Most General Winning Secure Equilibria Synthesis in Graph Games*

Satya Prakash Nayak(✉) and Anne-Kathrin Schmuck

Max Planck Institute for Software Systems, Kaiserslautern, Germany
{sanayak,akschmuck}@mpi-sws.org

Abstract This paper considers the problem of co-synthesis in k-player games over a finite graph where each player has an individual ω-regular specification ϕ_i. In this context, a secure equilibrium (SE) is a Nash equilibrium w.r.t. the lexicographically ordered objectives of each player to first satisfy their own specification, and second, to falsify other players' specifications. A *winning secure equilibrium* (WSE) is an SE strategy profile $(\pi_i)_{i\in[1;k]}$ that ensures the specification $\phi := \bigwedge_{i\in[1;k]} \phi_i$ if no player deviates from their strategy π_i. Distributed implementations generated from a WSE make components *act rationally* by ensuring that a deviation from the WSE strategy profile is *immediately punished* by a *retaliating strategy* that makes the involved players lose.

In this paper, we move from *deviation punishment* in WSE-based implementations to a *distributed, assume-guarantee based realization* of WSE. This shift is obtained by generalizing WSE from *strategy profiles* to *specification profiles* $(\varphi_i)_{i\in[1;k]}$ with $\bigwedge_{i\in[1;k]} \varphi_i = \phi$, which we call *most general winning secure equilibria (GWSE)*. Such GWSE have the property that each player can individually pick a strategy π_i winning for φ_i (against all other players) and all resulting strategy profiles $(\pi_i)_{i\in[1;k]}$ are guaranteed to be a WSE. The obtained flexibility in players' strategy choices can be utilized for robustness and adaptability of local implementations. Concretely, our contribution is three-fold: (1) we formalize GWSE for k-player games over finite graphs, where each player has an ω-regular specification ϕ_i; (2) we devise an *iterative semi-algorithm* for GWSE synthesis in such games, and (3) obtain an *exponential-time algorithm* for GWSE synthesis with parity specifications ϕ_i.

Keywords: Distributed Synthesis, Parity Games, Secure Equilibria, Assume-Guarantee Contracts

1 Introduction

Games over graphs provide a well known abstraction for many challenging correct-by-construction synthesis problems for software and hardware in embedded cyber-physical applications. In particular, the correct-by-construction co-synthesis of

* Authors are supported by the DFG project 389792660 TRR 248-CPEC. Additionally A.-K. Schmuck is supported by the DFG project SCHM 3541/1-1.

B. Finkbeiner and L. Kovács (Eds.): TACAS 2024, LNCS 14572, pp. 173–193, 2024.
https://doi.org/10.1007/978-3-031-57256-2_9

multiple interacting (reactive) components – each with its own correctness specification – poses, as of today, severe challenges in automated system design.

While many of these challenges arise from the fact that not every component has the same information about all relevant variables in the system, even in the seemingly simple setting of *full information* – where all components see the valuation to all variables – finding the right balance between centralized and local reasoning for co-synthesis is surprisingly challenging. While assuming all players to cooperate might demand too much commitment from individual components, a fully adversarial setting where all other components are assumed to harm a local implementation (independently of their own objective) might not capture a realistic scenario either.

To address this issue, starting with the seminal work of Chatterjee et al. [13], the concept of *rationality* – stemming from classical game theory – was brought to graph games in order to formalize a more realistic model for interaction of multiple components in co-synthesis. The main conceptual contribution of [13] was the introduction of *secure equilibria* (SE) – a special sub-class of Nash equilibria – given as particular strategy profiles. Intuitively, an SE is a Nash equilibrium w.r.t. the lexicographically ordered objectives of each player to first satisfy their own specification, and only second, to falsify other players' specifications. More specifically, it is a strategy profile, i.e., a tuple $(\pi_i)_i$, with π_i being the strategy of Player i, such that no player can improve w.r.t. their lexicographically ordered objective by deviating from this strategy.

As stated by [13, p.68], an SE can thus be interpreted as a contract between the players which enforces cooperation: any unilateral selfish deviation by one player cannot put the other players at a disadvantage if they follow the SE. While this property makes SE very desirable, their main draw-back, as most prominently pointed out by [5], is their restriction to a *single* strategy profile. This, in combination with classical reactive synthesis engines typically preferring small and goal-oriented strategies, incentivizes "immediate punishment" of deviations from an SE strategy profile in the final implementation.

Motivating Example. To illustrate this effect, let us consider the game depicted in Fig. 1, taken from [13]. Here, an SE can be described as follows: if Player 1 always chooses $v_3 \to v_1$ (forming π_1) and Player 2 always chooses $v_0 \to v_2$ and $v_2 \to v_3$ (forming π_2), then they both satisfy their specifications; if Player 1 deviates by choosing $v_3 \to v_2$ (risking falsification of ϕ_2), then Player 2 can retaliate by choosing $v_2 \to v_4$ (ensuring falsification of both ϕ_i); similarly, if Player 2 deviates by choosing $v_0 \to v_3$ (risking falsification of ϕ_1), then Player 1 retaliate by choosing $v_3 \to v_4$ (ensuring falsification of both ϕ_i). Clearly, the strategy profile (π_1, π_2) is an SE. It is, in particular, a *winning* SE as both players sat-

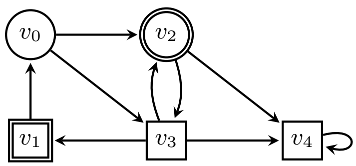

Figure 1: A two-player game with Player 1's vertices (squares), Player 2's vertices (circles) where Player i's specification $\phi_i = \Box\Diamond v_i$ is to visit v_i infinitely often.

isfy their specifications when following it. However, as the outlined retaliating strategies (π'_1, π'_2) are also part of the final implementation generated from this SE, any play that deviates from (π_1, π_2) *only once*, makes the game end up in a loop at v_4 resulting in neither player satisfying their objectives. Intuitively, this way of implementing SE-based strategies makes components *act rationally* by ensuring that a deviation from the contract is *immediately punished*.

Having the interpretation of an SE as a contract in mind, it is however very appealing to think about the *realization* of this contract in the final implementation in a more *permissive* way. Intuitively, in the game depicted in Fig. 1, both players can satisfy their specifications ϕ_i without the help by the other player, as long as the play does not go to v_4. In particular, whenever both players independently chose a strategy π_i which ensures that they (i) never take their edge to v_4 and (ii) satisfy ϕ_i for every strategy π_{-i} of the other player that also never takes their edge to v_4, forms an SE strategy profile (π_1, π_2). These *minimal cooperation obligations* for an SE can be interpreted as a *specification profile* (φ_1, φ_2), s.t. $\varphi_1 := \psi_1 \wedge (\psi_2 \Rightarrow \phi_1)$ and $\varphi_2 := \psi_2 \wedge (\psi_1 \Rightarrow \phi_2)$, where $\psi_1 = \square\neg(v_3 \wedge \bigcirc v_4)$ and $\psi_2 = \square\neg(v_2 \wedge \bigcirc v_4)$ express the above discussed assumption that Player i does not move to v_4 from their vertex. It turns out, that this new *specification profile* (φ_1, φ_2) has three nice properties: (i) it is *most general* meaning it does not lose any cooperative solution, i.e., $\phi_1 \wedge \phi_2 = \varphi_1 \wedge \varphi_2$, (ii) it is *realizable*, i.e., Player i has a strategy π_i that satisfies φ_i in a zero-sum sense, (i.e., no matter what the other player does) and, most importantly (iii) it is *secure (winning)*, i.e., every strategy profile (π_1, π_2), where Player i's strategy π_i satisfies φ_i (in a zero-sum sense) is a *winning* SE. While properties (i) and (iii) motivated us to call the set of new specifications a *most general winning secure equilibrium* (GWSE), property (ii) ensures that any specification φ_i from this tuple is locally and fully independently realizable by every component. Conceptually, this allows us to move from *deviation-punishment* in SE-based implementations to a *distributed, assume-guarantee based realization* of SE.

Contribution. By moving from *strategy profiles* (WSE) to *specification profiles* (GWSE) for SE realizations, our approach takes the conceptualisation of rationality for distributed synthesis to an extreme: as we are in the position to *design* every component (as it is a computer system not a human that actually acts rationally) we can enforce that implementations respect the new specifications φ_i. We only use the *concept of rationality* encoded in WSE to automatically obtain *meaningful and implementable* distributed specifications φ_i for this co-design process. Thereby the implementation of an accompanying *punishment mechanism* to enforce rationality of players becomes obsolete. The obtained flexibility in players' strategy choices can be utilized for robustness and adaptability of local implementations, which makes GWSE particularly suited for embedded systems applications.

Concretely, our contribution is three-fold: (1) We formalize GWSE for k-player games over finite graphs, where each player has an ω-regular specification. (2) We devise an *iterative semi-algorithm*[1] for GWSE synthesis under ω-regular

[1] A semi-algorithm is an algorithm that is not guaranteed to halt on all inputs.

specifications. (3) We give a (sound but incomplete) *exponential-time algorithm* for GWSE synthesis under *parity* specifications.

