Assume-Guarantee Synthesis*

Krishnendu Chatterjee[1] and Thomas A. Henzinger[1,2]

[1] University of California, Berkeley, USA
[2] EPFL, Switzerland
{c_krish,tah}@eecs.berkeley.edu

Abstract. The classical synthesis problem for reactive systems asks, given a proponent process A and an opponent process B, to refine A so that the closed-loop system $A||B$ satisfies a given specification Φ. The solution of this problem requires the computation of a winning strategy for proponent A in a game against opponent B. We define and study the *co-synthesis* problem, where the proponent A consists itself of two independent processes, $A = A_1||A_2$, with specifications Φ_1 and Φ_2, and the goal is to refine both A_1 and A_2 so that $A_1||A_2||B$ satisfies $\Phi_1 \wedge \Phi_2$. For example, if the opponent B is a fair scheduler for the two processes A_1 and A_2, and Φ_i specifies the requirements of mutual exclusion for A_i (e.g., starvation freedom), then the co-synthesis problem asks for the automatic synthesis of a mutual-exclusion protocol.

We show that co-synthesis defined classically, with the processes A_1 and A_2 either collaborating or competing, does not capture desirable solutions. Instead, the proper formulation of co-synthesis is the one where process A_1 competes with A_2 but not at the price of violating Φ_1, and vice versa. We call this *assume-guarantee synthesis* and show that it can be solved by computing secure-equilibrium strategies. In particular, from mutual-exclusion requirements the assume-guarantee synthesis algorithm automatically computes Peterson's protocol.

1 Introduction

The algorithmic *synthesis* (or control) of reactive systems is based on solving 2-player zero-sum games on graphs [11, 12]. Player 1 (representing the system or controller to be synthesized) attempts to satisfy a specification Φ; player 2 (representing the environment or plant) tries to violate the specification. Synthesis is successful if a strategy for player 1 can be found which ensures that Φ is satisfied no matter what player 2 does. These games are *zero-sum*, because the objective of player 2 is $\neg\Phi$, the negation of player 1's objective. In other words, synthesis assumes the worst-case scenario that player 2 is as obstructive as possible.

In many game situations in economics, the two players do not have strictly complementary objectives. Then the appropriate notion of rational behavior is that of a Nash equilibrium. One also encounters *non-zero-sum* situations in

* This research was supported in part by the Swiss National Science Foundation and by the NSF grants CCR-0225610 and CCR-0234690.

computer science applications [10]. In this paper, we demonstrate that non-zero-sum situations arise in the *co-synthesis* problem. In co-synthesis, we are not asked to synthesize a single reactive process, but a system composed of several processes P_i, each with its own specification Φ_i. For instance, the design of a mutual-exclusion protocol is a co-synthesis question: each one of two processes P_1 and P_2 is supposed to satisfy certain requirements, such as mutual exclusion, bounded overtaking, and starvation freedom. In such a situation, the processes are neither collaborating nor are they strictly competitive: they are not collaborating because process P_1 cannot assume that P_2 will help establishing Φ_1; they are not strictly competitive because process P_2 will not obstruct Φ_1 at all costs, but only if doing so does not endanger Φ_2. In other words, the two processes are *conditionally competitive*: process P_1 can assume that P_2 will primarily try to satisfy Φ_2, and only secondarily try to violate Φ_1, and vice versa. This situation can be captured by 2-player games with lexicographic objectives, and Nash equilibria for such lexicographic objectives are called *secure equilibria* [4]. Formally, a pair of strategies for the two players is winning and secure if (1) both players satisfy their objectives by playing the strategies, and (2) if one player deviates from her strategy in order to harm the other player, then the other player can retaliate by violating the first player's objective. We refer to the resulting payoff profile, with both players winning, as a *winning* secure equilibrium.

We formally define the co-synthesis problem, using the automatic synthesis of a mutual-exclusion protocol as a guiding example. More precisely, we wish to synthesize two processes P_1 and P_2 so that the composite system $P_1||P_2||R$, where R is a scheduler that arbitrarily but fairly interleaves the actions of P_1 and P_2, satisfies the requirements of mutual exclusion and starvation freedom for each process. We show that traditional zero-sum game-theoretic formulations, where P_1 and P_2 either collaborate against R, or unconditionally compete, do not lead to acceptable solutions. We then show that for the non-zero-sum game-theoretic formulation, where the two processes compete conditionally, there exists an unique winning secure-equilibrium solution, which corresponds exactly to Peterson's mutual-exclusion protocol. In other words, Peterson's protocol can be synthesized automatically as the winning secure strategies of two players whose objectives are the mutual-exclusion requirements. This is to our knowledge the first application of non-zero-sum games in the synthesis of reactive processes. It is also, to our knowledge, the first application of Nash equilibria —in particular, the special kind called "secure"— in system design.

The new formulation of co-synthesis, with the two processes competing conditionally, is called *assume-guarantee synthesis*, because similar to assume-guarantee verification (e.g., [1]), in attempting to satisfy her specification, each process makes the assumption that the other process does not violate her own specification. The solution of the assume-guarantee synthesis problem can be obtained by computing secure equilibria in 3-player games, with the three players P_1, P_2, and R. Previously, meaningful (i.e., unique maximal) secure equilibria were known to exist only for 2-player games [4], and there it was also shown that in general such meaningful equilibria need not exist for three players. Here we

```
do                                  do
{                                   {
flag[1]:=true; turn:=2;             flag[2]:=true; turn:=1;

| while(flag[1]) nop;               | while(flag[1]) nop;           (C1)
| while(flag[2]) nop;               | while(flag[2]) nop;           (C2)
| while(turn=1) nop;                | while(turn=1) nop;            (C3)
| while(turn=2) nop;                | while(turn=2) nop;            (C4)
| while(flag[1] & turn=2) nop;      | while(flag[1] & turn=2) nop;  (C5)
| while(flag[1] & turn=1) nop;      | while(flag[1] & turn=1) nop;  (C6)
| while(flag[2] & turn=1) nop;      | while(flag[2] & turn=1) nop;  (C7)
| while(flag[2] & turn=2) nop;      | while(flag[2] & turn=2) nop;  (C8)

Cr1:=true; fin_wait; Cr1:=false;    Cr2:=true; fin_wait; Cr2:=false;
flag[1]:=false;                     flag[2]:=false;

wait[1]:=1;                         wait[2]:=1;
while(wait[1]=1)                    while(wait[2]=1)
  | nop;                              | nop;                         (C9)
  | wait[1]:=0;                       | wait[2]:=0;                  (C10)
} while(true)                       } while(true)
```

Fig. 1. Mutual-exclusion protocol synthesis

extend the theoretical results of [4] in two ways, in order to solve the assume-guarantee synthesis problem. First, we prove the existence of meaningful secure equilibria in the special case of 3-player games where the third player can win unconditionally. This special case arises in assume-guarantee synthesis, because the winning condition of the third player (i.e., the scheduler) is *fairness*. Second, we give an algorithm for answering the existence of a winning secure equilibrium (Theorem 2), and for computing the corresponding strategies (Theorem 3). These algorithms extend those of [4] from two to three players.

