothing]\neg\varphi \to [\mathbb{AG}]\varphi$	(4')	$L_\alpha\varphi \to L_1 L_\alpha\varphi$
(CL4)	$[G](\varphi\wedge\psi)\to[G]\varphi$	(5')	$\neg L_\alpha\varphi \to L_1\neg L_\alpha\varphi$
(CL5)	$[G_1]\varphi\wedge[G_2]\psi\to[G_1\cup G_2](\varphi\wedge\psi)$, where $G_1\cap G_2=\varnothing$	**Rules:**	
		(MP)	From φ and $\varphi\to\psi$, infer ψ
(CL6)	$\neg[\mathbb{AG}]init$	(Eq)	From $\varphi\leftrightarrow\psi$, infer $[G]\varphi\leftrightarrow[G]\psi$
($K_\boxminus$)	$\boxminus(\varphi\to\psi)\to(\boxminus\varphi\to\boxminus\psi)$	($\text{Nec}_\boxminus$)	From φ, infer $\boxminus\varphi$
($U_\boxminus$)	$\diamondminus\varphi\to\boxminus\varphi$	(A6)	From $\varphi\leftrightarrow\psi$ infer $L_\alpha\varphi\leftrightarrow L_\alpha\psi$
($1_\boxminus$)	$init\to\boxminus\bot$	(B)	From $(\varphi_1,\dots,\varphi_m)\leftrightarrow(\psi_1,\dots,\psi_n)$, infer $\bigwedge_{i=1}^{m} L_{\alpha_i}\varphi_i \wedge \bigwedge_{j=2}^{n} M_{\beta_j}\psi_j \to L_\gamma\psi_1$, where $\gamma=(\alpha_1+\dots+\alpha_m)-(\beta_2+\dots+\beta_n)$
($2_\boxminus$)	$\neg init\to\diamondminus\top$		
($3_\boxminus$)	$\neg init\wedge\varphi\to\boxminus[\mathbb{AG}]\varphi$		
(A1)	$L_0\varphi$		

Table 1. The proof system for GRR.

Definition 3 (Semantics). *Given a model $\mathcal{M}$ and a state $s\in S$ we define $\vDash$ relation in the following way:*
$\mathcal{M},s\vDash p$ *iff* $s\in V(p)$;
$\mathcal{M},s\vDash\neg\varphi$ *iff* $\mathcal{M},s\nvDash\varphi$;
$\mathcal{M},s\vDash\varphi\wedge\psi$ *iff* $\mathcal{M},s\vDash\varphi$ *and* $\mathcal{M},s\vDash\psi$;
$\mathcal{M},s\vDash[G]\varphi$ *iff there is a strategy* str_G *for* G, *such that for all* $s'\in o(s,str_G)$ *it holds that* $\mathcal{M},s'\vDash\varphi$;
$\mathcal{M},s\vDash L_\alpha\varphi$ *iff* $P(s)([\varphi]^{\mathcal{M}})\geq\alpha$[1];
$\mathcal{M},s\vDash\boxminus\varphi$ *iff* $\forall s'\in\mathsf{Past}(s): M,s'\vDash\varphi$.

Now, we can establish the following results. The detailed overview can be found in the extended version of this paper.

Theorem 1 (Completeness). *Logic* GRR *is complete, i.e.* $\vDash\varphi$ *iff* $\vdash_{\mathsf{GRR}}\varphi$.

Theorem 2 (Decidability). *The satisfiability problem for* GRR *is decidable.*

3 Conclusion and Future Work

Our work also has some limitations that inspire directions for future research. We build GRR over a Coalition logic CL which is essentially a Next-fragment of ATL logic [1]. The choice of the use of Coalition logic was to consider strategic ability in the simplest abstract setting. Other work, e.g., Yazdanpanah et al. [13], have used ATL to define a group of agents responsible for an outcome if the group had an alternative to prevent the outcome at some point in the past. Another direction of future research is incorporating imperfect information in the spirit of [7, 8, 4, 10]. This however involves solving the problem of axiomatising CL or ATL under the strongly uniform strategies semantics.

[1] Here $[\varphi]^{\mathcal{M}}$ abbreviates for $\{s\in S\mid\mathcal{M},s\vDash\varphi\}$.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (sep 2002). https://doi.org/10.1145/585265.585270, https://doi.org/10.1145/585265.585270
2. Aminof, B., Kwiatkowska, M., Maubert, B., Murano, A., Rubin, S.: Probabilistic strategy logic. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019. pp. 32–38 (2019)
3. Bulling, N., Jamroga, W.: What agents can probably enforce. Fundam. Informaticae **93**(1-3), 81–96 (2009)
4. Fervari, R., Herzig, A., Li, Y., Wang, Y.: Strategically knowing how. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. pp. 1031–1038 (2017). https://doi.org/10.24963/ijcai.2017/143, https://doi.org/10.24963/ijcai.2017/143
5. Heifetz, A., Mongin, P.: Probability logic for type spaces. Games and economic behavior **35**(1-2), 31–53 (2001)
6. Huang, X., Su, K., Zhang, C.: Probabilistic alternating-time temporal logic of incomplete information and synchronous perfect recall. In: Hoffmann, J., Selman, B. (eds.) Proceedings of the Twenty-Sixth AAAI. AAAI Press (2012)
7. Jamroga, W.: Some remarks on alternating temporal epistemic logic. In: Dunin-Keplicz, B., Verbrugge, R. (eds.) Proceedings of Formal Approaches to Multi-Agent Systems (FAMAS 2003). pp. 133–139 (2003)
8. Jamroga, W., van der Hoek, W.: Agents that know how to play. Fundamenta Informaticae **63**(2-3), 185–219 (2004)
9. Naumov, P., Tao, J.: Knowing-how under uncertainty. Artificial Intelligence **276**, 41–56 (2019). https://doi.org/10.1016/j.artint.2019.06.007, https://doi.org/10.1016/j.artint.2019.06.007
10. Naumov, P., Tao, J.: An epistemic logic of blameworthiness. Artificial Intelligence **283**, 103269 (2020). https://doi.org/https://doi.org/10.1016/j.artint.2020.103269
11. Novák, P., Jamroga, W.: Agents, actions and goals in dynamic environments. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011)
12. Pauly, M.: A Modal Logic for Coalitional Power in Games. Journal of Logic and Computation **12**(1), 149–166 (02 2002). https://doi.org/10.1093/logcom/12.1.149, https://doi.org/10.1093/logcom/12.1.149
13. Yazdanpanah, V., Dastani, M., Jamroga, W., Alechina, N., Logan, B.: Strategic responsibility under imperfect information. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. p. 592–600. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2019)

First order synthesis for data words revisited

Julien Grange[1][0009−0005−0470−1781] and Mathieu Lehaut[2][0000−0002−6205−0682]

[1] Univ Paris Est Creteil, LACL, F-94010 Creteil, France
julien.grange@lacl.fr
[2] University of Gothenburg, Sweden
lehaut@chalmers.se

Abstract. We carry on the study of the synthesis problem on data words for fragments of first order logic, and delineate precisely the border between decidability and undecidability.[3]

The reactive synthesis problem, dating back to Church [4], is about generating a *correct-by-construction* program with respect to a given specification. It is often formulated as a two-player game between an uncontrollable Environment and the System that alternate picking an input and an output letter respectively. This creates an infinite execution, and the goal of the System is to make every execution satisfy the specification, whatever Environment does. If the System has a strategy to ensure this result, it then corresponds to a program that is sure to respect the specification. The original problem is decidable and was solved by Büchi and Landweber [3], and several improvements and extensions have since been studied. However, it only encompasses finite alphabets, which are inadequate for representing executions of distributed systems involving a number of processes which is not fixed; this occurs in communication protocols, distributed algorithms, multi-agent systems, swarm robotics, or with ad-hoc networks.

We thus consider an extension of this problem that deals with alphabets whose size is not fixed. For that, we use data words, introduced in [2], to represent executions. System and Environment have disjoint finite sets of actions A_{S} and A_{E}. As for the processes, we distinguish between processes from $\mathbb{P}_{\mathrm{SE}}$ that can be activated by both players, and processes from $\mathbb{P}_{\mathrm{S}}$ (resp. $\mathbb{P}_{\mathrm{E}}$) that can only be played by System (resp. Environment). These sets of processes are finite, but their size is unbounded. A *data word* is a finite or infinite sequence $w = (a_0, p_0)(a_1, p_1)\ldots$ over $A_{\mathrm{S}} \times (\mathbb{P}_{\mathrm{S}} \cup \mathbb{P}_{\mathrm{SE}}) \cup A_{\mathrm{E}} \times (\mathbb{P}_{\mathrm{E}} \cup \mathbb{P}_{\mathrm{SE}})$.

It remains to choose a formalism to express the specification that should be satisfied by System. As always, there is a trade off between expressiveness of the formalism and tractability of the synthesis problem. Many specification languages for data words have been studied, and the synthesis problem has been studied for some of them, such as register automata [6] and the Logic of Repeating Values [5]. Here we follow the steps of [1] and consider first order logic FO and its fragment FO^2 where only two reusable variables are allowed. A data word is seen as a structure whose domain is the set of all the positions (a_i, p_i),

[3] A longer version of this paper is available on *arXiv*.

with a binary predicate $\sim$ such that $x \sim y$ if the two positions x and y belong to the same process. On top of that, we consider the binary relations $<$ and $+1$, corresponding to the order and the successor in the sequence. Moreover, a unary predicate is introduced for each action from A_{S} and A_{E}, to mark the positions where this action has been played. On this basis, we consider a variety of logics $\mathrm{FO}[\sim]$, $\mathrm{FO}^2[\sim, <, +1]$, $\mathrm{FO}^2[\sim, <]$, etc., depending on which binary predicates are allowed.

A *strategy* for System is a function $\mathcal{S}$ that, given a finite data word (representing the history of the game), outputs some move in $A_{\mathrm{S}} \times (\mathbb{P}_{\mathrm{S}} \cup \mathbb{P}_{\mathrm{SE}}) \cup \{\varepsilon\}$. A data word w is said to be an $\mathcal{S}$-compatible execution if each move from System in w is consistent with $\mathcal{S}$ wrt. the history preceding this position; furthermore if w is finite, $\mathcal{S}(w) = \varepsilon$. With this definition we intuitively allow Environment to block System from playing anytime Environment wants to play, which could potentially be forever. To prevent pathological cases, we will consider only *fair* executions: if during the execution, System asked infinitely often to make an action (different from ε), then the execution has infinitely many actions from System. Finally, we say that $\mathcal{S}$ is *winning* if all executions that are $\mathcal{S}$-compatible and fair satisfy φ.

