

# Environment Assumptions for Synthesis

*Krishnendu Chatterjee*<sup>1</sup>

*Thomas A. Henzinger*<sup>2</sup>

*Barbara Jobstmann*<sup>2</sup>

<sup>1</sup> University of California, Santa Cruz

<sup>2</sup> Ecole Polytechnique Federale de Lausanne

Technical report  
August 11, 2008

# Environment Assumptions for Synthesis

Krishnendu Chatterjee<sup>2</sup>, Thomas A. Henzinger<sup>1</sup>, and Barbara Jobstmann<sup>1</sup>

<sup>1</sup> EPFL, Lausanne

<sup>2</sup> University of California, Santa Cruz

**Abstract.** The synthesis problem asks to construct a reactive finite-state system from an  $\omega$ -regular specification. Initial specifications are often unrealizable, which means that there is no system that implements the specification. A common reason for unrealizability is that assumptions on the environment of the system are incomplete. We study the problem of correcting an unrealizable specification  $\varphi$  by computing an environment assumption  $\psi$  such that the new specification  $\psi \rightarrow \varphi$  is realizable. Our aim is to construct an assumption  $\psi$  that constrains only the environment and is as weak as possible. We present a two-step algorithm for computing assumptions. The algorithm operates on the game graph that is used to answer the realizability question. First, we compute a safety assumption that removes a minimal set of environment edges from the graph. Second, we compute a liveness assumption that puts fairness conditions on some of the remaining environment edges. We show that the problem of finding a minimal set of fair edges is computationally hard, and we use probabilistic games to compute a locally minimal fairness assumption.

## 1 Introduction

Model checking has become one of the most successful verification techniques in hardware and software design. Although the methods are automated, the success of a verification process highly depends on the quality of the specifications. Writing correct and complete specifications is a tedious task: it usually requires several iterations until a satisfactory specification is obtained. Specifications are often too weak (e.g., they may be vacuously satisfied [2, 14]); or too strong (e.g., they may allow too many environment behaviors), resulting in spurious counterexamples. In this work we automatically strengthen the environment constraints within specifications whose assumptions about the environment behavior are so weak as to make it impossible for a system to satisfy the specification.

Automatically deriving environment assumptions has been studied from several points of view. For instance, in circuit design one is interested in automatically constructing environment models that can be used in test-bench generation [21, 19]. In compositional verification, environment assumptions have been generated as the weakest input conditions under which a given software or hardware component satisfies a given specification [4, 7]. We follow a different path

by leaving the design completely out of the picture and deriving environment assumptions from the specification alone. Given a specification, we aim to compute a least restrictive environment that allows for an implementation of the specification.

The assumptions that we compute can assist the designer in different ways. They can be used as baseline necessary conditions in component-based model checking. They can be used in designing interfaces and generating test cases for components before the components themselves are implemented. They can provide insights into the given specification. And above all, in the process of automatically constructing an implementation for the given specification (“synthesis”), they can be used to correct the specification in a way that makes implementation possible.

While specifications of closed systems can be implemented if they are *satisfiable*, specifications of open systems can be implemented if they are *realizable*—i.e., there is a system that satisfies the specification without constraining the inputs. The key idea of our approach is that given a specification, if it is not realizable, cannot be complete and has to be weakened by introducing assumptions on the environment of the system. Formally, given an  $\omega$ -regular specification  $\varphi$  which is not realizable, we compute a condition  $\psi$  such that the new specification  $\psi \rightarrow \varphi$  is realizable. Our aim is to construct a condition  $\psi$  that does not constrain the system and is as weak as possible. The notion that  $\psi$  must constrain only the environment can be captured by requiring that  $\psi$  itself is realizable for the environment—i.e., there exists an environment that satisfies  $\psi$  without constraining the outputs of the system (in general, in a closed loop around system and environment—or controller and plant—both  $\psi$  and  $\varphi$  refer to inputs as well as outputs).

The notion that  $\psi$  be as weak as possible is more difficult to capture. We will show that in certain situations, there is no unique weakest environment-realizable assumption  $\psi$ , and in other situations, it is NP-hard to compute such an assumption.

**Example.** During our efforts of formally specifying certain hardware designs [5, 6], several unrealizable specifications were produced. One specification was particular difficult to analyze. Its structure can be simplified to the following example. Consider a reactive system with the signals `req`, `cancel`, and `grant`, where `grant` is the only output signal. The specification requires that (i) every request is eventually granted starting from the next time step, written in linear temporal logic as  $G(\text{req} \rightarrow X F \text{grant})$ ; and (ii) whenever the input `cancel` is received or `grant` is high, then `grant` has to stay low in the next time step, written  $G((\text{cancel} \vee \text{grant}) \rightarrow X \neg \text{grant})$ . This specification is not realizable because the environment can force, by sending `cancel` all the time, that the `grant` signal has to stay low forever (Part (ii)). If `grant` has to stay low, then a request cannot be answered and Part (i) of the specification is violated. One assumption that clearly makes this specification realizable is  $\psi_1 = G(\neg \text{cancel})$ . This assumption is undesirable because it completely forbids the environment to send `cancel`. A system synthesized with this assumption would simply ignore the signal `cancel`.

Assumption  $\psi_2 = G(F(\neg\text{cancel}))$  and  $\psi_3 = G(\text{req} \rightarrow F(\neg\text{cancel}))$  are more desirable but still not satisfactory:  $\psi_2$  forces the environment to lower `cancel` infinitely often even when no requests are sent and  $\psi_3$  is not strong enough to implement a system that in each step first produces an output and then reads the input: assume the system starts with output `grant` = 0 in time step 0, then receives the input `req` = 1 and `cancel` = 0, now in time step 1, it can choose between (a) `grant` = 1, or (b) `grant` = 0. If it chooses to set `grant` to high by (a), then the environment can provide the same inputs once more (`req` = 1 and `cancel` = 0) and can set all subsequent inputs to `req` = 0 and `cancel` = 1. Then the environment has satisfied  $\psi_3$  because during the two requests in time step 0 and 1 `cancel` was kept low but the system cannot fulfill Part (i) of its specification without violating Part (ii) due to `grant` = 1 in time step 1 and `cancel` = 1 afterwards. On the other hand, if the system decides to choose to set `grant` = 0 by (b), then the environment can choose to set the inputs to `req` = 0 and `cancel` = 1 and the system again fails to fulfill Part (i) without violating (ii). The assumption  $\psi_4 = G(\text{req} \rightarrow XF(\neg\text{cancel}))$ , which is a subset of  $\psi_3$ , is sufficient. However, there are infinitely many sufficient assumptions between  $\psi_3$  and  $\psi_4$ , e.g.,  $\psi'_3 = (\neg\text{cancel} \wedge X(\psi_3)) \vee \psi_3$ . The assumption  $\psi_5 = G(\text{req} \rightarrow XF(\neg\text{cancel} \vee \text{grant}))$  is also weaker than  $\psi_3$  and still sufficient because the environment only needs to lower `cancel` eventually if a request has not been answered yet. Finally, let  $\xi = \text{req} \rightarrow XF(\neg\text{cancel} \vee \text{grant})$ , consider the assumption  $\psi_6 = \xi W(\xi \wedge (\text{cancel} \vee \text{grant}) \wedge X\text{grant})$ , which is a sufficient assumption. It is desirable because it states that whenever a request is sent the environment has to eventually lower `cancel` if it has not seen a `grant`, but as soon as the system violates its specification (Part (ii)) all restrictions on the environment are dropped. If we replace  $\xi$  in  $\psi_6$  with  $\xi' = \text{req} \rightarrow F(\neg\text{cancel} \vee \text{grant})$ , we get again an assumption that is not sufficient for the specification to be realizable. This example shows that the notion of weakest and desirable are hard to capture.