On large state spaces, assume-guarantee synthesis, like all algorithmic methods, can be impractical. In Section 4, we provide an *abstraction* methodology for assume-guarantee synthesis. We show how a game structure can be abstracted, independently for player 1 and player 2, so that from certain winning strategies on the two abstract games, we can infer winning secure strategies on the concrete game. To our knowledge, this is the first abstraction methodology that works with two independent abstractions of a single game structure. Single-player abstractions suffice for zero-sum games (the abstraction weakens one player and strengthens the other). However, for non-zero-sum games, the *two-abstractions* methodology suggests itself, because each abstraction focuses on the objective of a different player and may thus omit different details. In this way, both abstractions may have smaller state spaces than a combined abstraction would. Specifically, we provide proof rules for inferring winning secure strategies on a concrete 3-player non-zero-sum game from classical winning strategies on two abstract 2-player zero-sum games, for the cases of safety and Büchi objectives. In fact, in the safety case, our proof rule corresponds closely to the assume-guarantee rule of [1]. In the Büchi case, our rule provides a novel assume-guarantee rule for the verification of specifications under weak fairness.

Related work. We use non-zero-sum games in a perfect-information setting to restrict the power of an adversary in the synthesis of reactive systems. Another way to restrict the power of the adversary is to allow the adversary only a partial view of the state space. The resulting class of imperfect-information games [3, 13], and more generally, distributed games [8, 9], have been studied extensively in the literature, but only with zero-sum (strictly competitive) objectives. The computational complexity of imperfect-information games is typically much higher than of the perfect-information analogues, and several problems become undecidable in the distributed setting. As illustrated with the mutual-exclusion example, we believe that non-zero-sum games have their place in system synthesis, for synthesizing components with different specifications. They restrict the behaviors of the players in a natural way, by focusing on non-zero-sum objectives, without the exponential (or worse) cost of limiting information.

2 Co-synthesis

In this section we define processes, refinement, schedulers, and specifications. We consider the traditional co-operative [5] and strictly competitive [11, 12] versions of the co-synthesis problem; we refer to them as *weak co-synthesis* and *classical co-synthesis*, respectively. We show the drawbacks of these formulations and then present a new formulation of co-synthesis, namely, *assume-guarantee synthesis.*

Variables, valuations, and traces. Let X be a finite set of variables such that each variable $x \in X$ has a finite domain D_x. A *valuation* v on X is a function $v : X \to \bigcup_{x \in X} D_x$ that assigns to each variable $x \in X$ a value $v(x) \in D_x$. We write V for the set of valuations on X. A *trace* on X is an infinite sequence $(v_0, v_1, v_2, \ldots) \in V^\omega$ of valuations on X. Given a valuation $v \in V$ and a subset $Y \subseteq X$ of the variables, we denote by $v \upharpoonright Y$ the restriction of the valuation v to the variables in Y. Similarly, for a trace $\tau = (v_0, v_1, v_2, \ldots)$ on X, we write $\tau \upharpoonright Y = (v_0 \upharpoonright Y, v_1 \upharpoonright Y, v_2 \upharpoonright Y, \ldots)$ for the restriction of τ to the variables in Y. The restriction operator is lifted to sets of valuations, and to sets of traces.

Processes and refinement. For $i \in \{1, 2\}$, a *process* $P_i = (X_i, \delta_i)$ consists of a finite set X_i of variables and a nondeterministic transition function $\delta_i : V_i \to 2^{V_i} \setminus \emptyset$, where V_i is the set of valuations on X_i. The transition function maps a present valuation to a nonempty set of possible successor valuations. We write $X = X_1 \cup X_2$ for the set of variables of both processes; note that some variables may be shared by both processes. A *refinement* of process $P_i = (X_i, \delta_i)$ is a process $P'_i = (X'_i, \delta'_i)$ such that (1) $X_i \subseteq X'_i$, and (2) for all valuations v' on X'_i, we have $\delta'_i(v') \upharpoonright X_i \subseteq \delta_i(v' \upharpoonright X_i)$. In other words, the refined process P'_i has possibly more variables than the original process P_i, and every possible update of the variables in X_i by P'_i is a possible update by P_i. We write $P'_i \preceq P_i$ to denote that P'_i is a refinement of P_i. Given two refinements P'_1 of P_1 and P'_2 of

P_2, we write $X' = X_1' \cup X_2'$ for the set of variables of both refinements, and we denote the set of valuations on X' by V'.

Schedulers. Given two processes P_1 and P_2, a *scheduler* R for P_1 and P_2 chooses at each computatiuon step whether it is process P_1's turn or process P_2's turn to update its variables. Formally, the scheduler R is a function $R : V^* \to \{1, 2\}$ that maps every finite sequence of global valuations (representing the history of a computation) to $i \in \{1, 2\}$, signaling that process P_i is next to update its variables. The scheduler R is *fair* if it assigns turns to both P_1 and P_2 infinitely often; i.e., for all traces $(v_0, v_1, v_2, \ldots) \in V^\omega$, there exist infinitely many $j \geq 0$ and infinitely many $k \geq 0$ such that $R(v_0, \ldots, v_j) = 1$ and $R(v_0, \ldots, v_k) = 2$. Given two processes $P_1 = (X_1, \delta_1)$ and $P_2 = (X_2, \delta_2)$, a scheduler R for P_1 and P_2, and a start valuation $v_0 \in V$, the set of possible traces is $[\![(P_1 \mid\mid P_2 \mid\mid R)(v_0)]\!] = \{(v_0, v_1, v_2, \ldots) \in V^\omega \mid \forall j \geq 0.\ R(v_0, \ldots, v_j) = i \text{ and } v_{j+1} \upharpoonright (X \setminus X_i) = v_j \upharpoonright (X \setminus X_i) \text{ and } v_{j+1} \upharpoonright X_i \in \delta_i(v_j \upharpoonright X_i)\}$. Note that during turns of one process P_i, the values of the private variables $X \setminus X_i$ of the other process remain unchanged.

Specifications. A *specification* Φ_i for processs P_i is a set of traces on X; that is, $\Phi_i \subseteq V^\omega$. We consider only ω-regular specifications [14]. We define boolean operations on specifications using logical operators such as $\wedge$ (conjunction) and $\to$ (implication).

Weak co-synthesis. In all formulations of the co-synthesis problem that we consider, the input to the problem is given as follows: two processes $P_1 = (X_1, \delta_1)$ and $P_2 = (X_2, \delta_2)$, two specifications Φ_1 for process 1 and Φ_2 for process 2, and a start valuation $v_0 \in V$. The *weak co-synthesis* problem is defined as follows: do there exist two processes $P_1' = (X_1', \delta_1')$ and $P_2' = (X_2', \delta_2')$, and a valuation $v_0' \in V'$, such that (1) $P_1' \preceq P_1$ and $P_2' \preceq P_2$ and $v_0' \upharpoonright X = v_0$, and (2) for all fair schedulers R for P_1' and P_2', we have $[\![(P_1' \mid\mid P_2' \mid\mid R)(v_0')]\!] \upharpoonright X \subseteq (\Phi_1 \wedge \Phi_2)$.