In this paper we focus on two specific configurations for processes: when all processes are shared, and when they are partitioned between players. We say that the processes are *shared* when $\mathbb{P}_{\mathrm{S}} = \mathbb{P}_{\mathrm{E}} = \emptyset$, i.e. when all processes can be affected both by System and Environment. The synthesis problem for logic $\mathcal{L}$ with shared processes is denoted by $\textsc{SharedSynth}(\mathcal{L})$: it amounts, given an alphabet $A_{\mathrm{S}} \cup A_{\mathrm{E}}$ of actions and a formula $\varphi \in \mathcal{L}$, to decide whether there exists a finite set of processes $\mathbb{P}_{\mathrm{SE}}$ and a winning strategy $\mathcal{S}$ for φ. Since the identity of the shared processes does not matter and only the cardinality of $\mathbb{P}_{\mathrm{SE}}$ is relevant, we say in that case that $\mathcal{S}$ is a $|\mathbb{P}_{\mathrm{SE}}|$-winning strategy for φ. On the other hand, we say that the processes are *partitioned* if $\mathbb{P}_{\mathrm{SE}} = \emptyset$. In that case, each player has their own pool of processes on which they can play, but that their opponent cannot use. The synthesis problem for logic $\mathcal{L}$ with partitioned processes is denoted $\textsc{PartSynth}(\mathcal{L})$. As above, it is the problem of deciding, given $A_{\mathrm{S}} \cup A_{\mathrm{E}}$ and $\varphi \in \mathcal{L}$, whether System has a winning strategy for φ with $\mathbb{P}_{\mathrm{SE}} = \emptyset$ and some arbitrary finite sets $\mathbb{P}_{\mathrm{S}}$ and $\mathbb{P}_{\mathrm{E}}$. Similarly, we say in that case that $\mathcal{S}$ is a $(|\mathbb{P}_{\mathrm{S}}|, |\mathbb{P}_{\mathrm{E}}|)$-winning strategy for φ.

Bojańczyk et al. [2] proved that the satisfiability problem for $\mathrm{FO}^2[\sim, <, +1]$ on data words is decidable. Note that this corresponds to the synthesis problem for $\mathrm{FO}^2[\sim, <, +1]$ when both $\mathbb{P}_{\mathrm{E}}$ and $\mathbb{P}_{\mathrm{SE}}$ are empty. They also showed that as soon as a third variable is available, this problem becomes undecidable, even without the order (i.e. for $\mathrm{FO}^3[\sim, +1]$). Decidability of the satisfiability problem for the two-variable logic in this setting is what prompted Bérard et al. to consider the synthesis problem on data words, for several fragments of first order logic [1]. They proved that the synthesis problem for $\mathrm{FO}[\sim]$ is decidable in the partitioned case [1], but where the number of Environment processes is fixed. In contrast, they established undecidability results for $\mathrm{FO}[\sim]$ and $\mathrm{FO}^2[\sim, <, +1]$ when processes are shared.

We summarize the contributions of this paper in the following table, in bold font. Results from [1] are also mentioned. As can be seen, we bridge all the gaps left open by [1].

Logic \ Processes	Partitioned	Shared
$\mathrm{FO}^2[\sim]$	**decidable** (Th 1)	**undecidable** (Th 3)
$\mathrm{FO}[\sim]$	**decidable** (Th 1)	undecidable [1]
$\mathrm{FO}^2[\sim, <]$	**undecidable** (Th 2)	**undecidable** (Th 3)
$\mathrm{FO}^2[\sim, +1]$	**undecidable** (Th 2)	**undecidable** (Th 3)
$\mathrm{FO}^2[\sim, <, +1]$	**undecidable** (Th 2)	undecidable [1]

FO[∼] with processes partitioned between players

First, we turn to the case where processes are partitioned between System and Environment. It has been shown in [1] that in that case, the synthesis problem for $\mathrm{FO}[\sim]$ is decidable when System has an arbitrary number of processes, but Environment only has access to a fixed number of processes. We extend this result by lifting this restriction:

Theorem 1. *PartSynth($FO[\sim]$) is decidable.*

The idea of the proof is to show that beyond a certain threshold (which depends only on the formula), having access to more processes in $\mathbb{P}_{\mathrm{E}}$ is always a boon for the Environment. Note that this is note true for small cardinalities of $\mathbb{P}_{\mathrm{E}}$: it is not hard to design a game where Environment wins if $\mathbb{P}_{\mathrm{E}} = \emptyset$ but loses as soon as $|\mathbb{P}_{\mathrm{E}}| \geq 1$. We then use a result from [1] stating that for a fixed n_E, if System has an (n_S, n_E)-winning strategy then they already has such a strategy for a small n_S wrt. n_E. Combining both results, we reduce the search for a winning strategy for φ to some finite (and computable) set $[0, n_S] \times [0, n_E]$. To conclude the proof, note that when the number of processes is fixed, the synthesis problem on data words reduces to the (decidable [3]) synthesis problem on words, by duplicating each letter from A_{S} (resp. A_{E}) for each process in $\mathbb{P}_{\mathrm{S}}$ (resp. $\mathbb{P}_{\mathrm{E}}$).

Undecidability results

As soon as we are able to compare the relative positions of two processes, synthesis becomes undecidable, even when restricting ourselves to the two-variable setting, and when having access only to one positional relation ($<$ or $+1$):

Theorem 2. *Both PartSynth($FO^2[\sim, <]$) and PartSynth($FO^2[\sim, +1]$) are undecidable.*

We prove this theorem by reduction from the halting problem for two-counter Minsky machines. Given such a machine M, we exhibit a formula φ_M, computable from M, such that there exists a halting run for M iff System has an $(n_S, 1)$-winning strategy for φ_M for some $n_S \in \mathbb{N}$.

Environment has two letters ok_E and ko_E, while System has a letter for each state and transition of M, as well as letters inc_0, dec_0, inc_1, dec_1, noop, ok_S and

ko_S. Let us give an example of a data word encoding a halting run. Suppose that M has two counters c_0 and c_1, states $\{q_0, q_1, q_2, q_h\}$ and $t_0 : q_0 \xrightarrow{c_0++} q_0$, $t_1 : q_0 \xrightarrow{c_0--} q_1$, $t_2 : q_1 \xrightarrow{c_0--} q_2$ and $t_3 : q_2 \xrightarrow{c_0==0} q_h$

The halting run $(t_0 \cdot t_0 \cdot t_1 \cdot t_2 \cdot t_3)$ of M could be represented as the following data word, where we denote System's processes by integers, and Environment's process as $\bullet$.

$$\begin{aligned}(0, \text{ok}_S)(\bullet, \text{ok}_E)(0, q_0)(0, t_0)(0, \text{inc}_0)(0, \text{ok}_S)(\bullet, \text{ok}_E)\\ (0, q_0)(0, t_0)(1, \text{inc}_0)(0, \text{ok}_S)(\bullet, \text{ok}_E)\\ (0, q_0)(0, t_1)(0, \text{dec}_0)(0, \text{ok}_S)(\bullet, \text{ok}_E)\\ (0, q_1)(0, t_2)(1, \text{dec}_0)(0, \text{ok}_S)(\bullet, \text{ok}_E)\\ (0, q_2)(0, t_3)(0, \text{noop})(0, \text{ok}_S)(\bullet, \text{ok}_E)(0, q_h)\end{aligned}$$

We encode the run with a succession of $(\_, q)(\_, t)(\_, l)(\_, \text{ok}_S)(\_, \text{ok}_E)$ where l is either noop, an inc_i or a dec_i. Eventually, the data word stops in the halting state q_h. At any point during the run, the value of c_0 is equal to the number of System processes on which an inc_0 has been played, but no dec_0. Thus, following a transition increasing a counter, φ_M will force System to play an inc_i on a new process. Similarly, after each transition decreasing a counter, System will have to play a dec_i on a process on which an inc_i has been played, but no dec_i yet. When the transition is a zero-check, φ_M grants Environment an immediate win if there exists a process on which an inc_i and no dec_i have been played.

If System tries to cheat, then Environment immediately responds by playing ko_E, and conversely System plays ko_S when detecting a fraud from Environment. Once a ko_E or ko_S has been played, φ_M checks whether the ko is justified. If the other player was indeed cheating, then the player who ko'ed wins, otherwise they lose. With a bit of care, one can write sentences enforcing these rules both in $\text{FO}^2[\sim, <]$ and $\text{FO}^2[\sim, +1]$.

When processes are shared, things are worse: SharedSynth($\text{FO}[\sim]$) was shown to be undecidable in [1]. This result actually extends to $\text{FO}^2[\sim]$ as well.

Theorem 3. *SharedSynth($FO^2[\sim]$) is undecidable.*

Conclusion

In this paper, we have answered the questions left open in [1]. It appears that when positions between two processes can be compared, the synthesis problem quickly becomes undecidable. As a next step, it thus seems natural to consider the case of partitioned processes for an intermediate logic between $\text{FO}^2[\sim]$ and $\text{FO}^2[\sim, <]$: $\text{FO}^2[\lesssim]$, where one can compare only positions pertaining to the same process.

References

1. Bérard, B., Bollig, B., Lehaut, M., Sznajder, N.: Parameterized synthesis for fragments of first-order logic over data words. In: Foundations of Software Science and Computation Structures FOSSACS. Springer (2020)

2. Bojanczyk, M., Muscholl, A., Schwentick, T., Segoufin, L., David, C.: Two-variable logic on words with data. In: 21th IEEE Symposium on Logic in Computer Science LICS. IEEE Computer Society (2006)
3. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Transactions of the American Mathematical Society **138**, 295–311 (Apr 1969)
4. Church, A.: Applications of recursive arithmetic to the problem of circuit synthesis. In: Summaries of the Summer Institute of Symbolic Logic – Volume 1. pp. 3–50. Institute for Defense Analyses (1957)
5. Figueira, D., Praveen, M.: Playing with repetitions in data words using energy games. In: Dawar, A., Grädel, E. (eds.) Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018. pp. 404–413. ACM (2018). https://doi.org/10.1145/3209108.3209154
6. Khalimov, A., Maderbacher, B., Bloem, R.: Bounded synthesis of register transducers. In: Lahiri, S.K., Wang, C. (eds.) Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11138, pp. 494–510. Springer (2018). https://doi.org/10.1007/978-3-030-01090-4_29

Failure Handling in BDI Plans via Runtime Enforcement

Angelo Ferrando[1][0000−0002−8711−4670] and Rafael C. Cardoso[2][0000−0001−6666−6954]

[1] University of Genoa, Italy
angelo.ferrando@unige.it
[2] University of Aberdeen, United Kingdom
rafael.cardoso@abdn.ac.uk

1 Introduction

Engineering a software system can be a complex process. This is especially true when the system under consideration presents some degree of autonomy. In the context of Multi-Agent Systems (MAS), multiple entities called agents are programmed and deployed in a distributed fashion to solve various types of tasks.