**Contributions.** The realizability problem (and synthesis problem) can be reduced to emptiness checking for tree automata, or equivalently, to solving turn-based two-player games on graphs. More specifically, an  $\omega$ -regular specification  $\varphi$  is realizable iff there exists a winning strategy in a certain parity game constructed from  $\varphi$ . If  $\varphi$  is not realizable, then we construct an environment assumption  $\psi$  such that  $\psi \rightarrow \varphi$  is realizable, in two steps. First, we compute a safety assumption that removes a minimal set of environment edges from the graph. Second, we compute a liveness assumption that puts fairness conditions on some of the remaining environment edges of the game graph: if these edges can be chosen by the environment infinitely often, then they need to be chosen infinitely often. While the problem of finding a minimal set of fair edges is shown to be NP-hard, a local minimum can be found in polynomial time (in the size of the game graph) for Büchi and co-Büchi specifications, and in  $NP \cap \text{coNP}$  for parity specifications. The algorithm for checking the sufficiency of a set of fair edges is of independent theoretical interest, as it involves a novel reduction of deterministic parity games to probabilistic parity games.

We show that the resulting conjunction of safety and liveness assumptions is sufficient to make the specification realizable, and itself realizable by the environment. We also illustrate the algorithm on several examples, showing that it computes natural assumptions.

**Related works.** There are some related works that consider games that are not winning, methods of restricting the environment, and constructing most general winning strategies in games. The work of [11] considers games that are not winning, and considers *best-effort* strategies in such games. However, relaxing the winning objective to make the game winning is not considered. In [8], a notion of non-zero-sum game is proposed, where the strategies of the environment are restricted according to a given objective, but the paper does not study how to obtain an environment objective that is sufficient to transform the game to a winning one. A minimal assumption on a player with an objective can be captured by the most general winning strategy for the objective. The result of [3] shows that such most general winning strategies exist only for safety games, and also presents an approach to compute a strategy, called a *permissive strategy*, that subsumes behavior of all memoryless winning strategies. Our approach is different, as it attempts to construct the minimal assumption for the environment that makes the game winning, and we derive assumptions from the specification alone.

**Outline.** In Section 2, we introduce the necessary theoretical background for defining and computing environment assumptions. Section 3 discusses environment assumptions and why they are difficult to capture. In Section 4 and 5, we compute, respectively, safety and liveness assumptions, which are then combined in Section 6.

## 2 Preliminaries

**Words, Languages, Safety, and Liveness.** Given a finite alphabet  $\Sigma$  and an infinite word  $w \in \Sigma^\omega$ , we use  $w_i$  to denote the  $(i + 1)^{th}$  letter of  $w$ , and  $w^i$  to denote the finite prefix of  $w$  of length  $i + 1$ . Note that the first letter of a word has index 0. Given a word  $w \in \Sigma^\omega$ , we write  $\text{even}(w)$  for the subsequence of  $w$  consisting of the even positions ( $\forall i \geq 0 : \text{even}(w)_i = w_{2i}$ ). Similarly,  $\text{odd}(w)$  denotes the subsequence of the odd positions. Given a set  $L \subseteq \Sigma^\omega$  of infinite words, we define the set of finite prefixes by  $\text{prefixes}(L) = \{v \in \Sigma^* \mid \exists w \in L, i \geq 0 : v = w^i\}$ . Given a set  $L \subseteq \Sigma^*$  of finite words, we define the set of infinite limits by  $\text{safety}(L) = \{w \in \Sigma^\omega \mid \forall i \geq 0 : w^i \in L\}$ . We consider languages of infinite words. A language  $L \subseteq \Sigma^\omega$  is a *safety* language if  $L = \text{safety}(\text{prefixes}(L))$ . A language  $L \subseteq \Sigma^\omega$  is a *liveness* language if  $\text{prefixes}(L) = \Sigma^*$ . Every language  $L \subseteq \Sigma^\omega$  can be presented as the intersection of the safety language  $L_S = \text{safety}(\text{prefixes}(L))$  and the liveness language  $L_L = (\Sigma^\omega \setminus L_S) \cup L$  [1].

**Transducers.** We model reactive systems as deterministic finite-state transducers. We fix a finite set  $P$  of atomic propositions, and a partition of  $P$  into a set  $O$

of output propositions and a set  $I$  of input propositions. We use the corresponding alphabets  $\Sigma = 2^P$ ,  $\mathcal{O} = 2^O$ , and  $\mathcal{I} = 2^I$ . A *Moore transducer* with input alphabet  $\mathcal{I}$  and output alphabet  $\mathcal{O}$  is a tuple  $\mathcal{T} = (Q, q_I, \delta, \kappa)$ , where  $Q$  is a finite set of states,  $q_I \in Q$  is the initial state,  $\delta: Q \times \mathcal{I} \rightarrow Q$  is the transition function, and  $\kappa$  is a state labeling function  $\kappa: Q \rightarrow \mathcal{O}$ . A *Mealy transducer* is like a Moore transducer, except that  $\kappa: Q \times \mathcal{I} \rightarrow \mathcal{O}$  is a transition labeling function. A Moore transducer describes a reactive system that reads words over the alphabet  $\mathcal{I}$  and writes words over the alphabet  $\mathcal{O}$ . The environment of the system, in turn, can be described by a Mealy transducer with input alphabet  $\mathcal{O}$  and output alphabet  $\mathcal{I}$ . We extend the definition of the transition function  $\delta$  to finite words  $w \in \mathcal{I}^*$  inductively by  $\delta(q, w) = \delta(\delta(q, w^{|w|-1}), w_{|w|})$  for  $|w| > 0$ . Given an input word  $w \in \mathcal{I}^\omega$ , the run of  $\mathcal{T}$  over  $w$  is the infinite sequence  $\pi \in Q^\omega$  of states such that  $\pi_0 = q_I$ , and  $\pi_{i+1} = \delta(\pi_i, w_i)$  for all  $i \geq 0$ . The run  $\pi$  over  $w$  generates the infinite word  $\mathcal{T}(w) \in \Sigma^\omega$  defined by  $\mathcal{T}(w)_i = \kappa(\pi_i) \cup w_i$  for all  $i \geq 0$  in the case of Moore transducers; and  $\mathcal{T}(w)_i = \kappa(\pi_i, w_i) \cup w_i$  for all  $i \geq 0$  in the Mealy case. The *language* of the transducer  $\mathcal{T}$  is the set  $L(\mathcal{T}) = \{\mathcal{T}(w) \mid w \in \mathcal{I}^\omega\}$  of infinite words generated by runs of  $\mathcal{T}$ .

**Specifications and Realizability.** A *specification* of a reactive system is an  $\omega$ -regular language  $L \subseteq \Sigma^\omega$ . We use Linear Temporal Logic (LTL) formulae over the atomic proposition  $P$ , as well as  $\omega$ -automata with transition labels from  $\Sigma$ , to define specifications. Given an LTL formula (resp.  $\omega$ -automaton)  $\phi$ , we write  $L(\phi) \subseteq \Sigma^\omega$  for the set of infinite words that satisfy (resp. are accepted by)  $\phi$ . A transducer  $\mathcal{T}$  *satisfies* a specification  $L(\phi)$ , written  $\mathcal{T} \models \phi$ , if  $L(\mathcal{T}) \subseteq L(\phi)$ . Given an LTL formula (resp.  $\omega$ -automaton)  $\phi$ , the *realizability problem* asks if there exists a transducer  $\mathcal{T}$  with input alphabet  $\mathcal{I}$  and output alphabet  $\mathcal{O}$  such that  $\mathcal{T} \models \phi$ . The specification  $L(\phi)$  is *Moore realizable* if such a Moore transducer  $\mathcal{T}$  exists, and *Mealy realizable* if such a Mealy transducer  $\mathcal{T}$  exists. Note that for an LTL formula, the specification  $L(\phi)$  is Mealy realizable iff  $L(\phi')$  is Moore realizable, where the LTL formula  $\phi'$  is obtained from  $\phi$  by replacing all occurrences of  $o \in O$  by  $Xo$ . The process of constructing a suitable transducer  $\mathcal{T}$  is called *synthesis*. The synthesis problem can be solved by computing winning strategies in graph games.