Other Related Work. After the introduction of *secure equilibria* (SE) by Chatterjee et al. [13], there has been several efforts on extending the notion to other classes of games, e.g., games with sup, inf, lim sup, lim inf, and mean-payoff measures [9], multi-player games with probabilistic transitions [17] or quantitative reachability games [8]. Furthermore, a variant of secure equilibria, called *Doomsday equilibria* was studied in [12], where if any coalition of players deviates and violates one players' objective, then the objective of every player is violated. Moreover, the notion of secure equilibria has been applied effectively in the synthesis of mutual-exclusion protocols [15,4] and fair-exchange protocols [21,23].

Motivated by similar insights, other concepts of rationality have also been introduced in multi-player games, e.g. subgame perfect equilibria [29,7,28,10,6] or rational synthesis [20,22,18]. Similar to the implementations of SE by [13], these works restrict implementations to a *single* strategy profile. In contrast, our work introduces a more flexible concept of rationality that is closely related to contract-based distributed synthesis, as in [24,19,16,2]. Here, an assume-guarantee contract is synthesized, such that every strategy realizing the guarantee is ensured to win whenever the other players satisfy the assumption. While this is conceptually similar to our synthesis of GWSE, these works do not consider the players to be adversarial, and hence, there is no notion of *equilibria*.

To the best of our knowledge, the only other work that also combines flexibility with equilibria is *assume-admissible (AA) synthesis* [5]. Their work utilizes a different, incomparable definition of rationality based on a dominance order. Both approaches are incomparable – there exist co-synthesis problems where our approach successfully synthesizes a GWSE and no AA contract exists, and vice versa (see Ex. 1 for details). Conceptually, AA contracts still require *rational* behaviour of players within the contract, while our approach only uses rationality as a *concept* to synthesize meaningful local specifications which can then be implemented in an arbitrary (non-rational) manner. We believe that this is a superior strength of our approach compared to AA synthesis.

2 Preliminaries

Notation. We use $\mathbb{N}$ to denote the set of natural numbers including zero. Given $a, b \in \mathbb{N}$ with $a < b$, we use $[a;b]$ to denote the set $\{n \in \mathbb{N} \mid a \leq n \leq b\}$. For any given set $[a;b]$, we write $i \in_{\text{even}} [a;b]$ and $i \in_{\text{odd}} [a;b]$ as short hand for $i \in [a;b] \cap \{0,2,4,\ldots\}$ and $i \in [a;b] \cap \{1,3,5,\ldots\}$ respectively. For a finite alphabet Σ, Σ^* and Σ^ω denote the set of finite and infinite words over Σ, respectively.

Linear Temporal Logic (LTL). Given a finite set $\mathtt{AP}$ of atomic propositions, *linear temporal logic (LTL)* formulas over $\mathtt{AP}$ are defined by the grammar:

$$\phi \coloneqq p \in \mathtt{AP} \mid \phi \vee \phi \mid \neg\phi \mid \bigcirc\phi \mid \phi \; \mathcal{U} \; \phi,$$

where $\vee$, $\neg$, $\bigcirc$, and $\mathcal{U}$ denotes the operators *disjunction*, *negation*, *next*, and *until*, respectively. Furthermore, we use the usual derived operators, $\mathsf{True} = p \vee \neg p$,

$\mathsf{False} = \neg\mathsf{True}$, *conjunction* $\phi \wedge \phi' = \neg(\neg\phi \vee \neg\phi')$, *implication* $\phi \Rightarrow \phi' = \neg\phi \vee \phi'$, and other temporal operators such as *finally* $\Diamond\phi = \mathsf{True}\,\mathcal{U}\,\phi$ and *globally* $\Box\phi = \neg\Diamond\neg\phi$. The semantics of LTL formulas are defined as usual (see standard textbooks [3]).

Game Graphs. A *k-player (turn-based) game graph* is a tuple $G = (V, E, v_0)$ where (V, E, v_0) is a finite directed graph with *vertices* V and *edges* E, and $v_0 \in V$ is an initial vertex. For such a game graph, let $\mathtt{P} = [1;k]$ be the set of players such that $V = \bigcup_{i\in\mathtt{P}} V_i$ is partioned into vertices of k players in $\mathtt{P}$. We write E_i, $i \in \mathtt{P}$, to denote the edges from Player i's vertices, i.e., $E_i = E \cap (V_i \times V)$. Further, we write V_{-i} and E_{-i} to denote the set $\bigcup_{j\neq i} V_j$ and $\bigcup_{j\neq i} E_j$, respectively. A *play* from a vertex u_0 is a finite or infinite sequence of vertices $\rho = u_0 u_1 \ldots$ with $(u_j, u_{j+1}) \in E$ for all $j \geq 0$.

Specifications. Given a game graph G, we consider *specifications* specified using a LTL formula Φ over the vertex set V, that is, we consider LTL formulas whose atomic propositions are sets of vertices V. In this case the set of desired infinite plays is given by the semantics of ϕ over G, which is an ω-regular language $\mathcal{L}(G, \phi) \subseteq V^\omega$. We just write $\mathcal{L}(\phi)$ to denote this language when the game graph G is clear in the context. Every game graph with an arbitrary ω-regular set of desired infinite plays can be reduced to a game graph (possibly with an extended set of vertices) with an LTL objective, as above. The standard definitions of ω-regular languages are omitted for brevity and can be found in standard textbooks [3]. To simplify notation we use $e = (u, v)$ in LTL formulas as syntactic sugar for $u \wedge \bigcirc v$.

Games and Strategies. A *k-player game* is a pair $\mathcal{G} = (G, (\phi_i)_{i\in\mathtt{P}})$ where G is a k-player game graph and each ϕ_i is an *objective* for Player i over G. A strategy of Player i, $i \in \mathtt{P}$, is a function $\pi_i\colon V^* V_i \to V$ such that for every $\rho v \in V^* V_i$, it holds that $(v, \pi_i(\rho v)) \in E$. A strategy profile for a set of players $\mathtt{P}' \subseteq \mathtt{P}$ is a tuple $\Pi = (\pi_i)_{i\in\mathtt{P}'}$ of strategies, one for each player in $\mathtt{P}'$. To simplify notation, we write $\mathtt{P}_{-i}$ and π_{-i} to denote the set $\mathtt{P} \setminus \{i\}$ and their strategy profile $(\pi_j)_{j\in\mathtt{P}\setminus\{i\}}$, respectively. Given a strategy profile $(\pi_i)_{i\in\mathtt{P}'}$, we say that a play $\rho = u_0 u_1 \ldots$ is $(\pi_i)_{i\in\mathtt{P}'}$*-play* if for every $i \in \mathtt{P}'$ and for all $\ell \geq 1$, it holds that $u_{\ell-1} \in V_i$ implies $u_\ell = \pi_i(u_0 \ldots u_{\ell-1})$.

Satisfying Specifications. Given a game graph G and a specification ϕ, a play ρ *satisfies* ϕ if $\rho \in \mathcal{L}(\phi)$. A strategy profile $(\pi_i)_{i\in\mathtt{P}'}$ satisfies/winning w.r.t. a specification ϕ, from a vertex v, denoted by $(\pi_i)_{i\in\mathtt{P}'} \vDash_v \phi$, if every $(\pi_i)_{i\in\mathtt{P}'}$-play from v satifies ϕ. We just write $(\pi_i)_{i\in\mathtt{P}'} \vDash \phi$ if v is the initial vertex. We collect all vertices from which there exists a strategy profile for players in $\mathtt{P}'$ that satisfies ϕ in the winning region[2] $\langle\!\langle \mathtt{P}' \rangle\!\rangle(G, \phi)$. We just write $\langle\!\langle \mathtt{P}' \rangle\!\rangle\phi$ to denote this set if game graph G is clear in the context. Furthermore, we write ϕ_{-i} to denote $\bigwedge_{j\in\mathtt{P}_{-i}} \phi_j$.

Parity Specifications. Give a game graph $G = (V, E, v_0)$, a specification ϕ is called *parity* if $\phi = \mathit{Parity}(\Omega) \coloneqq \bigwedge_{i\in_{\mathrm{odd}}[0;d]} \left(\Box\Diamond\Omega_i \Rightarrow \bigvee_{j\in_{\mathrm{even}}[i+1;d]} \Box\Diamond\Omega_j\right)$, with $\Omega_i = \{v \in V \mid \Omega(v) = i\}$ for some priority function $\Omega : V \to [0;d]$ that assigns each vertex a priority. A play satisfies such a specification if the maximum of priorities seen infinitely often is even.

[2] Slightly abusing notation, we write $\langle\!\langle i \rangle\!\rangle\phi$ for singleton sets of players $\mathtt{P}' = \{i\}$.

3 Most General Winning Secure Equilibria

This section formalizes most general winning secure equilibria (GWSE). In order to do so, we first recall the notion of secure equilibria from [13].

Secure Equilibria. Given a *k-player game* $\mathcal{G} = (G, (\phi_i)_{i\in\mathsf{P}})$ and a strategy profile $\Pi := (\pi_i)_{i\in\mathsf{P}}$ one can define a payoff profile, denoted by $\mathsf{payoff}(\Pi)$, as the tuple $(p_i)_{i\in\mathsf{P}}$ s.t. $p_i = 1$ iff $\Pi \models \phi_i$. With this, we can define a Player j preference order $\prec_j$ on payoff profiles lexicographiacally, s.t.

$$(p_i)_{i\in\mathsf{P}} \prec_j (p'_i)_{i\in\mathsf{P}} \text{ iff } (p_j < p'_j) \vee \big((p_j = p'_j) \wedge (\forall i \neq j.p_i \geq p'_i) \wedge (\exists i \neq j.p_i > p'_i)\big).$$

Intuitively, this preference order captures the fact that every player's main objective is to satisfy their own specification ϕ_i, and, as a secondary objective, falsify the specifications of the other players.

