Assume Guarantee Synthesis

Krishnendu Chatterjee
Thomas A. Henzinger

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-32
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-32.html

April 4, 2006

Acknowledgement

This research was supported in part by the AFOSR MURI grant F49620-00-1-0327, and the NSF ITR grant CCR-0225610.

Assume Guarantee Synthesis *

Krishnendu Chatterjee[†] Thomas A. Henzinger[†,‡]

[†] EECS, University of California, Berkeley, USA
[‡] EPFL, Switzerland
{c_krish,tah}@eecs.berkeley.edu

April 4, 2006

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 (such as mutex, bounded overtaking, and 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 our assume-guarantee synthesis algorithm automatically computes Peterson's protocol.

1 Introduction

The algorithmic *synthesis* (or control) of reactive systems is based on solving two-player zero-sum games on graphs [8, 9]. 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 computer science applications [7]. We demonstrate, for

*This research was supported in part by the AFOSR MURI grant F49620-00-1-0327, and the NSF ITR grant CCR-0225610.

the first time, 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 mutex, 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 that other player can retaliate by violating the first player's objective. We refer to such strategies as winning secure equilibrium strategies.

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 $P_1||P_2||S$, where S is a scheduler that arbitrarily but fairly interleaves the actions of P_1 and P_2, satisfies the mutual-exclusion requirements of Dijkstra [10]. We show that traditional zero-sum game-theoretic formulations, where P_1 and P_2 either collaborate against S, 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, the secure-equilibrium solution is exactly Peterson's mutual-exclusion protocol. In other words, Peterson's protocol can be synthesized automatically as the secure winning 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. This is also, to our knowledge, the first application of Nash equilibria —in particular, the special kind called "secure"— in system design.

The correct formulation of co-synthesis, with the two processes competing conditionally, is called *assume-guarantee synthesis*, because similar to assume-guarantee verification (see e.g. [6]), 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 requires the computation of secure equilibria in 3-player games, with the three players P_1, P_2, and S. Previously, meaningful (i.e., unique maximal) secure equilibria were known to exist only for certain 2-player games [4], and there it was also shown that in general such meaningful equilibria need not exist for three players. Here we extend the theoretical results of [4] in two ways, in order to solve the assume-guarantee synthesis problem. First, we provide unique characterization of winning secure equilibrium in 3-player games *provided* the third player can win unconditionally. In assume-guarantee synthesis this condition is satisfied, 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 secure equilibrium 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 secure equilibrium 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.

2 Co-synthesis

2.1 Processes and schedulers

Let X be a *finite* set of variables such that each variable $x \in X$ has a finite domain D_x. We denote by $\theta[X] \in \prod_{x \in X} D_x$ a *valuation* on X that assigns each variable $x \in X$ a value $\theta(x) \in D_x$. We denote by $\Theta[X]$ the set of all valuations on X. For $A \subseteq X$ and a valuation $\theta[X]$ we denote by $\theta[X] \restriction A$ the restriction of the valuation on the set A of variables.

Processes. For $i \in \{ 1, 2 \}$, a *process* $P_i = (X_i, \delta_i)$ consists of a *finite* set X_i of variables and a non-deterministic transition function $\delta_i : \Theta[X_i] \to 2^{\Theta[X_i]} \setminus \emptyset$, i.e., the transition function given a present valuation specifies the possible updates. We denote by $X = X_1 \cup X_2$ the set of all variables.

Scheduler. A *scheduler* Sc at each round chooses whether it is process P_1's or process P_2's turn to update the variables. Formally, the scheduler Sc is a function $\mathsf{Sc} : \Theta[X]^* \to \{1, 2\}$. A *fair scheduler* assigns turns to both process P_1 and P_2 infinitely often, i.e., let $\tau \in (\Theta[X])^\omega$ be an infinite sequence of valuations and let τ_i denote the prefix of length i of τ; for a fair scheduler Sc for all τ there exists infinitely many $i \geq 0$ and $j \geq 0$ such that $\mathsf{Sc}(\tau_i) = 1$ and $\mathsf{Sc}(\tau_j) = 2$.

Refinement of processes. A *refinement* of a 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 $\theta[X'_i] \in \Theta[X'_i]$ we have $\delta'_i(\theta[X'_i]) \restriction X_i \subseteq \delta_i(\theta[X'_i] \restriction X_i)$;

i.e., the process P'_i has possibly more variables and every possible update of variables of X_i in P'_i is a possible update in P_i. We denote by $P'_i \preceq P_i$ that P'_i is a refinement of P_i.

Traces. Given a set X of variables, a *trace* $\tau(X) = (v_0, v_1, \ldots) \in \Theta[X]^\omega$ is an infinite sequence of valuations on X. Given a trace $\tau(X) = (v_0, v_1, v_2, \ldots)$ and $A \subseteq X$ we denote by $\tau(X) \restriction A = (v_0 \restriction A, v_1 \restriction A, v_2 \restriction A, \ldots)$ the restriction of $\tau(X)$ on the set A of variables. The notation for a set of traces is similar.

Traces of processes and scheduler. Given process $P_1 = (X_1, \delta_1)$, $P_2 = (X_2, \delta_2)$, a scheduler Sc and a starting valuation $v_0 \in \Theta[X]$, where $X = X_1 \cup X_2$, the set of all possible traces are

$$\begin{aligned} [\![(P_1 \parallel P_2 \parallel \mathsf{Sc})(v_0)]\!] \;=\; & \{\, (v_0, v_1 \ldots) \in \Theta[X]^\omega \mid \forall i \geq 0.\ \mathsf{Sc}(v_0, v_1, \ldots, v_i) = j; \\ & v_{i+1} \restriction (X \setminus X_j) = v_i \restriction (X \setminus X_j);\ \ v_{i+1} \restriction X_j \in \delta_j(v_i \restriction X_j) \,\} \end{aligned}$$

Specifications. A *specification* $\Phi_i(X)$ is a set of traces on X, i.e., $\Phi_i(X) \subseteq \Theta[X]^\omega$. We consider only ω-regular specifications in this paper [11].

2.2 Weak co-synthesis

The *weak co-synthesis* problem is formulated as follows:

1. *Input.* Processes $P_1 = (X_1, \delta_1)$, $P_2 = (X_2, \delta_2)$; and specifications $\Phi_1(X)$ for player 1 and $\Phi_2(X)$ for player 2, and a starting valuation $v_0 \in \Theta[X]$, where $X = X_1 \cup X_2$.

