Towards Strategy Repair for Adjustable Autonomy

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

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

Abstract

Strategy Repair is the problem of finding a minimal amount of modifications to turn a strategy for a game on graphs from losing into winning. This allows for minimally changing the programmed behavior of an agent, in response to unexpected changes or additional requirements one might need to fulfill at execution time. In this paper, we report on an extension of previous work concerning Strategy Repair for Reachability Games, and discuss the usefulness of Strategy Repair in addressing some issues related to Adjustable Autonomy, i.e., the idea that agents, in order to be autonomous, might also need to stop execution and ask for external (typically human) intervention.

Keywords

Strategic Reasoning, Formal Games,

1. Introduction

Strategy Repair is the problem of finding a minimal amount of modifications to turn a strategy for a game from losing into winning. This problem has been recently studied [1] in the setting of *Reachability Games* [2], i.e., finite-state games played by two players, where one, Player 0, aims at reaching a state from a *target* set, no matter how the other, Player 1, behaves. To fulfill its purpose, Player 0 needs a *strategy*, i.e., a function telling the player which move to perform next, in order to eventually achieve a target state. Synthesizing a *winning* strategy (for Player 0) in reachability games is a very well-studied problem in several areas, including Formal Synthesis [3, 4, 5, 6] and Fully Observable Nondeterministic (FOND) Planning [7], with the decision version known to be P-complete wrt the game size [2, 8, 9, 10].

Synthesized strategies may become invalid at runtime, as the result of, e.g., deviations from the model, goal changes consequent to unexpected situations, or the detection of a critical situation, where, e.g., safety or ethical aspects are crucial, possibly calling for human assistance. In these situations, a new strategy needs to be defined, which guarantees goal achievement, while possibly fulfilling a number of additional (soft) requirements possibly emerged at execution time and not explicitly modeled, which might require a significant effort to be satisfied, or not be automatically addressable at all.

This relates to the notion of *Adjustable Autonomy* (see, e.g., [11]), i.e., the idea that autonomous agents sometime need to be assisted, typically by humans, in carrying out their tasks. In other words, the agent's autonomy might need to be *adjusted* at execution time, in order to obtain the desired behavior, according not only to the primary goal the agent was designed for, such as reaching a desired position, but also to other requirements involving aspects that are difficult to model or guarantee, such as never harm people, act fairly, etc.

In general, the human might intervene directly by taking control of the agent at execution time or indirectly, by revising the strategy under execution (in a way that the agent could not do) and then, once done, let the agent execute it. However, in certain conditions, modifying a strategy (either at synthesis or at execution time) to a large extent may negatively affect the quality of the delivered solution, for a number of reasons. For instance, the agent might have been designed or optimized for the original strategy (think, e.g., of when the agent is equipped with specific hardware tailored on the actions prescribed by the original strategy), and deviating from it might require an additional overhead

AAPEI '24: 1st International Workshop on Adjustable Autonomy and Physical Embodied Intelligence, October 20, 2024, Santiago de Compostela, Spain.

0009-0009-7054-6360 (P. Gaillard); 0000-0002-9116-251X (F. Patrizi); 0000-0002-8687-6323 (G. Perelli)

at execution time in order to fulfill the objectives; or there might be communication inefficiencies due to low bandwidth or noisy channel, which prevent efficient communication of the revised strategy from the human to the agent. Obviously, many more situations exist where a strategy change may have a negative impact on the actual solution.

This work makes three contributions. Firstly, it presents a motivating scenario for Strategy Repair in the context of Adjustable Autonomy, for an environment where communication is critical and the amount of transmitted data should be minimized. Second, it reports a sound and complete solution approach for Strategy Repair in Reachability Games, together with a greedy (polynomial) approach which, while sub-optimal, returns solutions that much similar to the original strategy than those obtained from scratch. This second contribution appeared in [1]. Finally, we present preliminary results concerning the extension of the problem to Büchi Games, i.e., a more general setting, where the agent's goal consists in guaranteeing that some states from a target set can be visited infinitely often.

2. Preliminaries

A *2-player reachability game*, or simply *game* is a tuple $\mathcal{G} = \langle V, V_0, V_1, \mathrm{E}, \mathcal{T} \rangle$, where V is the set of *nodes*, or *vertices*, with $V = V_0 \cup V_1$ and $V_0 \cap V_1 = \emptyset$, $\mathrm{E} \subseteq V \times V$ is the set of *edges*, and $\mathcal{T} \subseteq V$ is a subset of nodes, sometimes called *target*. We say that V_0 is the set of nodes controlled by player 0 (P_0), whereas V_1 is the set of nodes controlled by player 1 (P_1). A structure $\mathcal{A} = \langle V, V_0, V_1, \mathrm{E} \rangle$ satisfying these properties is called an *arena*.