Definition 1. *Given a* k-player game $\mathcal{G} = (G, (\phi_i)_{i\in\mathsf{P}})$*, a strategy profile* $\Pi := (\pi_i)_{i\in\mathsf{P}}$ *is a* secure equilibrium *(SE) if for all* $i \in \mathsf{P}$*, there does not exist a strategy* π'_i *of Player* i *such that* $\mathsf{payoff}(\Pi) \prec_i \mathsf{payoff}(\pi'_i, \pi_{-i})$.

It is well known that every secure equilibrium is also a nash equilibrium in the classical sense. Within this paper, we only consider *winning secure equilibria* (WSE) i.e., SE with the payoff profile $(p_i = 1)_{i\in\mathsf{P}}$. As WSE have a trivial payoff profile, they can be characterized without referring to payoffs as formalized next.

Definition 2. *Give a k-player game* $(G, (\phi_i)_{i\in\mathsf{P}})$*, a* winning secure equilibrium (WSE) *is a strategy profile* $(\pi_i)_{i\in\mathsf{P}}$ *such that (i)* $(\pi_i)_{i\in\mathsf{P}} \models \bigwedge_{i\in\mathsf{P}} \phi_i$*; and (ii) for every strategy* π'_i *of Player* i*, if* $(\pi'_i, \pi_{-i}) \not\models \phi_{-i}$ *holds, then* $(\pi'_i, \pi_{-i}) \not\models \phi_i$ *holds.*

Intuitively, item i ensures that the strategy profile satisfies all player's objective, whereas item ii ensures that no player can improve, i.e., falsify another player's objective without falsifying their own objective, by deviating from the prescribed strategy.

Most General Winning Secure Equilibria. As illustrated by the motivating example in Sec. 1, we aim at generalizing WSE from single *strategy profiles* to *specification profiles* that capture an infinite number of WSE. These specification profiles $(\varphi_i)_{i\in\mathsf{P}}$, which we call *most general winning secure equilibria* (GWSE), allow each player to locally (and fully independently) pick a strategy π_i that is winning for φ_i (in a zero-sum sense). It is then guaranteed that any resulting strategy profile $(\pi_i)_{i\in\mathsf{P}}$ is indeed a WSE. This is formalized next.

Definition 3. *Give a k-player game* $(G, (\phi_i)_{i\in\mathsf{P}})$*, a tuple* $(\varphi_i)_{i\in\mathsf{P}}$ *of specifications is said to be a* most general winning secure equilibrium (GWSE) *if it is*

(i) (most) general: $\mathcal{L}(\bigwedge_{i\in\mathsf{P}} \varphi_i) = \mathcal{L}(\bigwedge_{i\in\mathsf{P}} \phi_i)$*;*
(ii) realizable: $v_0 \in \langle\langle i \rangle\rangle \varphi_i$ *for all* $i \in \mathsf{P}$*; and*
(iii) secure (winning): every strategy profile $(\pi_i)_{i\in\mathsf{P}}$ *with* $\pi_i \models \varphi_i$ *is a WSE.*

Intuitively, generality ensures that the transformation of the specifications $(\phi_i)_{i\in\mathsf{P}}$ into new specifications $(\varphi_i)_{i\in\mathsf{P}}$ does not lose any winning play. Further, realizability ensures that every single player can enforce φ_i (without the help of other players) from the initial vertex. Finally, security ensures that any locally chosen strategy π_i winning for φ_i fors a strategy profile which is indeed a WSE.

4 Computing GWSE in ω-regular Games

This section proposes an *iterative semi-algorithm*[3] to compute GWSE in this paper which utilizes the concept of *adequately permissive assumptions* (APA) introduced by Anand et al. [1]. Given a k-player game $(G, (\phi_i)_{i \in P})$, an APA is a specification ψ_i that collects all Player i strategies which allow for a cooperative solution if other players cooperate. It therefore overapproximates the set of all Player i strategies which could possibly form a WSE with the other players. As a consequence, the intersection $\bigwedge_{i \in P} \psi_i$ is an overapproximation of a GWSE. In order to refine this approximation, the next computation round can now use the APA's of other players when computing new local APA's. In order to properly formalize this idea, we first recall the concept of APA's from [1].

4.1 Adequately Permissive Assumptions

Following [1], we define an *adequately permissive assumption* (APA) as follows.

Definition 4. *Given a k-player game graph $G = (V, E, v_0)$ and a specification ϕ, we say that a specification ψ_i is an* adequately permissive assumption (APA) *on Player i for ϕ if it is:*

(i) sufficient*: there exists a strategy profile π_{-i} such that for every Player i strategy π_i with $\pi_i \models \psi_i$, we have $(\pi_i, \pi_{-i}) \models \phi$;*
(ii) implementable*: $\langle\langle i \rangle\rangle \psi_i = V$; and*
(iii) permissive*: $\mathcal{L}(\psi_i) \supseteq \mathcal{L}(\phi)$.*

The intuition behind an APA is that even if a player can not realize a specification ϕ, they should at least satisfy an APA on them as it will allow them to realize ϕ if the other players are willing to help (sufficiency). Further, such a behavior by Player i does not prevent any WSE (permissiveness), and Player i can individually choose to follow an APA (implementability).

Remark 1. While Def. 4 is an almost direct adaptation from [1, Def. 2-5] to k-player games, it has a couple of noteable differences. First, Anand et al. define APA's for 2-player games and, conceptually, use APA's to constraint the opponents moves. While we can simply view the k-player game as a 2-player game between the protagonist Player i and (the collection of) its opponents $\mathtt{P}_{-i}$, we will use the computed assumption ψ_i to constrain the *protagonist's* moves (not the opponent) in Def. 4. Second, the sufficiency condition for an APA in [1, Def. 2] does not depend on an initial vertex. An APA always exists in their setting (possibly being True when $\langle\langle \mathtt{P} \rangle\rangle \phi = \varnothing$). In contrast, the k-player games in this paper have a designated initial vertex, hence, an APA only exists iff $v_0 \in \langle\langle \mathtt{P} \rangle\rangle \phi$.

With this insight, we can use the algorithm from [1] to compute APA's for parity specificatios $\phi = \mathit{Parity}(\Omega)$ in polynomial time.

[3] A semi-algorithm is an algorithm that is not guaranteed to halt on all inputs.

Lemma 1 ([1, Thm. 4]). *Given a k-player game graph $G = (V, E, v_0)$ and a parity specification $\phi = Parity(\Omega)$, an APA on Player i for ϕ can be computed, if one exists, in time $\mathcal{O}(|V|^4)$.*

Let us write ComputeAPA(G, ϕ, i) to denote the procedure that returns this APA if it exists; otherwise, it returns False.

Remark 2. We note that *Lem.* 1 also gives a method to compute APA's for games with LTL- or ω-regular specifications as such games can be converted into parity games (possibly with an extended game graph) by standard methods [3]. Therefore, with a slight abuse of notation, we will also call the algorithm ComputeAPA(G, ϕ, i) if ϕ is not a parity specification, which the understanding, that the game is always converted into a parity game first. This might incur an exponential blowup of the state space. As we call ComputeAPA repeatedly to compute GWSE's, this blowup might cause non-termination (see Sec. 4.6 for details). In order to obtain a (non-optimal but) terminating algorithm for GWSE computation, we will mitigate this blowup later in Sec. 5.

4.2 Iterative Computation of APA's

Given the results of the previous section, we can use the algorithm ComputeAPA on a given game $(G, (\phi_i)_{i\in P})$ to compute APA's for each player, i.e., $\psi_i :=$ ComputeAPA(G, ϕ_i, i). Intuitively, ψ_i overapproximates the set of all Player i strategies which could possibly form a WSE with the other players. As a consequence, the intersection $\bigwedge_{i\in P} \psi_i$ is an overapproximation of the GWSE.

As outlined previously, we will iteratively refine these computed APA's to finally compute the GWSE. In order to do so, we want to condition the computation of the next-round APA ψ'_i on the previous-round APA's of all other players ψ_{-i}, as any secure strategy of players in P_{-i} is incentivized to comply with ψ_{-i}. The most intuitive method to do this is to simply consider $\psi_{-i} \Rightarrow \phi_i$ as the specification for APA computation in the next round. However, the way sufficiency is formulated for APA's prevents this approach, as the implication $\psi_{-i} \Rightarrow \phi_i$ is true if ψ_{-i} is false. As there obviously exists a strategy profile π_{-i} which violates ψ_{-i}, the sufficiency condition becomes meaningless for this specification.

However, as we know that ψ_{-i} are APA's, their implementability constraint (Def. 4.ii) ensures that Player i can neither enforce nor falsify them. Therefore, a new specification $\phi'_i := \psi_{-i} \wedge \phi_i$ still puts all the burden of satisfying ψ_{-i} to players in P_{-i} and hence, implicitly constrains the choices of P_{-i} to strategies complying with ψ_{-i} for sufficiency of the new APA. However, using $\phi'_i := \psi_{-i} \wedge \phi_i$ indeed weakens the permissiveness requirement $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\phi \wedge \psi_{-i})$, i.e., the new APA ψ'_i needs to be more general than the specification ϕ, only when the assumption ψ_{-i} holds. With these refined conditions for sufficiency and permissiveness, it becomes evident that an APA for specification ϕ under assumption ψ_{-i} is equivalent to an APA for the modified specification $\psi_{-i} \wedge \phi$, as formalized below.