2. *Solution problem.* Does there exist $P_1' = (X_1', \delta_1')$ and $P_2' = (X_2', \delta_2')$ and a valuation $v_0' \in \Theta[X_1' \cup X_2']$ such that $P_1' \preceq P_1$ and $P_2' \preceq P_2$; $v_0' \restriction X = v_0$ and for all fair schedulers Sc we have

$$[\![(P_1' \mid\mid P_2' \mid\mid \mathsf{Sc})(v_0')]\!] \restriction X \subseteq \Phi_1(X) \cap \Phi_2(X).$$

Example 1 (Mutual exclusion protocol synthesis) *We will consider the synthesis of the mutual exclusion (mutex) protocol of two processes as shown in Fig 1. We use the short-hand notations* $\Box$ *and* $\Diamond$ *to denote* always (safety) *and* eventually (reachability) *specifications, respectively. A process i places a request to enter the critical section by setting* $\mathsf{flag[i]} = 1$ *and the entering of the process i in the critical section is denoted by* $\mathsf{Cri} = \mathsf{true}$*. The process can stay in the critical section for an arbitrary finite amount of time (denoted as* $\mathsf{fin\_wait}$*), and then comes out of the critical section and assigns* $\mathsf{Cri} = \mathsf{false}$*. The choice* $(\mathsf{C9}, \mathsf{C10})$ *expresses that the process can arbitrarily wait (for possibly infinitely long) without any further request to enter the critical section or proceed with a request to enter the critical section. The specification for process 1 is consists of two parts:*

$$\Phi_1^{\mathsf{progress}} = \Box\big((\mathsf{flag[1]} == 1) \Rightarrow \Diamond(\mathsf{Cr1} == \mathsf{true})\big);\ \textit{and}$$

$$\Phi_1^{\mathsf{mutex}} = \Box\big(\neg(\mathsf{Cr1} == \mathsf{true} \wedge \mathsf{Cr2} == \mathsf{true})\big).$$

The specification $\Phi_1^{\mathsf{progress}}$ *denotes that if process 1 wishes to enter the critical section (denoted by setting* $\mathsf{flag[1]} = 1$*), then process 1 eventually enters the critical section; and the specification* Φ_1^{mutex} *denotes that both the processes are not in the critical section. The specification* Φ_1 *for process 1 is the conjunction of* $\Phi_1^{\mathsf{progress}}$ *and* Φ_1^{mutex}*. The specification* Φ_2 *for process 2 is symmetric.* ∎

Example 2 (Undesirable behavior synthesis) *A solution of the weak co-synthesis formulation (i.e.,* P_1' *and* P_2'*) is given in Fig 2. However, the solution is not satisfactory because process 1 depends on the fact that process 2 requests for the critical section infinitely often to make progress. If process 2 only makes a single request for the critical section and then never makes further request, then the progress specification of process 1 is easily violated.* ∎

2.3 Classical co-synthesis

The *classical co-synthesis* problem is formulated as follows:

1. *Input.* Processes $P_1 = (X_1, \delta_1)$, $P_2 = (X_2, \delta_2)$; and specifications $\Phi_1(X)$ for player 1 and $\Phi_2(X)$ for player 2, and a starting valuation $v_0 \in \Theta[X]$, where $X = X_1 \cup X_2$.

```
do                                      do
{                                       {
flag[1]:=1; turn:=2;                    flag[2]:=1; 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)          | while (flag[1] && turn ==2)         (C5)
   nop;                                   nop;
 | while (flag[1] && turn ==1)          | while (flag[1] && turn ==1)         (C6)
   nop;                                   nop;
 | while (flag[2] && turn ==2)          | while (flag[2] && turn ==2)         (C7)
   nop;                                   nop;
 | while (flag[2] && turn ==1)          | while (flag[2] && turn ==1)         (C8)
   nop;                                   nop;
Cr1:=true; fin_wait;                    Cr2:=true; fin_wait;
Cr1:=false;                             Cr2:=false;
flag[1]:=false;                         flag[2]:=false;

X(1):=1;                                X(2):=1;
while(X(1)==1)                          while(X(2)==1)
 | nop;                                  | nop;                              (C9)
 | X(1):=0;                              | X(2):=0;                          (C10)
} while(1)                              } while(1)
```

Figure 1: A mutual exclusion protocol synthesis

2. *Solution problem.* Does there exist $P_1' = (X_1', \delta_1')$ and $P_2' = (X_2', \delta_2')$ and a valuation $v_0' \in \Theta[X_1' \cup X_2']$ such that $P_1' \preceq P_1$ and $P_2' \preceq P_2$; $v_0' \restriction X = v_0$ and for all fair schedulers Sc we have

$$[\![(P_1' \parallel P_2 \parallel \mathsf{Sc})(v_0')]\!] \restriction X \subseteq \Phi_1(X);$$

$$[\![(P_1 \parallel P_2' \parallel \mathsf{Sc})(v_0')]\!] \restriction X \subseteq \Phi_2(X).$$

Example 3 (Mutex protocol synthesis) *The answer to the classical synthesis problem for Example 1 is "NO". We will argue later (in Example 5) why the answer to the given problem is negative.* ∎

2.4 Assume-guarantee synthesis

We now present a new formulation of the co-synthesis problem, and the idea of the formulation is derived from the notion of *secure equilibrium* [3]. We refer to this new formulation as the *assume-guarantee synthesis* problem.

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

 while (turn==2) nop;                while (turn==1) nop;