Example 1 (Mutual-exclusion protocol synthesis). Consider the two processes shown in Fig. 1. Process P_1 (on the left) places a request to enter its critical section by setting `flag[1]:=true`, and the entry of P_1 into the critical section is signaled by `Cr1:=true`; and similarly for process P_2 (on the right). The two variables `flag`[1] and `flag`[2] are boolean, and in addition, both processes may use a shared variable `turn` that takes two values 1 and 2. There are 8 possible conditions `C1`–`C8` for a process to guard the entry into its critical section.[1] The figure shows all 8×8 alternatives for the two processes; any refinement without additional variables will choose a subset of these. Process P_1 may stay in its critical section for an arbitrary finite amount of time (indicated by `fin_wait`), and then exit by setting `Cr1:=false`; and similarly for process P_2. The `while` loop with the two alternatives `C9` and `C10` expresses the fact that a process may wait arbitrarily long (possibly infinitely long) before a subsequent request to enter its critical section.

[1] Since a guard may check any subset of the three 2-valued variables, there are 256 possible guards; but all except 8 can be discharged immediately as not useful.

```
do                                      do
{                                       {
flag[1]:=true; turn:=2;                 flag[2]:=true; turn:=1;

while (flag[2] & turn=1) nop;           while (flag[1] & turn=2) nop;   (C8+C5)

Cr1:=true; fin_wait; Cr1:=false;        Cr2:=true; fin_wait; Cr2:=false;
flag[1]:=false;                         flag[2]:=false;

wait[1]:=1;                             wait[2]:=1;
while(wait[1]=1)                        while(wait[2]=1)
  | nop;                                  | nop;                        (C9)
  | wait[1]:=0;                           | wait[2]:=0;                 (C10)
} while(true)                           } while(true)
```

Fig. 2. Peterson's mutual-exclusion protocol

We use the notations $\Box$ and $\Diamond$ to denote *always* (safety) and *eventually* (reachability) specifications, respectively. The specification for process P_1 consists of two parts: a safety part $\Phi_1^{\texttt{mutex}} = \Box\neg(\texttt{Cr1} = \texttt{true} \ \wedge \ \texttt{Cr2} = \texttt{true})$ and a liveness part $\Phi_1^{\texttt{prog}} = \Box(\texttt{flag}[1] = \texttt{true} \ \rightarrow \ \Diamond(\texttt{Cr1} = \texttt{true}))$. The first part $\Phi_1^{\texttt{mutex}}$ specifies that both processes are not in their critical sections simultaneously (*mutual exclusion*); the second part $\Phi_1^{\texttt{prog}}$ specifies that if process P_1 wishes to enter its critical section, then it will eventually enter (*starvation freedom*). The specification Φ_1 for process P_1 is the conjunction of $\Phi_1^{\texttt{mutex}}$ and $\Phi_1^{\texttt{prog}}$. The specification Φ_2 for process P_2 is symmetric. ∎

The answer to the weak co-synthesis problem for Example 1 is "Yes." A solution of the weak co-synthesis formulation are two refinements P_1' and P_2' of the two given processes P_1 and P_2, such that the composition of the two refinements satisfies the specifications Φ_1 and Φ_2 for every fair scheduler. One possible solution is as follows: in P_1', the alternatives C4 and C10 are chosen, and in P_2', the alternatives C3 and C10 are chosen. This solution is not satisfactory, because process P_1's starvation freedom depends on the fact that process P_2 requests to enter its critical section infinitely often. If P_2 were to make only a single request to enter its critical section, then the progress part of Φ_1 would be violated.

Classical co-synthesis. The *classical co-synthesis* problem is defined as follows: do there exist two processes $P_1' = (X_1', \delta_1')$ and $P_2' = (X_2', \delta_2')$, and a valuation $v_0' \in V'$, such that (1) $P_1' \preceq P_1$ and $P_2' \preceq P_2$ and $v_0' \upharpoonright X = v_0$, and (2) for all fair schedulers R for P_1' and P_2', we have (a) $[\![(P_1' \mid\mid P_2 \mid\mid R)(v_0')]\!] \upharpoonright X \subseteq \Phi_1$ and (b) $[\![(P_1 \mid\mid P_2' \mid\mid R)(v_0')]\!] \upharpoonright X \subseteq \Phi_2$.

The answer to the classical co-synthesis problem for Example 1 is "No." We will argue later (in Example 2) why this is the case.

Assume-guarantee synthesis. We now present a new formulation of the co-synthesis problem. The main idea is derived from the notion of secure equilibria [4]. We refer to this new formulation as the *assume-guarantee synthesis* problem; it is defined as follows: do there exist two refiements $P_1' = (X_1', \delta_1')$ and

$P_2' = (X_2', \delta_2')$, and a valuation $v_0' \in V'$, such that (1) $P_1' \preceq P_1$ and $P_2' \preceq P_2$ and $v_0' \upharpoonright X = v_0$, and (2) for all fair schedulers R for P_1' and P_2', we have (a) $[\![(P_1' \mid\mid P_2 \mid\mid R)(v_0')]\!] \upharpoonright X \subseteq (\Phi_2 \to \Phi_1)$ and (b) $[\![(P_1 \mid\mid P_2' \mid\mid R)(v_0')]\!] \upharpoonright X \subseteq (\Phi_1 \to \Phi_2)$ and (c) $[\![(P_1' \mid\mid P_2' \mid\mid R)(v_0')]\!] \upharpoonright X \subseteq (\Phi_1 \wedge \Phi_2)$.

The answer to the assume-guarantee synthesis problem for Example 1 is "Yes." A solution P_1' and P_2' is shown in Fig. 2. We will argue the correctness of this solution later (in Example 3). The two refined processes P_1' and P_2' present exactly Peterson's solution to the mutual-exclusion problem. In other words, Peterson's protocol can be derived automatically as an answer to the assume-guarantee synthesis problem for the requirements of mutual exclusion and starvation freedom. The success of assume-guarantee synthesis for the mutual-exclusion problem, together with the failure of the classical co-synthesis, suggests that the classical formulation of co-synthesis is too strong.

3 Game Algorithms for Co-synthesis

We reduce the three formulations of the co-synthesis problem to problems about games played on graphs with three players.

Game graphs. A *3-player game graph* $G = ((S, E), (S_1, S_2, S_3))$ consists of a directed graph (S, E) with a finite set S of states and a set $E \subseteq S^2$ of edges, and a partition (S_1, S_2, S_3) of the state space S into three sets. The states in S_i are player-i states, for $i \in \{1, 2, 3\}$. For a state $s \in S$, we write $E(s) = \{t \in S \mid (s, t) \in E\}$ for the set of successor states of s. We assume that every state has at least one outgoing edge; i.e., $E(s)$ is nonempty for all states $s \in S$. Beginning from a start state, the three players move a token along the edges of the game graph. If the token is on a player-i state $s \in S_i$, then player i moves the token along one of the edges going out of s. The result is an infinite path in the game graph; we refer to such infinite paths as plays. Formally, a *play* is an infinite sequence $(s_0, s_1, s_2, \ldots)$ of states such that $(s_k, s_{k+1}) \in E$ for all $k \geq 0$. We write Ω for the set of plays.