Definition 5. *Given a k-player game, a specification ϕ_i and an assumption ψ_{-i}, we say that the specification ψ_i is an* APA on Player i for ϕ_i under ψ_{-i} *if it is an APA on Player i for specification $\psi_{-i} \wedge \phi$.*

Following Rem. 2, we denote by $\textsc{ComputeAPA}(G, \psi_{-i} \wedge \phi, i)$ the algorithm which computes *APA's* on Player i for ϕ *under assumptions* ψ_{-i}, even though $\psi_{-i} \wedge \phi$ is typically not a parity specification over G anymore.

4.3 Computing GWSE

Using all the intuition discussed before, we now give a semi-algorithm in Algo. 1 to compute GWSE for k-player games with ω-regular specifications for all players. The main idea is to iteratively compute assumptions $(\psi_i)_{i \in \mathtt{P}}$ on every player and check if they are stable enough so that every player can satisfy their actual specification ϕ_i under the assumption ψ_{-i}. If not, then, in the next iteration, we compute new assumptions $(\psi'_i)_{i \in \mathtt{P}}$ that are stricter than earlier ones, i.e., $\mathcal{L}(\psi'_i) \subseteq \mathcal{L}(\psi_i)$ but still more general than their specifications under the earlier assumption, i.e., $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\psi_{-i} \wedge \phi_i)$.

Algorithm 1 $\textsc{ComputeGE}(\mathcal{G})$

Require: A k-player game $\mathcal{G}$ with game graph $G = (V, E, v_0)$ and parity specifications $(\phi_i)_{i \in \mathtt{P}}$.
Ensure: Either a GWSE $(\varphi_i)_{i \in \mathtt{P}}$ or False.
1: $\psi_i \leftarrow \mathsf{True}\ \forall i \in \mathtt{P}$
2: **return** $\textsc{RecursiveGE}(\mathcal{G}, (\psi_i)_{i \in \mathtt{P}})$

3: **procedure** $\textsc{RecursiveGE}(\mathcal{G}, (\psi_i)_{i \in \mathtt{P}})$
4: $\quad\varphi_i \leftarrow \psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)\ \forall i \in \mathtt{P}$
5: $\quad$**if** $v_0 \in \bigcap_{i \in \mathtt{P}} \langle\langle i \rangle\rangle \varphi_i$ **then**
6: $\quad\quad$**return** $(\varphi_i)_{i \in \mathtt{P}}$
7: $\quad\psi'_i \leftarrow \psi_i \wedge \textsc{ComputeAPA}(G, \psi_{-i} \wedge \phi_i, i)\ \forall i \in \mathtt{P}$
8: $\quad$**if** $\psi'_i = \psi_i$ for all $i \in \mathtt{P}$ **then**
9: $\quad\quad$**return** False
10: $\quad$**return** $\textsc{RecursiveGE}(\mathcal{G}, (\psi'_i)_{i \in \mathtt{P}})$

More specifically, we start with $\psi_i = \mathsf{True}$ for each $i \in \mathtt{P}$ in the first iteration (line 1), and then in every iteration, we want each player to satisfy $\varphi_i = \psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)$ (computed in line 4) *by themselves*, i.e., always satisfy their assumption ψ_i and satisfy specification ϕ_i whenever others satisfy their assumptions ψ_{-i}. Note that, in this part of the algorithm it is correct to use this implication-style specification, as it is used for solving a *zero-sum* 2*-player game* between Player i and its opponent (i.e., the collection of all other players in $\mathtt{P}_{-i}$) for the specification φ_i. The winning regions $\langle\langle i \rangle\rangle \varphi_i$ for each such zero-sum 2-player game are then intersected in line 5 to obtain the winning region that is achievable by any strategy profile $(\pi_i)_{i \in \mathtt{P}}$ where π_i is a winning strategy of Player i w.r.t. φ_i (in a zero-sum sense). If this resulting winning region contains the initial vertex, we return the specification $(\varphi_i)_{i \in \mathtt{P}}$ (line 6), which is proven to indeed be a GWSE in Thm. 1.

If this is not the case, we keep on strengthening APA's, as discussed in Sec. 4.2, to make the above mentioned zero-sum 2-player games easier to solve (as they can rely on tighter assumptions now). Hence, we call ComputeAPA with the modified specifications $\phi_i' := \psi_{-i} \wedge \phi_i$ for all players (line 7). If this assumption refinement step was unsuccessful, i.e., assumptions have not changed (line 8), we give up and return False. Otherwise, we recheck the termination condition for the newly computed APA's.

Example 1. Before proving the correctness of the (semi) Algo. 1, let us first illustrate the steps using an example depicted in Fig. 2. In line 1, we begin with $\psi_1 = \psi_2 = \mathsf{True}$ and run the recursive procedure RecursiveGE in line 2.

Within the first iteration of RecursiveGE, in line 4, we set $\varphi_i = \phi_i$ as $\psi_i = \mathsf{True}$ for all $i \in [1;2]$. Then, in line 5, we check whether each player can satisfy $\varphi_i = \phi_i$ without cooperation (i.e., in a zero-sum sense), from the initial vertex v_0. As no player can ensure that, we move to line 7. Here, as $\psi_i = \mathsf{True}$ for $i \in [1;2]$, the new assumptions ψ_i' is an APA computed by ComputeAPA(G, ϕ_i, i). This gives us $\psi_1' = \Box\neg(e_{12} \wedge e_{34}) \wedge \Diamond\Box\neg e_{10}$ and $\psi_2' = \Diamond\Box\neg e_{00}$, where $e_{ij} = v_i \wedge \bigcirc v_j$. Intuitively, ψ_1' ensures that edges, i.e., $v_1 \to v_2$ and $v_3 \to v_4$, leading to the region from which it is not possible to satisfy ϕ_1 are never taken; and the edge, i.e., $v_1 \to v_0$, restricting the play to progress towards target vertex v_5 (as in ϕ_1) is eventually not taken. Similarly, ψ_2 ensures that the edge $v_0 \to v_0$ is eventually not taken that ensures progress towards ϕ_2's target vertices $\{v_4, v_5\}$. As $\psi_i' \neq \psi_i$ for all $i \in [1;2]$ in line 8, we go to the next iteration of RecursiveGE.

In the second iteration, we again compute the new potential GWSE (φ_1, φ_2) with $\varphi_i = \psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)$ in line 4. In line 5, we find that $v_0 \notin \langle\langle 1 \rangle\rangle \varphi_1$. That is because Player 1 cannot ensure satisfying ϕ_1 even when Player 2 satisfies ψ_2 as Player 2 can always use edge $v_0 \to v_3$ leading to the play $(v_0 v_3)^\omega \not\models \phi_2$. Hence, in line 7, the APA under ψ_1 gives a more restricted assumptions on Player 2: $\psi_2' = \Diamond\Box\neg(e_{00} \wedge e_{03})$. As the assumption ψ_2 on Player 2 was very weak, the APA for Player 1 under ψ_2 results in the same assumption as ψ_1, and hence, $\psi_1' = \psi_1$. Then, we move to the third iteration.

In this iteration, we find that both players can indeed satisfy their new specification φ_i from the initial vertex in line 5. Hence, we finally return a GWSE (φ_1, φ_2) with $\varphi_i = \psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)$ where $\psi_2 = \Diamond\Box\neg(e_{00} \wedge e_{03})$ and $\psi_1 = \Box\neg(e_{12} \wedge e_{34}) \wedge \Diamond\Box\neg(e_{10})$.

Remark 3. Let us remark that for the game depicted in Fig. 2, assume-admissible (AA) synthesis [5] has no solution. AA-synthesis utilizes a different, incomparable definition of rationality based on a dominance order. In their framework, a

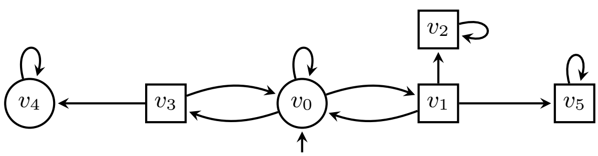

Figure 2: A two-player game with initial vertex v_0, Player 1's vertices (squares), Player 2's vertices (circles) and specifications $\phi_1 = \Diamond\Box\{v_5\}$ and $\phi_2 = \Diamond\Box\{v_4, v_5\}$.

Player i strategy π_i is said to be *dominated* by π'_i if the set of strategy profiles that π' is winning against (i.e., satisfies Player i's specification) is strictly larger than that of π. A strategy not dominated by any other strategy is called *admissible.* In AA-synthesis, one needs to find an admissible strategy π_i for Player i such that for every admissible strategy π'_{-i} for the other player, $(\pi_i, \pi'_{-i}) \models \phi_i$. In this example, Player 1 has only one admissible strategy π_1 that always uses $v_1 \to v_5$ and $v_3 \to v_0$. However, with the admissible strategy π'_2 of Player 2 that always uses $v_0 \to v_3$, we have $(\pi_1, \pi'_2) \not\models \phi_1$.