**Graph games.** We consider two classes of turn-based games on graphs, namely, two-player probabilistic games and two-player deterministic games. The probabilistic games are not needed for synthesis, but we will use them for constructing environment assumptions. For a finite set  $A$ , a probability distribution on  $A$  is a function  $\delta: A \rightarrow [0, 1]$  such that  $\sum_{a \in A} \delta(a) = 1$ . We denote the set of probability distributions on  $A$  by  $\mathcal{D}(A)$ . Given a distribution  $\delta \in \mathcal{D}(A)$ , we write  $\text{Supp}(\delta) = \{x \in A \mid \delta(x) > 0\}$  for the support of  $\delta$ . A *probabilistic game graph*  $G = ((S, E), (S_1, S_2, S_P), \delta)$  consists of a finite directed graph  $(S, E)$ , a partition  $(S_1, S_2, S_P)$  of the set  $S$  of states, and a probabilistic transition function  $\delta: S_P \rightarrow \mathcal{D}(S)$ . The states in  $S_1$  are *player-1* states, where player 1 decides the successor state; the states in  $S_2$  are *player-2* states, where player 2 decides the successor state; and the states in  $S_P$  are *probabilistic* states, where the successor state is chosen according to the probabilistic transition function. We require that

for all  $s \in S_P$  and  $t \in S$ , we have  $(s, t) \in E$  iff  $\delta(s)(t) > 0$ , and we often write  $\delta(s, t)$  for  $\delta(s)(t)$ . For technical convenience we also require that every state has at least one outgoing edge. Given a subset  $E' \subseteq E$  of edges, we write  $\text{Source}(E')$  for the set  $\{s \in S \mid \exists t \in S : (s, t) \in E'\}$  of states that have an outgoing edge in  $E'$ . The *deterministic game graphs* are the special case of the probabilistic game graphs with  $S_P = \emptyset$ , that is, a deterministic game graph  $G = ((S, E), (S_1, S_2))$  consist of of a directed graph  $(S, E)$  together with the partition of the state space  $S$  into player-1 states  $S_1$  and player-2 states  $S_2$ .

**Plays and Strategies.** An infinite path, or *play*, of the game graph  $G$  is an infinite sequence  $\pi = s_0 s_1 s_2 \dots$  of states such that  $(s_k, s_{k+1}) \in E$  for all  $k \geq 0$ . We write  $\Pi$  for the set of plays, and for a state  $s \in S$ , we write  $\Pi_s \subseteq \Pi$  for the set of plays that start from  $s$ . A *strategy* for player 1 is a function  $\alpha: S^* \cdot S_1 \rightarrow S$  that for all finite sequences of states ending in a player-1 state (the sequence represents a prefix of a play), chooses a successor state to extend the play. A strategy must prescribe only available moves, that is,  $\alpha(\tau \cdot s) \in E(s)$  for all  $\tau \in S^*$  and  $s \in S_1$ . The strategies for player 2 are defined analogously. Note that we have only pure (i.e., nonprobabilistic) strategies. We denote by  $\mathcal{A}$  and  $\mathcal{B}$  the set of strategies for player 1 and player 2, respectively. A strategy  $\alpha$  is *memoryless* if it does not depend on the history of the play but only on the current state. A memoryless player-1 strategy can be represented as a function  $\alpha: S_1 \rightarrow S$ , and a memoryless player-2 strategy is a function  $\beta: S_2 \rightarrow S$ . We denote by  $\mathcal{A}^M$  and  $\mathcal{B}^M$  the set of memoryless strategies for player 1 and player 2, respectively.

Once a starting state  $s \in S$  and strategies  $\alpha \in \mathcal{A}$  and  $\beta \in \mathcal{B}$  for the two players are fixed, the outcome of the game is a random walk  $\pi_s^{\alpha, \beta}$  for which the probabilities of events are uniquely defined, where an *event*  $\mathcal{E} \subseteq \Pi$  is a measurable set of plays. Given strategies  $\alpha$  for player 1 and  $\beta$  for player 2, a play  $\pi = s_0 s_1 s_2 \dots$  is *feasible* if for all  $k \geq 0$ , we have  $\alpha(s_0 s_1 \dots s_k) = s_{k+1}$  if  $s_k \in S_1$ , and  $\beta(s_0 s_1 \dots s_k) = s_{k+1}$  if  $s_k \in S_2$ . Given two strategies  $\alpha \in \mathcal{A}$  and  $\beta \in \mathcal{B}$ , and a state  $s \in S$ , we denote by  $\text{Outcome}(s, \alpha, \beta) \subseteq \Pi_s$  the set of feasible plays that start from  $s$ . Note that for deterministic game graphs, the set  $\text{Outcome}(s, \alpha, \beta)$  contains a single play. For a state  $s \in S$  and an event  $\mathcal{E} \subseteq \Pi$ , we write  $\Pr_s^{\alpha, \beta}(\mathcal{E})$  for the probability that a play belongs to  $\mathcal{E}$  if the game starts from the state  $s$  and the two players follow the strategies  $\alpha$  and  $\beta$ , respectively.

**Objectives.** An *objective* for a player is a set  $\Phi \subseteq \Pi$  of winning plays. We consider  $\omega$ -regular sets of winning plays, which are measurable. For a play  $\pi = s_0 s_1 s_2 \dots$ , let  $\text{Inf}(\pi)$  be the set  $\{s \in S \mid s = s_k \text{ for infinitely many } k \geq 0\}$  of states that appear infinitely often in  $\pi$ .

1. *Reachability and safety objectives.* Given a set  $F \subseteq S$  of states, the reachability objective  $\text{Reach}(F)$  requires that some state in  $F$  be visited, and dually, the safety objective  $\text{Safe}(F)$  requires that only states in  $F$  be visited. Formally, the sets of winning plays are  $\text{Reach}(F) = \{s_0 s_1 s_2 \dots \in \Pi \mid \exists k \geq 0 : s_k \in F\}$  and  $\text{Safe}(F) = \{s_0 s_1 s_2 \dots \in \Pi \mid \forall k \geq 0 : s_k \in F\}$ .
2. *Büchi and co-Büchi objectives.* Given a set  $F \subseteq S$  of states, the Büchi objective  $\text{Buchi}(F)$  requires that some state in  $F$  be visited infinitely often, and

dually, the co-Büchi objective  $\text{coBüchi}(F)$  requires that only states in  $F$  be visited infinitely often. Thus, the sets of winning plays are  $\text{Büchi}(F) = \{\pi \in \Pi \mid \text{Inf}(\pi) \cap F \neq \emptyset\}$  and  $\text{coBüchi}(F) = \{\pi \in \Pi \mid \text{Inf}(\pi) \subseteq F\}$ .

3. *Parity objectives.* Given a function  $p: S \rightarrow \{0, 1, 2, \dots, d-1\}$  that maps every state to a *priority*, the parity objective  $\text{Parity}(p)$  requires that of the states that are visited infinitely often, the least priority be even. Formally, the set of winning plays is  $\text{Parity}(p) = \{\pi \in \Pi \mid \min\{p(\text{Inf}(\pi))\} \text{ is even}\}$ . The dual, co-parity objective has the set  $\text{coParity}(p) = \{\pi \in \Pi \mid \min\{p(\text{Inf}(\pi))\} \text{ is odd}\}$  of winning plays.

The parity objectives are closed under complementation: given a function  $p: S \rightarrow \{0, 1, \dots, d-1\}$ , consider the function  $p+1: S \rightarrow \{1, 2, \dots, d\}$  defined by  $p+1(s) = p(s) + 1$  for all  $s \in S$ ; then  $\text{Parity}(p+1) = \text{coParity}(p)$ . The Büchi and co-Büchi objectives are special cases of parity objectives with two priorities, namely,  $p: S \rightarrow \{0, 1\}$  for Büchi objectives with  $F = p^{-1}(0)$ , and  $p: S \rightarrow \{1, 2\}$  for co-Büchi objectives with  $F = p^{-1}(2)$ . The reachability and safety objectives can be turned into Büchi and co-Büchi objectives, respectively, on slightly modified game graphs.