A *path* in the game is a sequence $\pi = v_0 \cdot v_1 \cdot v_2 \ldots \in V^\omega$ such that $(v_i, v_{i+1}) \in \mathrm{E}$ for each $i \in \mathbb{N}$. As usual, by π_i, we denote the i-th node occurring in the sequence π, whereas by $\pi_{\leq i}$ we denote the prefix of π up to node π_i, also called *partial path*. We say that a path π is *winning* for player 0 if $\pi_i \in \mathcal{T}$ for some $i \in \mathbb{N}$, otherwise it is winning for player 1. A strategy for player 0 is a function $\sigma_0 : V^* \cdot V_0 \to \mathrm{E}$ mapping partial paths to edges, such that $\sigma_0(v_0 \ldots v_n)$ is an edge outgoing from v_n, for each partial path in $V^* \cdot V_0$. A strategy σ_1 for player 1 is defined accordingly. A path π is *compatible* with strategy σ_0 if $\sigma_0(\pi_{\leq i}) = (\pi_i, \pi_{i+1})$ for each $\pi_i \in V_0$. Analogously, it is *compatible* with strategy σ_1 if $\sigma_1(\pi_{\leq i}) = (\pi_i, \pi_{i+1})$ for each $\pi_i \in V_1$.

We say that a strategy σ_0 is *winning* for player 0 from v, if every path π starting from v and compatible with σ_0 is *winning*. We say that a node v is winning for player 0 if there exists a strategy σ_0 winning from v. We denote by $\mathrm{Win}_0(\mathcal{G})$ and $\mathrm{Win}_1(\mathcal{G})$ the sets of nodes in $\mathcal{G}$ that are winning for player 0 and 1, respectively. Finally, a strategy is said to be simply *winning* if it is winning from every vertex in $\mathrm{Win}_0(\mathcal{G})$. It is well known that reachability games are *memoryless determined* [12], that is, every node v is either winning for player 0 or winning for player 1 and that there always exists a memoryless winning strategy. Therefore, from now on we restrict our attention to only memoryless strategies, that are defined as $\sigma_0 : V_0 \to \mathrm{E}$ mapping each node belonging to an agent to an outgoing edge. Such restriction allows us to define a very natural distance between two player 0 strategies σ_0 and σ_0' over the same game, that is $\mathsf{dist}(\sigma_0, \sigma_0') = |\{v \in V_0 \mid \sigma_0(v) \neq \sigma_0'(v)\}|$. Intuitively, the distance between two strategies counts the number of nodes on which they map to a different outgoing edge [1].

We conclude this section by introducing some useful notation. For a given subset $\mathrm{E}' \subseteq \mathrm{E}$ of edges, by $\mathcal{G}_{\mathrm{E}'}$ we denote the game where we remove every edge (v_1', v_2') incompatible with E', that is, such that $v_1' = v_1$ and $v_2' \neq v_2$, for some $(v_1, v_2) \in \mathrm{E}'$. Notice that a (memoryless) strategy σ_0 can be regarded as a subset of edges, one for each node in V_0, therefore $\mathcal{G}_{\sigma_0}$ denotes the game induced from $\mathcal{G}$ by removing every edge (v, v') incompatible with σ_0, that is, such that $v \in V_0$ and $(v, v') \neq \sigma_0(v)$. Note that every vertex of V_0 has only one successor in $\mathcal{G}_{\sigma_0}$, which means that player 0 has only strategy σ_0 available in the game.

3. The Strategy Repair Problem

We now introduce the *strategy repair* problem for reachability games. First, for a given reachability game $\mathcal{G}$ and a player 0 strategy σ_0, define $\mathrm{Win}_0(\mathcal{G}, \sigma_0)$ to be the set of nodes from which σ_0 is winning. It is

not hard to show that $\mathrm{Win}_0(\mathcal{G}, \sigma_0) = \mathrm{Win}_0(\mathcal{G}_{\sigma_0})$, that is, the nodes that are winning for player 0 when it is using strategy σ_0 can be obtained by considering the game $\mathcal{G}_{\sigma_0}$ where the choices incompatible with σ_0 have already been ruled out. Observe that it always holds that $\mathrm{Win}_0(\mathcal{G}, \sigma_0) \subseteq \mathrm{Win}_0(\mathcal{G})$, with $\mathrm{Win}_0(\mathcal{G}, \sigma_0) = \mathrm{Win}_0(\mathcal{G})$ if, and only if, σ_0 is winning for player 0. We define the strategy repair problem as follows.