In this paper, we consider MAS designed and developed following the principles of the Belief Desire Intention (BDI) model [10]. We choose the BDI model because it is one of the most popular architectures for the development of MAS. The BDI model is part of the symbolic approaches to Artificial Intelligence (AI) development, hence it expects the developer to fully specify how an agent behaves. This is obtained by defining, *beliefs*, *goals*, and especially *plans*, which denote – step by step – the agent's reasoning process. Through such plans, the developer has complete control over the agent. However, the resulting programming process is not trivial. BDI languages, such as AgentSpeak(L) [9], are notoriously different from traditional programming languages and usually come with a steep learning curve. The process of testing [12], debugging [13], and verifying [7], such systems can be quite complex. When these BDI languages are applied to safety-critical scenarios, in which an error can be costly, any solution which may make the BDI development more reliable is of uttermost importance.

The main idea of this work is to use Runtime Verification (RV) [2] as a way to enforce safety properties [1] on BDI agents. These properties can only be violated at runtime, which means the resulting monitor can only report negative and inconclusive verdicts. This is due to the fact that safety properties are satisfied only by infinite traces of events, and at runtime we only have access to finite traces. BDI agents can be applied to dynamic scenarios, where it may be difficult to guarantee that their behaviour will always be consistent with the developers' expectations. Runtime verification is usually more focused on detecting unexpected behaviours, rather than enforcing the system to actually behave in a correct way. Enforcing a behaviour leads to Runtime Enforcement [8].

We synthesise runtime monitors (called safety shields) to enforce the correct behaviour of existing BDI agents. In this paper, we summarise the main features

of such safety shields, along with their generation and integration into the BDI architecture. A safety shield works as a sandbox for the agent. Every command (actions, addition/removal of beliefs, and so on) performed in the agent's shielded plans are checked by their respective safety shields before being executed. In this way, in case the command would violate the safety specification, the safety shield can intervene and stop such command from being completed.

2 AgentSpeak(L) operational Semantics

An AgentSpeak(L) configuration C is a tuple $\langle I, E, A, R, Ap, \iota, \rho, \epsilon \rangle$ where: I is the set of intentions $\{i, i', \ldots\}$. Each intention i is a stack of partially instantiated plans $[p_1|p_2 \ldots p_n]$. We use the | symbol to separate plans in an intention. E is a set of events $\{\langle te, i\rangle, \langle te', i'\rangle, \ldots\}$. Each event is a pair $\langle te, i\rangle$, where te is a triggering event and the intention i are plans associated with te. A is a set of actions $\{\langle a, i\rangle, \langle a', i'\rangle, \ldots\}$. Each event is a pair $\langle a, i\rangle$, where a is an action and the intention i are plans associated with a. R is a set of relevant plans. Ap is a set of applicable plans. ι, ϵ and ρ keep the record of a particular intention, event and applicable plan (respectively) being considered in the current agent's reasoning cycle. This notation is similar to the ones presented in [9, 11, 5].

To keep the notation compact, we adopt the following notations: (i) if C is an AgentSpeak(L) configuration, we write C_E to make reference to the component E of C (same for the other components of C); (ii) we write $C_\iota = \_$ to indicate there is no intention considered in the agent's execution (same for C_ρ and C_ϵ); (iii) we write $i[p]$ to denote the intention that has p on its top.

3 Safety Shields

In this section, we introduce the notion of safety shields for the BDI model. Specifically, we extend the standard AgentSpeak(L) operational semantics (*i.e.*, the inference rules). Due to space constraints, we present only some of the rules that need to be extended.

A shield is a component which can be attached to an agent's plan to check whether such plan violates a formal specification during its execution. In such case, the shield enforces the plan to conform.

Safety Shield Specification The first aspect to tackle is how, and when, a safety shield is specified. We achieve this by annotating the plans which we want to "shield". Annotating plans is a common practice in existing BDI programming languages and can be found for example in [4, 6]. An annotation is a structured label attached to a plan. More formally, a shield annotation can be specified as follows: $@shield[\varphi_1, \ldots, \varphi_n]$ with $(n \geq 1)$ where *shield* is a custom label to identify that a shield annotation is being added, and φ_i (with $1 \leq i \leq n$) is the formal property the shield will look out for. By design, annotations do not have any specific semantics. The agent's reasoning cycle does not consider them, unless the developer explicitly modifies it to do so.

Adding and Removing Safety Shields This is obtained by extending the inference rules in the agent's reasoning cycle. First, we have to consider where the shields are stored. Since each shield is attached to a certain plan and each plan is executed as an intention, then a shield can be attached to such intention. Thus, the shield is used to analyse events concerning the corresponding intention.

Catching Violations (Failure Detection) Since the entire agent's reasoning cycle depends on which plans are selected as relevant[3] and, consequently, applicable. One possible way to enforce the satisfaction of a formal property is by extending the standard *RelPlans* function. The goal of such an extension is to take a property into consideration while selecting the relevant plans for a triggering event. The updated version is as follows:

$$RelPlans(plans, te, S) =$$
$$\{p\sigma \mid p \in plans \ \wedge \ \sigma = mgu(te, TE(p)) \wedge \ \nexists_{s \in S}.s_\sigma \cdot te \models s_\varphi\}$$

where S denotes the set of shields associated to the current selected intention, and $\cdot$ denotes the concatenation amongst trace of events. In this way, we can check whether the triggering event te violates at least one shield s in S (with s_σ the trace observed up to now by s, and s_φ the property checked by s). If that is the case, *RelPlans* returns the empty set.

Besides updating the *RelPlans* function, we also need to update the corresponding rule that makes use of it in the operational semantics. In particular the Rel_1 rule, which is defined as follows:

$$(Rel_1)\frac{RelPlans(plans, te) \neq \emptyset}{C, beliefs \rightarrow C', beliefs} \ {\scriptstyle C_\epsilon = \langle te, i\rangle,\ C_{Ap} = C_R = \emptyset}$$

$$where \quad C'_R = RelPlans(plans, te)$$

Rel_1 takes the current event in C_ϵ, and extracts the set of relevant plans for the specific triggering event te. Its extension, which uses the new version of *RelPlans*, is defined as follows:

$$(Rel_{1'})\frac{RelPlans(plans, te, S) \neq \emptyset}{C, beliefs \rightarrow C', beliefs} \ {\scriptstyle C_\epsilon = \langle te, i\rangle,\ C_{Ap} = C_R = \emptyset,\ \langle i, S\rangle \in C_I}$$

$$\begin{aligned} where \quad & C'_R = RelPlans(plans, te, S) \\ & C'_I = (C_I \ \setminus \ \{\langle i, S\rangle\}) \ \cup \ \{\langle i, S'\rangle\} \\ & S' = \{\langle \sigma', \varphi, i'\rangle \mid \langle \sigma, \varphi, i'\rangle \in S \ \wedge \ \sigma' = \sigma \cdot te\} \end{aligned}$$

The updated rule is necessary to keep track of the events into S's shields. Each time an event is considered in the agent's reasoning cycle, it is also stored in every active shield in S for the corresponding intention i, to be evaluated in future executions.

Note that, when the triggering event (te) violates at least one shield in S, *RelPlans* returns the empty set. Thus, no relevant plan is available ($C_R = \emptyset$), as well as no applicable plan ($C_{Ap} = \emptyset$); since *AppPlans* is defined on top of

[3] A plan is relevant for a triggering event if the triggering event can successfully be unified with the plan's head.

RelPlans. Consequently, no plan can be selected and the resulting plan failure handling is triggered; as shown in *Appl* rule, this is achieved by adding the corresponding plan deletion event ($-\%at$).

$$(Appl)\ \frac{AppPlans(C_R, beliefs) = \emptyset}{C, beliefs \rightarrow C', beliefs}\ {\scriptstyle C_\epsilon = \langle te, i\rangle,\ C_{Ap} = \emptyset,\ C_R \neq \emptyset}$$

$$where \quad C'_E = \begin{cases} C_E \cup \{\langle -\%at, i\rangle\} & \text{if } te = +\%at \text{ with } \% \in \{!, ?\} \\ C_E \cup \{C_\epsilon\} & \text{otherwise} \end{cases}$$

By updating *RelPlans* to consider a formal specification in the plan selection, we can enforce the reasoning cycle to only consider events which do not violate a certain property.

4 Implementation

As a proof of concept, we implemented a prototype[4] in the JaCaMo multi-agent development framework [3]. Jason [4], which is the implementation of AgentSpeak(L) used in JaCaMo, is one of the most used and well-known BDI programming languages.

Specifically, we decided to use JaCaMo instead of Jason, because the former supports artifacts which are well-suited for implementing the shields and interfacing with the monitors. Artifacts allows agents to have better control over their shields, while in Jason this would have to be done in a shared Java environment. The artifact maintains all the information on the shields, and it is the object consulted when a shield needs to be added, removed, or updated.

5 Conclusions and Future Work

In this extended abstract, we summarise the design and implementation of safety shields for BDI agents. We formally specify how to enhance the agent's reasoning cycle to enforce the satisfaction of safety properties through shields. Some resulting extended inference rules are reported. The contribution is not only theoretical, but it comprises a practical component as well. A prototype of our approach is proposed, along with its integration in the JaCaMo platform.

For future work, we are interested in improving the integration in JaCaMo. The current implementation is based on instrumentation, which is a less invasive way to approach the problem at the implementation level. However, instrumentation has implications at the engineering level, and it is less ideal in the long run w.r.t. the actual agent's reasoning cycle modification (as proposed in the theory of this work). Also on the implementation side, we are interested in extending the work from using one single artifact per agent, to one artifact per shield. This extension should bring to better performances, above all in the case of nested shields.