Strategies. A strategy for a player is a recipe that specifies how to extend plays. Formally, a *strategy* σ_i for player i is a function $\sigma_i : S^* \cdot S_i \to S$ that, given a finite sequence of states (representing the history of the play so far) which ends in a player-i state, chooses the next state. The strategy must choose an available successor state; i.e., for all $w \in S^*$ and $s \in S_i$, if $\sigma_i(w \cdot s) = t$, then $t \in E(s)$. We write Σ_i for the set of strategies for player i. Strategies in general require memory to remember some facts about the history of a play. An equivalent definition of strategies is as follows. Let M be a set called *memory.* A strategy $\sigma = (\sigma^u, \sigma^n)$ can be specified as a pair of functions: (1) a *memory-update* function $\sigma^u : S \times M \to M$ that, given the memory and the current state, updates the memory; and (2) a *next-state* function $\sigma^n : S \times M \to S$ that, given the memory and the current state, determines the successor state. The strategy σ is *finite-memory* if the memory M is finite. The strategy σ is *memoryless* if the memory M is a singleton set. Memoryless strategies do not depend on the history of a play, but only on the current state. A memoryless strategy for player i can be specified as a function $\sigma_i : S_i \to S$ such that $\sigma_i(s) \in E(s)$ for all $s \in S_i$. Given a start state $s \in S$ and three strategies $\sigma_i \in \Sigma_i$, one for each of the three players $i \in \{1, 2, 3\}$, there is an unique play, denoted $\omega(s, \sigma_1, \sigma_2, \sigma_3) = (s_0, s_1, s_2, \ldots)$, such that $s_0 = s$ and for all $k \geq 0$, if $s_k \in S_i$, then $\sigma_i(s_0, s_1, \ldots, s_k) = s_{k+1}$; this play is the outcome of the game starting at s *given* the three strategies σ_1, σ_2, and σ_3.

Winning. An *objective* Ψ is a set of plays; i.e., $\Psi \subseteq \Omega$. The following notation is derived from ATL [2]. For an objective Ψ, the set of *winning states* for player 1 in the game graph G is $\langle\langle 1\rangle\rangle_G(\Psi) = \{s \in S \mid \exists \sigma_1 \in \Sigma_1.\ \forall \sigma_2 \in \Sigma_2.\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Psi\}$; a witness strategy σ_1 for player 1 for the existential quantifier is referred to as a *winning strategy.* The winning sets $\langle\langle 2\rangle\rangle_G(\Psi)$ and $\langle\langle 3\rangle\rangle_G(\Psi)$ for players 2 and 3 are defined analogously. The set of winning states for the *team* consisting of player 1 and player 2, playing against player 3, is $\langle\langle 1, 2\rangle\rangle_G(\Psi) = \{s \in S \mid \exists \sigma_1 \in \Sigma_1.\ \exists \sigma_2 \in \Sigma_2.\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Psi\}$. The winning sets $\langle\langle I\rangle\rangle_G(\Psi)$ for other teams $I \subseteq \{1, 2, 3\}$ are defined similarly. The following determinacy result follows from [6].

Theorem 1 (Finite-memory determinacy [6]). *Let Ψ be an ω-regular objective, let G be a 3-player game graph, and let $I \subseteq \{1, 2, 3\}$ be a set of the players. Let $J = \{1, 2, 3\} \setminus I$. Then (1) $\langle\langle I\rangle\rangle_G(\Psi) = S \setminus \langle\langle J\rangle\rangle_G(\neg\Psi)$, and (2) there exist finite-memory strategies for the players in I such that against all strategies for the players in J, for all states in $s \in \langle\langle I\rangle\rangle_G(\Psi)$, the play starting at s given the strategies lies in Ψ.*

Game solutions to weak and classical co-synthesis. Given two processes $P_1 = (X_1, \delta_1)$ and $P_2 = (X_2, \delta_2)$, we define the 3-player game graph $\widehat{G} = ((S, E), (S_1, S_2, S_3))$ as follows: let $S = V \times \{1, 2, 3\}$; let $S_i = V \times \{i\}$ for $i \in \{1, 2, 3\}$; and let E contain (1) all edges of the form $((v, 3), (v, 1))$ for $v \in V$, (2) all edges of the form $((v, 3), (v, 2))$ for $v \in V$, and (3) all edges of the form $((v, i), (u, 3))$ for $i \in \{1, 2\}$ and $u \restriction X_i \in \delta_i(v \restriction X_i)$ and $u \restriction (X \setminus X_i) = v \restriction (X \setminus X_i)$. In other words, player 1 represents process P_1, player 2 represents process P_2, and player 3 represents the scheduler. Given a play of the form $\omega = ((v_0, 3), (v_0, i_0), (v_1, 3), (v_1, i_1), (v_2, 3), \ldots)$, where $i_j \in \{1, 2\}$ for all $j \geq 0$, we write $[\omega]_{1,2}$ for the sequence of valuations $(v_0, v_1, v_2, \ldots)$ in ω (ignoring the intermediate valuations at player-3 states). A specification $\Phi \subseteq V^\omega$ defines the objective $[\![\Phi]\!] = \{\omega \in \Omega \mid [\omega]_{1,2} \in \Phi\}$. In this way, the specifications Φ_1 and Φ_2 for the processes P_1 and P_2 provide the objectives $\Psi_1 = [\![\Phi_1]\!]$ and $\Psi_2 = [\![\Phi_2]\!]$ for players 1 and 2, respectively. The objective for player 3 (the scheduler) is the fairness objective $\Psi_3 = \mathsf{Fair}$ that both S_1 and S_2 are visited infinitely often; i.e., Fair contains all plays $(s_0, s_1, s_2, \ldots) \in \Omega$ such that $s_j \in S_1$ for infinitely many $j \geq 0$, and $s_k \in S_2$ for infinitely many $k \geq 0$.

Proposition 1. *Given two processes $P_1 = (X_1, \delta_1)$ and $P_2 = (X_2, \delta_2)$, two specifications Φ_1 for P_1 and Φ_2 for P_2, and a start valuation $v_0 \in V$, the answer to the weak co-synthesis problem is "Yes" iff* $(v_0, 3) \in \langle\langle 1, 2\rangle\rangle_{\widehat{G}}(\mathsf{Fair} \to ([\![\Phi_1]\!] \wedge$

$\llbracket \Phi_2 \rrbracket$)); *and the answer to the classical co-synthesis problem is "Yes" iff both* $(v_0, 3) \in \langle\langle 1 \rangle\rangle_{\widehat{G}}(\mathsf{Fair} \rightarrow \llbracket \Phi_1 \rrbracket)$ *and* $(v_0, 3) \in \langle\langle 2 \rangle\rangle_{\widehat{G}}(\mathsf{Fair} \rightarrow \llbracket \Phi_2 \rrbracket)$.