The next theorem shows that Algo. 1 is indeed sound.

Theorem 1. *Let $\mathcal{G}$ be a k-player game with game graph $G = (V, E, v_0)$ and parity specifications $(\phi_i)_{i\in\mathsf{P}}$ such that $(\varphi_i^*)_{i\in\mathsf{P}} = \textsc{Compute}GE(\mathcal{G})$, then $(\varphi_i^*)_{i\in\mathsf{P}}$ is a GWSE for $\mathcal{G}$.*

Proof. First, observe that ComputeGE did not return False by the premise of the theorem. So, if ComputeAPA returned False in line 7, i.e., $\psi'_i = \mathsf{False}$ for some $i \in \mathsf{P}$, in some n-th iteration, then in the $n+1$-th iteration, we have $\psi_i = \mathsf{False}$ and $\psi_{-j} = \mathsf{False}$ for all $j \in \mathsf{P}_{-i}$. So, it holds that $v_0 \notin \langle\langle i \rangle\rangle \varphi_i = \langle\langle i \rangle\rangle \mathsf{False} = \varnothing$ and hence, it does not return in line 6. Furthermore, as $\psi_{-j} \wedge \phi_j = \mathsf{False}$ for all $j \in \mathsf{P}_{-i}$, by sufficiency, ComputeAPA returns False for all $j \in \mathsf{P}_{-i}$. Hence, $\psi'_j = \mathsf{False}$ for all $j \in \mathsf{P}$. This would imply (by similar arguments), in $(n+2)$-th iteration, $\psi'_j = \psi_j = \mathsf{False}$ for all $j \in \mathsf{P}$ and hence, the algorithm would return False. Therefore, we can assume ComputeAPA never returned False in any iteration.

Now, let us claim that in every iteration of RecursiveGE, for all $i \in \mathsf{P}$:

(claim 1) $\mathcal{L}(\psi_i) \supseteq \mathcal{L}(\bigwedge_{j\in\mathsf{P}} \phi_j)$, and (claim 2) $\mathcal{L}(\psi_i) \supseteq \mathcal{L}(\psi_{-i} \wedge \phi_i)$.

We will prove the claim using induction on the number of itereative calls to RecursiveGE. For the base case, observe $\psi_i = \mathsf{True}$ for all $i \in \mathsf{P}$, hence, the claim holds trivially. For the induction step, assume that claim 1+2 hold in the n-th iteration. Then, for all $i \in \mathsf{P}$, as ψ'_i (computed in line 7) is ψ in the next iteration, it suffices to show that $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\bigwedge_{j\in\mathsf{P}} \phi_j)$ and $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\psi'_{-i} \wedge \phi_i)$.

By permissiveness of APA (as in Def. 4), for all $i \in \mathsf{P}$, we have $\mathcal{L}(\textsc{ComputeAPA}(G, \psi_{-i} \wedge \phi_i, i)) \supseteq \mathcal{L}(\psi_{-i}) \cap \mathcal{L}(\phi_i)$. Hence, by line 7, for all $i \in \mathsf{P}$, we have $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\psi_i) \cap \mathcal{L}(\psi_{-i}) \cap \mathcal{L}(\phi_i) = \left(\bigcap_{j\in\mathsf{P}} \mathcal{L}(\psi_j)\right) \cap \mathcal{L}(\phi_i)$, and hence, by claim 1, $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\bigwedge_{j\in\mathsf{P}} \phi_j)$.

Similarly, for all $i \in \mathsf{P}$, as $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\psi_i) \cap \mathcal{L}(\psi_{-i}) \cap \mathcal{L}(\phi_i)$, by claim 2, we also have $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\psi_{-i}) \cap \mathcal{L}(\phi_i)$. Furthermore, by line 7, for all $j \in \mathsf{P}$, we have $\mathcal{L}(\psi_j) \supseteq \mathcal{L}(\psi'_j)$, and hence, $\mathcal{L}(\psi_{-i}) = \bigcap_{j\neq i} \mathcal{L}(\psi_j) \supseteq \bigcap_{j\neq i} \mathcal{L}(\psi'_j) = \mathcal{L}(\psi'_{-i})$. Therefore, for all $i \in \mathsf{P}$, we have $\mathcal{L}(\psi'_i) \supseteq \mathcal{L}(\psi'_{-i}) \cap \mathcal{L}(\phi_i) = \mathcal{L}(\psi'_{-i} \wedge \phi_i)$.

Now, we show that Def. 3 (i)-(ii) indeed holds for the tuple $(\varphi_i^*)_{i\in\mathsf{P}}$.
(i) (general) By construction, $\varphi_i^* = \psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)$ for the specifications $(\psi_i)_{i\in\mathsf{P}}$ computed in last iteration. Hence, it holds that

$$\mathcal{L}\left(\bigwedge_{i\in\mathsf{P}} \varphi_i^*\right) = \bigcap_{i\in\mathsf{P}} \mathcal{L}(\psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)) = \bigcap_{i\in\mathsf{P}} \mathcal{L}(\psi_i) \cap \bigcap_{i\in\mathsf{P}} \mathcal{L}(\psi_{-i} \Rightarrow \phi_i)$$

$$= \bigcap_{i\in\mathsf{P}} \mathcal{L}(\psi_{-i}) \cap \bigcap_{i\in\mathsf{P}} \mathcal{L}(\psi_{-i} \Rightarrow \phi_i) = \bigcap_{i\in\mathsf{P}} \mathcal{L}(\psi_{-i} \wedge (\psi_{-i} \Rightarrow \phi_i)) \subseteq \bigcap_{i\in\mathsf{P}} \mathcal{L}(\phi_i) = \mathcal{L}\left(\bigwedge_{i\in\mathsf{P}} \phi_i\right).$$

For the other direction, it holds that

$$\mathcal{L}(\varphi_i^*) = \mathcal{L}(\psi_i) \wedge \mathcal{L}(\psi_{-i} \Rightarrow \phi_i) \supseteq \mathcal{L}(\psi_i) \cap \mathcal{L}(\phi_i) \tag{1}$$

Then, by claim 1, for all $i \in \mathsf{P}$, we have $\mathcal{L}(\varphi_i^*) \supseteq \mathcal{L}\left(\bigwedge_{i\in\mathsf{P}} \phi_i\right)$, and hence, $\mathcal{L}\left(\bigwedge_{i\in\mathsf{P}} \varphi_i^*\right) \supseteq \mathcal{L}\left(\bigwedge_{i\in\mathsf{P}} \phi_i\right)$. Therefore, $(\varphi_i^*)_{i\in\mathsf{P}}$ is general.
(ii) (realizable) Holds trivially by line 5.
(iii) (secure) Let $(\pi_i)_{i\in\mathsf{P}}$ be a strategy profile with $\pi_i \models \varphi_i^*$. Then, every $(\pi_i)_{i\in\mathsf{P}}$-play from v_0 satisfies φ_i^* for all $i \in \mathsf{P}$, and hence, $(\pi_i)_{i\in\mathsf{P}} \models \bigwedge_{i\in\mathsf{P}} \varphi_i^*$. So, by generality, we have $(\pi_i)_{i\in\mathsf{P}} \models \bigwedge_{i\in\mathsf{P}} \phi_i$.

Now, to prove item ii of Def. 2, let π_i' be a strategy of Player i, and let ρ be the (π_i', π_{-i})-play from v_0. As before, for all $j \in \mathsf{P}$, we have $\varphi_j^* = \psi_j \wedge (\psi_{-j} \Rightarrow \phi_j)$. So, for every $j \neq i$, $\rho \in \mathcal{L}(\varphi_j^*) \subseteq \mathcal{L}(\psi_j)$. Hence, we have $\rho \in \bigcap_{j\neq i} \mathcal{L}(\psi_j) = \mathcal{L}(\psi_{-i})$.

Now, if $\rho \in \mathcal{L}(\phi_i)$, then $\rho \in \mathcal{L}(\psi_{-i} \wedge \phi_i)$. Then, by (1) and claim 2, we have $\rho \in \mathcal{L}(\varphi_i^*)$. Furthermore, as $\pi_{-i} \models \varphi_{-i}^*$, we have $\rho \in \mathcal{L}(\varphi_{-i}^*)$. Therefore, $\rho \in \mathcal{L}(\varphi_i^* \wedge \varphi_{-i}^*)$, and by generality, $\rho \in \mathcal{L}(\phi_i \wedge \phi_{-i}) \subseteq \mathcal{L}(\phi_{-i})$. Then, by contraposition, item ii of Def. 2 holds for $(\pi_i)_{i\in\mathsf{P}}$. Hence, $(\pi_i)_{i\in\mathsf{P}}$ is an SE, and hence, $(\varphi_i^*)_{i\in\mathsf{P}}$ is secure. □

4.4 Games with an Environment Player

Up to this point, we have only considered games played between k players, each representing a distinct system. However, in the context of reactive synthesis problems, a different setup is often encountered. Here, the system players play against an environment player, who is considered as being adversarial toward all the system players. Consequently, the system players must fulfill their objectives against all possible strategies employed by the environment player.