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

X(1):=1                             X(2):=1
while(X(1)==1)                      while(X(2)==1)
 X(1):=0;                            X(2):=0;
} while(1)                          } while(1)
```

Figure 2: Weak co-synthesis

1. *Input.* Processes $P_1 = (X_1, \delta_1)$, $P_2 = (X_2, \delta_2)$; and specifications $\Phi_1(X)$ for player 1 and $\Phi_2(X)$ for player 2, and a starting valuation $v_0 \in \Theta[X]$, where $X = X_1 \cup X_2$.

2. *Solution problem.* Does there exist $P_1' = (X_1', \delta_1')$ and $P_2' = (X_2', \delta_2')$ and a valuation $v_0' \in \Theta[X_1' \cup X_2']$ such that $P_1' \preceq P_1$ and $P_2' \preceq P_2$; $v_0' \restriction X = v_0$ and for all fair schedulers Sc we have

$$[\![(P_1' \mid\mid P_2 \mid\mid \mathsf{Sc})(v_0')]\!] \restriction X \subseteq \Phi_2(X) \rightarrow \Phi_1(X);$$

$$[\![(P_1 \mid\mid P_2' \mid\mid \mathsf{Sc})(v_0')]\!] \restriction X \subseteq \Phi_1(X) \rightarrow \Phi_2(X);$$

$$[\![(P_1' \mid\mid P_2' \mid\mid \mathsf{Sc})(v_0')]\!] \restriction X \subseteq \Phi_1(X) \cap \Phi_2(X).$$

Example 4 (Assume-guarantee synthesis of mutex protocol) *A solution to the assume-guarantee synthesis problem for Example 1 is shown in Fig 3. We will argue the correctness of the solution as assume-guarantee synthesis problem later (in Example 6). The synthesis of the processes shown in Fig 3 is the Peterson mutual exclusion protocol.* ∎

The success of the assume-guarantee synthesis problem for the mutex protocol of Example 1 and the failure of the classical co-synthesis problem suggests the classical co-synthesis formulation is too strong.

3 Game Algorithms for Co-synthesis

3.1 Three player games

We consider games played on game graphs with three players.

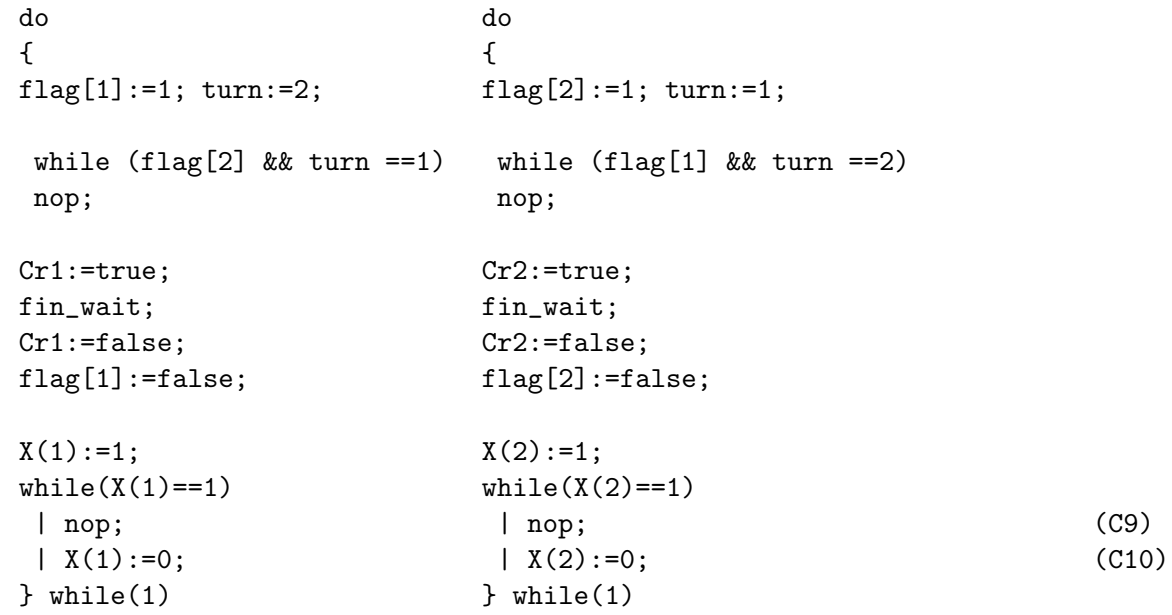

Figure 3: Peterson mutual exclusion protocol synthesis

Game graphs. A *game graph* $G = ((S, E), (S_1, S_2, S_3))$ consists of a directed graph (S, E) with a finite state space S and a set E of edges, and a partition (S_1, S_2, S_3) of the state space S into three sets. The states in S_1 are player 1 states, the states in S_2 are player 2 states, and the states in S_3 are player 3 states. 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 out-going edge, i.e., $E(s)$ is non-empty for all states $s \in S$.

Plays. A game is played by three players: player 1, player 2, and player 3, who form an infinite path in the game graph by moving a token along edges. They start by placing the token on an initial state, and then they take moves indefinitely in the following way. If the token is on a state in S_1, then player 1 moves the token along one of the edges going out of the state. If the token is on a state in S_2 (or S_3), then player 2 (resp. player 3) does likewise. 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 $\langle s_0, s_1, s_2, \ldots \rangle$ of states such that $(s_k, s_{k+1}) \in E$ for all $k \geq 0$. We write Ω for the set of all 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 $\sigma_i\colon\ S^* \cdot S_i \to S$ 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 only available successors, 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 all strategies for player i. Strategies in general require memory to remember the history of plays. An equivalent definition of strategies is as follows. Let M be a set called *memory*. A strategy with memory can be described as a pair of functions: (a) a *memory-update* function $\sigma_u\colon\ S \times M \to M$ that, given the memory and the current state,

updates the memory; and (b) a *next-state* function σ_n: $S \times M \to S$ that, given the memory and the current state, specifies the successor state. The strategy is *finite-memory* if the memory M is finite. The strategy is *memoryless* if the memory M is a singleton set. The memoryless strategies do not depend on the history of a play, but only on the current state. Each memoryless strategy for player i can be specified as a function σ_i: $S_i \to S$ such that $\sigma_i(s) \in E(s)$ for all $s \in S_i$. Given a starting state $s \in S$, strategies $\sigma_i \in \Sigma_i$, for player 1, player 2 and player 3, there is a unique play, denoted $\omega(s, \sigma_1, \sigma_2, \sigma_3) = \langle s_0, s_1, s_2, \ldots \rangle$, which is defined as follows: $s_0 = s$ and for all $k \geq 0$, if $s_k \in S_i$, then $\sigma_i(s_0, s_1, \ldots, s_k) = s_{k+1}$.

Notations. Objectives Φ in G are subsets of plays, i.e., $\Phi \subseteq \Omega$. We define the notion of winning for an objective, for a player and a set of players as follows: the notations are derived from ATL [2]. For an objective Φ the set of winning states for player 1 in a game graph G is as follows:

$$\langle\langle 1 \rangle\rangle_G(\Phi) = \{\, s \in S \mid \exists \sigma_1 \in \Sigma_1.\ \forall \sigma_2 \in \Sigma_2.\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Phi \,\};$$

and a witness strategy σ_1 for player 1 for the existential quantification is referred to as a *winning* strategy. The notations for winning states $\langle\langle 2 \rangle\rangle_G(\Phi)$ and $\langle\langle 3 \rangle\rangle_G(\Phi)$ for player 2 and player 3 are similar. The notion of the set of winning states for objective Φ in a game graph G, for player 1 and player 2 combined against player 3 is as follows:

$$\langle\langle 1, 2 \rangle\rangle_G(\Phi) = \{\, s \in S \mid \exists \sigma_1 \in \Sigma_1.\ \exists \sigma_2 \in \Sigma_2.\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Phi \,\};$$

and the notations for $\langle\langle 1, 3 \rangle\rangle_G(\Phi)$ and $\langle\langle 2, 3 \rangle\rangle_G(\Phi)$ are similar. The following determinacy results follows from the result of [5].

Theorem 1 (Finite-memory determinacy[5]) *For all ω-regular objectives Φ, for all three player game graphs G, for all $I \subseteq \{ 1, 2, 3 \}$ and $J = \{ 1, 2, 3 \} \setminus I$, the following assertions hold: (a) $\langle\langle I \rangle\rangle_G(\Phi) = S \setminus \langle\langle J \rangle\rangle_G(\neg\Phi)$; and (b) there exists finite-memory strategies for players in I such that against all strategies of the players in J, for all states in $s \in \langle\langle I \rangle\rangle_G(\Phi)$, the play starting at s given the strategies satisfy Φ.*

3.2 Game solution to weak and classical co-synthesis

Game graph. We associate with the processes and scheduler a game graph $G_{\mathsf{cosynthesis}} = ((S, E), (S_1, S_2, S_3))$ defined as follows:

1. The set of states S is as follows:

$$S = \Theta[X] \times \{ 1, 2, 3 \};\ \text{and}\ S_i = \Theta[X] \times \{ i \}.$$

2. The set of edges E is as follows:

$$\begin{array}{rcl} E & = & \{\, ((s,3),(s,1)) \mid s \in \Theta[X] \,\} \ \cup\ \{\, ((s,3),(s,2)) \mid s \in \Theta[X] \,\} \\ & \cup & \{\, ((s,i),(s',3)) \mid i \in \{ 1,2 \};\ s' \restriction X_i \in \delta_i(s \restriction X_i);\ s' \restriction (X \setminus X_i) = s \restriction (X \setminus X_i) \,\}. \end{array}$$