**Sure and Almost-Sure Winning.** Given an objective  $\Phi$ , a strategy  $\alpha \in \mathcal{A}$  is *sure winning* for player 1 from a state  $s \in S$  if for every strategy  $\beta \in \mathcal{B}$  for player 2, we have  $\text{Outcome}(s, \alpha, \beta) \subseteq \Phi$ . The strategy  $\alpha$  is *almost-sure winning* for player 1 from  $s$  for  $\Phi$  if for every player-2 strategy  $\beta$ , we have  $\text{Pr}_s^{\alpha, \beta}(\Phi) = 1$ . The sure and almost-sure winning strategies for player 2 are defined analogously. Given an objective  $\Phi$ , the *sure winning set*  $\langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi)$  for player 1 is the set of states from which player 1 has a sure winning strategy. Similarly, the *almost-sure winning set*  $\langle\langle 1 \rangle\rangle_{\text{almost}}(\Phi)$  for player 1 is the set of states from which player 1 has an almost-sure winning strategy. The winning sets  $\langle\langle 2 \rangle\rangle_{\text{sure}}(\Phi)$  and  $\langle\langle 2 \rangle\rangle_{\text{almost}}(\Phi)$  for player 2 are defined analogously. It follows from the definitions that for all probabilistic game graphs and all objectives  $\Phi$ , we have  $\langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi) \subseteq \langle\langle 1 \rangle\rangle_{\text{almost}}(\Phi)$ . In general the subset inclusion relation is strict. For deterministic games the notions of sure and almost-sure winning coincide [15], that is, for all deterministic game graphs and all objectives  $\Phi$ , we have  $\langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi) = \langle\langle 1 \rangle\rangle_{\text{almost}}(\Phi)$ , and in such cases we often omit the subscript. Given an objective  $\Phi$ , the *cooperative winning set*  $\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$  is the set of states  $s$  for which there exist a player-1 strategy  $\alpha$  and a player-2 strategy  $\beta$  such that  $\text{Outcome}(s, \alpha, \beta) \subseteq \Phi$ .

**Theorem 1 (Deterministic games [10]).** *For all deterministic game graphs and parity objectives  $\Phi$ , the following assertions hold: (i)  $\langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi) = S \setminus \langle\langle 2 \rangle\rangle_{\text{sure}}(\Pi \setminus \Phi)$ ; (ii) memoryless sure winning strategies exist for both players from their sure winning sets; and (iii) given a state  $s \in S$ , whether  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi)$  can be decided in  $NP \cap \text{coNP}$ .*

**Theorem 2 (Probabilistic games [9]).** *Given a probabilistic game graph  $G = ((S, E), (S_1, S_2, S_P), \delta)$  and a parity objective  $\Phi$  with  $d$  priorities, we can construct a deterministic game graph  $\widehat{G} = ((\widehat{S}, \widehat{E}), (\widehat{S}_1, \widehat{S}_2))$  with  $S \subseteq \widehat{S}$ ,*

and a parity objective  $\widehat{\Phi}$  with  $d + 1$  priorities such that (i)  $|\widehat{S}| = O(|S| \cdot d)$  and  $|\widehat{E}| = O(|E| \cdot d)$ ; and (ii) the set  $\langle\langle 1 \rangle\rangle_{\text{almost}}(\widehat{\Phi})$  in  $G$  is equal to the set  $\langle\langle 1 \rangle\rangle_{\text{sure}}(\widehat{\Phi}) \cap S$  in  $\widehat{G}$ . Moreover, memoryless almost-sure winning strategies exist for both players from their almost-sure winning sets in  $G$ .

**Realizability Games.** The realizability problem has the following game-theoretic formulation.

**Theorem 3 (Reactive synthesis [17]).** *Given an LTL formula (resp.  $\omega$ -automaton)  $\varphi$ , we can construct a deterministic game graph  $G$ , a state  $s_I$  of  $G$ , and a parity objective  $\Phi$  such that  $L(\varphi)$  is realizable iff  $s_I \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi)$ .*

The deterministic game graph  $G$  with parity objective  $\varphi$  referred to in Theorem 3 is called a *synthesis game* for  $\varphi$ . Starting from an LTL formula  $\varphi$ , we construct the synthesis game by first building a nondeterministic Büchi automaton that accepts  $L(\varphi)$  [20]. Then, following the algorithm of [16], we translate this automaton to a deterministic parity automaton that accepts  $L(\varphi)$ . By splitting every state of the parity automaton w.r.t. inputs  $I$  and outputs  $O$ , we obtain the synthesis game. Both steps involve an exponential blowup that is unavoidable: for LTL formulae  $\varphi$ , the realizability problem is 2EXPTIME-complete [18].

Synthesis games, by relating paths in the game graph to the specification  $\varphi$ , have the following special form. A *synthesis game*  $\mathcal{G}$  is a tuple  $(G, s_I, \lambda, \Phi)$ , where  $G = ((S, E), (S_1, S_2))$  is a deterministic bipartite game graph, in which player-1 and player-2 states strictly alternate (i.e.,  $E \subseteq (S_1 \times S_2) \cup (S_2 \times S_1)$ ), the initial state  $s_I \in S_1$  is a player-1 state, the labeling function  $\lambda: S \rightarrow \mathcal{O} \cup \mathcal{I}$  maps player-1 and player-2 states to letters in  $\mathcal{I}$  and  $\mathcal{O}$ , respectively (i.e.,  $\lambda(s) \in \mathcal{I}$  for all  $s \in S_1$ , and  $\lambda(s) \in \mathcal{O}$  for all  $s \in S_2$ ), and  $\Phi$  is a parity objective. Furthermore, synthesis games are deterministic w.r.t. input and output labels, that is, for all edges  $(s, s'), (s, s'') \in E$ , if  $\lambda(s') = \lambda(s'')$ , then  $s' = s''$ . Without loss of generality, we assume that synthesis games are complete w.r.t. input and output labels, that is, for all state  $s \in S_1$  ( $S_2$ ) and  $l \in \mathcal{O}$  ( $\mathcal{I}$ , respectively), there exists an edge  $(s, s') \in E$  such that  $\lambda(s') = l$ . We define a function  $w: \Pi \rightarrow \Sigma^\omega$  that maps each play to an infinite word such that  $w_i = \lambda(\pi_{2i+1}) \cup \lambda(\pi_{2i+2})$  for all  $i \geq 0$ . Note that we ignore the label of the initial state. Given a synthesis game  $\mathcal{G}$  for a specification formula or automaton  $\varphi$ , every Moore transducer  $\mathcal{T} = (Q, q_I, \delta, \kappa)$  that satisfies  $L(\varphi)$  represents a winning strategy  $\alpha$  of player 1 as follows: for all sequences  $\tau \in (S_1 S_2)^* \cdot S_1$ , let  $w$  be the finite word such that  $w_i = \lambda(\tau_{i+1})$  for all  $0 \leq i < |\tau|$ ; then, if there exists an edge  $(\tau_{|\tau|}, s') \in E$  with  $\lambda(s') = \kappa(\delta(q_I, \text{odd}(w)))$ , then  $\alpha(\tau) = s'$ , and otherwise  $\alpha(\tau)$  is arbitrary. Conversely, every memoryless winning strategy  $\alpha$  of player 1 represents a Moore transducer  $\mathcal{T} = (Q, q_I, \delta, \kappa)$  that satisfies  $L(\varphi)$  as follows:  $Q = S_1$ ,  $q_I = s_I$ ,  $\kappa(q) = \lambda(\alpha(q))$ , and  $\delta(q, l) = s'$  if  $\lambda(s') = l$  and  $(\alpha(q), s') \in E$ .

### 3 Assumptions

In this section, we discuss about environment assumptions in general, illustrating through several simple examples, and then identify conditions that every assumption has to satisfy.

Given a specification  $\varphi$  that describes the desired behavior of an open system  $\mathcal{S}$ , we search for assumptions on the environment of  $\mathcal{S}$  that are sufficient to ensure that  $\mathcal{S}$  exists and satisfies  $\varphi$ . The assumptions we study are independent of the actual implementation. They are derived from the given specification and can be seen as part of a correct specification. We first define what it means for an assumption to be sufficient.