Interestingly, this framework can be seen as equivalent to a $(k+1)$-player game with the original k system players and a $(k+1)$-th player, representing the environment. For this new player, the objective is simply $\phi_{k+1} = \mathsf{True}$. Then, it is easy to see that an APA for such specification ϕ_{k+1} under any assumption is True. Hence, in each iteration of RecursiveGE in Algo. 1, the associated assumption ψ_{k+1} is also True, and thus, $\varphi_{k+1} = \mathsf{True} \wedge ((\bigwedge_{i\in[1;k]} \psi_i) \Rightarrow \mathsf{True}) \equiv \mathsf{True}$. Consequently, if ComputeGE yields a GWSE $(\varphi_i^*)_{i\in[1;k+1]}$, the new objective of the environment player, $\varphi_{k+1}^* = \mathsf{True}$, doesn't impose any constraints on the environment's actions. Therefore, the tuple $(\varphi_i^*)_{i\in[1;k]}$ remains secure (as in Def. 3) for the k system players because the environment player can never violate its new specification φ_{k+1}. In sum, games featuring an environment player can be effectively handled as a special case, as formally summarized below:

Corollary 1. *Let $G = (V, E)$ be a game graph with k system players, i.e., $\mathsf{P} = [1; k]$, and an environment player* $\mathtt{env}$ *such that $V = (\bigcup_{i\in\mathsf{P}} V_i) \uplus V_{\mathtt{env}}$. Let $(\phi_i)_{i\in\mathsf{P}}$ be the tuple of specifications, one for each system player. Then, a tuple $(\varphi_i)_{i\in\mathsf{P}}$ is a GWSE for $(G, (\phi_i)_{i\in\mathsf{P}})$ if and only if $(\varphi_i)_{i\in[1;k+1]}$ with $\varphi_{k+1} = \mathsf{True}$ is a GWSE for the $k+1$-player game $(G, (\phi_i)_{i\in[1;k+1]})$ with $\phi_{k+1} = \mathsf{True}$.*

Furthermore, in synthesis problems, the choices of the environment are sometimes restricted based on a certain assumption $\phi_{\mathtt{env}}$. In such scenarios, a viable approach involves updating each system player's specification ϕ_i to $\phi_{\texttt{env}} \Rightarrow \phi_i$ and subsequently utilizing Cor. 1 to compute a GWSE. An alternative approach is to consider a $(k+1)$-player game with specification $\phi_{k+1} = \phi_{\texttt{env}}$ for the $(k+1)$-th player. With this approach, the solution becomes more meaningful, as any strategy profile for the system players satisfying the resulting GWSE allows the environment to satisfy its own assumptions $\phi_{\texttt{env}}$. This approach nicely complements existing works [14,25] that aim to synthesize strategies for systems while allowing the environment to fulfill its own requirement.

4.5 Partially Winning GWSE

In the preceding sections, we have presented a method for computing *winning* SE, i.e., equilibria where all players satisfy their objectives. However, it's worth noting that in certain scenarios, WSE might not exist (see e.g. [13] for a detailed discussion). In such cases, a subset $\mathtt{P}'$ of players can still form a coalition, which serves their interests by enabling them to compute a GWSE for their coalition only, while treating the remaining players in $\mathtt{P} \setminus \mathtt{P}'$ as part of the environment. This can be accomplished by computing a GWSE with updated specifications denoted as $(\phi'_i)_{i \in \mathtt{P}}$, wherein $\phi'_i = \phi_i$ for all $i \in \mathtt{P}'$ and $\phi'_i = \mathsf{True}$ for all $i \notin \mathtt{P}'$. This scenario aligns with the concept of considering an environment from Sec. 4.4.

It is important to emphasize that for instances where no WSE exists, there might not even exist a *unique maximal* outcome for which an SE is feasible, see [13, Sec. 5] for a simple example. As a result, there may be multiple coalitions that can offer different advantages to individual players from the initial vertex. This scenario presents an intriguing, unexplored challenge for future research.

4.6 Computational Tractability and Termination

While Algo. 1 has multiple desirable properties, additionally supported by the possible extensions discussed in Sec. 4.4 and 4.5, its computational tractability and termination is questionable for the full class of ω-regular games.

As pointed out in Rem. 2, the application of ComputeAPA might require changing the game graph for if the input is not a parity specification. While the *language* of the computed APA is guarantee to shrink in every iteration (see the proof of Thm. 1), this does not guarantee termination of Algo. 1 as such a language still contains an infinite number of words. Due to the possibly repeated changes in the game graph for APA computation, the finiteness of the underlying model can also not be used as a termation argument.

In addition, the need to change game graphs induces a severe computational burden. While this might be not so obvious for the polynomial time algorithm ComputeAPA, this is actually also the case for the (zero-sum) game solver that needs to be invoked line 5 of Algo. 1. As the specification for these games also keeps changing in each iteration, a new parity game needs to be constructed in each iteration, which might be increasingly harder to solve, depending on the nature of the added assumptions. We will see in Sec. 5 how these problems can be resolved by a suitable restriction of the considered assumption class.

5 Optimized Computation of GWSE in Parity Games

As discussed in Sec. 4.6, the potential need to repeatedly change game graphs in the computations of lines 5 and 7 in Algo. 1 might incur increasing computational costs and prevents a termination guarantee. To circumvent these problems, this section proposes a different algorithm for GWSE synthesis which overapproximates APA's by a simpler assumption class, called UCA's. The resulting algorithm is computationally more tractable and ensured to terminate. Nevertheless, unlike the semi-algorithm discussed in the previous section, this algorithm may not be able to compute a GWSE in all scenarios where the semi-algorithm can.

5.1 From APA's to UCA's

One of the main features of APA's on Player i computed by ComputeAPA from [1], is the fact that they can be expressed by well structured templates using Player i's edges, namely *unsafe-edge-*, *colive-edge-*, and *(conditional)-live-group-templates*. Unsafe- and colive-edge-templates are structurally very simple. Given a set of unsafe edges $S \subseteq E_i$ and colive edges $C \subseteq E_i$ the respective assumption templates $\psi_{\text{UNSAFE}}(S) := \bigwedge_{e \in S} \square \neg e$ and $\psi_{\text{COLIVE}}(C) := \bigwedge_{e \in C} \lozenge \square \neg e$ simply assert that unsafe (resp. colive) edges should never (resp. only finitely often) be taken. We call an assumption which can be expressed by these two types of templates an **U**nsafe- and **C**olive-edge-template **A**ssumption (UCA), as defined next.

Definition 6. *Given a k-player game graph $G = (V, E)$, a specification ψ is called an unsafe- and colive-edge-template assumption (UCA) for Player i, if there exist sets $S, C \subseteq E_i$ s.t. $\psi := \psi_{\text{UNSAFE}}(S) \wedge \psi_{\text{COLIVE}}(C)$. We write $\psi^{[S,C]}$ to denote such assumptions.*

It was recently shown by Schmuck et al. [27] that two-player (zero-sum) parity games under UCA assumptions, i.e., games $(G, \psi \Rightarrow \phi)$ where ψ is an UCA and ϕ is a parity specification over G, can be directly solved over G without computational overhead, compared to the non-augmented version (G, ϕ) of the same game. Interestingly, the synthesis problem under assumptions becomes proveably harder if live-group-templates ψ_{COND} are needed to express an assumption, requiring a change of the game graph in most cases. Conditional-live-group-templates ψ_{COND}, are structurally more challenging than UCA's, as they impose a Streett-type fairness conditions on edges in G (see [1, Sec.4] for details).

Motivated by this result, we will restrict the assumption class used for GWSE computation to UCA's in this section. Unfortunately, UCA's are typically not expressive enough to capture APA's for parity games. This follows from one of the main results of Anand et al., which shows that APA's computed by ComputeAPA for parity games are expressible by a conjunctions of all *three* template types, as re-stated in the following proposition.

Proposition 1 ([1, Thm. 3]). *Given the premisses of Lem. 1, the APA computed by ComputeAPA on Player i can be written as the conjunction $\psi := \psi_{\text{UNSAFE}}(S) \wedge \psi_{\text{COLIVE}}(C) \wedge \psi_{\text{COND}}$ where $S, C \subseteq E_i$.*

We therefore need to overapproximate APA's by UCA's, by simply dropping the ψ_{COND}-term from their defining conjunction, as formalized next.

Definition 7. *Given the premisses of Lem. 1, let* $\psi := \textsc{ComputeAPA}(G, \phi, i) = \psi_{\textsc{unsafe}}(S) \wedge \psi_{\textsc{colive}}(C) \wedge \psi_{\textsc{cond}}$. *Then we denote by* $\textsc{ApproxAPA}(G, \phi, i)$ *the algorithm that computes* $\psi^{[S,C]}$ *by first executing* $\textsc{ComputeAPA}(G, \phi, i)$ *and then dropping all* $\psi_{\textsc{cond}}$*-terms from the resulting APA.*

It is easy to see that $\mathcal{L}(\psi) \subseteq \mathcal{L}(\psi^{[S,C]})$. Therefore, it also follows that $\psi^{[S,C]}$ is implementable and permissive (i.e., Def. 4(ii) and (iii) holds). Unfortunatly, $\psi^{[S,C]}$ is in general no longer sufficient (i.e., Def. 4(i) does not necessarily hold). As the proof of Thm. 1 only uses permissiveness of APA, even though sufficiency is lost for UCA's, replacing ComputeAPA by ApproxAPA in Algo. 1 does not mitigate soundness, i.e., whenever ComputeGE terminates in line 6 with a specification profile $(\varphi_i)_{i \in \mathtt{P}}$, this profile is indeed a GWSE, even if APA's are over-approximated by UCA's. This is formalized next.