[4] https://github.com/AngeloFerrando/SafetyShieldsBDI

References

1. Alpern, B., Schneider, F.B.: Recognizing safety and liveness. Distributed Comput. **2**(3), 117–126 (1987). https://doi.org/10.1007/BF01782772
2. Bartocci, E., Falcone, Y., Francalanza, A., Reger, G.: Introduction to runtime verification. In: Lectures on Runtime Verification - Introductory and Advanced Topics, Lecture Notes in Computer Science, vol. 10457, pp. 1–33. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5_1, https://doi.org/10.1007/978-3-319-75632-5_1
3. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Science of Computer Programming **78**(6), 747–761 (Jun 2013). https://doi.org/10.1016/j.scico.2011.10.004
4. Bordini, R., Hübner, J., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason, vol. 8. John Wiley & Sons, Ltd, United Kingdom (10 2007). https://doi.org/10.1002/9780470061848
5. Bordini, R.H., Hübner, J.F.: Semantics for the Jason variant of AgentSpeak (plan failure and some internal actions). In: 19th European Conference on Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 635–640. IOS Press (2010). https://doi.org/10.3233/978-1-60750-606-5-635, https://doi.org/10.3233/978-1-60750-606-5-635
6. Cranefield, S., Winikoff, M., Dignum, V., Dignum, F.: No pizza for you: Value-based plan selection in BDI agents. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. pp. 178–184. IJCAI (2017). https://doi.org/10.24963/ijcai.2017/26, https://doi.org/10.24963/ijcai.2017/26
7. Dennis, L.A., Fisher, M., Webster, M.P., Bordini, R.H.: Model checking agent programming languages. Autom. Softw. Eng. **19**(1), 5–63 (2012). https://doi.org/10.1007/s10515-011-0088-x, https://doi.org/10.1007/s10515-011-0088-x
8. Falcone, Y., Mounier, L., Fernandez, J., Richier, J.: Runtime enforcement monitors: composition, synthesis, and enforcement abilities. Formal Methods Syst. Des. **38**(3), 223–262 (2011). https://doi.org/10.1007/s10703-011-0114-4, https://doi.org/10.1007/s10703-011-0114-4
9. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, January 22-25, 1996. Lecture Notes in Computer Science, vol. 1038, pp. 42–55. Springer (1996). https://doi.org/10.1007/BFb0031845, https://doi.org/10.1007/BFb0031845
10. Rao, A.S., Georgeff, M.P.: BDI agents: From theory to practice. In: Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA. pp. 312–319. The MIT Press (1995)
11. Vieira, R., Moreira, Á.F., Wooldridge, M.J., Bordini, R.H.: On the formal semantics of speech-act based communication in an agent-oriented programming language. J. Artif. Intell. Res. **29**, 221–267 (2007). https://doi.org/10.1613/jair.2221, https://doi.org/10.1613/jair.2221
12. Winikoff, M.: BDI agent testability revisited. Auton. Agents Multi Agent Syst. **31**(5), 1094–1132 (2017). https://doi.org/10.1007/s10458-016-9356-2, https://doi.org/10.1007/s10458-016-9356-2
13. Winikoff, M.: Debugging agent programs with why?: Questions. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017. pp. 251–259. ACM (2017), http://dl.acm.org/citation.cfm?id=3091166

Lorenzen-style strategies as proof-search strategies

Matteo Acclavio[1] and Davide Catta[2]

[1] University of Southern Denmark, Odense, Denmark
[2] Università degli studi di Napoli, Federico II, Naples, Italy

Abstract. Dialogical logic, originated in the work of Lorenzen and his student Lorenz, is an approach to logic in which the validity of a certain formula is defined as the existence of a winning strategy for a particular kind of turn-based two-players games. This paper studies the relationship between winning strategies for Lorenzen-style dialogical games and sequent calculus derivations. We define three different classes of dialogical logic games for the implicational fragment of intuitionistic logic, showing that winning strategies for such games naturally correspond to classes of derivations defined by uniformly restraining the rules of the sequent calculus.

Keywords: Dialogical Logic · Sequent Calculus · Game Semantics

1 Introduction

Dialogical logic is an approach to the study of logical reasoning, introduced by Lorenzen and his student Lorenz [12, 13], in which the validity of a formula is defined as the existence of a winning strategy for a turn-based two-player game. These games are articulated as argumentative dialogues in which the *Proponent* player **P** (she/her) aims at showing that a given formula is valid, while the *Opponent* player **O** (he/him) aims at finding possible fallacies disproving it. More precisely, each play starts with **P** asserting a certain formula. **O** takes his turn and attacks the claim made by **P** according to its logical form. The player **P** can, either, defend his previous claim or counter-attack. The debate evolves following this pattern. The player **P** wins whenever she has the last word, i.e., when **O** cannot attack anymore without violating the game's rules.

Dialogical logic was initially conceived as a foundation for the meaning of the connectives and quantifiers of *intuitionistic logic*, and it has gradually become detached from its connection with intuitionism over the years, becoming a subject of research in the formal semantics of natural language [5, 4], in proof theory [2, 9, 7, 15, 8, 18] and inspiring a series of work in formal argumentation theory and multi-agent systems [16, 17, 14, 11]. In proof theory, the soundness and completeness of a dialogical system is proved by establishing the equivalence between the existence of a winning strategy in specific games and the notion of validity in a given logic. This result is typically attained by defining a procedure that reconstructs a formal derivation from a winning strategy (and vice versa) in a sound and complete system for a given logic [7, 6, 2]. In this paper, we study the correspondence between certain classes of winning strategies for a given dialogic system and the structure of the corresponding formal derivations in the sequent calculus. We study winning strategies in which **P** moves are restricted according to **O** precedent moves (e.g., if **O** plays a move $A \to B$ as a response to a move of **P**

of a special kind, then the **P** has to immediately reply to this move). We prove that for each of the classes of winning strategies we consider, we have a correspondence with a proof-search strategy in the sequent calculus GKi for the $\rightarrow$-fragment of intuitionistic logic [19]. This latter result is obtained by showing that it is possible to narrow the proof-search space in sequent calculus without losing the soundness and completeness of the sequent system (as, e.g., in focusing [3]) and that there is a straightforward correspondence between such focused proofs and winning strategies.

2 Dialogical Logic

Notation and terminology. We denote by $|\sigma|$ the *length* of a countable[3] *sequence* $\sigma = \sigma_1, \sigma_2, \dots$, by $\sigma_{\leq i}$ the *prefix* $\sigma_1, \dots, \sigma_i$. The *parity* of an element σ_i in σ is the parity of i. It is *even* or *odd* iff i is. Given two sequences σ and τ, we write $\sigma \sqsubseteq \tau$ if $\sigma = \tau_{\leq i}$ for a given $i \leq |\tau|$ and we denote by $\sigma \cdot \tau$ their concatenation. We consider *formulas* generated from a countable non-empty set of atomic propositions $\mathcal{A} = \{a, b, c, \dots\}$ and the implication connective $\rightarrow$ (and the parenthesis symbols). In the following, we may write $(A_1 \cdots A_n) \rightarrow c$ as a shortcut for $A_1 \rightarrow (\cdots \rightarrow (A_n \rightarrow c) \cdots)$. The *implication fragments of intuitionistic logic* $\mathsf{IL}^{\rightarrow}$, is the smallest set of formulas containing each instance of the two axioms $A \rightarrow (B \rightarrow A)$ and $A \rightarrow (B \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$ and closed for *modus ponens*, that is: if $A \in \mathsf{IL}^{\rightarrow}$ and $A \rightarrow B \in \mathsf{IL}^{\rightarrow}$ then $B \in \mathsf{IL}^{\rightarrow}$. We say that a formula F is *valild* if and only if $F \in \mathsf{IL}^{\rightarrow}$.

Dialogical Games. A *challenge* is a pair $\langle ?, s \rangle$ where s is either an occurrence of the symbol $\bullet$, in which case such a challenge is said *atomic*, or where s is formula F. A *defense* is a pair $\langle !, F \rangle$ where F is a formula. An *assertion (of F)* is a non-atomic challenge $\langle ?, F \rangle$ or a defense $\langle !, F \rangle$. A *move* is an assertion or an atomic-challenge. An *augmented sequence* is a pair $\langle \sigma, \phi \rangle$ where σ is a non-empty sequence of moves, and ϕ is a function mapping any σ_i with $i > 1$ to a $\sigma_j = \phi(\sigma_i)$ with opposite parity and such that $j < i$. A move σ_i in σ is called **P**-*move* (denoted $\sigma_i^{\mathbf{P}}$) if i is odd, and **O**-*move* (denoted $\sigma_i^{\mathbf{O}}$) if i is even. It is a *repetition* if there is $j < i$ such that i and j have opposite parity and σ_i and σ_j are assertions of the same formula.

Definition 1. *Let $\langle \sigma, \phi \rangle$ be an augmented sequence and $i \leq |\sigma|$.*

1. *A challenge σ_i is* justified *whenever:*
   (a) *either σ_i is an atomic-challenge and $\phi(\sigma_i)$ is an assertion of an atomic formula;*
   (b) *or $\sigma_i = \langle ?, A \rangle$ and $\phi(\sigma_i)$ is an assertion of a formula $A \rightarrow B$.*
2. *A defense is σ_i is* justified *whenever:*
   (a) *either σ_i and $\phi(\phi(\sigma_i))$ are assertions of a same atomic formula $a \in \mathcal{A}$ and $\phi(\sigma_i)$ is an atomic challenge;*
   (b) *or σ_i is an assertion of a formula B, $\phi(\sigma_i)$ is a justified challenge of the form $\langle ?, A \rangle$, and $\phi(\phi(\sigma_i))$ is an assertion of $A \rightarrow B$.*

If σ_i is a justified move, we say $\phi(\sigma_i)$ justifies *σ_i and that σ_i* is justified *by $\phi(\sigma_i)$. A challenge σ_i is* unanswered *if there is no defense σ_k such that σ_k is justified by σ_i. A justified challenge σ_i is a* counterattack *if $\phi(\sigma_i)$ is a challenge. A* ***justified sequence*** *is an augmented sequence in which any move except the first one is justified.*

[3] We use the adjective *countable* in the standard mathematical sense: a set is countable iff it is in a one-to-one correspondence with a (finite or infinite) subset of the set of natural numbers.

Definition 2 (Play). *A play for F is a justified sequence $\mathsf{p} = \langle \sigma, \phi \rangle$ starting with* **P** *defending F, that is, $\sigma_1 = \langle !, F \rangle$ and such that the following holds for any $1 < i \leq |\sigma|$:*

1. *each* **O***-move is justified by the immediately preceding* **P***-move, that is, $\phi(\sigma_{2k}) = \sigma_{2k-1}$ for any $2k \leq |\sigma|$;*
2. *if* **P** *states a defense, such defense is justified by the last unanswered challenge stated by* **O***, that is, if $\sigma_{2k_{+1}} = \langle !, F \rangle$, then $\phi(\sigma_{2k+1}) = \sigma_{2h}$ is the unanswered challenge with maximal $2h \leq 2k$;*
3. *if* **P** *state a defense and such a defense is an assertion of an atomic formula, then there must be another preceding* **O** *assertion of the same atomic formula. That is, if $\sigma_{2k+1} = \langle !, a \rangle$ with $a \in \mathcal{A}$, then σ_{2k+1} is a repetition;*
4. *Only* **O** *can challenge assertions of atomic formulas and these assertions must be challenges. That is, if $\sigma_i = \langle ?, \bullet \rangle$, then i must be even and $\phi(\sigma_i)$ is a challenge.*

A play $\mathsf{p} = \langle \sigma, \phi \rangle$ is finite if σ is. and its length $|\mathsf{p}|$ *is the length σ. A move m is* legal *for* p *if $\langle \sigma \cdot m, \phi \cup \{\langle m, \sigma_i \rangle\} \rangle$ is a play for a $i \leq |\sigma|$. The player* **P** wins *a play $\mathsf{p} = \langle \sigma, \phi \rangle$ if σ is finite and ends with a* **P***-move $\langle !, a \rangle$ with $a \in \mathcal{A}$. Otherwise,* **O** *wins.*