Restriction of plays. We define the restriction of plays to valuations as follows: given a play $\omega = \langle (v_0, 3), (v_0, i_0), (v_1, 3), (v_1, i_1), (v_2, 3), \ldots \rangle \in \Omega$, where for all $j \geq 0$ we have $i_j \in \{ 1, 2 \}$, we denote by $\omega \restriction (S_1 \cup S_2)$ the sequence of valuations $(v_0, v_1, v_2, \ldots)$ in ω.

Objectives. Objectives Φ in the game $G_{\mathsf{cosynthesis}}$ are subsets of plays, i.e., $\Phi \subseteq S^\omega$. We only consider ω-regular objectives [11]. Given a specification $\Phi(X) \subseteq \Theta[X]^\omega$ we denote by $[\![\Phi(X)]\!] = \{\, \omega \in \Omega \mid \omega \restriction (S_1 \cup S_2) \in \Phi(X) \,\}$. For the rest of the section the objective of player 3 is the fairness objective Φ_3 to satisfy that both S_1 and S_2 are visited infinitely often.

Lemma 1 (Game solution of weak co-synthesis) *Given processes $P_1 = (X_1, \delta_1)$, $P_2 = (X_2, \delta_2)$; and specifications $\Phi_1(X)$ for player 1 and $\Phi_2(X)$ for player 2, and a starting valuation $v_0 \in \Theta[X]$, where $X = X_1 \cup X_2$, let $\Phi_1 = [\![\Phi_1(X)]\!]$ and $\Phi_2 = [\![\Phi_2(X)]\!]$. The answer to the weak co-synthesis problem is "YES" if and only if $(v_0, 3) \in \langle\!\langle 1,2 \rangle\!\rangle_{G_{\mathsf{cosynthesis}}}(\Phi_3 \rightarrow (\Phi_1 \wedge \Phi_2))$.*

Proof. *(Sketch).* We first note that for games with ω-regular objectives, finite-memory winning strategies suffices (Theorem 1). The proof follows by the following case analysis.

1. Given a finite-memory strategy σ_1, a witness $P_1' = (X_1', \delta_1')$ for the weak co-synthesis problem can be obtained as follows: the variables $X_i' \setminus X_i$ encodes the finite-memory information of the strategy σ_1 and the next-state function of the strategy is then captured by a deterministic update function δ_1'. A similar construction holds for player 2.

2. Given a witness $P_1' = (X_1', \delta_1')$ as a witness for the weak co-synthesis problem, we first observe that any deterministic restriction of P_1' (i.e., the transition function δ_1' is made deterministic) is also a witness to the weak co-synthesis problem. A witness strategy σ_1 in $G_{\mathsf{cosynthesis}}$ is obtained as follows: the variables in $X_1' \setminus X_1$ is encoded as the finite-memory information of σ_1 and the deterministic update is captured by the next-state function. The construction of witness strategies for player 2 is similar. ∎

Lemma 2 (Game solution of classical co-synthesis) *Given processes $P_1 = (X_1, \delta_1)$, $P_2 = (X_2, \delta_2)$; and specifications $\Phi_1(X)$ for player 1 and $\Phi_2(X)$ for player 2, and a starting valuation $v_0 \in \Theta[X]$, where $X = X_1 \cup X_2$, let $\Phi_1 = [\![\Phi_1(X)]\!]$ and $\Phi_2 = [\![\Phi_2(X)]\!]$. The answer to the classical co-synthesis problem is "YES" if and only if $(v_0, 3) \in \langle\!\langle 1 \rangle\!\rangle_{G_{\mathsf{cosynthesis}}}(\Phi_3 \rightarrow \Phi_1)$ and $(v_0, 3) \in \langle\!\langle 2 \rangle\!\rangle_{G_{\mathsf{cosynthesis}}}(\Phi_3 \rightarrow \Phi_2)$.*

The proof for Lemma 2 is similar to Lemma 1.

Example 5 (Failure of classical co-synthesis) *We now argue the failure of the classical co-synthesis for Example 1. We show that for every strategy for process 1, there exists* spoiling *strategy for process 2 and scheduler such that specification of process 1 is violated. We first consider the choices of process 1 ($C1$ — $C8$) and produce spoiling strategies for process 2 and the scheduler.*

1. Choice C1. *Given choice $C1$ for process 1, we construct spoiling strategy for process 2 and the scheduler that satisfies process 2's specification and not process 1's. The sequence of valuations is as follows:*

   Process 2: flag[2]:=1; turn:=1;
   Process 1: flag[1]:=1; turn:=2;
   Process 2: Enters critical section by condition while(flag[2] && turn==1) nop; and then it always does nop ($C9$); so process 1 does not enter the critical section.

2. Choice C2. *Process 2 and the scheduler follow the same spoiling strategy till entering the critical section as for the case of choice C1. Then instead of choice* $(C9)$*, process 2 comes out of the loop, sets flag[2]:=1; turn:=1, and then the scheduler assigns turn to process 1.*

3. Choice C3. *Same spoiling strategy as for choice C2.*

4. Choice C4. *A spoiling strategy of process 2 and the scheduler that satisfies process 2's specification and not process 1's is as follows:*

   Process 1: flag[1]:=1; turn:=2;
   Process 2: flag[2]:=1; turn:=1;
   Process 2: Enters critical section by condition while(flag[2] && turn==2) nop; and then it always does nop; so process 1 does not enter the critical section.

5. Choice C5 and C6. *Reduces to C3 and C4, respectively.*

6. Choice C7. *The spoiling strategy is as follows:*

   Process 1: flag[1]:=1; turn:=2;
   Process 2: flag[2]:=1; turn:=1;
   Process 2: Enters critical section by condition while(flag[2] && turn==2) nop; and then process 2 follows choice $(C10)$*, enters the loop, sets flag[2]:=1; turn:=1; and then the scheduler assigns turn to process 1; and so process 1 does not enter the critical section.*

7. Choice C8. *The spoiling strategies are as follows:*

   Process 2: flag[2]=1; turn=1;
   Process 1: flag[1]=1; turn=2;
   Process 2: while(flag[2]) nop; hence process 2 does not enter the critical section and neither does process 1.

   Observe that in this case process 2 cannot falsify process 1's specification satisfying her own specification.

It is easy to argue from the above construction, that for every strategy for process 1, there exists strategy for process 2 and the scheduler that falsify process 1's specification. Hence the answer to the classical co-synthesis problem is negative. ∎

3.3 Secure equilibrium and game solution to assume-guarantee synthesis

In this subsection we consider three player games with lexicographic ordering of objectives and the notion of equilibrium is formalized as the notion of secure equilibrium [4]. We consider the special class of games where player 3 can win unconditionally, i.e., given the objective Φ_3 for player 3 we have $\langle\langle 3\rangle\rangle_G(\Phi_3) = S$. In the setting of processes and scheduler (as player 3) with the fairness

objective Φ_3 for the scheduler, the restriction that $\langle\langle 3 \rangle\rangle_G(\Phi_3) = S$ means the scheduler has a fair scheduling strategy from all states. In the special case when the scheduler is not restricted to be fair, the objective Φ_3 of the scheduler is the set of all paths and the restriction is satisfied trivially. We first present the notion of secure equilibrium and then provide unique characterization of the winning secure equilibrium states (Theorem 2). We also establish the existence of finite-memory winning secure equilibrium strategies (Theorem 3) and relate the assume-guarantee synthesis problem to solving three player games with winning secure equilibrium (Theorem 4).

Values and payoff-profiles. Given objectives Φ_i for player i, strategies σ_i for player i, for $i \in \{ 1, 2, 3 \}$ and a state s, we denote by val_i the value for player i and is defined as follows:

$$val_i(s, \sigma_1, \sigma_2, \sigma_3) = \begin{cases} 1 & \text{if } \omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Phi_i; \\ 0 & \text{otherwise;} \end{cases}$$

and the payoff-profile $(val_1(s, \sigma_1, \sigma_2, \sigma_3), val_2(s, \sigma_1, \sigma_2, \sigma_3), val_3(s, \sigma_1, \sigma_2, \sigma_3))$ consists of the value for each player, and i-th component represents the value for player i. We now present the notion of *secure equilibrium* [3]. Since $\langle\langle 3 \rangle\rangle_G(\Phi_3) = S$, any equilibrium payoff-profile will assign value 1 to player 3. Hence we will focus on payoff-profiles with value 1 for player 3.

Payoff-profile ordering. The preference order $\preceq_i$ for player i is defined as follows:

$$\begin{array}{lll} (val_1, val_2, val_3) & \prec_i & (val'_1, val'_2, val'_3) \\ & \text{iff} & (val_i < val'_i) \vee (val_i = val'_i \wedge val_j + val_k > val'_j + val'_k); \\ & \text{for} & j \neq k, j, k \in \{ 1, 2, 3 \} \setminus \{ i \}. \end{array}$$

In the special case where the value for player 3 is 1, the player 1 preference order $\preceq_1$ and the player 2 preference order $\preceq_2$ on payoff profiles are obtained lexicographically:

$$(val_1, val_2, 1) \prec_1 (val'_1, val'_2, 1) \text{ iff } (val_1 < val'_1) \vee (val_1 = val'_1 \wedge val_2 > val'_2),$$

that is, player 1 prefers a payoff profile which gives her greater payoff, and if two payoff profiles match in the first component, then she prefers the payoff profile in which the player 2's payoff is minimized; symmetrically,

$$(val_1, val_2, 1) \prec_2 (val'_1, val'_2, 1) \text{ iff } (val_2 < val'_2) \vee (val_2 = val'_2 \wedge val_1 > val'_1).$$

The preference order for player 3 is as follows:

$$(val_1, val_2, 1) \prec_3 (val'_1, val'_2, 1) \text{ iff } val_1 + val_2 > val'_1 + val'_2.$$

The preference ordering for each player is shown in Fig 4. Given two payoff profiles (val_1, val_2, val_3) and (val'_1, val'_2, val'_3), we write $(val_1, val_2, val_3) = (val'_1, val'_2, val'_3)$ iff $val_i = val'_i$ for $i \in \{1, 2, 3\}$ and $(val_1, val_2, val_3) \preceq_i (val_1, val'_2, val'_3)$ iff either $(val_1, val_2, val_3) \prec_i (val'_1, val'_2, val'_3)$ or $(val_1, val_2, val_3) = (val'_1, val'_2, val'_3)$.

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

$$\forall \sigma'_1 \in \Sigma_1.\ val_1(s, \sigma'_1, \sigma_2, \sigma_3) \preceq_1 val_1(s, \sigma_1, \sigma_2, \sigma_3);$$

$(0,1,1) \prec_1 (0,0,1) \prec_1 (1,1,1) \prec_1 (1,0,1)$ Player 1 preference order.

$(1,0,1) \prec_2 (0,0,1) \prec_2 (1,1,1) \prec_2 (0,1,1)$ Player 2 preference order.

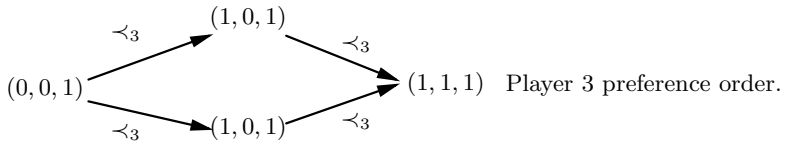

Figure 4: Payoff-profile preference ordering.

$$\forall \sigma_2' \in \Sigma_2.\ val_2(s, \sigma_1, \sigma_2', \sigma_3) \preceq_2 val_2(s, \sigma_1, \sigma_2, \sigma_3);$$
$$\forall \sigma_3' \in \Sigma_3.\ val_3(s, \sigma_1, \sigma_2, \sigma_3') \preceq_3 val_3(s, \sigma_1, \sigma_2, \sigma_3),$$

i.e., $(\sigma_1, \sigma_2, \sigma_3)$ is a Nash equilibrium with respect to the payoff-profile orderings $\preceq_i$ for player i, where $i \in \{ 1, 2, 3 \}$. For $v, w \in \{0, 1\}$, we write $S_{vw1} \subseteq S$ to denote the set of states s such that a secure equilibrium with the payoff profile $(v, w, 1)$ exists in the game G at s with objectives Φ_i for player i, that is, $s \in S_{vw1}$ iff there is a secure equilibrium $(\sigma_1, \sigma_2, \sigma_3)$ at s such that $(val_1(s, \sigma_1, \sigma_2, \sigma_3), val_2(s, \sigma_1, \sigma_2, \sigma_3), val_3(s, \sigma_1, \sigma_2, \sigma_3)) = (v, w, 1)$. Similarly, $\mathsf{MS}_{vw1} \subseteq S_{vw1}$ denotes the set of states s such that the payoff profile $(v, w, 1)$ is a *maximal secure equilibrium payoff profile* at s, that is, $s \in \mathsf{MS}_{vw1}$ iff (1) $s \in S_{vw1}$ and (2) for all $v', w' \in \{0, 1\}$, if $s \in S_{v'w'1}$, then $(v', w', 1) \preceq_1 (v, w, 1)$ and $(v', w', 1) \preceq_2 (v, w, 1)$. The set of states MS_{111} is referred to as the winning secure equilibrium states and witness secure equilibrium strategies as winning secure equilibrium strategies.

Characterization of MS_{111}. Let

$$U_1 = \langle\langle 1 \rangle\rangle(\Phi_3 \rightarrow \Phi_1); \qquad U_2 = \langle\langle 2 \rangle\rangle(\Phi_3 \rightarrow \Phi_2).$$
$$Z_1 = \langle\langle 1, 3 \rangle\rangle_{(G \restriction U_1)}(\Phi_1 \wedge \Phi_3 \wedge \neg\Phi_2); \qquad Z_2 = \langle\langle 2, 3 \rangle\rangle_{(G \restriction U_2)}(\Phi_2 \wedge \Phi_3 \wedge \neg\Phi_1).$$

Lemma 3 *In Z_1 the unique secure equilibrium is $(1, 0, 1)$.*

Proof. Since $Z_1 = \langle\langle 1, 3 \rangle\rangle_{(G \restriction U_1)}(\Phi_1 \wedge \Phi_3 \wedge \neg\Phi_2)$, 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 $\omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Phi_1 \wedge \Phi_3 \wedge \neg\Phi_2$. 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 $\Phi_3 \rightarrow \Phi_1$. Such a strategy exists since $Z_1 \subseteq U_1 = \langle\langle 1 \rangle\rangle_G(\Phi_3 \rightarrow \Phi_1)$.

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 Φ_3. Such a strategy exists since $\langle\langle 3 \rangle\rangle_G(\Phi_3) = S$.

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. ∎

Lemma 4 *In Z_2 the unique secure equilibrium is* $(0, 1, 1)$.

Theorem 2 (Characterization of MS_{111}) *Let $Z = Z_1 \cup Z_2$, and $W = \langle\langle 1, 2 \rangle\rangle_{G\restriction(S\setminus Z)}(\Phi_3 \rightarrow (\Phi_1 \wedge \Phi_2))$. Then $W = \mathsf{MS}_{111}$.*

Proof. By Theorem 1 we have $S \setminus W = \langle\langle 3 \rangle\rangle_G(\Phi_3 \wedge (\neg\Phi_1 \vee \neg\Phi_2))$ and there is a player 3 strategy σ_3 that satisfies $\Phi_3 \wedge (\neg\Phi_1 \vee \neg\Phi_2)$ 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., $\mathsf{MS}_{111} \subseteq W$. We now show that in W there is a secure equilibrium with payoff-profile $(1, 1, 1)$. The following construction ensures.

1. In $W \cap U_1$, player 1 plays a winning strategy for objective $\Phi_3 \rightarrow \Phi_1$, and player 2 plays a winning strategy for objective $(\Phi_3 \wedge \Phi_1) \rightarrow \Phi_2$. Observe that $S \setminus Z_1 = \langle\langle 2 \rangle\rangle_G(\neg\Phi_1 \vee \neg\Phi_3 \vee \Phi_2)$, 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 $\Phi_3 \rightarrow \Phi_2$, and player 1 plays a winning strategy for objective $(\Phi_2 \wedge \Phi_3) \rightarrow \Phi_1$. Observe that $S \setminus Z_2 = \langle\langle 1 \rangle\rangle_G(\neg\Phi_2 \vee \neg\Phi_3 \vee \Phi_1)$, and hence such a winning strategy exists for player 1.

3. By Theorem 1 we have $W \setminus U_1 = \langle\langle 2, 3 \rangle\rangle_G(\neg\Phi_1 \wedge \Phi_3)$ and $W \setminus U_2 = \langle\langle 1, 3 \rangle\rangle_G(\neg\Phi_2 \wedge \Phi_3)$. 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 $\Phi_1 \wedge \Phi_2$ against all strategies of player 3, and player 3 plays a winning strategy for Φ_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 $\Phi_3 \wedge \neg\Phi_1$ 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 $\Phi_3 \wedge \neg\Phi_2$ 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.