Theorem 2. *Let AComputeGE be the algorithm obtained by replacing procedure ComputeAPA by ApproxAPA in Algo. 1. Then, given a k-player game* $\mathcal{G}$ *with parity specifications such that* $(\varphi_i^*)_{i \in \mathtt{P}} = \textsc{AComputeGE}(\mathcal{G})$, *the tuple* $(\varphi_i^*)_{i \in \mathtt{P}}$ *is a GWSE for* $\mathcal{G}$.

The rest of this section will now show how the restriction to UCA's allows to execute lines 5 and 7 in Algo. 1 efficiently and allows to prove termination of the resulting algorithm for GWSE computation.

5.2 Iterative Computation of UCA's

We have seen in the previous section that UCA's can be computed by utilizing ComputeAPA and dropping all ψ_{COND} terms (called ApproxAPA). Of course, this can be done in every iteration of ComputeGE. However, ComputeAPA expects a party game as an input, and from the second iteration of ComputeGE onward the input to ComputeAPA is given by $(G, \psi_{-i} \wedge \phi_i, i)$, where ψ_{-i} is an assumption on players in $\mathtt{P}_{-i}$, which is not necessarily a parity game.

This section therefore provides a new algorithm, called ComputeUCA and given in Algo. 2 which computes UCA's for Player i directly on the game graph G for games $(G, \psi \wedge \phi)$ where $\psi = \psi^{[S,C]}$ is an UCA for $\mathtt{P}_{-i}$ with unsafe edges $S \subseteq E_{-i}$ and colive edges $C \subseteq E_{-i}$, and ϕ is a parity specification, both over G. Intuitively, ComputeUCA first slightly modifies G to a new two-player game graph $\hat{G}$ (lines 1 and 2) s.t. the specification $\psi \wedge \phi$ can be directly expressed as a parity specification $\hat{\phi}$ on $\hat{G}$ (line 4). This allows to apply ApproxAPA to construct and return an UCA for Player 1 on $\hat{G}$ (line 5). As the resulting UCA is for Player i, the unsafe edge and colive edge sets are subsets of E_i. Further, due to the mild modifications from G to $\hat{G}$, the edges of Player i are retained in $\hat{G}$ as E_1, hence, the resulting UCA is a well-defined UCA for Player i in G.

We have the following soundness result for showing equivalence between the UCA's computed by ComputeUCA and ApproxAPA for UCA assumptions, proven in extended version of this paper [26, App. A].

Algorithm 2 COMPUTEUCA$(G, \psi^{[S,C]} \wedge \phi, i)$

Require: A k-player game graph $G = (V, E, v_0)$ and specification $\psi \wedge \phi$ with UCA $\psi = \psi^{[S,C]}$ for $\mathtt{P}_{-i}$, i.e., $S, C \subseteq E_{-i}$, and $\phi = \mathit{Parity}(\Omega)$ s.t. $\Omega : V \to [0; 2d+1]$.
Ensure: An UCA $\psi^{[S',C']}$ for Player i.
1: $\hat{V}_1 \leftarrow V_i$ and $\hat{V}_2 \leftarrow V_{-i} \uplus C$
2: $\hat{E}_1 \leftarrow E_i$ and $\hat{E}_2 \leftarrow E_{-i} \setminus (S \cup C) \cup \{(u,c),(c,v) \mid c = (u,v) \in C\}$
3: $\hat{\Omega} = \begin{cases} \Omega(v) & \text{if } v \in V \\ 2d+1 & \text{otherwise.} \end{cases}$
4: $\hat{G} = (\hat{V}_1 \uplus \hat{V}_2, \hat{E}_1 \uplus \hat{E}_2, v_0)$; $\hat{\phi} \leftarrow \mathit{Parity}(\hat{\Omega})$
5: **return** APPROXAPA$(\hat{G}, \hat{\phi}, 1)$

Proposition 2. *Given game graph $G = (V, E, v_0)$ with parity specification ϕ and an UCA $\psi = \psi^{[S,C]}$ for $\mathtt{P}_{-i}$, let $\psi' :=$ APPROXAPA$(G, \psi \wedge \phi, i)$ and $\psi'' :=$ COMPUTEUCA$(G, \psi \wedge \phi, i)$ then $\mathcal{L}(\psi') = \mathcal{L}(\psi'')$. Furthermore, COMPUTEUCA terminates in time $\mathcal{O}((|V| + |E|)^4)$.*

The proof of this result is given in extended version [26, App. A], and essentially relies on the observation that the parity specification $\hat{\phi}$ in $\hat{G}$ expresses the language $\mathcal{L}(\psi \wedge \phi)$ when restricted to V, i.e, $\mathcal{L}(\hat{G}, \hat{\phi})|_V = \mathcal{L}(G, \phi \wedge \psi)$ and the fact that every UCA for Player 1 in $\hat{G}$ is also an UCA for Player i in G.

The usefulness of expressing the computed assumptions as unsafe and colive edge sets S, C over the input game graph G is that there are only a finite number of edges in that graph. Therefore, there obviously also exists only a finite number of unsafe or colive edge sets, which could all be enumerated in the worst case. Therefore, computing UCA's on the same game graph in every iteration, will ensure termination of the overall computation of GWSE.

5.3 Solving Parity Games under UCA's

As the final step towards an optimized version of Algo. 1, we now address the computations required in line 5 of Algo. 1. Observe that this line requires to check $v_0 \in \bigcap_{i \in \mathtt{P}} \langle\langle i \rangle\rangle \varphi_i$ for $\varphi_i = \psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)$. If this check returns True the algorithm terminates, if it returns False new assumptions are computed. In both cases, the game graph used to check this conditional will not have any effect on the future behavior of the algorithm.

Nevertheless, we utilize the recent result by Schmuck et al. [27] to compute $\langle\langle i \rangle\rangle \varphi_i$ more efficiently if ψ_i and ψ_{-i} are UCA's on Player i and $\mathtt{P}_{-i}$, respectively. The construction uses the same idea as presented in Algo. 2 to encode UCA's into a new, slightly modified two-player parity game $(\hat{G}, \hat{\phi})$ which can then be solved by a standard parity solver, such as Zielonka's algorithm [30], which return the winning region $\mathcal{W}$ of Player 1 in this new game that corresponds to the winning region of Player i in G. The resulting algorithm is called COMPUTEWIN given in the extended version [26, Algo. 3] and has the property that $v_0 \in \langle\langle i \rangle\rangle (G, \varphi)$ if and only if $v_0 \in \mathcal{W}$. This is formalized and proven in the extended version [26, Prop. 3].

5.4 Computation of GWSE via UCA's

With the previously discussed algorithms in place, we are now in the position to propose an optimized, surely terminating algorithm to compute GWSE, called OComputeGE. Within ComputeGE the recursive procedure RecursiveGE is replaced by one which uses the algorithms ComputeUCA and ComputeWin for UCA's from Sec. 5.2 and 5.3, as follows

procedure RecursiveGE($\mathcal{G}, (\psi_i)_{i \in \mathtt{P}}$)
  $\varphi_i \leftarrow \psi_i \wedge (\psi_{-i} \Rightarrow \phi_i)\ \forall i \in \mathtt{P}$
  $\mathcal{W}_i \leftarrow$ ComputeWin($\mathcal{G}, \varphi_i, i$)
  if $v_0 \in \bigcap_{i \in \mathtt{P}} \mathcal{W}_i$ **then**
    return $(\varphi_i)_{i \in \mathtt{P}}$
  $\psi'_i \leftarrow \psi_i \wedge$ ComputeUCA($G, \psi_{-i} \wedge \phi_i, i$) $\forall i \in \mathtt{P}$
  if $\psi'_i = \psi_i$ for all $i \in \mathtt{P}$ **then**
    return False
  return RecursiveGE($\mathcal{G}, (\psi'_i)_{i \in \mathtt{P}}$)

We have the following main result of this section.

Theorem 3. *Let $\mathcal{G}$ be a k-player game with game graph $G = (V, E, v_0)$ and parity specifications $(\phi_i)_{i \in \mathtt{P}}$ such that $(\varphi^*_i)_{i \in \mathtt{P}} = OComputeGE(\mathcal{G})$, then $(\varphi^*_i)_{i \in \mathtt{P}}$ is a GWSE for $\mathcal{G}$. Moreover, OComputeGE terminates in time $\mathcal{O}(k^2\,|E| \cdot (2\,|V| + 2\,|E|)^{d+2})$, where d is the number of priorities used in the parity specifications.*

Proof. Combining results from Thm. 1 with Thm. 2 and Prop. 2 gives us that $(\varphi^*_i)_{i \in \mathtt{P}}$ is indeed a GWSE for $\mathcal{G}$. Furthermore, as ψ_i (for all $i \in \mathtt{P}$) in each iteration of the algorithm either remains the same or add more unsafe/colive edges, it can only change $2\,|E|$ times. Hence, as there are k players, the algorithm OComputeGE will terminate within $2k\,|E|$ iterations. Moreover, each iteration involves k calls to both ComputeWin and ComputeUCA. Using Zielonka's algorithm[4] [30] for solving parity games, each iteration will take $\mathcal{O}((2\,|V| + 2\,|E|)^{d+2})$ time for d priorities (by Prop. 2). In total, this gives us that OComputeGE terminates in time $\mathcal{O}(k^2\,|E| \cdot (2\,|V| + 2\,|E|)^{d+2})$. □

Remark 4. As Anand et al. show that APA's for games with co-Büchi specifications (i.e., $\phi = \Diamond \Box T$ for some $T \subseteq V$) are always expressible by UCA's [1, Thm. 3], we note that ComputeAPA and ApproxAPA coincide for such games. This implies that no over approximation of assumptions is needed in this case an the optimizations discussed for ComputeUCA and ComputeWin can be directly applied for APA's.