Definition 3. *Let $\mathsf{p} = \langle \sigma, \phi \rangle$ be a play.*

1. p *is a* Lorenzen-Felscher play *(or* LF-play*) if any* **P***-assertion of an atomic formula is a repetition. That is, if $\sigma_{2k+1} \in \{\langle !, a \rangle, \langle ?, a \rangle \mid a \in \mathcal{A}\}$, then there is $h \leq k$ such that $\sigma_{2h} = \langle \star, a \rangle$ for $\star \in \{?, !\}$*
2. p *is a* Stubborn play *(or* ST-play*) if the following hold:*
   (a) *whenever* **O** *assert a complex formula $A \to B$ as a defense from a preceding challenge, then* **P***'s next move is a challenge of such a formula. That is, if $\sigma' \cdot m^{\mathbf{O}} \sqsubseteq \sigma$ and $m = \langle !, A \to B \rangle$, then $\sigma' \cdot m^{\mathbf{O}} \cdot n^{\mathbf{P}} \sqsubseteq \sigma$ for a $n = \langle ?, A \rangle$ justified by m.*
   (b) *whenever* **O** *assert an atomic formula c as a defense from a preceding challenge, then* **P***'s next move is a defense asserting c. That is, if $\sigma' \cdot m^{\mathbf{O}} \sqsubseteq \sigma$ and $m = \langle !, c \rangle$, then $\sigma = \sigma' \cdot m^{\mathbf{O}} \cdot n^{\mathbf{P}}$ where $n = \langle !, c \rangle$.*

Definition 4. *Let A be a formula. The* game *for A is a pair $\mathcal{G}_A = \langle \mathcal{R}_A, \phi_A \rangle$ where $\mathcal{R}_A = \langle \mathcal{N}_A, \mathcal{E}_A \rangle$ is a tree of moves and $\phi : \mathcal{N}_A \to \mathcal{N}_A$ is a map such that:*

1. *for each path $\mathcal{P}$ of $\mathcal{R}_A$, the pair $\langle \mathcal{P}, \phi_A|_{\mathcal{P}} \rangle$ is a play for A;*
2. *for each node v of $\mathcal{R}_A$, all and only the children of v are legal move of the play in $\mathcal{G}_A$ ending with v.*

A node v of $\mathcal{G}$ is a **P**-node *(resp.* **O**-node*) if is its height is odd (resp. even). A* strategy *for A is a pair $\mathcal{S} = \langle \mathcal{T}, \psi \rangle$ such that $\mathcal{T}$ is a subtree of $\mathcal{R}_A$ (and ψ is defined as the restriction of ϕ_A on the nodes in $\mathcal{T}$) in which every* **O***-node has at most one child. It is* winning *when $\mathcal{T}$ is finite and any of its branch is a play won by* **P***. A* Lorenzen-Felscher strategy *(resp.* Stubborn strategy*) is a strategy such that each branch of $\mathcal{S}$ is is a* LF*-play (resp. a* ST*-play).*

3 Sequent Calculus

A *sequent* is an expression $\Gamma \vdash C$ where C is a formula and Γ is a finite (possibly empty) multiset of formulas. A derivation $\mathscr{D}$ is a finite tree of sequents constructed using the rules in Figure 1 in which each leaf is obtained by an Ax-rule and each non-leaf sequent is obtained by $\to^{\mathsf{R}}$-rule or a $\to^{\mathsf{L}}$-rule. A sequent $\Gamma \vdash C$ is GKi-provable if it admits a derivation in the sequent calculus GKi, whose root (or conclusion) is $\Gamma \vdash C$.

$$\frac{}{\Gamma, \underline{a} \vdash a}\,\mathsf{Ax} \qquad \frac{\Gamma, A \to B \vdash \underline{A} \quad \Gamma, A \to B, \underline{B} \vdash C}{\Gamma, \underline{A \to B} \vdash C} \to^{\mathsf{L}} \qquad \frac{\Gamma, \underline{A} \vdash \underline{B}}{\Gamma \vdash \underline{A \to B}} \to^{\mathsf{R}}$$

Fig. 1. Rules for the sequent calculus GKi. In each rule we have underlined its *principal* formula in the conclusion, and the *active* formulas in each premise.

Definition 5. *Let $\mathscr{D}$ be a derivation of some sequent $\Delta \vdash F$ in* GKi*. We say that:*

1. *$\mathscr{D}$ is a* strategic derivation *(or* **S**-derivation*) when each left-hand side premise of $\to^{\mathsf{L}}$-rule of the form $\Gamma \vdash A \to B$ is the conclusion of a $\to^{\mathsf{R}}$-rule;*
2. *$\mathscr{D}$ is a* **LF**-derivation *if the left-hand side premise of each $\to^{\mathsf{L}}$-rule is always the conclusion of a $\to^{\mathsf{R}}$-rule or an* Ax*-rule;*
3. *$\mathscr{D}$ is a* **ST**-derivation *if is a **S**-derivation and the active formula of the right-hand premise of each $\to^{\mathsf{L}}$-rule in $\mathscr{D}$ is the principal formula of this premise. That is, if $\Gamma, A \to B, B \vdash C$ is the right-hand premise of a $\to^{\mathsf{L}}$-rule, then either it is the conclusion of a* Ax *if $B = C$ is atomic, or it is the conclusion of a $\to^{\mathsf{L}}$-rule. In both cases B is the principal formula of $\Gamma, A \to B, B \vdash C$*

Theorem 1. *Let F be a formula, It is valid iff it admits a **S**-derivation iff it admits a **LF**-derivation iff it admits a **ST**-derivation.*

Theorem 2. *The following statements hold:*

1. *The set of **S**-derivations is in one-to-one correspondence with the set of winning strategies;*
2. *The set of **LF**-derivations is in one-to-one correspondence with the set of Lorenzen-Felscher winning strategies*
3. *The set of **ST**-derivations is in one-to-one correspondence with the set of Stubborn winning strategies.*

4 Conclusion and Future Work

We have defined different classes of Lorenzen-style dialogical plays for intuitionistic logic by restricting the way in which **P** can play during a game. Winning strategies for such games naturally corresponds to particular GKi derivations obtained by limiting the application of GKis-rules in proof search procedures.

In future work, we want to extend our definitions to full intuitionistic propositional logic with disjunction $\vee$, conjunction $\wedge$, and absurdity $\bot$. In addition, we want to study the correspondence between the class of **ST**-derivations and terms of the simply typed lambda calculus [10]. Moreover, the results in [1] would suggest a way to define a dialogical system for the constructive modal logic CK.

The semantics of formal argumentation systems are often specified through the help of concepts originated in dialogic logic (e.g. E-strategies see [14]). We think it would be interesting to study a more abstract version of our stubborn strategies in the context of formal argumentation.

Acknowledgments. The first author is supported by Villum Fonden, grant no. 50079. The second author is supported by the PRIN project RIPER (No. 20203FFYLK)

References

1. Acclavio, M., Catta, D., Straßburger, L.: Game semantics for constructive modal logic. In: , TABLEAUX 2021. Springer International Publishing (2021)
2. Alama, J., Knoks, A., Uckelman, S.: Dialogues games for classical logic (short paper), pp. 82–86. Universiteit Bern (2011)
3. Andreoli, J.M.: Logic programming with focusing proofs in linear logic. Journal of Logic and Computation **2**, 297–347 (1992)
4. Catta, D., Moot, R., Retoré, C.: Dialogical argumentation and textual entailment. In: Loukanova, R. (ed.) Natural Language Processing in Artificial Intelligence—NLPinAI 2020., Studies in Computational Intelligence, vol. 939. Springer (2021)
5. Catta, D., Stevens-Guille, S.: Lorenzen won the game, lorenzen did too: Dialogical logic for anaphora and ellipsis resolution. In: WoLLIC 2021. Springer (2021)
6. Felscher, W.: Dialogues, strategies, and intuitionistic provability. Annals of Pure and Applied Logic **28**(3), 217 – 254 (1985). https://doi.org/https://doi.org/10.1016/0168-0072(85)90016-8
7. Fermüller, C.G.: Parallel dialogue games and hypersequents for intermediate logics. In: Cialdea Mayer, M., Pirri, F. (eds.) Automated Reasoning with Analytic Tableaux and Related Methods. pp. 48–64. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
8. Fermüller, C.G.: Connecting sequent calculi with lorenzen-style dialogue games. Paul Lorenzen–Mathematician and Logician pp. 115–141 (2021)
9. Herbelin, H.: Séquents qu'on calcule : de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes. Phd thesis, Université Paris 7 (Janvier 1995), https://tel.archives-ouvertes.fr/tel-00382528/file/These-Her95.pdf
10. Hindley, J.R.: Basic Simple Type Theory, Cambridge Tracts in Theoretical Computer Science, vol. 42. Cambridge University Press (1997), corrected edition, 2008
11. Kacprzak, M., Budzynska, K.: Reasoning about dialogical strategies. In: KES 2012,. Springer (2012). https://doi.org/10.1007/978-3-642-37343-5_18, https://doi.org/10.1007/978-3-642-37343-5_18
12. Lorenzen, P.: Logik und agon. Atti Del XII Congresso Internazionale di Filosofia **4**, 187–194 (1958)
13. Lorenzen, P., Lorenz, K.: Dialogische Logik. Wissenschaftliche Buchgesellschaft, [Abt. Verlag] (1978)
14. Modgil, S., Caminada, M.: Proof theories and algorithms for abstract argumentation frameworks. In: Simari, G.R., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 105–129. Springer (2009). https://doi.org/10.1007/978-0-387-98197-0_6, https://doi.org/10.1007/978-0-387-98197-0_6
15. Pavlova, A.: Dialogue games for minimal logic. Logic and Logical Philosophy **30**(2), 281–309 (Nov 2020). https://doi.org/10.12775/LLP.2020.022, https://apcz.umk.pl/LLP/article/view/LLP.2020.022
16. Prakken, H.: Coherence and flexibility in dialogue games for argumentation. J. Log. and Comput. **15**(6), 1009–1040 (dec 2005). https://doi.org/10.1093/logcom/exi046, https://doi.org/10.1093/logcom/exi046
17. Prakken, H., Sartor, G.: A dialectical model of assessing conflicting arguments in legal reasoning. Artif. Intell. Law **4**(3-4), 331–368 (1996). https://doi.org/10.1007/BF00118496, https://doi.org/10.1007/BF00118496
18. Sticht, M.: Multi-agent dialogue games and dialogue sequents for proof search and scheduling. In: Proceedings of the 31st Italian Conference on Computational Logic. CEUR-WS.org (2016), https://ceur-ws.org/Vol-1645/paper_20.pdf
19. Troelstra, A., Schwichtenberg, H.: Basic Proof Theory. Cambridge University Press, USA (1996)

Strategy Repair in Reachability Games *

Pierre Gaillard[1], Fabio Patrizi[2], and Giuseppe Perelli[2]

[1] ENS Paris-Saclay, University Paris-Saclay
[2] Sapienza University of Rome

Reachability Games (RGs) [2] can serve as semantic models for reasoning about dynamic domains, with the resulting strategy representing the behavior that an agent can execute, in order to achieve a desired state. Typically, however, at execution time, models deviate from the actual trajectory that stems from strategy execution, resulting in a situation where the actual state does not match that of the model. There may also be situations where the goal changes during strategy execution. In both these examples, the agent is unable to keep executing the computed strategy (which was originally winning) and take appropriate actions to achieve the desired goal. Thus, the problem arises of coming up with a new strategy that guarantees goal achievement.