The result follows. ∎

Strategic characterization of MS_{111}. To obtain a characterization of MS_{111} in terms of winning strategies we define the notion of *retaliating strategies*. The set of retaliation strategies for player 1 and player 2 as follows:

$$\mathsf{Re}_1(s) = \{\, \sigma_1 \in \Sigma_1 \mid \forall \sigma_2 \in \Sigma_2.\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in ((\Phi_3 \wedge \Phi_2) \rightarrow \Phi_1) \,\};$$

$$\mathsf{Re}_2(s) = \{\, \sigma_2 \in \Sigma_2 \mid \forall \sigma_1 \in \Sigma_1.\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in ((\Phi_3 \wedge \Phi_1) \rightarrow \Phi_2) \,\}.$$

Theorem 3 *Let*

$$U = \{\, s \in S \mid \exists \sigma_1 \in \mathsf{Re}_1(s).\ \exists \sigma_2 \in \mathsf{Re}_2(s).\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Phi_3 \rightarrow (\Phi_1 \wedge \Phi_2) \,\}.$$

Then $U = \mathsf{MS}_{111}$.

Proof. We first show that $U \subseteq \mathsf{MS}_{111}$. For a state $s \in U$, pick $\sigma_1 \in \mathsf{Re}_1(s)$ and $\sigma_2 \in \mathsf{Re}_2(s)$ such that for all $\sigma_3 \in \Sigma_3$ we have $\omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Phi_3 \rightarrow (\Phi_1 \wedge \Phi_2)$. Then fixing (σ_1, σ_2) for player 1 and player 2, and a winning strategy for player 3 we obtain a secure equilibrium of $(1, 1, 1)$.

We now show that $\mathsf{MS}_{111} \subseteq U$. The result follows from the proof of Theorem 2. We proved that for $s \in G \setminus (Z_1 \cup Z_2)$ we have $\mathsf{Re}_1(s) \neq \emptyset$ and $\mathsf{Re}_2(s) \neq \emptyset$. The strategy construction in Theorem 2 in W is a witness that for all $s \in \mathsf{MS}_{111}$ we have $s \in U$. ∎

Observe that the construction of secure equilibrium strategies in Theorem 2 are finite-memory strategies for ω-regular objectives. The existence of finite-memory secure equilibrium strategies and argument similar to Lemma 1 establishes the following theorem.

Theorem 4 (Game solution of assume-guarantee synthesis) *Given processes $P_1 = (X_1, \delta_1)$, $P_2 = (X_2, \delta_2)$; and specifications $\Phi_1(X)$ for player 1 and $\Phi_2(X)$ for player 2, and a starting valuation $v_0 \in \Theta[X]$, where $X = X_1 \cup X_2$, let $\Phi_1 = [\![\Phi_1(X)]\!]$ and $\Phi_2 = [\![\Phi_2(X)]\!]$. The answer to the assume-guarantee synthesis problem is "YES" if and only if $(v_0, 3) \in \mathsf{MS}_{111}$ in $G_{\mathsf{cosynthesis}}$.*

Example 6 (Assume-guarantee synthesis of mutex protocol) *It follows from the analysis in Example 5 that all choices other than the choice $C8$ for process 1 cannot be witness for the assume-guarantee synthesis problem. The choice $C8$ for process 1 and the symmetric choice $C5$ for process 2 yield a witness for the assume-guarantee synthesis problem.* ∎

4 Abstraction-based Co-synthesis

The results of Section 3 provide game algorithms for the co-synthesis problems. However the state space of the original game structure can be large and algorithmic analysis on the original game structure can be impractical. In this section we present sound proof rules for game solutions for the co-synthesis problems from simpler game structures (*abstractions*). In Section 3 we established correspondence of the weak, classical and assume-guarantee synthesis problem with game solutions for the respective co-synthesis problem. In this section we focus on sound proof rules for the game solutions. We first present the notion of abstractions and review the sound proof rules for the weak and classical co-synthesis problem. We then present sound proof rules for the assume-guarantee synthesis problem for safety and Büchi (liveness) objectives. In particular we show how to solve zero-sum simpler game structures (abstractions) and obtain witness of winning secure equilibrium strategies in the original game structure from the winning strategies of the simpler games.

4.1 Abstractions

Abstractions. An *abstraction* $G^A_\Xi(I)$, for $I \subseteq \{1,2,3\}$, for a game structure $G = ((S,E),(S_1,S_2,S_3))$ is a game structure $G^A_\Xi(I) = ((S^A,E^A)(S^A_1,S^A_2,S^A_3))$ with a *concretization* function $\Xi : S^A \to 2^S \setminus \emptyset$ such that the following conditions hold.

- The abstraction preserves the player structures: for all $i \in \{1,2,3\}$, for all $s^A \in S^A_i$ we have $\Xi(s^A) \subseteq S_i$.
- The abstraction partitions the concrete state space: $\bigcup_{s^A \in S^A} \Xi(s^A) = S$ and for every $s \in S$ there is an unique $s^A \in S^A$ such that $s \in \Xi(s^A)$.
- Universal abstraction of edges for players in I and existential abstraction of edges for players in $J = \{1,2,3\} \setminus I$:

$$\begin{array}{rcl} E^A & = & \{\, (s^A,t^A) \mid \exists i \in I.\ s^A \in S^A_i.\ \forall s \in \Xi(s^A).\ \exists t \in \Xi(t^A).\ (s,t) \in E \,\} \\ & \cup & \{\, (s^A,t^A) \mid \exists i \in J.\ s^A \in S^A_i.\ \exists s \in \Xi(s^A).\ \exists t \in \Xi(t^A).\ (s,t) \in E \,\} \end{array}$$

Mapping of plays and strategies. Given a play $\omega = \langle s_0, s_1, s_2, \ldots \rangle \in \Omega$ in G we obtain the play $\Xi(\omega) = \langle s^A_0, s^A_1, s^A_2, \ldots \rangle$ in G^A_Ξ such that for all $i \geq 0$ we have $s_i \in \Xi(s^A_i)$. The notation for prefix of plays (i.e., history of a play) is similar. For $i \in I$, and a strategy σ^A_i we denote by $\sigma_i = \Xi(\sigma^A_i)$ the strategy defined as follows: for a history $w \in S^*$ and $s \in S_i$, the strategy σ_i chooses a successor t in

$\Xi(\sigma_i^A(\Xi(w \cdot s)))$; observe that since the abstraction of edges for player i is universal, the construction of the strategy is sound.

Abstraction of objectives. Given an objective Φ in G we use the following notations for objectives in $G^A_\Xi(I)$

$$\begin{array}{rrcl} \text{Existential projection:} & \Xi(\Phi)_\exists & = & \{\, \omega^A \mid \exists \omega \in \Phi.\ \omega^A = \Xi(\omega) \,\}; \\ \text{Universal projection:} & \Xi(\Phi)_\forall & = & \{\, \omega^A \mid \forall \omega.\ \text{if } \omega^A = \Xi(\omega) \text{ then } \omega \in \Phi \,\}. \end{array}$$

For abstraction $G^A_\Xi(I)$ the objectives of players in I are obtained by universal projection and objectives of players in $\{1,2,3\} \setminus I$ are obtained by existential projection. For singleton i, in sequel we write $G^A_\Xi(i)$ to denote $G^A_\Xi(\{\, i \,\})$.

4.2 Proof rules

The following theorem is the basic principle of obtaining sound proof rules for weak and classical co-synthesis problem.

Theorem 5 *Given a game structure G, an objective Φ, and an abstraction $G^A_\Xi(1)$ if $s^A \in \langle\langle 1 \rangle\rangle_{G^A_\Xi(1)}(\Xi(\Phi)_\forall)$, then for all $s \in \Xi(s^A)$ we have $s \in \langle\langle 1 \rangle\rangle_G(\Phi)$. Similar result hold for player 2 abstractions.*

We now present sound proof rules for secure equilibrium for safety and Büchi objectives (as a special case the result can be also obtained for reachability objectives).