Example 2 (Failure of classical co-synthesis). We now demonstrate the failure of classical co-synthesis for Example 1. We show that for every strategy for process P_1, there exist *spoiling* strategies for process P_2 and the scheduler such that (1) the scheduler is fair and (2) the specification Φ_1 of process P_1 is violated. With any fair scheduler, process P_1 will eventually set `flag[1]:=true`. Whenever process P_1 enters its critical section (setting `Cr1:=true`), the scheduler assigns a finite sequence of turns to process P_2. During this sequence, process P_2 enters its critical section: it may first choose the alternative `C10` to return to the beginning of the the main loop, then set `flag[2]:=true; turn:=1;` then pass the guard `C4:` (since ($\mathtt{turn} \neq 2$)), and enter the critical section (setting `Cr2:=true`). This violates the mutual-exclusion requirement $\Phi_1^{\mathtt{mutex}}$ of process P_1. On the other hand, if process P_1 never enters its critical section, this violates the starvation-freedom requirement $\Phi_1^{\mathtt{prog}}$ of process P_1. Thus the answer to the classical co-synthesis problem is "No." ∎

Game solution to assume-guarantee synthesis. We extend the notion of secure equilibria [4] from 2-player games to 3-player games where player 3 can win unconditionally; i.e., $\langle\langle 3 \rangle\rangle_G(\Psi_3) = S$ for the objective Ψ_3 for player 3. In the setting of two processes and a scheduler (player 3) with a fairness objective, the restriction that $\langle\langle 3 \rangle\rangle_G(\Psi_3) = S$ means that the scheduler has a fair strategy from all states; this is clearly the case for $\Psi_3 = \mathsf{Fair}$. (Alternatively, the scheduler may not required to be fair; then Ψ_3 is the set of all plays, and the restriction is satisfied trivially.) The concept of secure equilibria is based on a lexicographic preference ordering of payoff profiles, which can be extended naturally from two to three players under the restriction that player 3 can win unconditionally. We first present the definition of secure equilibria and then characterize the winning secure equilibrium states as the winning states of certain subgames with zero-sum objectives (Theorem 2); this result is a non-trivial generalization of [4] from two to three players. We then establish the existence of finite-memory winning secure strategies (Theorem 3). This will allow us to solve the assume-guarantee synthesis problem by computing winning secure equilibria (Theorem 4).

Payoffs. In the following, we fix a 3-player game graph G and objectives Ψ_1, Ψ_2, and Ψ_3 for the three players such that $\langle\langle 3 \rangle\rangle_G(\Psi_3) = S$. Given strategies σ_i for the three players $i \in \{1, 2, 3\}$, and a state $s \in S$, the *payoff* $p_i(s, \sigma_1, \sigma_2, \sigma_3)$ for player i is 1 if $\omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Psi_i$, and 0 otherwise. The payoff *profile* $(p_1(s, \sigma_1, \sigma_2, \sigma_3), p_2(s, \sigma_1, \sigma_2, \sigma_3), p_3(s, \sigma_1, \sigma_2, \sigma_3))$ consists of the payoff for each player. Since $\langle\langle 3 \rangle\rangle_G(\Psi_3) = S$, any equilibrium payoff profile will assign payoff 1 to player 3. Hence we focus on payoff profiles whose third component is 1.

Payoff-profile ordering. The preference order $\prec_i$ for player i on payoff profiles is defined by $(p_1, p_2, p_3) \prec_i (p'_1, p'_2, p'_3)$ iff either (1) $p_i < p'_i$, or (2) $p_i = p'_i$ and $p_j + p_k > p'_j + p'_k$ for $j, k \in \{1, 2, 3\} \setminus \{i\}$ with $j \neq k$. In the case where the payoff for player 3 is 1, the player-1 preference order $\prec_1$ on payoff profiles is lexicographic:

$(p_1, p_2, 1) \prec_1 (p'_1, p'_2, 1)$ iff either (1) $p_1 < p'_1$, or (2) $p_1 = p'_1$ and $p_2 > p'_2$; that is, player 1 prefers a payoff profile that gives her greater payoff, and if two payoff profiles match in the first component, then she prefers the payoff profile in which player 2's payoff is smaller. The preference order for player 2 is symmetric. The preference order for player 3 is such that $(p_1, p_2, 1) \prec_3 (p'_1, p'_2, 1)$ iff $p_1 + p_2 > p'_1 + p'_2$. Given two payoff profiles (p_1, p_2, p_3) and (p'_1, p'_2, p'_3), we write $(p_1, p_2, p_3) = (p'_1, p'_2, p'_3)$ iff $p_i = p'_i$ for all $i \in \{1, 2, 3\}$, and we write $(p_1, p_2, p_3) \preceq_i (p'_1, p'_2, p'_3)$ iff $(p_1, p_2, p_3) \prec_i (p'_1, p'_2, p'_3)$ or $(p_1, p_2, p_3) = (p'_1, p'_2, p'_3)$.

Secure equilibria. A strategy profile $(\sigma_1, \sigma_2, \sigma_3)$ is a *secure equilibrium* at a state $s \in S$ iff the following three conditions hold:

$$\forall \sigma'_1 \in \Sigma_1.\ (p_1(s, \sigma'_1, \sigma_2, \sigma_3), p_2(s, \sigma'_1, \sigma_2, \sigma_3), p_3(s, \sigma'_1, \sigma_2, \sigma_3)) \preceq_1 p;$$
$$\forall \sigma'_2 \in \Sigma_2.\ (p_1(s, \sigma_1, \sigma'_2, \sigma_3), p_2(s, \sigma_1, \sigma'_2, \sigma_3), p_3(s, \sigma_1, \sigma'_2, \sigma_3)) \preceq_2 p;$$
$$\forall \sigma'_3 \in \Sigma_3.\ (p_1(s, \sigma_1, \sigma_2, \sigma'_3), p_2(s, \sigma_1, \sigma_2, \sigma'_3), p_3(s, \sigma_1, \sigma_2, \sigma'_3)) \preceq_3 p;$$

where $p = (p_1(s, \sigma_1, \sigma_2, \sigma_3), p_2(s, \sigma_1, \sigma_2, \sigma_3), p_3(s, \sigma_1, \sigma_2, \sigma_3))$. In other words, $(\sigma_1, \sigma_2, \sigma_3)$ is a Nash equilibrium with respect to the payoff-profile orderings $\preceq_i$ for the three players $i \in \{1, 2, 3\}$. For $u, w \in \{0, 1\}$, we write $S_{uw1} \subseteq S$ for the set of states s such that a secure equilibrium with the payoff profile $(u, w, 1)$ exists at s; that is, $s \in S_{uw1}$ iff there is a secure equilibrium $(\sigma_1, \sigma_2, \sigma_3)$ at s such that $(p_1(s, \sigma_1, \sigma_2, \sigma_3), p_2(s, \sigma_1, \sigma_2, \sigma_3), p_3(s, \sigma_1, \sigma_2, \sigma_3)) = (u, w, 1)$. Moreover, we write $\mathsf{MS}_{uw1}(G) \subseteq S_{uw1}$ for the set of states s such that the payoff profile $(u, w, 1)$ is a *maximal* secure equilibrium payoff profile at s; that is, $s \in \mathsf{MS}_{uw1}(G)$ iff (1) $s \in S_{uw1}$, and (2) for all $u', w' \in \{0, 1\}$, if $s \in S_{u'w'1}$, then both $(u', w', 1) \preceq_1 (u, w, 1)$ and $(u', w', 1) \preceq_2 (u, w, 1)$. The states in $\mathsf{MS}_{111}(G)$ are referred to as *winning secure equilibrium states*, and the witnessing secure equilibrium strategies as *winning secure strategies*.