We further note that OComputeGE also efficiently computes GWSE for games with more expressive specifications than co-Büchi . For instance, all games discussed in this paper as well as the mutual exclusion protocol discussed in [15] can be solved by OComputeGE.

[4] We note that the time complexity is exponential as we use Zielonka's algorithm [30] to solve parity games. One can also use a quasi-polynomial algorithm [11] for solving parity games to get a quasi-polynomial time complexity for OComputeGE.

References

1. Anand, A., Mallik, K., Nayak, S.P., Schmuck, A.: Computing adequately permissive assumptions for synthesis. In: Sankaranarayanan, S., Sharygina, N. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13994, pp. 211–228. Springer (2023). https://doi.org/10.1007/978-3-031-30820-8_15, https://doi.org/10.1007/978-3-031-30820-8_15
2. Anand, A., Nayak, S.P., Schmuck, A.: Contract-based distributed synthesis in two-objective parity games. CoRR **abs/2307.06212** (2023). https://doi.org/10.48550/ARXIV.2307.06212, https://doi.org/10.48550/arXiv.2307.06212
3. Baier, C., Katoen, J.P.: Principles of model checking. MIT press (2008)
4. Bloem, R., Chatterjee, K., Jacobs, S., Könighofer, R.: Assume-guarantee synthesis for concurrent reactive programs with partial information. In: Baier, C., Tinelli, C. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9035, pp. 517–532. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_50, https://doi.org/10.1007/978-3-662-46681-0_50
5. Brenguier, R., Raskin, J., Sankur, O.: Assume-admissible synthesis. Acta Informatica **54**(1), 41–83 (2017). https://doi.org/10.1007/s00236-016-0273-2, https://doi.org/10.1007/s00236-016-0273-2
6. Brice, L., Raskin, J., van den Bogaard, M.: Subgame-perfect equilibria in mean-payoff 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. 8:1–8:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.CONCUR.2021.8, https://doi.org/10.4230/LIPIcs.CONCUR.2021.8
7. Brice, L., Raskin, J., van den Bogaard, M.: Rational verification for nash and subgame-perfect equilibria in graph games. In: Leroux, J., Lombardy, S., Peleg, D. (eds.) 48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France. LIPIcs, vol. 272, pp. 26:1–26:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.MFCS.2023.26, https://doi.org/10.4230/LIPIcs.MFCS.2023.26
8. Brihaye, T., Bruyère, V., De Pril, J.: On equilibria in quantitative games with reachability/safety objectives. Theory Comput. Syst. **54**(2), 150–189 (2014). https://doi.org/10.1007/s00224-013-9495-7, https://doi.org/10.1007/s00224-013-9495-7
9. Bruyère, V., Meunier, N., Raskin, J.: Secure equilibria in weighted games. In: Henzinger, T.A., Miller, D. (eds.) Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014. pp. 26:1–26:26. ACM (2014). https://doi.org/10.1145/2603088.2603109, https://doi.org/10.1145/2603088.2603109
10. Bruyère, V., Roux, S.L., Pauly, A., Raskin, J.: On the existence of weak subgame perfect equilibria. Inf. Comput. **276**, 104553 (2021). https://doi.org/10.1016/j.ic.2020.104553, https://doi.org/10.1016/j.ic.2020.104553

11. Calude, C.S., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: Hatami, H., McKenzie, P., King, V. (eds.) Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017. pp. 252–263. ACM (2017). https://doi.org/10.1145/3055399.3055409, https://doi.org/10.1145/3055399.3055409
12. Chatterjee, K., Doyen, L., Filiot, E., Raskin, J.: Doomsday equilibria for omega-regular games. Inf. Comput. **254**, 296–315 (2017). https://doi.org/10.1016/j.ic.2016.10.012, https://doi.org/10.1016/j.ic.2016.10.012
13. Chatterjee, K., Henzinger, T.A., Jurdzinski, M.: Games with secure equilibria. Theor. Comput. Sci. **365**(1-2), 67–82 (2006). https://doi.org/10.1016/j.tcs.2006.07.032, https://doi.org/10.1016/j.tcs.2006.07.032
14. Chatterjee, K., Horn, F., Löding, C.: Obliging games. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6269, pp. 284–296. Springer (2010). https://doi.org/10.1007/978-3-642-15375-4_20, https://doi.org/10.1007/978-3-642-15375-4_20
15. Chatterjee, K., Raman, V.: Assume-guarantee synthesis for digital contract signing. Formal Aspects Comput. **26**(4), 825–859 (2014). https://doi.org/10.1007/s00165-013-0283-6, https://doi.org/10.1007/s00165-013-0283-6
16. Damm, W., Finkbeiner, B.: Automatic compositional synthesis of distributed systems. In: Jones, C.B., Pihlajasaari, P., Sun, J. (eds.) FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8442, pp. 179–193. Springer (2014). https://doi.org/10.1007/978-3-319-06410-9_13, https://doi.org/10.1007/978-3-319-06410-9_13
17. De Pril, J., Flesch, J., Kuipers, J., Schoenmakers, G., Vrieze, K.: Existence of secure equilibrium in multi-player games with perfect information. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II. Lecture Notes in Computer Science, vol. 8635, pp. 213–225. Springer (2014). https://doi.org/10.1007/978-3-662-44465-8_19, https://doi.org/10.1007/978-3-662-44465-8_19
18. Filiot, E., Gentilini, R., Raskin, J.: Rational synthesis under imperfect information. 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. 422–431. ACM (2018). https://doi.org/10.1145/3209108.3209164, https://doi.org/10.1145/3209108.3209164
19. Finkbeiner, B., Passing, N.: Compositional synthesis of modular systems. Innov. Syst. Softw. Eng. **18**(3), 455–469 (2022). https://doi.org/10.1007/s11334-022-00450-w, https://doi.org/10.1007/s11334-022-00450-w
20. Fisman, D., Kupferman, O., Lustig, Y.: Rational synthesis. In: Esparza, J., Majumdar, R. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6015, pp. 190–204. Springer (2010). https://doi.org/10.1007/978-3-642-12002-2_16, https://doi.org/10.1007/978-3-642-12002-2_16

21. Kremer, S., Raskin, J.: A game-based verification of non-repudiation and fair exchange protocols. J. Comput. Secur. **11**(3), 399–430 (2003). https://doi.org/10.3233/jcs-2003-11307, https://doi.org/10.3233/jcs-2003-11307
22. Kupferman, O., Shenwald, N.: The complexity of LTL rational synthesis. In: Fisman, D., Rosu, G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13243, pp. 25–45. Springer (2022). https://doi.org/10.1007/978-3-030-99524-9_2, https://doi.org/10.1007/978-3-030-99524-9_2
23. Li, X., Li, X., Xu, G., Hu, J., Feng, Z.: Formal analysis of fairness for optimistic multiparty contract signing protocol. J. Appl. Math. **2014**, 983204:1–983204:10 (2014). https://doi.org/10.1155/2014/983204, https://doi.org/10.1155/2014/983204
24. Majumdar, R., Mallik, K., Schmuck, A., Zufferey, D.: Assume-guarantee distributed synthesis. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **39**(11), 3215–3226 (2020). https://doi.org/10.1109/TCAD.2020.3012641, https://doi.org/10.1109/TCAD.2020.3012641
25. Majumdar, R., Piterman, N., Schmuck, A.: Environmentally-friendly GR(1) synthesis. In: Vojnar, T., Zhang, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11428, pp. 229–246. Springer (2019). https://doi.org/10.1007/978-3-030-17465-1_13, https://doi.org/10.1007/978-3-030-17465-1_13
26. Nayak, S.P., Schmuck, A.K.: Most general winning secure equilibria synthesis in graph games. CoRR **abs/2401.09957** (2024). https://doi.org/10.48550/arXiv.2401.09957, https://doi.org/10.48550/arXiv.2401.09957
27. Schmuck, A.K., Thejaswini, K.S., Sağlam, I., Nayak, S.P.: Solving two-player games under progress assumptions. In: Dimitrova, R., Lahav, O., Wolff, S. (eds.) Verification, Model Checking, and Abstract Interpretation. pp. 208–231. Springer Nature Switzerland, Cham (2024)
28. Steg, J.: On identifying subgame-perfect equilibrium outcomes for timing games. Games Econ. Behav. **135**, 74–78 (2022). https://doi.org/10.1016/j.geb.2022.05.012, https://doi.org/10.1016/j.geb.2022.05.012
29. Ummels, M.: Rational behaviour and strategy construction in infinite multiplayer games. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4337, pp. 212–223. Springer (2006). https://doi.org/10.1007/11944836_21, https://doi.org/10.1007/11944836_21
30. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. **200**(1-2), 135–183 (1998). https://doi.org/10.1016/S0304-3975(98)00009-7, https://doi.org/10.1016/S0304-3975(98)00009-7