The original strategy might have been designed to guarantee not only goal achievement, but also a number of additional properties, such as cost minimization, reward maximization, or forbidden-state avoidance, which might yield a significant additional computational effort. Thus, when the unexpected changes are small and yield only a slightly different problem wrt the original one, i.e., only few target states are added or removed and state mismatches occur rarely, it is reasonable to seek for a solution obtained as a slight modification of the original one, under the assumption that the new strategy will retain all (or part of) the properties featured by the initial strategy, without needing the computational overhead required to achieve such properties.

This paper investigates this approach from the general perspective of RGs. We introduce a problem, called *Strategy Repair*, which requires, given a *losing* strategy σ_0, to find a minimal amount of modifications which turn σ_0 into a winning strategy.

We make the following contributions. Firstly, we formally define the problem by introducing a notion of *distance* between two strategies, which intuitively corresponds to the number of states over which the strategies differ. Then, based on this notion, we devise a solution algorithm and characterize its complexity. Specifically, we prove, by reduction from Vertex Cover, that the decision version of Strategy Repair is NP-complete. We then investigate more efficient, but sub-optimal, alternatives, devising a polynomial greedy algorithm with an effective heuristic, called MustFix, which can be integrated also in the optimal algorithm. Finally, we report on an experimental analysis, which shows that the polynomial algorithm, together with the MustFix heuristic, yields impressive results in terms of running time, scalability and accuracy (measured as distance from the optimal solution). Also the optimal algorithm greatly benefits from the MustFix heuristic, outperforming the running times of the basic version by orders of magnitude.

* Extended version to appear at ECAI-23 [1]

References

1. Gaillard, P., Patrizi, F., Perelli, G.: Strategy repair in reachability games. In: 26th European Conference on Artificial Intelligence, ECAI (2023), to appear
2. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], Lecture Notes in Computer Science, vol. 2500. Springer (2002). https://doi.org/10.1007/3-540-36387-4, https://doi.org/10.1007/3-540-36387-4

Incentive Engineering for Concurrent Games

David Hyland
University of Oxford
Oxford, United Kingdom
david.hyland@cs.ox.ac.uk

Julian Gutierrez
Monash University
Melbourne, Australia
julian.gutierrez@monash.edu

Michael Wooldridge
University of Oxford
Oxford, United Kingdom
mjw@cs.ox.ac.uk

We consider the problem of incentivising desirable behaviours in multi-agent systems by way of taxation schemes. Our study employs the concurrent games model: in this model, each agent is primarily motivated to seek the satisfaction of a goal, expressed as a Linear Temporal Logic (LTL) formula; secondarily, agents seek to minimise costs, where costs are imposed based on the actions taken by agents in different states of the game. In this setting, we consider an external principal who can influence agents' preferences by imposing taxes (additional costs) on the actions chosen by agents in different states. The principal imposes taxation schemes to motivate agents to choose a course of action that will lead to the satisfaction of their goal, also expressed as an LTL formula. However, taxation schemes are limited in their ability to influence agents' preferences: an agent will always prefer to satisfy its goal rather than otherwise, no matter what the costs. The fundamental question that we study is whether the principal can impose a taxation scheme such that, in the resulting game, the principal's goal is satisfied in at least one or all runs of the game that could arise by agents choosing to follow game-theoretic equilibrium strategies. We consider two different types of taxation schemes: in a *static* scheme, the same tax is imposed on a state-action profile pair in all circumstances, while in a *dynamic* scheme, the principal can choose to vary taxes depending on the circumstances. We investigate the main game-theoretic properties of this model as well as the computational complexity of the relevant decision problems.

1 Introduction

Rational verification is the problem of establishing which temporal logic properties will be satisfied by a multi-agent system, under the assumption that agents in the system choose strategies that form a game-theoretic equilibrium [18, 51, 25]. Thus, rational verification enables us to verify which desirable and undesirable behaviours could arise in a system through individually rational choices. This article, however, expands beyond verification and studies methods for incentivising outcomes with favourable properties while mitigating undesirable consequences.

We take as our starting point the work of [50], who considered the possibility of influencing one-shot Boolean games by introducing taxation schemes, which impose additional costs onto a game at the level of individual actions. In the model of preferences considered in [50], agents are primarily motivated to achieve a goal expressed as a (propositional) logical formula, and only secondarily motivated to minimise costs. This logical component limits the possibility to influence agent preferences: an agent can never be motivated by a taxation scheme away from achieving its goal. In related work, Wooldridge et al. defined the following implementation problem: given a game G and an objective Υ, expressed as a propositional logic formula, does there exists a taxation scheme τ that could be imposed upon G such that, in the resulting game G^{τ}, the objective Υ will be satisfied in at least one Nash equilibrium [50].

We develop these ideas by applying models of finite-state automata to introduce and motivate the use of history-dependent incentives in the context of *concurrent games* [2]. In a concurrent game, play continues for an infinite number of rounds, where at each round, each agent simultaneously chooses an

action to perform. Preferences in such a multiplayer game are defined by associating with each agent i a Linear Temporal Logic (LTL) goal γ_i, which agent i desires to see satisfied. In this work, we also assume that actions incur costs, and that agents seek to minimise their limit-average costs.

Since, in contrast to the model of [50], play in our games continues for an infinite number of rounds, we find there are two natural variations of taxation schemes for concurrent games. In a *static* taxation scheme, we impose a fixed cost on state-action profiles so that the same state-action profile will always incur the same tax, no matter when it is performed. In a *dynamic* taxation scheme, the same state-action profile may incur different taxes in different circumstances: it is history-dependent. We first show that dynamic taxation schemes are strictly more powerful than static taxation schemes, making them a more appropriate model of incentives in the context of concurrent games, characterise the conditions under which an LTL objective Υ can be implemented in a game using dynamic taxation schemes, and begin to investigate the computational complexity of the corresponding decision problems.

References

[1] Natasha Alechina, Giuseppe De Giacomo, Brian Logan & Giuseppe Perelli (2022): *Automatic Synthesis of Dynamic Norms for Multi-Agent Systems*. In: *19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022: KR 2022*, doi:10.24963/kr.2022/2.

[2] Rajeev Alur, Thomas A. Henzinger & Orna Kupferman (2002): *Alternating-time temporal logic*. *J. ACM* 49(5), pp. 672–713, doi:10.1145/585265.585270.

[3] Robert J. Aumann (1961): *The core of a cooperative game without side payments*. *Transactions of the American Mathematical Society* 98(3), pp. 539–552, doi:10.2307/1993348.

[4] Jan Balaguer, Raphael Koster, Christopher Summerfield & Andrea Tacchetti (2022): *The Good Shepherd: An Oracle Agent for Mechanism Design*. *arXiv preprint arXiv:2202.10135*, doi:10.48550/arXiv.2202.10135.

[5] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger & Barbara Jobstmann (2009): *Better Quality in Synthesis through Quantitative Objectives*. In Ahmed Bouajjani & Oded Maler, editors: *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, *Lecture Notes in Computer Science* 5643, Springer, pp. 140–156, doi:10.1007/978-3-642-02658-4-14. Available at https://doi.org/10.1007/978-3-642-02658-4_14.

[6] Michael Bräuning, Eyke Hüllermeier, Tobias Keller & Martin Glaum (2017): *Lexicographic preferences for predictive modeling of human decision making: A new machine learning method with an application in accounting*. *European Journal of Operational Research* 258(1), pp. 295–306, doi:10.1016/j.ejor.2016.08.055.

[7] Nils Bulling & Mehdi Dastani (2016): *Norm-based mechanism design*. *Artificial Intelligence* 239, pp. 97–142, doi:10.1016/j.artint.2016.07.001.

[8] Henrique Lopes Cardoso & Eugénio Oliveira (2009): *Adaptive Deterrence Sanctions in a Normative Framework*. In: *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 2, pp. 36–43, doi:10.1109/WI-IAT.2009.123.

[9] Roberto Centeno & Holger Billhardt (2011): *Using incentive mechanisms for an adaptive regulation of open multi-agent systems*. In: *Twenty-Second International Joint Conference on Artificial Intelligence*, doi:10.5591/978-1-57735-516-8/IJCAI11-035.

[10] Georgios Chalkiadakis, Edith Elkind & Michael J. Wooldridge (2011): *Computational Aspects of Cooperative Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, doi:10.2200/S00355ED1V01Y201107AIM016. Available at http://dx.doi.org/10.2200/S00355ED1V01Y201107AIM016.

[11] Krishnendu Chatterjee, Thomas A. Henzinger & Marcin Jurdzinski (2005): *Mean-Payoff Parity Games*. In: *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, IEEE Computer Society, pp. 178–187, doi:10.1109/LICS.2005.26.

[12] Davide Dell'Anna, Mehdi Dastani & Fabiano Dalpiaz (2020): *Runtime Revision of Sanctions in Normative Multi-Agent Systems*. *Autonomous Agents and Multi-Agent Systems* 34(2), doi:10.1007/s10458-020-09465-8.

[13] Paul E. Dunne, Wiebe van der Hoek, Sarit Kraus & Michael J. Wooldridge (2008): *Cooperative Boolean games*. In Lin Padgham, David C. Parkes, Jörg P. Müller & Simon Parsons, editors: *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Volume 2*, IFAAMAS, pp. 1015–1022, doi:10.1145/1402298.1402363. Available at https://dl.acm.org/citation.cfm?id=1402363.

[14] Andrzej Ehrenfeucht & Jan Mycielski (1979): *Positional strategies for mean payoff games*. *International Journal of Game Theory* 8(2), pp. 109–113, doi:10.1007/BF01768705.

[15] Mahmoud Elbarbari, Florent Delgrange, Ivo Vervlimmeren, Kyriakos Efthymiadis, Bram Vanderborght & Ann Nowé (2022): *A framework for flexibly guiding learning agents*. *Neural Computing and Applications*, pp. 1–17, doi:10.1007/s00521-022-07396-x.