Theorem 2. *Let G be a 3-player game graph G with the objectives Ψ_1, Ψ_2, and Ψ_3 for the three players such that $\langle\langle 3 \rangle\rangle_G(\Psi_3) = S$. Let*

$$U_1 = \langle\langle 1 \rangle\rangle_G(\Psi_3 \ \rightarrow\ \Psi_1);$$
$$U_2 = \langle\langle 2 \rangle\rangle_G(\Psi_3 \ \rightarrow\ \Psi_2);$$
$$Z_1 = \langle\langle 1, 3 \rangle\rangle_{G \upharpoonright U_1}(\Psi_1 \ \wedge\ \Psi_3 \ \wedge\ \neg\Psi_2);$$
$$Z_2 = \langle\langle 2, 3 \rangle\rangle_{G \upharpoonright U_2}(\Psi_2 \ \wedge\ \Psi_3 \ \wedge\ \neg\Psi_1);$$
$$W = \langle\langle 1, 2 \rangle\rangle_{G \upharpoonright (S \setminus (Z_1 \cup Z_2))}(\Psi_3 \ \rightarrow\ (\Psi_1 \ \wedge\ \Psi_2)).$$

Then the following assertions hold: (1) at all states in Z_1 the only secure equilibrium payoff profile is $(1, 0, 1)$; (2) at all states in Z_2 the only secure equilibrium payoff profile is $(0, 1, 1)$; and (3) $W = \mathsf{MS}_{111}(G)$.

Proof. We prove parts (1) and (3); the proof of part (2) is similar to part (1).

Part (1). Since $\langle\langle 3 \rangle\rangle_G(\Psi_3) = S$ and $Z_1 \subseteq U_1 = \langle\langle 1 \rangle\rangle_G(\Psi_3 \rightarrow \Psi_1)$, it follows that any secure equilibrium profile in Z_1 has payoff profile of the form $(1, \_, 1)$. Since $(1, 1, 1) \prec_1 (1, 0, 1)$ and $(1, 1, 1) \prec_3 (1, 0, 1)$, to prove uniqueness it suffices to show that player 1 and player 3 can fix strategies to ensure secure equilibrium payoff profile $(1, 0, 1)$. Since $Z_1 = \langle\langle 1, 3 \rangle\rangle_{G \upharpoonright U_1}(\Psi_1 \wedge \Psi_3 \wedge \neg\Psi_2)$, consider the strategy pair (σ_1, σ_3) such that against all player 2 strategies σ_2 and for all states $s \in Z_1$, we have $\omega(s, \sigma_1, \sigma_2, \sigma_3) \in (\Psi_1 \wedge \Psi_3 \wedge \neg\Psi_2)$. The secure equilibrium strategy pair (σ_1^*, σ_3^*) for player 1 and player 3 (along with any strategy σ_2 for player 2) is constructed as follows.

1. The strategy σ_1^* is as follows: player 1 plays σ_1 and if player 3 deviates from σ_3, then player 1 switches to a winning strategy for $\Psi_3 \to \Psi_1$. Such a strategy exists since $Z_1 \subseteq U_1 = \langle\langle 1\rangle\rangle_G(\Psi_3 \to \Psi_1)$.
2. The strategy σ_3^* is as follows: player 3 plays σ_3 and if player 1 deviates from σ_1, then player 3 switches to a winning strategy for Ψ_3. Such a strategy exists since $\langle\langle 3\rangle\rangle_G(\Psi_3) = S$.

Hence objective of player 1 is always satisfied, given objective of player 3 is satisfied. Thus player 3 has no incentive to deviate. Similarly, player 1 also has no incentive to deviate. The result follows.

Part (3). By Theorem 1 we have $S \setminus W = \langle\langle 3\rangle\rangle_G(\Psi_3 \wedge (\neg\Psi_1 \vee \neg\Psi_2))$ and there is a player 3 strategy σ_3 that satisfies $\Psi_3 \wedge (\neg\Psi_1 \vee \neg\Psi_2)$ against all strategies of player 1 and player 2. Hence the equilibrium $(1, 1, 1)$ cannot exist in the complement set of W, i.e., $\mathsf{MS}_{111}(G) \subseteq W$. We now show that in W there is a secure equilibrium with payoff profile $(1, 1, 1)$. The following construction completes the proof.

1. In $W \cap U_1$, player 1 plays a winning strategy for objective $\Psi_3 \to \Psi_1$, and player 2 plays a winning strategy for objective $(\Psi_3 \wedge \Psi_1) \to \Psi_2$. Observe that $S \setminus Z_1 = \langle\langle 2\rangle\rangle_G(\neg\Psi_1 \vee \neg\Psi_3 \vee \Psi_2)$, and hence such a winning strategy exists for player 2.
2. In $W \cap (U_2 \setminus U_1)$, player 2 plays a winning strategy for objective $\Psi_3 \to \Psi_2$, and player 1 plays a winning strategy for objective $(\Psi_2 \wedge \Psi_3) \to \Psi_1$. Observe that $S \setminus Z_2 = \langle\langle 1\rangle\rangle_G(\neg\Psi_2 \vee \neg\Psi_3 \vee \Psi_1)$, and hence such a winning strategy exists for player 1.
3. By Theorem 1 we have $W \setminus U_1 = \langle\langle 2, 3\rangle\rangle_G(\neg\Psi_1 \wedge \Psi_3)$ and $W \setminus U_2 = \langle\langle 1, 3\rangle\rangle_G(\neg\Psi_2 \wedge \Psi_3)$. The strategy construction in $W \setminus (U_1 \cup U_2)$ is as follows: player 1 and player 2 play a strategy (σ_1, σ_2) to satisfy $\Psi_1 \wedge \Psi_2$ against all strategies of player 3, and player 3 plays a winning strategy for Ψ_3; if player 1 deviates, then player 2 and player 3 switches to a strategy $(\overline{\sigma}_2, \overline{\sigma}_3)$ such that against all strategies for player 1 the objective $\Psi_3 \wedge \neg\Psi_1$ is satisfied; and if player 2 deviates, then player 1 and player 3 switches to a strategy $(\overline{\sigma}_1, \overline{\sigma}_3)$ such that against all strategies for player 2 the objective $\Psi_3 \wedge \neg\Psi_2$ is satisfied. Hence neither player 1 and nor player 2 has any incentive to deviate according to the preference order $\preceq_1$ and $\preceq_2$, respectively. ∎

Alternative characterization of winning secure equilibria. In order to obtain a characterization of the set $\mathsf{MS}_{111}(G)$ in terms of strategies, we define retaliation strategies following [4]. Given objectives Ψ_1, Ψ_2, and Ψ_3 for the three players, and a state $s \in S$, the sets of *retaliation strategies* for players 1 and 2 at s are

$$\mathsf{Re}_1(s) = \{\sigma_1 \in \Sigma_1 \mid \forall \sigma_2 \in \Sigma_2. \forall \sigma_3 \in \Sigma_3. \, \omega(s, \sigma_1, \sigma_2, \sigma_3) \in ((\Psi_3 \wedge \Psi_2) \to \Psi_1)\};$$
$$\mathsf{Re}_2(s) = \{\sigma_2 \in \Sigma_2 \mid \forall \sigma_1 \in \Sigma_1. \forall \sigma_3 \in \Sigma_3. \, \omega(s, \sigma_1, \sigma_2, \sigma_3) \in ((\Psi_3 \wedge \Psi_1) \to \Psi_2)\}.$$