Let  $\varphi$  be a specification. A language  $\psi \subseteq \Sigma$  is a *sufficient environment assumption* for  $\varphi$  if  $(\Sigma^\omega \setminus \psi) \cup \varphi$  is realizable.

*Example 1.* Consider the specification  $\varphi = \text{out} \cup \text{in}$ . There exists no system  $\mathcal{S}$  with input  $\text{in}$  and output  $\text{out}$  such that  $\mathcal{S} \models \varphi$ , because  $\mathcal{S}$  cannot control the value of  $\text{in}$  and  $\varphi$  is satisfied only if  $\text{in}$  eventually becomes true. We have to weaken the specification to make it realizable. A candidate  $\psi$  for the assumption is  $\text{F in}$  because it forces the environment to assert the signal  $\text{in}$  eventually, which allows the system to fulfill  $\varphi$ . Further candidates are  $\text{false}$ , which makes the specification trivially realizable,  $\text{X in}$ , which forces the environment to assert the signal  $\text{in}$  in the second step,  $\text{F out}$ , or  $\text{F } \neg \text{out}$ . The last two assumptions lead to new specifications of the following form  $\varphi' = \psi \rightarrow \varphi = \text{F out} \rightarrow \varphi = \text{G}(\neg \text{out}) \vee \varphi$ . The system can implement  $\varphi'$  independent of  $\varphi$  simply by keeping  $\text{out}$  low all the time.

Example 1 shows that there are several assumptions that allow to implement the specification but not all of them are satisfactory. For example, the assumption  $\text{false}$  does not provide the desired information. Similarly, the assumption  $\text{F out}$  is not satisfactory, because it cannot be satisfied by any environment that controls  $\text{in}$ . Intuitively, assumptions that are false or that can be falsified by the system correspond to a new specification  $\psi \rightarrow \varphi$  that can be satisfied vacuously [2, 14] by the system. In order to exclude those assumptions, we require that an assumption fulfills the following condition:

- (1) *Realizable for the environment:* The system cannot trivially falsify the assumption, so there exists an implementation of the environment that satisfies  $\psi$ . Formally,  $\psi$  is Mealy realizable<sup>3</sup>.

Note that Condition 1 induces that  $\varphi$  has to be satisfiable for  $\psi$  to exist. If  $\varphi$  is not satisfiable there exists only the trivially solution  $\psi = \text{false}$ . We assume from now on that  $\psi$  is satisfiable. Apart from Condition 1, we ask for a condition to compare or order different assumptions. We aim to restrict the environment “as little as possible”. An obvious candidate for this order is language inclusion:

---

<sup>3</sup> Note that we ask here for a Mealy transducer, since the system is a Moore transducer.

- (2) *Minimum*: An assumption  $\psi$  is minimal if there exists no other sufficient assumption that includes  $\psi$ . There is no language  $\psi' \subseteq \Sigma$  such that  $\psi \subset \psi'$  and  $(\Sigma^\omega \setminus \psi') \cup \varphi$  is realizable.

The following example shows that using language inclusion we cannot ask for a unique minimal assumption.

*Example 2.* Consider the specification  $\varphi = (\text{out} \text{ U } \text{in}_1) \vee (\neg \text{out} \text{ U } \text{in}_2)$ , where  $\text{in}_1$  and  $\text{in}_2$  are inputs and  $\text{out}$  is an output. Again,  $\varphi$  is not realizable. Consider the assumptions  $\psi_1 = \text{F in}_1$  and  $\psi_2 = \text{F in}_2$ . Both are sufficient because assuming  $\psi_1$  the system can keep the signal  $\text{out}$  constantly high and assuming  $\psi_2$  it can keep  $\text{out}$  constantly low. However, if we assume the disjunction  $\psi = \psi_1 \vee \psi_2$ , the system does not know, which of the signals  $\text{in}_1$  and  $\text{in}_2$  the environment is going to assert eventually. Since a unique minimal assumption has to subsume all other sufficient assumptions and  $\psi$  is not sufficient, it follows that there exists no unique minimal assumption that is sufficient.

Let us consider another example to illustrate the difficulties that arise when comparing environment assumptions w.r.t. language inclusion.

*Example 3.* Assume the specification  $\varphi = \text{G}(\text{in} \rightarrow \text{X out}) \wedge \text{G}(\text{out} \rightarrow (\text{X } \neg \text{out}))$  with input signal  $\text{in}$  and output signal  $\text{out}$ . The specification is not realizable because whenever  $\text{in}$  is set to true in two consecutive steps, the system cannot produce a value for  $\text{out}$  such that  $\varphi$  is satisfied. One natural assumption is  $\psi = \text{G}(\text{in} \rightarrow \text{X } \neg \text{in})$ . Another assumption is  $\psi' = \psi \vee \text{F}(\neg \text{in} \wedge \text{X out})$ , which is weaker than  $\psi$  w.r.t. language inclusion and still realizable. Looking at the resulting system specification  $\psi' \rightarrow \varphi = (\psi \vee \text{F}(\neg \text{in} \wedge \text{X out})) \rightarrow \varphi = \psi \rightarrow (\text{G}(\neg \text{in} \rightarrow \text{X } \neg \text{out}) \wedge \varphi)$ , we see that  $\psi'$  restricts the system instead of the environment.

Intuitively, using language inclusion as ordering notion, results in minimal environment assumptions that allow only a single implementation for the system. We aim for an assumption that does not restrict the system if possible. One may argue that  $\psi$  should talk only about input signals. Let us consider the specification of Example 3 once more. Another sufficient assumption is  $\psi'' = (\text{in} \rightarrow \text{X } \neg \text{in}) \text{ W}(\text{out} \wedge \text{X out})$ , which is weaker than  $\psi$ . Intuitively,  $\psi''$  requires that the environment guarantees  $(\text{in} \rightarrow \text{X } \neg \text{in})$  as long as the system did not make a mistake (by setting  $\text{out}$  to true in two consecutive steps), which clearly means the intuition of an environment assumption. The challenge is to find an assumption that (a) is sufficient, (b) does not restrict the system, and (c) gives the environment maximal freedom.

Note that the assumptions  $\psi$  and  $\psi''$  are safety assumptions, while the assumptions in Example 2 are liveness assumptions. In general, every language can be split into a safety and a liveness component. We use this separation to provide a way to compute environment assumption that fulfills our criteria.

We consider restriction on game graphs of synthesis games to find sufficient environment assumptions. More precisely, we propose to put restrictions on player-2 edges, because they correspond to decisions the environment can make. If the given specification is satisfiable, this choice of restrictions leads to assumptions that fulfill the realizability for the environment.

## 4 Safety Assumptions

In this section, we define and compute an assumption that restricts the safety behavior of the environment.

### 4.1 Non-restrictive Safety Assumption on Games

Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$  and the winning objective  $\Phi$  for player 1. A *safety assumption* on the set  $E_s \subseteq E_2$  of edges requires that player 2 chooses only edges outside from  $E_s$ . A synthesis game  $\mathcal{G} = (G, s_I, \lambda)$  with a safety assumption on  $E_s$  defines an environment assumption  $\psi_{E_s}$  as the set of words  $w \in \Sigma^\omega$  such that there exists a play  $\pi \in \Pi_{s_I}$  with  $w = w(\pi)$ , where for all  $i \geq 0$ , we have  $(\pi_i, \pi_{i+1}) \notin E_s$ .

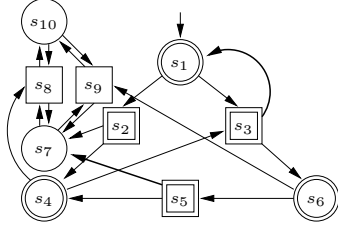
The set  $E_s$  can be seen as a set of forbidden edges of player 2. A natural order on safety assumptions is the number of edges in a safety assumption. We write  $E_s \leq E_{s'}$  if  $|E_s| \leq |E_{s'}|$  holds. A safety assumption refers to the safety component of a winning objective, which can be formulated as  $\Phi_S = \text{Safe}(\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi))$ . Formally, the winning objective of player 1 is modified to  $\text{AssumeSafe}(E_s, \Phi) = \{\pi = s_0 s_1 s_2 \dots \mid \text{either (i) there exists } i \geq 0 \text{ such that } (s_i, s_{i+1}) \in E_s, \text{ or (ii) } \pi \in \Phi_S\}$  denoting the set of all plays in which either one of the edges in  $E_s$  is chosen, or that satisfies the safety component of  $\Phi$ .

Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$ , a winning objective  $\Phi$  for player 1, and a safety assumption  $E_s$ , the safety assumption on  $E_s$  is *safe-sufficient for state*  $s \in S$  and  $\Phi$  if player 1 has a winning strategy from  $s$  for the objective  $\text{AssumeSafe}(E_s, \Phi)$ .

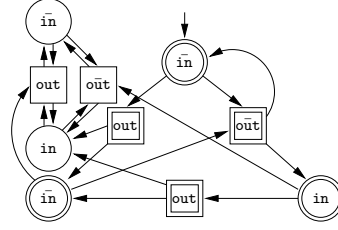
**Theorem 4.** *Let  $\varphi$  be a specification and let  $\mathcal{G}_\varphi = (G, s_I, \lambda)$  be a synthesis game for  $\varphi$  with the winning objective  $\Phi$ . An environment assumption  $\psi_{E_s}$  defined by a safety assumption  $E_s$  on  $\mathcal{G}_\varphi$  that is safe-sufficient for  $s_I$  and  $\Phi$ , is sufficient for  $\varphi' = \text{safety}(\text{prefixes}(L(\varphi)))$ . Note that if  $\varphi$  is a safety language, then  $\psi_{E_s}$  is sufficient for  $\varphi$ .*

*Proof.* Since  $E_s$  is safe-sufficient for  $s_I$  and  $\Phi$ , player 1 has a memoryless winning strategy  $\alpha$  for  $s_I$  and  $\text{AssumeSafe}(E_s, \Phi)$ . We know from Theorem 3 that  $\alpha$  corresponds to a transducer  $\mathcal{T}$ . We need to show that the language of the transducer  $L(\mathcal{T})$  is a subset of the new specification  $(\Sigma \setminus \psi_{E_s}) \cup \text{safety}(\text{prefixes}(L(\varphi)))$ . A run of  $\mathcal{T}$  on a word  $w \in \Sigma^\omega$  corresponds to a winning play  $\pi \in \text{AssumeSafe}(E_s, \Phi)$  of  $\mathcal{G}_\varphi$ . A play  $\pi' = s_0 s_1 \dots \in \text{AssumeSafe}(E_s, \Phi)$  either has an edge  $(s_i, s_{i+1}) \in E_s$ , then  $w(\pi') \in (\Sigma \setminus \psi_{E_s})$ , or  $\pi' \in \text{Safe}(\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi))$ , then we have that  $w(\pi') \in \text{safety}(\text{prefixes}(L(\varphi)))$ .  $\square$

In the following example, we show that there exist safety games such that for some state  $s$  there is no unique smallest assumption that is safe-sufficient for  $s$ .



**Fig. 1.** Game with two equally small safe-sufficient assumptions for  $s_1$ :  $E_s = \{(s_3, s_1)\}$  and  $E_s' = \{(s_5, s_7)\}$ .



**Fig. 2.** Synthesis game for  $G(\text{in} \rightarrow \text{X out}) \wedge G(\text{out} \rightarrow \text{X } \neg\text{out})$ .

*Example 4.* Consider the game shown in Figure 1. Circles denote states of player 1, boxes denote states of player 2. The winning objective of player 1 is to stay in the set  $\{s_1, s_2, s_3, s_4, s_5, s_6\}$  denoted by double lines. Player 1 has no winning strategy for  $s_1$ . There are two equally small safety assumptions that are safe-sufficient for  $s_1$ :  $E_s = \{(s_3, s_1)\}$  and  $E_s' = \{(s_5, s_7)\}$ . In both cases, player 1 has a winning strategy from state  $s_1$ .

If we consider a specification, where the corresponding synthesis game has this structure, neither of these assumptions is satisfactory. Figure 2 shows such a synthesis game for the specification  $G(\text{in} \rightarrow \text{X out}) \wedge G(\text{out} \rightarrow \text{X } \neg\text{out})$  with input signal  $\text{in}$  and output signal  $\text{out}$  (cf. Example 3). Assuming the safety assumption  $E_s$ , the corrected specification would allow only the single implementation, where  $\text{out}$  is constantly keep low. The second assumption  $E_s'$  leads to a corrected specification that additionally enforces  $G(\neg\text{in} \rightarrow \text{X } \neg\text{out})$ .

Besides safe-sufficient, we also ask for an assumption that does not restrict player 1. This condition can be formulated as follows. Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$ , a winning objective  $\Phi$  for player 1, and a safety assumption  $E_s$ . We call the safety assumption on  $E_s$  *restrictive* for state  $s \in S$  and  $\Phi$ , if there exist strategies  $\alpha \in \mathcal{A}$  and  $\beta \in \mathcal{B}$  of player 1 and 2, respectively such that the play  $\text{Outcome}(s, \alpha, \beta)$  contains an edge from  $E_s$  and is in  $\Phi_S$ . A non-restrictive safety assumption should allow any edge that does not lead to an immediate violation of the safety component of the winning objective of player 1.

**Theorem 5.** *Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$ , a winning objective  $\Phi$  for player 1, and a state  $s \in S$ , if  $s \in \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$ , then there exists a unique minimal safety assumption  $E_s$  that is non-restrictive and safe-sufficient for state  $s$  and  $\Phi$ . Let  $s \in \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$  and let  $E_s$  be this unique minimal safety assumption for  $s$  and  $\Phi$ , then player 2 has winning strategy for  $s$  and the objective to avoid the edges in  $E_s$ .*

Applying this theorem to environment assumptions, we get the following theorem.

**Theorem 6.** *Let  $\varphi$  be a satisfiable specification and let  $\mathcal{G}_\varphi = (G, s_I, \lambda)$  be a synthesis game for  $\varphi$  with winning objective  $\Phi$ , then there exists a unique minimal safety assumption  $E_s$  that is non-restrictive and safe-sufficient for state  $s$*

and  $\Phi$  and the corresponding environment assumption  $\psi_{E_s}$  is realizable for the environment.

## 4.2 Computing Non-restrictive Safety Assumptions

Given a deterministic game graph  $G$  and a winning objective  $\Phi$ , we compute a non-restrictive safety assumption  $E_s$  as follows: first, we compute the set  $\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$ . Note that for this set the players cooperate. We can compute  $\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$  in polynomial time for all objectives we consider. In particular, if  $\Phi$  is a parity condition,  $\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$  can be computed by reduction to Büchi [13]. The safety assumption  $E_s$  is the set of all player-2 edges  $(s, t) \in E_2$  such that  $s \in \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$  and  $t \notin \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$ .

**Theorem 7.** *Consider a deterministic game graph  $G$  with a winning objective  $\Phi$ . The safety assumption  $E_s = \{(s, t) \in E_2 \mid s \in \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi) \text{ and } t \notin \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)\}$  is the unique minimal safety assumption that is non-restrictive and safe-sufficient for all states  $s \in \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$ . The set  $E_s$  can be computed in polynomial time for all parity objectives  $\Phi$ .*

For the game show in Figure 1, we obtain the safety assumption  $E_s = \{(s_3, s_1), (s_5, s_7)\}$ . For the corresponding synthesis game in Figure 2,  $E_s$  defines the environment assumption  $\psi_{E_s} = (\neg \text{in} \vee \neg \text{out}) \mathbf{W}((\neg \text{in} \vee \neg \text{out}) \wedge (\text{in} \wedge \mathbf{X}(\neg \text{out})) \wedge (\text{out} \wedge \mathbf{X}(\text{out})))$ . This safety assumption meets our intuition of a minimal environment assumption, since it states that the environment has to ensure that either **in** or **out** is low as long as the system makes no obvious fault by either violating  $\mathbf{G}(\text{in} \rightarrow \mathbf{X} \text{out})$  or  $\mathbf{G}(\text{out} \rightarrow \mathbf{X} \neg \text{out})$ .