Safety objectives. For a set $F \subseteq S$, the *safety* objective requires the set F is never left. Formally, the safety objective $\Box(F)$ defines the set of plays $\{\, \langle s_0, s_1, s_2, \ldots \rangle \mid \forall i \geq 0.\ s_i \in F \,\}$. Given safety objectives for both players, whether the scheduler is a fair scheduler or not is immaterial, since if a safety objective can be violated, then it can also be violated by a fair scheduler. Hence in proof rules for safety objectives for simplicity the objective of player 3 is assumed to be the set of all plays. A sound proof rule for secure equilibrium with safety objectives is presented in the following theorem. The classical assume-guarantee rule can be obtained as a special case of the following result by restricting the strategies of player 1 and player 2 to singletons and interpreting player 3 as the universal player that resolves all non-determinism (observe that player 3 is not restricted to be fair in the following theorem).

Theorem 6 *Let G be a game structure with safety objectives Φ_1 and Φ_2 for player 1 and player 2, respectively. Let $G^A_\Xi(1)$ and $G^A_\Xi(2)$ be player 1 and player 2 abstractions, respectively. Let the objectives for the players in $G^A_\Xi(1)$ be $\Phi^1_1 = \Xi(\Phi_1)_\forall$, $\Phi^1_2 = \Xi(\Phi_2)_\exists$, respectively; and in $G^A_\Xi(2)$ be $\Phi^2_1 = \Xi(\Phi_1)_\exists$, $\Phi^2_2 = \Xi(\Phi_2)_\forall$, respectively. Then if $s^A \in \mathsf{MS}_{111}$ in $G^A_\Xi(1)$ and $s^A \in \mathsf{MS}_{111}$ in $G^A_\Xi(2)$, then for all $s \in \Xi(s^A)$ we have $s \in \mathsf{MS}_{111}$ in G.*

Proof. *(Sketch).* Let $(\overline{\sigma}_1, \sigma_2)$ be retaliation strategies in $G^A_\Xi(1)$ such that for all strategies σ_3 we have $\omega^A(s^A, \overline{\sigma}_1, \sigma_2, \sigma_3) \in (\Phi^1_1 \wedge \Phi^1_2)$. Let $(\sigma_1, \overline{\sigma}_2)$ be retaliation strategies in $G^A_\Xi(2)$ such that for all strategies σ_3 we have $\omega^A(s, \sigma_1, \overline{\sigma}_2, \sigma_3) \in (\Phi^2_1 \wedge \Phi^2_2)$. The strategies $\sigma^*_1 = \Xi(\overline{\sigma}_1)$ and $\sigma^*_2 = \Xi(\overline{\sigma}_2)$ are retaliation strategies in G, and for all σ_3 we have $\omega(s, \sigma^*_1, \sigma^*_2, \sigma_3) \in \Phi_1 \wedge \Phi_2$. ∎

Liberal retaliation strategies. Given safety objectives $\Box(F_1)$ and $\Box(F_2)$ for player 1 and player 2, respectively, a strategy σ_1 for player 1 is a *liberal retaliation strategy* if (a) σ_1 is a retaliation strategy; and (b) for all histories $\langle s_0, s_1, s_2, \ldots, s_k \rangle$ such that for all $i \leq k, s_i \in F_1$, such that the history can

arise by playing σ_1 and some strategies σ_2, σ_3, there is a strategy σ_2, such that for all σ_3 we have $\omega(s, \sigma_1, \sigma_2, \sigma_3) \in \Box(F_2)$. It means that any threat strategy is not invoked as long as the player is safe. We denote by LRe_1 the set of liberal retaliation strategies for player 1, and the definition for LRe_2 is similar. The next lemma follows since if both plays play any liberal retaliation strategies, then for all strategies σ_3 the play never leaves $F_1 \cap F_2$.

Lemma 5 *Given a game structure G with safety objectives Φ_1 and Φ_2 for player 1 and player 2, respectively, if $s \in \mathsf{MS}_{111}$, then $\forall \sigma_1 \in \mathsf{LRe}_1.\ \forall \sigma_2 \in \mathsf{LRe}_2.\ \forall \sigma_3 \in \Sigma_3.\ \omega(s, \sigma_1, \sigma_2, \sigma_3) \in (\Phi_1 \wedge \Phi_2)$.*

Büchi objectives. For a set $B \subseteq S$, the *Büchi* objective requires the set B is visited infinitely often. Formally, the Büchi objective $\Box\Diamond(B)$ defines the set of plays $\{ \langle s_0, s_1, s_2, \ldots \rangle \mid \forall i \geq 0.\ \exists j \geq i.\ s_i \in B \}$. A sound proof rule for secure equilibrium with Büchi objectives is presented in the following theorem.

Theorem 7 *Let G be a game structure with Büchi objectives $\Phi_1 = \Box\Diamond(B_1)$ and $\Phi_2 = \Box\Diamond(B_2)$ for player 1 and player 2, respectively, and the fairness objective Φ_3 for player 3. Let $G^A_\Xi(1)$ and $G^A_\Xi(2)$ be player 1 and player 2 abstractions, respectively, and $G^A_\Xi(\{1,2\})$ be player 1 and 2 abstraction. Let the objectives for the players be*

$$\Phi^1_1 = \Xi(\Phi_1)_\forall, \qquad \Phi^1_2 = \Xi(\Phi_2)_\exists, \qquad \Phi^1_3 = \Xi(\Phi_3)_\exists, \qquad \text{in } G^A_\Xi(1);$$

$$\Phi^2_1 = \Xi(\Phi_1)_\exists, \qquad \Phi^2_2 = \Xi(\Phi_2)_\forall, \qquad \Phi^2_3 = \Phi^1_3 = \Xi(\Phi_3)_\exists, \qquad \text{in } G^A_\Xi(2).$$

Let

$$C^A_1 = \langle\langle 1 \rangle\rangle_{G^A_\Xi(1)}((\Phi^1_3 \wedge \Phi^1_2) \rightarrow \Phi^1_1); \qquad C^A_2 = \langle\langle 2 \rangle\rangle_{G^A_\Xi(2)}((\Phi^2_3 \wedge \Phi^2_1) \rightarrow \Phi^2_2).$$

Let $C_1 = \{ s \in S \mid s \in \Xi(s^A), s^A \in C^A_1 \}$ and $C_2 = \{ s \in S \mid s \in \Xi(s^A), s^A \in C^A_2 \}$; and consider the objectives $\Phi^3_1 = \Xi(\Phi_1 \cap \Box(C_1))_\forall$ and $\Phi^3_2 = \Xi(\Phi_2 \cap \Box(C_2))_\forall$. Let

$$C^A_3 = \langle\langle 1, 2 \rangle\rangle_{G^A_\Xi(\{1,2\})}(\Phi^1_3 \rightarrow (\Phi^3_1 \wedge \Phi^3_2)).$$

For all $s^A \in C^A_3$ and for all $s \in \Xi(s^A)$ we have $s \in \mathsf{MS}_{111}$ in G.

Proof. *(Sketch).* There exist strategies (σ^A_1, σ^A_2) for player 1 and player 2 in $G^A_\Xi(\{1,2\})$ such that for all $s^A \in C^A_3$ and for all strategies σ^A_3 we have $\omega^A(s^A, \sigma^A_1, \sigma^A_2, \sigma^A_3) \in \Phi^1_3 \rightarrow (\Phi^3_1 \wedge \Phi^3_2)$, and the strategies also ensure that once B_1 is reached a state in B_2 is reached within at most $|S|$-visits to B_1 and that once B_2 is reached a state in B_1 is reached within at most $|S|$-visits to B_2. Let $\widehat{\sigma}^A_1$ be a strategy in $G^A_\Xi(1)$ such that for all $s^A \in C^A_1$ and for all strategies σ^A_2, σ^A_3 we have $\omega^A(s^A, \widehat{\sigma}^A_1, \sigma^A_2, \sigma^A_3) \in (\Phi^1_3 \wedge \Phi^1_2) \rightarrow \Phi^1_1$; and let $\widehat{\sigma}^A_2$ be a strategy in $G^A_\Xi(2)$ such that for all $s^A \in C^A_2$ and for all strategies σ^A_1, σ^A_3 we have $\omega^A(s^A, \sigma^A_1, \widehat{\sigma}^A_2, \sigma^A_3) \in (\Phi^1_3 \wedge \Phi^2_1) \rightarrow \Phi^2_2$. The witness for retaliation strategies for the conclusion is as follows.

- Strategy σ^*_1: player 1 starts playing the co-operating strategy $\Xi(\sigma^A_1)$ obtained from $G^A_\Xi(\{1,2\})$ to satisfy $\Phi_3 \rightarrow (\Phi_1 \wedge \Phi_2)$ and also ensures C_1 is not left; whenever a state in B_2 is reached after a visit to B_1, then player 1 continues to play $\Xi(\sigma^A_1)$ for at most $|S|$-visits to B_2 and if B_1 is not reached within $|S|$ visit to B_2, then player 1 switches to the strategy $\Xi(\widehat{\sigma}^A_1)$ obtained from $G^A_\Xi(1)$ to satisfy $(\Phi_3 \wedge \Phi_2) \rightarrow \Phi_1$, and if B_1 is reached continue with the co-operating strategy $\Xi(\sigma^A_1)$ obtained from $G^A_\Xi(\{1,2\})$.