Theorem 3. *Let G be a 3-player game graph G with the objectives Ψ_1, Ψ_2, and Ψ_3 for the three players such that $\langle\langle 3\rangle\rangle_G(\Psi_3) = S$. Let $U = \{s \in S \mid \exists\sigma_1 \in \mathsf{Re}_1(s).\ \exists\sigma_2 \in \mathsf{Re}_2(s).\ \forall\sigma_3 \in \Sigma_3.\ \omega(s,\sigma_1,\sigma_2,\sigma_3) \in (\Psi_3 \to (\Psi_1 \wedge \Psi_2))\}$. Then $U = \mathsf{MS}_{111}(G)$.*

Proof. We first show that $U \subseteq \mathsf{MS}_{111}(G)$. For a state $s \in U$, choose $\sigma_1 \in \mathsf{Re}_1(s)$ and $\sigma_2 \in \mathsf{Re}_2(s)$ such that for all $\sigma_3 \in \Sigma_3$, we have $\omega(s,\sigma_1,\sigma_2,\sigma_3) \in (\Psi_3 \to (\Psi_1 \wedge \Psi_2))$. Fixing the strategies σ_1 and σ_2 for players 1 and 2, and a winning strategy for player 3, we obtain the secure equilibrium payoff profile $(1,1,1)$. We now show that $\mathsf{MS}_{111}(G) \subseteq U$. This follows from the proof of Theorem 2. In Theorem 2 we proved that for all states $s \in (S \setminus (Z_1 \cup Z_2))$, we have $\mathsf{Re}_1(s) \neq \emptyset$ and $\mathsf{Re}_2(s) \neq \emptyset$; and the winning secure strategies constructed for the set $W = \mathsf{MS}_{111}(G)$ are witness strategies to prove that $\mathsf{MS}_{111}(G) \subseteq U$. ∎

Observe that for ω-regular objectives, the winning secure strategies of Theorem 3 are finite-memory strategies. The existence of finite-memory winning secure strategies establishes the following theorem.

Theorem 4 (Game solution of assume-guarantee synthesis). *Given two processes $P_1 = (X_1, \delta_1)$ and $P_2 = (X_2, \delta_2)$, two specifications Φ_1 for P_1 and Φ_2 for P_2, and a start valuation $v_0 \in V$, the answer to the assume-guarantee synthesis problem is "Yes" iff $(v_0, 3) \in \mathsf{MS}_{111}(\widehat{G})$ for the 3-player game graph $\widehat{G}$ with the objectives $\Psi_1 = [\![\Phi_1]\!]$, $\Psi_2 = [\![\Phi_2]\!]$, and $\Psi_3 = \mathsf{Fair}$.*

Example 3 (Assume-guarantee synthesis of mutual-exclusion protocol). We consider the 8 alternatives `C1`–`C8` of process P_1, and the corresponding spoiling strategies for process P_2 and the scheduler to violate P_1's specification. We denote by $[\to]$ a switch between the two processes (decided by the scheduler).

`C1` The spoiling strategies for process P_2 and the scheduler cause the following sequence of updates:
- P_1: `flag[1]:=true; turn:=2;` $[\to]$;
- P_2: `flag[2]:=true; turn:=1;`
- P_2: enters the critical section by passing the guard `C8:` (since (`turn` $\neq$ `2`)). After exiting its critical section, process P_2 chooses the alternative `C10` to enter the beginning of the main loop, sets `flag[2]:=true; turn:=1;` and then the scheduler assigns the turn to process P_1, which cannot enter its critical section. The scheduler then assigns turn to P_2 and then P_2 enters the critical section by passing guard `C8` and this sequence is repeated forever.

The same spoiling strategies work for choices `C2, C3, C6` and `C7`.

`C4` The spoiling strategies cause the following sequence of updates:
- P_2: `flag[2]:=true; turn:=1;` $[\to]$;
- P_1: `flag[1]:=true; turn:=2;` $[\to]$;
- P_2: enters the critical section by passing the guard `C3:` (since (`turn` $\neq$ `1`)). After exiting its critical section, process P_2 continues to choose the alternative `C9` forever, and the scheduler alternates turn between P_1 and P_2; and process P_1 cannot enter its critical section.

The same spoiling strategies work for the choice `C5`.

`C8` The spoiling strategies cause the following sequence of updates:

P_2: `flag[2]:=true; turn:=1;` $[\rightarrow]$;
P_1: `flag[1]:=true; turn:=2;` $[\rightarrow]$;
P_2: `while(flag[2]) nop;`

Then process P_2 does not enter its critical section, and neither can process P_1 enter. In this case P_2 cannot violate P_1's specification without violating her own specification.

It follows from this case analysis that no alternatives except `C8` for process P_1 can witness a solution to the assume-guarantee synthesis problem. The alternative `C8` for process P_1 and the symmetric alternative `C6` for process P_2 provide winning secure strategies for both processes. In this example, we considered refinements without additional variables; but in general refinements can have additional variables. ∎

4 Abstraction-based Co-synthesis

In Section 3 we provided game-based algorithms for the three formulations of the co-synthesis problem. However, the state space of the game graph can be very large, making an algorithmic analysis often impractical. In this section we present sound proof rules (i.e., sufficient conditions) for deriving solutions to the three co-synthesis problems from the analysis of simpler game graphs, which abstracts the original game graph. We first review the appropriate notion of *game abstraction* and the corresponding proof rules for the weak and classical versions of co-synthesis. We then give proof rules for assume-guarantee synthesis in the two special but common cases where the processes have safety and Büchi objectives. In particular, we show that the solution of zero-sum games on simpler, abstract game graphs is sufficient for solving a given assume-guarantee synthesis problem: the winning strategies of two different abstract zero-sum games provide winning secure strategies for the original non-zero-sum game.

Abstraction of game graphs. Let $I \subseteq \{1,2,3\}$ be a set of players, and let $J = \{1,2,3\} \setminus I$. An *I-abstraction* for a 3-player game graph $G = ((S,E),(S_1,S_2,S_3))$ consists of a 3-player game graph $G^A = ((S^A,E^A),(S_1^A,S_2^A,S_3^A))$ and a *concretization* function $\gamma : S^A \to 2^S \setminus \emptyset$ such that the following conditions hold.

1. The abstraction preserves the player structure: for all $i \in \{1,2,3\}$ and $a \in S_i^A$, we have $\gamma(a) \subseteq S_i$.
2. The abstraction partitions the concrete state space: $\bigcup_{a \in S^A} \gamma(a) = S$, and for every $s \in S$ there is a unique $a \in S^A$ such that $s \in \gamma(a)$.
3. The edges for players in I are abstracted universally, and the edges for players in J are abstracted existentially:

$$\begin{aligned} E^A = \; & \{(a,b) \mid \exists i \in I.\ a \in S_i^A \ \wedge\ \forall s \in \gamma(a).\ \exists t \in \gamma(b).\ (s,t) \in E\} \\ \cup \; & \{(a,b) \mid \exists i \in J.\ a \in S_i^A \ \wedge\ \exists s \in \gamma(a).\ \exists t \in \gamma(b).\ (s,t) \in E\}. \end{aligned}$$

The *abstraction* function $\alpha : S \to S^A$ is defined such that $s \in \gamma(\alpha(s))$ for all states $s \in S$. For a play $\omega = (s_0, s_1, s_2, \ldots)$ in G, the abstraction $\alpha(\omega) = (\alpha(s_0), \alpha(s_1), \alpha(s_2), \ldots)$ is a play in G^A.