## 5 Liveness Assumptions

### 5.1 Strongly Fair Assumptions on Games

Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$  and the objective  $\Phi$  for player 1, a *strongly fair* assumption is a set  $E_l \subseteq E_2$  of edges requiring that player 2 plays in a way such that if a state  $s \in \text{Source}(E_l)$  is visited infinitely often, then for all  $t \in S$  such that  $(s, t) \in E_l$ , the edge  $(s, t)$  is chosen infinitely often.

This notion is formalized by modifying the objective of player 1 as follows. Let  $\text{AssumeFair}(E_l, \Phi) = \{\pi = s_0 s_1 s_2 \dots \mid \text{either (i) } \exists (s, t) \in E_l, \text{ such that } s_k = s \text{ for infinitely many } k\text{'s, and } s_j = s \text{ and } s_{j+1} = t \text{ for finitely many } j\text{'s, or (ii) } \pi \in \Phi\}$  denote the set of paths  $\pi$  such that either (i) there is a state  $s \in \text{Source}(E_l)$  that appears infinitely often in  $\pi$  and there is a  $(s, t) \in E_l$  and  $t$  appears only finitely often in  $\pi$ , or (ii)  $\pi$  belongs to the objective  $\Phi$ . In other words, part (i) specifies that the strong fairness assumption on  $E_l$  is violated, and part (ii) specifies that  $\Phi$  is satisfied. The property that player 1 can ensure  $\Phi$  against player-2 strategies respecting the strongly fair assumption on edges is

formalized by requiring that player 1 can satisfy  $\text{AssumeFair}(E_l, \Phi)$  against all player-2 strategies.

Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$  and the objective  $\Phi$  for player 1, a strongly fair assumption on  $E_l \subseteq E_2$  is *sufficient* for state  $s \in S$  and  $\Phi$ , if player 1 has a winning strategy for  $s$  for the objective  $\text{AssumeFair}(E_l, \Phi)$ . Furthermore, given a deterministic game graph  $G = ((S, E), (S_1, S_2))$  and the objective  $\Phi$  for player 1, a state  $s \in S$  is *live for player 1*, if she has a winning strategy from  $s$  for the winning objective  $\text{Safe}(\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi))$ .

**Theorem 8.** *Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$  and a Reachability, Safety, or Büchi objective  $\Phi$ . If  $s \in S$  is live for player 1, then there exists a strongly fair assumption  $E_l \subseteq E_2$  that is sufficient for state  $s \in S$  and  $\Phi$ .*

A synthesis game  $\mathcal{G} = (G, s_I, \lambda)$  with a strongly fair assumption on  $E_l$  defines an environment assumption  $\psi_{E_l}$  as the set of words  $w \in \Sigma^\omega$  such that there exists a play  $\pi \in \Pi_{s_I}$  with  $w = w(\pi)$  and for all edges  $(s, t) \in E_l$  either there exists  $i \geq 0$  s.t. for all  $j > i$  we have  $\pi_j \neq s$ , or there exist infinitely many  $k$ 's such that  $\pi_k = s$  and  $\pi_{k+1} = t$ . Note that this definition and the structure of synthesis games ensure that  $\psi_{E_l}$  is realizable. These definitions together with Theorem 3 and 8 lead to the following theorem.

**Theorem 9.** *Let  $\varphi$  be a specification and  $\mathcal{G} = (G, s_I, \lambda)$  a synthesis game for  $\varphi$  with winning objective  $\Phi$ . If a strongly fair assumption on  $E_l$  is sufficient for  $s_I$  and  $\Phi$ , then the environment assumption  $\psi_{E_l}$  is sufficient for  $\varphi$  and realizable for the environment. Furthermore, if  $\Phi$  is a Reachability, Safety, or Büchi objective and  $s_I$  is live for player 1, then if there exists some sufficient assumption  $\psi \neq \emptyset$ , then there exists a strongly fair assumption that is sufficient.*

## 5.2 Computing Strongly Fair Assumptions

We now focus on solution of deterministic player games with objectives  $\text{AssumeFair}(E_l, \Phi)$ , where  $\Phi$  is a parity objective. Given a deterministic game graph  $G$ , an objective  $\Phi$ , and a strongly fair assumption  $E_l$  on edges, we first observe that the objective  $\text{AssumeFair}(E_l, \Phi)$  can be expressed as an implication: a strong fairness condition implies  $\Phi$ . Hence given  $\Phi$  as a Büchi, coBüchi or a parity objective, the solution of games with objective  $\text{AssumeFair}(E_l, \Phi)$  can be reduced to deterministic Rabin games. However, since deterministic Rabin games are NP-complete we would obtain NP solution (i.e., a NP upper bound), even for the case when  $\Phi$  is a Büchi or coBüchi objective. We now present an efficient reduction to probabilistic games and show that we can solve deterministic games with objectives  $\text{AssumeFair}(E_l, \Phi)$  in  $\text{NP} \cap \text{coNP}$  for parity objectives  $\Phi$ , and if  $\Phi$  is Büchi or coBüchi objectives, then the solution is achieved in polynomial time.

**Reduction.** Given a deterministic game graph  $G = ((S, E), (S_1, S_2))$ , a parity objective  $\Phi$  with a parity function  $p$ , and a set  $E_l \subseteq E_2$  of player-2 edges we construct a probabilistic game  $\tilde{G} = ((\tilde{S}, \tilde{E}), (\tilde{S}_1, \tilde{S}_2, \tilde{S}_P), \delta)$  as follows.

1. *State space.*  $\tilde{S} = S \cup \{\tilde{s} \mid s \in \text{Source}(E_l), E(s) \setminus E_l \neq \emptyset\}$ , i.e., along with states in  $S$ , there is a copy  $\tilde{s}$  of a state  $s$  in  $\text{Source}(E_l)$  such that not all out-going edges from  $s$  are contained in  $E_l$ .
2. *State space partition.*  $\tilde{S}_1 = S_1$ ;  $\tilde{S}_P = \text{Source}(E_l)$ ; and  $\tilde{S}_2 = \tilde{S} \setminus (\tilde{S}_1 \cup \tilde{S}_P)$ . The player-1 states in  $G$  and  $\tilde{G}$  coincide; every state in  $\text{Source}(E_l)$  is a probabilistic state and all other states are player-2 states.
3. *Edges and transition.* We explain edges for the three different kind of states.
  - (a) For  $s \in \tilde{S}_1$  we have  $\tilde{E}(s) = E(s)$ , i.e., the set of edges from player-1 states in  $G$  and  $\tilde{G}$  coincide.
  - (b) For  $s \in \tilde{S}_2$  we have the following cases: (i) if  $s \in S_2$  (i.e., the state is also a player-2 state in  $G$ , thus it is not in  $\text{Source}(E_l)$ ), then  $\tilde{E}(s) = E(s)$ , i.e, then the set of edges are same as in  $G$ ; and (ii) else  $s = \tilde{s}'$  and  $s' \in \text{Source}(E_l)$  and in this case  $\tilde{E}(s) = \{(s, t) \mid (s', t) \in E\}$ .
  - (c) For a state  $s \in \tilde{S}_P$  we have the following two sub-cases: (i) if  $E(s) \subseteq E_l$ , then  $\tilde{E}(s) = E(s)$  and the transition function chooses all states in  $E(s)$  uniformly at random; (ii) else  $\tilde{E}(s) = (E_l \cap E(s)) \cup \{(s, \tilde{s})\}$ , and the transition function is uniform over its successors.

Intuitively, the edges and transition function can be described as follows: all states  $s$  in  $\text{Source}(E_l)$  are converted to probabilistic states, and from  $s$  all edges in  $E(s)$  that are contained in  $E_l$  and the edge to  $\tilde{s}$ , which is a copy of  $s$ , are chosen uniformly at random. From  $\tilde{s}$  player 2 has the choice of the edges in  $E(s)$ .