Definition 1 (Strategy repair problem). *For a given reachability game $\mathcal{G}$ and a strategy σ_0, find a winning strategy σ_0' such that $\mathsf{dist}(\sigma_0, \sigma_0') \leq \mathsf{dist}(\sigma_0, \sigma_0'')$ for each winning strategy σ_0''.*

The problem introduced requires to minimize the number of modifications that are required to turn a strategy σ_0 into a strategy σ_0' winning for a given reachability game $\mathcal{G}$. The corresponding decision problem, instead, consists in fixing a given threshold $k \in \mathbb{N}$ and checking whether some winning strategy σ_0' exists with $\mathsf{dist}(\sigma_0, \sigma_0') \leq k$. In [1] we prove that the strategy repair problem for reachability games is NP-complete via a reduction from *vertex cover* [10].

Theorem 1. *The strategy repair problem for reachability games is NP-complete [1].*

4. Motivating Scenario

Consider a robot operating in a vast and distant environment with the objective of rescuing humans or finding a precious resource. The robot is autonomous but constantly connected to a control center which supervises the operations, collects data, and makes decisions at any time. Unfortunately, due to the nature of the environment the robot operates in, the communication channel is very noisy and requires a huge number of re-transmissions before data is actually received at the other end. As a result, in order to guarantee communication effectiveness and reliability, the amount of data to be transmitted, in both directions, must be minimized. The robot needs thus to be as much autonomous as possible.

The precise location of the objective (human or resource) is unknown, but the control center has identified a candidate search area and devised a respective search strategy that has been loaded onto the robot for execution, when at the control center, before starting the operation.

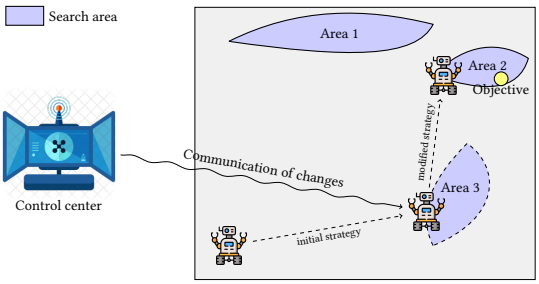

Figure 1: Robot Rescue

To the robot, the environment features many sources of uncertainty. For instance, some actions, such as *move forward*, might be imprecise and make the robot deviate from its nominal path, the robot might encounter obstacles along its path, and so on. The scenario can be modelled as a two-player Reachability Game where the target set corresponds to the search area and the opponent resolves the uncertainty about robot's moves. That is, the opponent controls the occurrence of the events over which the robot has uncertainty, such as, when moving forward yields drifting, or when an obstacle is present along the path.

Assume that the strategy guaranteeing reaching the search area, loaded onto the robot in the control center, is currently under execution and the robot is on the field. While the search progresses, data is sent to the control center, which collects and analyzes it, possibly revising the current target areas. For instance, after a while, the control center might realize that the initial area should be abandoned in favor of another, or that the robot is close to a human to rescue and thus particular care should be put in interacting with the person. In both these cases, the strategy followed by the robot needs to be adjusted. In theory, the control center could re-compute the new strategy and communicate it to the robot. Yet, the communication channel might prevent an entire strategy from being transmitted due to noise and losses. It becomes, thus, of crucial importance, reducing the amount of information to be sent to the robot, which can be done by computing a new strategy which achieves the (new) desired objective,

while remaining as similar as possible to the current one. In this case, indeed, only the differences wrt current strategy need to be communicated.

Figure 1 depicts this situation, where the robot initially follows the original strategy, which leads to area 3. However, when the control center realizes that the objective is located in a different place, it must adapt the strategy to reach the new area consisting of areas 1 and 2. To this aim, in order to minimize communication, the control center can repair the strategy, and then send only the adjustments to the robot. If time-efficiency is required, instead of computing an optimal solution, the control center might compute a sub-optimal with a greedy algorithm, as discussed later on in the work.

5. Algorithmic Solutions

We now present two algorithms for Strategy Repair, which we called `Opt` and `Greedy`, respectively. The former returns the optimal solution to the problem, but runs in exponential time. The latter, instead, returns a sub-optimal solution but runs in polynomial time. It is important to remark that they both produce correct winning strategies for the game. However, the algorithm `Greedy` does not provide the most optimal one in terms of distance from the original strategy.