Abstraction of objectives. Given an objective Ψ on the concrete game graph G, we define the following two objectives on the abstract game graph G^A:

–existential abstraction: $\alpha(\Psi) = \{\alpha(\omega) \mid \omega \in \Psi\}$;
–universal abstraction: $\beta(\Psi) = \{\tau \mid \forall \omega \in S^\omega.\ \text{if } \tau = \alpha(\omega) \text{ then } \omega \in \Psi\}$.

For the players in I, the abstract objectives are obtained by universal abstraction, and for the players in J, by existential abstraction.

Proof rules for weak and classical co-synthesis. The following proposition states the basic principle behind proof rules for weak and classical co-synthesis.

Proposition 2. [7] *Given a 3-player game graph G, a set $I \subseteq \{1,2,3\}$ of players, an I-abstraction (G^A, γ), and an objective Ψ, let $A = \langle\langle I \rangle\rangle_{G^A}(\beta(\Psi))$. Then $\gamma(A) \subseteq \langle\langle I \rangle\rangle_G(\Psi)$.*

Proof rules for assume-guarantee synthesis. We present proof rules for assume-guarantee synthesis in two cases: for safety objectives, and for Büchi objectives (which include reachability objectives as a special case).

Safety objectives. Given a set $F \subseteq S$ of states, the *safety* objective $\Box F$ requires that the set F is never left. Formally, the safety objective $\Box F$ contains all plays $(s_0, s_1, s_2, \ldots)$ such that $s_j \in F$ for all $j \geq 0$. Given safety objectives for players 1 and 2, it is immaterial whether the scheduler (player 3) is fair or not, because if a safety objective is violated, then it is violated by a finite prefix of a play. Hence, for simplicity, we assume that the objective of player 3 is trivial (i.e., the set of all plays). The following theorem states that winning secure equilibrium states in a game graph G can be derived from winning secure equilibrium states in two simpler graphs, a $\{1\}$-abstraction G_1^A and a $\{2\}$-abstraction G_2^A. The winning secure strategies on the concrete graph can likewise be derived from the winning secure strategies on the two abstract graphs.

Theorem 5. *Let G be a 3-player game graph with two safety objectives Ψ_1 and Ψ_2 for players 1 and 2, respectively. Let (G_1^A, γ_1) be a $\{1\}$-abstraction, and let (G_2^A, γ_2) be a $\{2\}$-abstraction. Let the objective for player 1 in G_1^A and G_2^A be $\beta_1(\Psi_1)$ and $\alpha_2(\Psi_1)$, respectively. Let the objective for player 2 in G_1^A and G_2^A be $\alpha_1(\Psi_2)$ and $\beta_2(\Psi_2)$, respectively. Let the objective for player 3 in G, G_1^A, and G_2^A be the set of all plays. Let $A_1 = \mathsf{MS}_{111}(G_1^A)$ and $A_2 = \mathsf{MS}_{111}(G_2^A)$. Then $(\gamma_1(A_1) \cap \gamma_2(A_2)) \subseteq \mathsf{MS}_{111}(G)$.*

The classical assume-guarantee rule for safety specifications [1] can be obtained as a special case of Theorem 5 where all states are player-3 states (in this case, player 3 is not only a scheduler, but also resolves all nondeterminism in the two processes P_1 and P_2).

Büchi objectives. Given a set $B \subseteq S$ of states, the *Büchi* objective $\Box\Diamond B$ requires that the set B is visited infinitely often. Formally, the Büchi objective $\Box\Diamond B$ contains all plays $(s_0, s_1, s_2, \ldots)$ such that $s_j \in B$ for infinitely many $j \geq 0$. The following theorem states that winning secure equilibrium states (and the corresponding winning secure strategies) in a game graph G can be derived from a zero-sum analysis of three simpler graphs, a $\{1\}$-abstraction G_1^A, a $\{2\}$-abstraction G_2^A, and a $\{1,2\}$-abstraction $G_{1,2}^A$.

Theorem 6. *Let G be a 3-player game graph with two Büchi objectives Ψ_1 and Ψ_2 for player 1 and player 2, respectively, and the objective* Fair *for player 3. Let (G_1^A, γ_1) be a $\{1\}$-abstraction, let (G_2^A, γ_2) be a $\{2\}$-abstraction, and let $(G_{1,2}^A, \gamma_{1,2})$ be a $\{1,2\}$-abstraction. Let*

$$A_1 = \langle\langle 1 \rangle\rangle_{G_1^A}((\alpha_1(\mathsf{Fair}) \ \wedge \ \alpha_1(\Psi_2)) \ \rightarrow \ \beta_1(\Psi_1));$$
$$A_2 = \langle\langle 2 \rangle\rangle_{G_2^A}((\alpha_2(\mathsf{Fair}) \ \wedge \ \alpha_2(\Psi_1)) \ \rightarrow \ \beta_2(\Psi_2));$$
$$A_3 = \langle\langle 1,2 \rangle\rangle_{G_{1,2}^A}(\alpha_{1,2}(\mathsf{Fair}) \ \rightarrow \ (\beta_{1,2}(\Psi_1') \ \wedge \ \beta_{1,2}(\Psi_2')));$$

where $\Psi_1' = (\Psi_1 \wedge \Box\gamma_1(A_1))$ and $\Psi_2' = (\Psi_2 \wedge \Box\gamma_2(A_2))$. Then $\gamma_{1,2}(A_3) \subseteq \mathsf{MS}_{111}(G)$.

References

1. R. Alur and T.A. Henzinger. Reactive modules. In *Formal Methods in System Design*, 15:7–48, 1999.
2. R. Alur, T.A. Henzinger, O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.
3. K. Chatterjee and T.A. Henzinger. Semiperfect-information games. In *FSTTCS'05*, LNCS 3821, pages 1–18. Springer, 2005.
4. K. Chatterjee, T.A. Henzinger, M. Jurdziński. Games with secure equilibria. In *LICS'04*, pages 160–169. IEEE, 2004.
5. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs'81*, LNCS 131, pages 52–71. Spinger, 1982.
6. Y. Gurevich and L. Harrington. Trees, automata, and games. In *STOC'82*, pages 60–65. ACM, 1982.
7. T.A. Henzinger, R. Majumdar, F.Y.C. Mang, J.-F. Raskin. Abstract interpretation of game properties. In *SAS'00*, LNCS 1824, pages 220–239. Springer, 2000.
8. P. Madhususan and P.S. Thiagarajan. Distributed controller synthesis for local specifications. In *ICALP'01*, LNCS 2076, pages 396–407. Springer, 2001.
9. S. Mohalik and I. Walukiewicz. Distributed games. In *FSTTCS'03*, LNCS 2914, pages 338–351. Springer, 2003.
10. C.H. Papadimitriou. Algorithms, games, and the internet. In *STOC'01*, pages 749–753. ACM, 2001.
11. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL'89*, pages 179–190. ACM, 1989.
12. P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25:206–230, 1987.
13. J.H. Reif. The complexity of 2-player games of incomplete information. *Journal of Computer and System Sciences*, 29:274–301, 1984.
14. W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.