[16] E. Allen Emerson (1990): *Temporal and Modal Logic*. In Jan van Leeuwen, editor: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, Elsevier and MIT Press, pp. 995–1072, doi:10.1016/b978-0-444-88074-1.50021-4.

[17] E. Allen Emerson & Charanjit S. Jutla (1991): *Tree automata, mu-calculus and determinacy*, doi:10.1109/SFCS.1991.185392.

[18] Dana Fisman, Orna Kupferman & Yoad Lustig (2010): *Rational synthesis*. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, pp. 190–204, doi:10.1007/978-3-642-12002-2_16.

[19] Sanford J. Grossman & Oliver D. Hart (1992): *An analysis of the principal-agent problem*. In: *Foundations of insurance economics*, Springer, pp. 302–340, doi:10.1007/978-94-015-7957-5_16.

[20] Julian Gutierrez, Paul Harrenstein & Michael J. Wooldridge (2017): *Reasoning about equilibria in game-like concurrent systems*. *Annals of Pure and Applied Logic* 169(2), pp. 373–403, doi:10.1016/j.apal.2016.10.009.

[21] Julian Gutierrez, Sarit Kraus & Michael J. Wooldridge (2019): *Cooperative Concurrent Games*. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, p. 1198–1206, doi:10.1016/j.artint.2022.103806.

[22] Julian Gutierrez, Aniello Murano, Giuseppe Perelli, Sasha Rubin, Thomas Steeples & Michael J. Wooldridge (2021): *Equilibria for games with combined qualitative and quantitative objectives*. *Acta Informatica* 58(6), pp. 585–610, doi:10.1007/s00236-020-00385-4.

[23] Julian Gutierrez, Aniello Murano, Giuseppe Perelli, Sasha Rubin & Michael Wooldridge (2017): *Nash Equilibria in Concurrent Games with Lexicographic Preferences*. doi:10.24963/ijcai.2017/148.

[24] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli & Michael J. Wooldridge (2019): *Equilibrium Design for Concurrent Games*. In: *30th International Conference on Concurrency Theory*, doi:10.4230/LIPIcs.CONCUR.2019.22.

[25] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli & Michael J. Wooldridge (2020): *Automated temporal equilibrium analysis: Verification and synthesis of multi-player games*. *Artificial Intelligence* 287, p. 103353, doi:10.1016/j.artint.2020.103353.

[26] Paul Harrenstein, Paolo Turrini & Michael J. Wooldridge (2014): *Hard and soft equilibria in boolean games*. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio & Paul Scerri, editors: *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, IFAAMAS/ACM, pp. 845–852, doi:10.5555/2615731.2615867. Available at http://dl.acm.org/citation.cfm?id=2615867.

[27] Paul Harrenstein, Paolo Turrini & Michael J. Wooldridge (2017): *Characterising the Manipulability of Boolean Games*. In Carles Sierra, editor: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, ijcai.org, pp. 1081–1087, doi:10.24963/ijcai.2017/150.

[28] Bengt Holmstrom (1982): *Moral hazard in teams*. The Bell journal of economics, pp. 324–340, doi:10.2307/3003457.

[29] Xiaowei Huang, Ji Ruan, Qingliang Chen & Kaile Su (2016): *Normative Multiagent Systems: A Dynamic Generalization*. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, AAAI Press, p. 1123–1129.

[30] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano & Sheila McIlraith (2018): *Using reward machines for high-level task specification and decomposition in reinforcement learning*. In: International Conference on Machine Learning, PMLR, pp. 2107–2116.

[31] Kenneth S. Lyon & Dug Man Lee (2001): *Pigouvian tax and the congestion externality: a benefit side approach*. Economics Research Institute Study Paper 10, p. 1.

[32] Moamin A. Mahmoud, Mohd Sharifuddin Ahmad, Mohd Zaliman Mohd Yusoff & Aida Mustapha (2014): *A review of norms and normative multiagent systems*. The Scientific World Journal 2014, doi:10.1155/2014/684587.

[33] N. Gregory Mankiw (2009): *Smart taxes: An open invitation to join the pigou club*. Eastern Economic Journal 35(1), pp. 14–23, doi:10.1057/EEJ.2008.43.

[34] Bastien Maubert, Munyque Mittelmann, Aniello Murano & Laurent Perrussel (2021): *Strategic reasoning in automated mechanism design*. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, 18, pp. 487–496, doi:10.24963/kr.2021/46.

[35] David Mguni, Joel Jennings, Emilio Sison, Sergio Valcarcel Macua, Sofia Ceppi & Enrique Munoz de Cote (2019): *Coordinating the Crowd: Inducing Desirable Equilibria in Non-Cooperative Systems*. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, p. 386–394.

[36] David Mguni & Marcin Tomczak (2019): *Efficient reinforcement dynamic mechanism design*. In: GAIW: Games, agents and incentives workshops, at AAMAS, Montreal, Canada.

[37] Munyque Mittelmann, Bastien Maubert, Aniello Murano & Laurent Perrussel (2022): *Automated synthesis of mechanisms*. In: 31st International Joint Conference on Artificial Intelligence (IJCAI-22), International Joint Conferences on Artificial Intelligence Organization, pp. 426–432, doi:10.24963/ijcai.2022/61.

[38] Sai Kiran Narayanaswami, Swarat Chaudhuri, Moshe Vardi & Peter Stone (2022): *Automating Mechanism Design with Program Synthesis*. Proc. of the Adaptive and Learning Agents Workshop (ALA 2022).

[39] Cyrus Neary, Zhe Xu, Bo Wu & Ufuk Topcu (2020): *Reward Machines for Cooperative Multi-Agent Reinforcement Learning*. doi:10.5555/3463952.3464063.

[40] David C Parkes, Ruggiero Cavallo, Florin Constantin & Satinder Singh (2010): *Dynamic incentive mechanisms*. Ai Magazine 31(4), pp. 79–94, doi:10.1609/aimag.v31i4.2316.

[41] Giuseppe Perelli (2019): *Enforcing equilibria in multi-agent systems*. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pp. 188–196, doi:10.5555/3306127.3331692.

[42] Arthur C. Pigou & Nahid Aslanbeigui (2017): *The Economics of Welfare*. Routledge, doi:10.4324/9781351304368.

[43] Amir Pnueli (1977): *The Temporal Logic of Programs*. In: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, pp. 46–57, doi:10.1109/SFCS.1977.32.

[44] Amir Pnueli & Roni Rosner (1989): *On the Synthesis of an Asynchronous Reactive Module*. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini & Simona Ronchi Della Rocca, editors: Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, July 11-15, 1989, Proceedings, Lecture Notes in Computer Science 372, Springer, pp. 652–671, doi:10.1007/BFb0035790.

[45] Lillian J Ratliff, Roy Dong, Shreyas Sekar & Tanner Fiez (2019): *A perspective on incentive design: Challenges and opportunities*. Annual Review of Control, Robotics, and Autonomous Systems 2, pp. 305–338, doi:10.1146/ANNUREV-CONTROL-053018-023634.

[46] Lillian J Ratliff & Tanner Fiez (2020): *Adaptive incentive design*. IEEE Transactions on Automatic Control 66(8), pp. 3871–3878, doi:10.1109/tac.2020.3027503.

[47] Weiran Shen, Pingzhong Tang & Song Zuo (2019): *Automated Mechanism Design via Neural Networks*. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, p. 215–223, doi:10.5555/3306127.3331696.

[48] A. Prasad Sistla & Edmund M. Clarke (1985): *The Complexity of Propositional Linear Temporal Logics*. J. ACM 32(3), pp. 733–749, doi:10.1145/3828.3837.

[49] Michael Ummels & Dominik Wojtczak (2011): *The complexity of Nash equilibria in limit-average games*. In: International Conference on Concurrency Theory, Springer, pp. 482–496, doi:10.1007/978-3-642-23217-6_32.

[50] Michael J. Wooldridge, Ulle Endriss, Sarit Kraus & Jérôme Lang (2013): *Incentive engineering for Boolean games*. Artif. Intell. 195, pp. 418–439, doi:10.1016/j.artint.2012.11.003.

[51] Michael J. Wooldridge, Julian Gutierrez, Paul Harrenstein, Enrico Marchioni, Giuseppe Perelli & Alexis Toumi (2016): *Rational Verification: From Model Checking to Equilibrium Checking*. In Dale Schuurmans & Michael P. Wellman, editors: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, AAAI Press, pp. 4184–4191, doi:10.1016/J.ARTINT.2017.04.003. Available at http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12268.

[52] Jiachen Yang, Ethan Wang, Rakshit Trivedi, Tuo Zhao & Hongyuan Zha (2021): *Adaptive Incentive Design with Multi-Agent Meta-Gradient Reinforcement Learning*. arXiv preprint arXiv:2112.10859, doi:10.5555/3535850.3536010.

[53] Ningyuan Zhang, Wenliang Liu & Calin Belta (2022): *Distributed Control using Reinforcement Learning with Temporal-Logic-Based Reward Shaping*. In: Learning for Dynamics and Control Conference, PMLR, pp. 751–762, doi:10.48550/arXiv.2203.04172.

Learning Task Automata for Reinforcement Learning using Hidden Markov Models

Alessandro Abate, Yousif Almulla, James Fox, David Hyland, Michael Wooldridge

University of Oxford

aabate@cs.ox.ac.uk, yousif.almulla38@gmail.com, james.fox@cs.ox.ac.uk, david.hyland@cs.ox.ac.uk, mjw@cs.ox.ac.uk

Abstract

Training reinforcement learning (RL) agents using scalar reward signals is often infeasible when an environment has sparse and non-Markovian rewards. Moreover, handcrafting these reward functions before training is prone to misspecification. This work proposes a novel pipeline for learning non-Markovian finite task specifications as finite-state 'task automata' from episodes of agent experience within unknown environments. First, we learn a product MDP, a model composed of the specification's automaton and the environment's MDP (both initially unknown), by treating it as a partially observable MDP and employing algorithms for hidden Markov models. Second, we propose a novel method for distilling the task automaton (assumed to be a deterministic finite automaton - DFA) from the learnt product MDP. Our learnt task automaton enables a task to be decomposed into sub-tasks, so an RL agent can later synthesise an optimal policy more efficiently. It is also an interpretable encoding of high-level task features, so a human can verify that the agent has learnt tasks with no misspecifications. We also take steps towards ensuring that the learnt automaton is environment-agnostic, making it well-suited for use in transfer learning.

1 Description of Work

Reinforcement Learning (RL) can be prohibitively sample inefficient at learning an optimal policy when the reward signal is sparse and non-Markovian because of the credit assignment problem (e.g., see Mnih et al. (2015) 's score on Montezuma's Revenge). Nevertheless, this setting is common for real-world applications, where tasks can involve a sequence of sequential sub-tasks such that no reward is given until all sub-tasks are completed. As an example, consider a house robot, which must collect coffee for the guest on the couch before turning on the TV and then ascending the stairs (Figure 1).