We now proceed with the description of Algorithm `Opt`. In order to do so, we first introduce some useful definition. For a given game $\mathcal{G}$ and a set $X \subseteq V$ of nodes, the *Frontier* of X, denoted $\text{Frontier}_0(X) = ((V_0 \setminus X) \times X) \cap \text{E}$, is the set of edges that are outgoing from a Player 0 node and incoming to a node in X. Intuitively, the edges in Frontier_0 can be used by Player 0 to enter in a single step the region X. Consider a game $\mathcal{G}$ and a strategy σ_0, and let $X = \text{Win}_0(\mathcal{G}, \sigma_0)$ be the set of nodes that are winning for strategy σ_0. Observe that for an edge $(v, v') \in \text{Frontier}_0(X)$, it holds that $\sigma_0(v) \neq (v, v')$, otherwise v would have been winning for σ_0 in the first place. Moreover, it is trivial to show that the strategy $\sigma_0' = \sigma_0[v \mapsto (v, v')]$ is such that $\text{Win}_0(\mathcal{G}, \sigma_0) \subsetneq \text{Win}_0(\mathcal{G}, \sigma_0')$, with the inclusion being proper because $v \in \text{Win}_0(\mathcal{G}, \sigma_0') \setminus \text{Win}_0(\mathcal{G}, \sigma_0)$.

Algorithm 1: `Opt`.

Input: $\mathcal{G}$ a reachability game, σ_0 a strategy for player 0
Output: Winning strategy minimizing the distance from σ_0

```
Fix(G, σ0) :
T' ← Win0(G, σ0)
if T' = Win0(G) then
    return (σ0, 0)
else
    select (v, v') from Frontier(T')
    (σ0', β') ← Fix(G, σ0[v ↦ (v, v')])
    G' ← G_σ0(v)
    if v ∈ Win0(G') then
        (σ0'', β'') ← Fix(G', σ0)
        if β'' < β' + 1 then
            return (σ0'', β'')
        end
    end
    return (σ0', β' + 1)
end
```

We are now ready to present the algorithm `Opt`, which is reported in Algorithm 1. The algorithm works as follow. First, it computes the winning region following σ_0 $T' = \text{Win}_0(\mathcal{G}, \sigma_0)$ and compares it with the winning region of the game $\text{Win}_0(\mathcal{G})$. If the two sets are equal, it means that σ_0 is already winning, so it returns the optimal solution $(\sigma_0, 0)$, with the second component denoting the cost of fixing. If that is not the case, the algorithm proceeds by first computing the frontier of $\text{Win}_0(\mathcal{G}, \sigma_0)$, in order to select an edge (v, v') from it, then it compares two possible solutions. The first is obtained by solving the problem where the initial strategy is $\sigma_0[v \mapsto (v, v')]$, obtained from σ_0 by diverting the choice on v with the frontier edge (v, v'). The second is obtained by solving the problem when Player 0 is forced to select edge $\sigma_0(v)$ in v. This is obtained by considering the game $\mathcal{G}' = \mathcal{G}_{\sigma_0(v)}$, where all other outgoing edges from v are removed. Both solutions are computed with their relative costs β' and β'', which are then compared to select the best between the two. Note that the latter solution might not exist, as the choice of σ_0 in v might lead, for instance, out of the winning region. The algorithm then first checks whether such solution is viable before making a useless recursive call on $(\mathcal{G}', \sigma_0)$. Observe that in the first case the total modification cost β' must be increased by 1, as the initial strategy $\sigma_0[v \mapsto (v, v')]$ is at distance 1 from σ_0 itself. We have the following.

Theorem 2. *The algorithm* `Opt` *returns the optimal solution to the Strategy Repair problem.*

The algorithm `Opt` presented in the previous section is of exponential complexity, as it requires two recursive calls at each iteration to compare the distances between the initial strategy and two candidate best solutions. Also, notice that the recursive call that makes use of the selected edge in the frontier always computes a correct solution, although it might not be the optimal one. Therefore, a suboptimal but polynomially computable solution could be found by just selecting the one obtained from such call, disregarding the other. This is how the algorithm `Greedy` is conceived.