```
Process 1.
do
{
 | z1:= 0;
 | z1:= 1;
 y1:=1;
 while(y1==1) {
  | y1:=1;
  | y1:=0;
 }
 if(y2==1)
  x1:=1;

 x1:=0;
} while(1)
```

```
Process 2.
do
{
 | z2:= 0;
 | z2:= 1;
 y2:=1;
 while(y2==1) {
  | y2:=1;
  | y2:=0;
 }
 if(y1==1)
  x2:=1;

 x2:=0;
} while(1)
```

Figure 5: Processes

- Strategy σ_2^*: player 2 starts playing the co-operating strategy $\Xi(\sigma_2^A)$ obtained from $G_\Xi^A(\{1,2\})$ to satisfy $\Phi_3 \rightarrow (\Phi_1 \wedge \Phi_2)$ and also ensures C_2 is not left; whenever a state in B_1 is reached after a visit to B_2, then player 2 continues to play $\Xi(\sigma_2^A)$ for at most $|S|$-visits to B_1 and if B_2 is not reached within $|S|$ visit to B_1, then player 2 switches to the strategy $\Xi(\widehat{\sigma}_2^A)$ obtained from $G_\Xi^A(2)$ to satisfy $(\Phi_3 \wedge \Phi_1) \rightarrow \Phi_2$, and if B_2 is reached continue with the co-operating strategy $\Xi(\sigma_2^A)$ obtained from $G_\Xi^A(\{\ 1,2\ \})$.

The strategies (σ_1^*, σ_2^*) are witness retaliation strategies to prove the result. ∎

Example 7 (Process abstractions) *Consider the processes P_1 and P_2 shown in Fig 5 and the corresponding abstractions A_1 and A_2 shown in Fig 6 obtained by ignoring variables z_1 and z_2, respectively. The objective for process i is $\Box\Diamond(x_i == 1)$, i.e., to set $x_i := 1$ infinitely often. The strategy for process 1 for assume-guarantee synthesis as obtained from the abstraction is as follows: process 1 sets y_1 to 1 unless x_2 is set to 1 and when x_2 is set to 1, then process 1 sets y_1 to 0. The strategy for process 2 is obtained symmetrically. The above strategies obtained from the abstractions are witness to the assume-guarantee synthesis problem.* ∎

References

[1] R. Alur and T.A. Henzinger. Reactive modules. In *Formal Methods in System Design*, pages 207–218. IEEE Computer Society Press, 1999.

[2] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

```
Abstraction 1.
do
{
 y1:=1;
 while(y1==1) {
  | y1:=1;
  | y1:=0;
 }
 if(y2==1)
  x1:=1;

 x1:=0;
} while(1)
```

```
Abstraction 2.
do
{
 y2:=1;
 while(y2==1) {
  | y2:=1;
  | y2:=0;
 }
 if(y1==1)
  x2:=1;

 x2:=0;
} while(1)
```

Figure 6: Process abstractions

[3] K. Chatterjee, T.A. Henzinger, and M. Jurdziński. Games with secure equilibria. In *LICS'04*, pages 160–169. IEEE, 2004.

[4] K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Quantitative stochastic parity games. In *SODA'04*, pages 121–130. SIAM, 2004.

[5] Y. Gurevich and L. Harrington. Trees, automata, and games. In *STOC'82*, pages 60–65. ACM Press, 1982.

[6] T.A. Henzinger, S. Qadeer, and S.K. Rajamani. You assume, we guarantee: Methodology and case studies. In *CAV'98*, LNCS 1427, pages 440–451. Springer, 1998.

[7] C.H. Papadimitriou. Algorithms, games, and the internet. In *STOC'01*, pages 749–753. ACM Press, 2001.

[8] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL'89*, pages 179–190. ACM Press, 1989.

[9] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[10] A. Silberschatz, P.B. Galvin, and G. Gagne. *Operating System Concepts (Seventh Edition)*. John Wiley and Sons, 2004.

[11] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.