Three existing approaches for improving learning in this setting are hierarchical RL (Sutton, Precup, and Singh 1999; Pateria et al. 2021), which allows agents to plan at various levels of abstraction; transfer learning, which expedites the learning rate by utilising knowledge learnt from similar tasks (Taylor and Stone 2007); and temporal logic planning approaches, which guide the agent's exploration by focusing it on the portion of the MDP that satisfies a linear temporal logic (LTL) property (Hasanbeig, Abate, and Kroening 2018; Camacho et al. 2019; Araki et al. 2021; Thiébaux et al. 2006; Jothimurugan et al. 2021; Neary et al. 2022). In the latter, an LTL property specifying a task is usually represented as an automaton, which makes it similar to work on reward machines (Icarte et al. 2022, 2018) – finite-state machines that represent non-Markovian reward functions.

Accepted as a full paper to ECAI 2023.

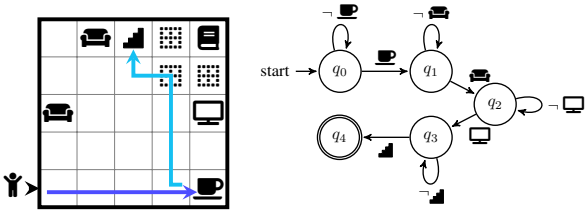

Figure 1: (a) A labelled MDP environment and (b) a 5-state TA (namely, a DFA) representing the task specification.

We concentrate on the following task: *how can one best learn the task automaton (TA) representing a task specification in sparse, non-Markovian environments?* Our approach learns a model of the underlying MDP along with the TA. Extensive work on temporal logic planning (Hasanbeig, Abate, and Kroening 2018; Camacho et al. 2019; Araki et al. 2021; Thiébaux et al. 2006; Jothimurugan et al. 2021; Neary et al. 2022), sample-efficient model-based RL approaches (Wang et al. 2019), or other methods (Baier and Katoen 2008; Tkachev et al. 2017) can then be used to synthesise an optimal policy.

Contributions: We devise an algorithmic pipeline to jointly learn both a task specification as a TA (encoded as a deterministic finite automaton(DFA)) and a model of the MDP, from episodes of agent experience within an unknown environment with non-Markovian reward. The reward is also sparse: it is only given when the full (and unknown) task is accomplished. This addresses existing sample inefficiency in these settings in three ways. First, a learnt TA exposes the sequential and separable nature of the task specification; sub-tasks can be independently solved (for an optimal policy) by the RL agent more efficiently (Hasanbeig et al. 2021; Icarte et al. 2022). Second, our approach is model-based and requires far fewer training episodes than related work

for learning automata; the learnt model also helps the agent learn, with the TA, an optimal policy more efficiently. Third, we take steps to remove environmental bias from our learned TA, making it more interpretable and better-suited for use in transfer learning (Taylor and Stone 2007).

We learn the TA via an intermediate product MDP structure, composed of the environment's '*spatial*' MDP and the task specification's TA, both of which are initially unknown. The product MDP is partially observable – the agent observes the spatial MDP's states and whether it has received a reward, but not the TA-state component (i.e., its progress through the unknown task). The product MDP is learnt in two steps. First, we learn an estimate of the spatial MDP using the Baum-Welch algorithm (Baum and Petrie 1966) and a uniform prior. Second, we use this learnt spatial MDP as an inductive bias for learning the full product MDP.

We then show that once the product MDP is learnt, distilling the TA is computationally efficient. This results in two further contributions. First, our learnt product MDP is useful for transfer learning because if the environment changes, the agent only needs to update the affected part of the product MDP before re-distilling the TA. Second, our efficient 'Cone Lumping' method can be used to determinise any product-MDP structure, which can also expedite (bi)simulation or model reduction studies (Larsen and Skou 1991).

References

Araki, B.; Li, X.; Vodrahalli, K.; DeCastro, J.; Fry, M.; and Rus, D. 2021. The logical options framework. In *International Conference on Machine Learning*, 307–317. PMLR.

Baier, C.; and Katoen, J.-P. 2008. *Principles of model checking*. MIT press.

Baum, L. E.; and Petrie, T. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, 37(6): 1554–1563.

Camacho, A.; Icarte, R. T.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *IJCAI*, volume 19, 6065–6073.

Hasanbeig, M.; Abate, A.; and Kroening, D. 2018. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*.

Hasanbeig, M.; Jeppu, N. Y.; Abate, A.; Melham, T.; and Kroening, D. 2021. Deepsynth: Automata synthesis for automatic task segmentation in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7647–7656.

Icarte, R. T.; Klassen, T.; Valenzano, R.; and McIlraith, S. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, 2107–2116. PMLR.

Icarte, R. T.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2022. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73: 173–208.

Jothimurugan, K.; Bansal, S.; Bastani, O.; and Alur, R. 2021. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34: 10026–10039.

Larsen, K. G.; and Skou, A. 1991. Bisimulation through probabilistic testing. *Information and computation*, 94(1): 1–28.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Neary, C.; Verginis, C.; Cubuktepe, M.; and Topcu, U. 2022. Verifiable and compositional reinforcement learning systems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, 615–623.

Pateria, S.; Subagdja, B.; Tan, A.-h.; and Quek, C. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5): 1–35.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Taylor, M. E.; and Stone, P. 2007. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, 879–886.

Thiébaux, S.; Gretton, C.; Slaney, J.; Price, D.; and Kabanza, F. 2006. Decision-theoretic planning with non-Markovian rewards. *Journal of Artificial Intelligence Research*, 25: 17–74.

Tkachev, I.; Mereacre, A.; Katoen, J.-P.; and Abate, A. 2017. Quantitative model-checking of controlled discrete-time Markov processes. *Information and Computation*, 253: 1–35.

Wang, T.; Bao, X.; Clavera, I.; Hoang, J.; Wen, Y.; Langlois, E.; Zhang, S.; Zhang, G.; Abbeel, P.; and Ba, J. 2019. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*.

LTL_f Best-Effort Synthesis in Nondeterministic Planning Domains

Giuseppe De Giacomo[a,b], Gianmarco Parretti[b] and Shufang Zhu[a]

[a]University of Oxford, UK
[b]University of Rome "La Sapienza", Italy

1 Introduction

Recently there has been quite some interest in synthesis [11, 9] for realizing goals (or tasks) φ against environment specifications $\mathcal{E}$ [2, 3], especially when both φ and $\mathcal{E}$ are expressed in Linear Temporal Logic on finite traces (LTL_f) [7, 8], the finite trace variant of LTL [12]. In this setting, synthesis amounts to finding an agent strategy that wins, i.e., generates a trace satisfying φ, whatever is the (counter-)strategy chosen by the environment, which in turn has to satisfy its specification $\mathcal{E}$.

Obviously, a winning strategy for the agent may not exist. To handle this possibility, the notion of strong cyclic plans was introduced [5]: if a (strong) plan does not exist, there may still exist a plan that could win assuming the environment is not strictly adversarial. Building on this intuition, Aminof et al. [1] proposed the notion of best-effort strategies (or plans), which formally capture the idea that the agent could do its best by adopting a strategy that wins against a maximal set (though not all) of possible environment strategies. Best-effort strategies have some notable properties: (*i*) they always exist, (*ii*) if a winning strategy exists, then best-effort strategies are exactly the winning strategies, (*iii*) best-effort strategies can be computed in 2EXPTIME as winning strategies (best-effort synthesis is indeed 2EXPTIME-complete) [4].

In [4] an algorithm for LTL_f best-effort synthesis has been presented. Using this technique, we can also capture best-effort synthesis in nondeterministic planning domains. In particular, one can simply re-express nondeterministic planing domains (FOND) in LTL_f [7, 2, 10] and then use the LTL_f best-effort synthesis approach directly. However, observe that in planning, while the (temporally extended) goal φ is typically small, the environment specification $\mathcal{E}$ is large, being the entire planning domain. This observation motivates our paper.

We study LTL_f best-effort synthesis directly in the context of nondeterministic (adversarial) planning domains and present a technique to solve it. Our technique consists of constructing and solving two variants of reachability games, namely adversarial and cooperative, played over a shared game arena obtained from composing the planning domain's transition systems and the agent goal's DFA, and combining their solutions. We implemented our technique by leveraging the symbolic LTL_f framework in [13, 6] and performed an empirical evaluation of the effectiveness of the approach. Our results show that computing best-effort strategies for LTL_f goals in nondeterministic domains is much more effective than using LTL_f best-effort synthesis directly. Our technique can be implemented quite efficiently, with only a small overhead wrt to computing winning strategies (i.e., strong plans) in FOND.

Acknowledgments

This work has been partially supported by the ERC-ADG WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No. 20203FFYLK), and the PNRR MUR project FAIR (No. PE0000013). This work has been carried out while Gianmarco Parretti was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome.

References

[1] Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, and Sasha Rubin, 'Synthesizing strategies under expected and exceptional environment behaviors', in *IJCAI*, pp. 1674–1680, (2020).
[2] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin, 'Planning and synthesis under assumptions', *arXiv*, (2018).
[3] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin, 'Planning under LTL environment specifications', in *ICAPS*, pp. 31–39, (2019).
[4] Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin, 'Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up', in *IJCAI*, pp. 1766–1772, (2021).
[5] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso, 'Weak, strong, and strong cyclic planning via symbolic model checking.', *AIJ*, **1–2**(147), 35–84, (2003).
[6] Giuseppe De Giacomo, Gianmarco Parretti, and Shufang Zhu, 'Symbolic LTLf Best-Effort Synthesis', in *EUMAS*, (2023). To appear.
[7] Giuseppe De Giacomo and Moshe Y. Vardi, 'Linear Temporal Logic and Linear Dynamic Logic on Finite Traces', in *IJCAI*, pp. 854–860, (2013).
[8] Giuseppe De Giacomo and Moshe Y. Vardi, 'Synthesis for LTL and LDL on Finite Traces', in *IJCAI*, pp. 1558–1564, (2015).
[9] Bernd Finkbeiner, 'Synthesis of Reactive Systems.', *Dependable Software Systems Eng.*, **45**, 72–98, (2016).
[10] Keliang He, Andrew M Wells, Lydia E Kavraki, and Moshe Y Vardi, 'Efficient Symbolic Reactive Synthesis for Finite-Horizon Tasks', in *ICRA*, pp. 8993–8999, (2019).
[11] A. Pnueli and R. Rosner, 'On the synthesis of a reactive module', in *POPL*, p. 179–190, (1989).
[12] Amir Pnueli, 'The temporal logic of programs', in *FOCS*, pp. 46–57, (1977).
[13] Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi, 'Symbolic LTL_f Synthesis', in *IJCAI*, pp. 1362–1369, (2017).