However, in order to improve the quality of the solution, i.e., the accuracy w.r.t. the optimum, we employ a selection criterion for the edge in the frontier set. Indeed, consider an instance $(\mathcal{G}, \sigma_0)$ of Strategy Repair, and an edge $(v, v') \in \text{Frontier}_0(\text{Win}_0(\mathcal{G}, \sigma_0))$. First, note that $\sigma_0(v) \neq (v, v')$, otherwise, the node v would be winning for σ_0 and (v, v') would not be in the frontier. Therefore, consider the set $\mathsf{Repair}_{\sigma_0}(v, v') = \text{Win}_0(\mathcal{G}, \sigma_0[v \mapsto (v, v')]) \setminus \text{Win}_0(\mathcal{G}, \sigma_0)$, that is, the set of nodes that are indirectly repaired by using the frontier edge (v, v') in the solution. When selecting the frontier edge, one might decide to greedily maximize the number of nodes that are indirectly repaired by such a selection. This is how the algorithm `Greedy` works, as it is presented in Algorithm 2.

Algorithm 2: `Greedy`.

Input: $\mathcal{G}$ a reachability game, σ_0 a strategy for player 0
$\mathsf{Fix}(\mathcal{G}, \sigma_0)$:
$T' \leftarrow \text{Win}_0(\mathcal{G}, \sigma_0)$
if $T' = \text{Win}_0(\mathcal{G})$ **then**
  return $(\sigma_0, 0)$
end
$F \leftarrow \text{Frontier}_0(T')$
$(v, v') \leftarrow \mathsf{argmax}\{|\mathsf{Repair}_{\sigma_0}(v, v')|; (v, v') \in F\}$
$(\sigma'_0, \beta') \leftarrow \mathsf{Fix}(\mathcal{G}, \sigma_0[v \mapsto (v, v')])$
return $(\sigma'_0, \beta' + 1)$

6. Strategy Repair for Büchi Games

We now study the strategy repair problem on Büchi games. A Büchi game is defined by $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle$ where $\mathcal{A} = \langle V, V_0, V_1, \text{E} \rangle$ is an arena and $\mathcal{T} \subseteq V$ is a target set. In a Büchi game, a path π is winning for P_0 when for all j, there exists $i \geqslant j$ such that $\pi_i \in \mathcal{T}$, *i.e.* when the target is reached infinitely many times.

The notion of winning strategy and winning region can be defined similarly as for the reachability games but with the Büchi winning condition. From the complexity analysis of RG, it immediately follows that the lower-bound for strategy repair in Büchi games is NP. In addition, notice that a guess-and-check approach for solving strategy repair with Büchi objectives still works. This is because also Büchi games can be solved in polynomial time. This provides us with the following result.

Theorem 3. *The strategy repair problem for Büchi games is NP-complete.*

Now, we turn to solving the problem in practice. To do so, we introduce the operator Pre_0 defined by $\mathsf{Pre}_0(X) = \{u \in V_0 \mid \exists (u, v) \in \text{E}, v \in X\} \cup \{u \in V_1 \mid \forall (u, v) \in \text{E}, v \in X\}$ for a given set of vertices $X \subseteq V$, which corresponds to the set of vertices from which P_0 can ensure to reach X at the next step.

The algorithm for strategy repair will use the following key lemma inspired from [13] :

Lemma 1. *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle$ be a Büchi game and W be its winning region. Then, W is equal to the winning region of the reachability game $\langle \mathcal{A}, \mathcal{T}' \rangle$ where $\mathcal{T}' = \mathcal{T} \cap \mathsf{Pre}_0(W)$.*

Intuitively, to reach $\mathcal{T}$ infinitely many times from W, P_0 must ensure to reach the part of the target from which it is possible to remain in W at the next step.

As a consequence, if $\mathcal{T}'$ stands for $\mathcal{T} \cap \mathsf{Pre}_0(W)$, note that any strategy ensuring player 0 to reach $\mathcal{T}'$ from $W \setminus \mathcal{T}'$ and going to W from $\mathcal{T}'$ (which is possible since $\mathcal{T}' \subseteq \mathsf{Pre}_0(W)$) is winning since any induced play would be of the form $((W \setminus \mathcal{T}')^*.\mathcal{T}')^\omega$ with $\mathcal{T}' \subseteq \mathcal{T}$, and computing such strategies is equivalent to solving a reachability game where the target is $\mathcal{T} \cap \mathsf{Pre}_0(W)$. This induces the algorithm 3, which uses the algorithm of strategy repair for reachability games.

Algorithm 3: Algorithm solving the strategy repair problem for Büchi games

Input: $\mathcal{G}$ a Büchi game, σ_0 a strategy for player 0
Output: Winning strategy for $\mathcal{G}$ minimizing the distance from σ_0

```
Fix_Buchi(G, σ0) :
if σ0 winning in G then
    return σ0
else
    W ← Win0(G)
    T' ← T ∩ Pre0(W)
    σ'0 ← Fix_Reachability(⟨A, T'⟩, σ0)        // minimal modifications to reach T'
    for v ∈ V0 ∩ T' do
        if σ0(v) ∉ W then
            find (v0, v') ∈ E such that v' ∈ W        // fixing the strategy in T'
            σ'0(v) ← (v, v')
        end
    end
    return σ'0
end
```

Note that both algorithms `Opt` and `Greedy` can be used for finding the minimal modifications to reach $\mathcal{T}'$, producing an exact solution in exponential time, or a suboptimal solution in polynomial time. As before with reachability games, note that in the second case, the strategy obtained is winning even if it does not minimize the distance.

7. Future Work

This work is an initial investigation into the problem of Strategy Repair and leaves at least two interesting questions open. Firstly, while the polynomial algorithm exhibits outstanding experimental performance, no approximation guarantee was obtained. For future work, we aim to study such a property. Secondly, it is interesting to go beyond simple reachability and Büchi to apply the repair approach to other problems. In particular, one immediate extension would be to investigate the applicability and effectiveness of the approach for *strong cyclic* [7] solutions.

Acknowledgments

The authors are supported by the PNRR MUR project PE0000013-FAIR, the Italian Ministry of University and Research (MUR) under PRIN grant B87G22000450001 (PINPOINT) and by Sapienza University of Rome under the Progetti Grandi di Ateneo programme, grant RG123188B3F7414A (ASGARD - Autonomous and Self-Governing Agent-Based Rule Design) and the Sapienza project MARLeN (Multi-layer Abstraction for Reinforcement Learning with Non-Markovian Rewards).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] P. Gaillard, F. Patrizi, G. Perelli, Strategy Repair in Reachability Games., in: K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, R. Radulescu (Eds.), ECAI'23, volume 372 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 780–787. URL: https://doi.org/10.3233/FAIA230344. doi:10.3233/FAIA230344.

[2] E. Grädel, W. Thomas, T. Wilke (Eds.), 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. URL: https://doi.org/10.1007/3-540-36387-4. doi:10.1007/3-540-36387-4.

[3] G. De Giacomo, M. Vardi, Synthesis for LTL and LDL on finite traces, in: Q. Yang, M. J. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 1558–1564. URL: http://ijcai.org/Abstract/15/223.

[4] A. Camacho, C. Muise, J. Baier, S. McIlraith, LTL realizability via safety and reachability games, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 4683–4691. URL: https://doi.org/10.24963/ijcai.2018/651. doi:10.24963/ijcai.2018/651.

[5] A. Camacho, J. Baier, C. Muise, S. A. McIlraith, Finite LTL synthesis as planning, in: M. de Weerdt, S. Koenig, G. Röger, M. T. J. Spaan (Eds.), Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018, AAAI Press, 2018, pp. 29–38. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17790.

[6] G. De Giacomo, M. Favorito, J. Li, M. Vardi, S. Xiao, S. Zhu, Ltlf synthesis as AND-OR graph search: Knowledge compilation at work, in: L. D. Raedt (Ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, ijcai.org, 2022, pp. 2591–2598. URL: https://doi.org/10.24963/ijcai.2022/359. doi:10.24963/ijcai.2022/359.

[7] M. Daniele, P. Traverso, M. Y. Vardi, Strong cyclic planning revisited, in: S. Biundo, M. Fox (Eds.), Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings, volume 1809 of *Lecture Notes in Computer Science*, Springer, 1999, pp. 35–48. URL: https://doi.org/10.1007/10720246_3. doi:10.1007/10720246_3.

[8] C. H. Papadimitriou, Computational complexity, Academic Internet Publ., 2007.

[9] S. Dasgupta, C. H. Papadimitriou, U. V. Vazirani, Algorithms, McGraw-Hill, 2008.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

[11] Proc. of the the AAAI Spring Symposium on Agents with Adjustable Autonomy, AAAI, 1999. URL: https://aaai.org/proceeding/spring-1999-06/.

[12] D. Perrin, J. Pin, Infinite words - automata, semigroups, logic and games, volume 141 of *Pure and applied mathematics series*, Elsevier Morgan Kaufmann, 2004.

[13] F. Horn, N. Fijalkow, Büchi games, in: N. Fijalkow (Ed.), Games on Graphs, Online, 2023.