Given the parity function  $p$ , we construct the parity function  $\tilde{p}$  on  $\tilde{S}$  as follows: for all states  $s \in S$  we have  $\tilde{p}(s) = p(s)$ , and for a state  $\tilde{s}$  in  $\tilde{S}$ , let  $\tilde{s}$  be a copy of  $s$ , then  $\tilde{p}(\tilde{s}) = p(s)$ . We refer to the above reduction as the edge assumption reduction and denote it by  $\text{AssRed}$ , i.e.,  $(\tilde{G}, \tilde{p}) = \text{AssRed}(G, E_l, p)$ . The following theorem states the connection about winning in  $G$  for the objective  $\text{AssumeFair}(E_l, \text{Parity}(p))$  and winning almost-surely in  $\tilde{G}$  for  $\text{Parity}(\tilde{p})$ . The key argument for the proof is as follows. A memoryless almost-sure winning strategy  $\tilde{\alpha}$  in  $\tilde{G}$  can be fixed in  $G$ , and it can be shown that the strategy in  $G$  is sure winning for the Rabin objective that can be derived from the objective  $\text{AssumeFair}(E_l, \text{Parity}(p))$ . Conversely, a memoryless sure winning strategy in  $G$  for the Rabin objective derived from  $\text{AssumeFair}(E_l, \text{Parity}(p))$  can be fixed in  $\tilde{G}$ , and it can be shown that the strategy is almost-winning for  $\text{Parity}(\tilde{p})$  in  $\tilde{G}$ . A key property useful in the proof is as follows: for a probability distribution  $\mu$  over a finite set  $A$  that assigns positive probability to each element in  $A$ , if the probability distribution  $\mu$  is sampled infinitely many times, then every element in  $A$  appears infinitely often with probability 1.

**Theorem 10.** *Let  $G$  be a deterministic game graph, with a parity objective  $\Phi$  defined by a parity function  $p$ . Let  $E_l \subseteq E_2$  be a subset of player-2 edges, and let  $(\tilde{G}, \tilde{p}) = \text{AssRed}(G, E_l, p)$ . Then  $\langle\langle 1 \rangle\rangle_{\text{almost}}(\text{Parity}(\tilde{p})) \cap S = \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l, \Phi))$ .*

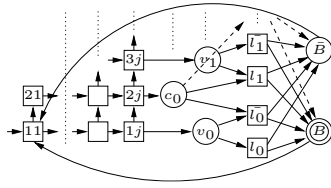
Theorem 10 presents a linear-time reduction for  $\text{AssumeFair}(E_l, \text{Parity}(p))$  to probabilistic games with parity objectives. Using the reduction of Theorem 2 and the results for deterministic parity games (Theorem 1) we obtain the following corollary.

**Corollary 1.** *Given a deterministic game graph  $G$ , an objective  $\Phi$ , a set  $E_l$  of edges, and a state  $s$ , whether  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l, \Phi))$  can be decided in quadratic time if  $\Phi$  is a Büchi or a coBüchi objective, and in  $NP \cap coNP$  if  $\Phi$  is a parity objective.*

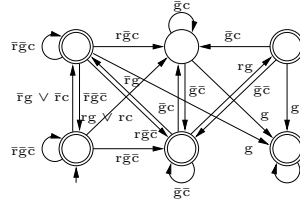
**Complexity of computing a minimal strongly fair assumptions.** We now discuss the problem of finding a minimal set of edges on which a strong fair assumption is sufficient. Given a deterministic game graph  $G$ , a Büchi objective  $\Phi$ , a number  $k \in \mathbb{N}$ , and a state  $s$ , we show that 3SAT can be reduced to the problem of deciding if there is a strongly fair assumption  $E_l$  with at most  $k$  edges ( $|E_l| \leq k$ ) that is sufficient for  $s$  and  $\Phi$ .

Given a CNF-formula  $f$ , we will construct a deterministic game graph  $G$ , give a Büchi objective  $\Phi$ , an initial state  $s$ , and a constant  $k$ , such that  $f$  is satisfiable if and only if there exists a strongly fair assumption  $E_l$  of size at most  $k$  that is sufficient for  $s$  and  $\Phi$ . In Figure 3 we show a sketch of how to construct  $G$ : for each variable  $v_i$  we build two player-2 states, one with the positive literal  $l_i$  and one with the negative literal  $\bar{l}_i$ . Each state has an edge to a Büchi state  $B$  and to non-Büchi state  $\bar{B}$ . Furthermore, for each variable we add a player-1 state  $v_i$  that connects the two states  $l_i$  and  $\bar{l}_i$  representing the literals. Similarly, for each clause  $c_i = l_{i_1} \vee l_{i_2} \vee l_{i_3}$  we have one player-1 state  $c_i$  connected to the state representing the literals  $l_{i_1}, l_{i_2}$ , and  $l_{i_3}$ . Let  $n$  be the number of variables and  $c$  be the number of clauses in  $f$ , and let  $j = n + c$ . Starting from the initial state 11 we have grid of player-2 states with  $j$  columns and from 2 up to  $j$  lines depending on the column. A grid state is connected to its right and to its upper neighbor. Each grid states in the last column is connected either to a state  $v_i$  representing a variable or a clause state  $c_i$ . The constructed game  $\mathcal{G}$  has  $3n + c + \frac{(j+1)(j+2)}{2}$  states and  $6n + 2c + (j+1)^2$  edges. We set  $\Phi = \text{Buchi}(\{B\})$ ,  $s = 11$ , and  $k = n$ .

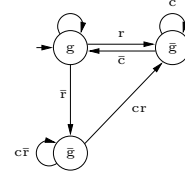
Given a satisfying assignment  $I$  for  $f$ , we build a strongly fair assumption  $E_l$  that includes for each variable  $v_i$  an edge such that if  $I(v_i) = \text{true}$ , then  $(l_i, B) \in E_l$ , else  $(\bar{l}_i, B) \in E_l$ . ( $|E_l| = n = k$ .) The memoryless strategy  $\alpha$  that sets  $\alpha(v_i)$  to  $l_i$  if  $I(v_i) = \text{true}$ , otherwise to  $\bar{l}_i$ , and  $\alpha(c_i)$  to the state representing the literal that is satisfied in  $c_i$  w.r.t.  $I$ , is a winning strategy for player 1 for state 11 and the objective  $\text{AssumeFair}(E_l, \Phi)$ . It follow that  $E_l$  is sufficient for 11 and  $\Phi$ . For the other direction, we observe that any assumption  $E_l$  of size smaller or equal to  $k$  that includes edges from grid states is not sufficient for state 11 and  $\Phi$ . Assume that there is some grid edge  $(s, t)$  in  $E_l$ . Since  $|E_l| \leq k$  there is some variable  $v$  for which neither the edge  $(l, B)$  nor  $(\bar{l}, B)$  is in  $E_l$ . Due to the structure of the grid, player 2 can pick a strategy that results in plays that avoids all except for one player-1 states. From this state  $v$ , player 1 has the two choices to go to  $l$  or  $\bar{l}$  but from both states player 2 can avoid  $B$ . Player 2 has a winning strategy from 11 and so  $E_l$  is not sufficient. An assumption of size



**Fig. 3.** Idea of the NP-hardness proof.



**Fig. 4.** Constructed environment assumption for the specification  $G(\text{req} \rightarrow F \text{grant}) \wedge G(\text{cancel} \rightarrow X \neg \text{grant})$ .



**Fig. 5.** System constructed with assumption shown in Figure 4.

$k$  that is sufficient for  $\mathbb{1}$  and  $\Phi$  only includes edges from literal states  $l_i$  or  $\bar{l}_i$ . Given an assumption  $E_l$  of size  $k$  that is sufficient for  $\mathbb{1}$  and  $\Phi$ . Since  $E_l$  include only edges from literal states, we can easily map  $E_l$  to a satisfying assignment  $I$  for  $f$ : If  $(l_i, B) \in E_l$ , then  $I(v_i) = \text{true}$ , and if  $(\bar{l}_i, B) \in E_l$ , then  $I(v_i) = \text{false}$ .

**Theorem 11.** *Given a deterministic game graph  $G$ , a Büchi objective  $\Phi$ , a number  $k \in \mathbb{N}$ , and a state  $s$ , deciding if there is a strongly fair assumption  $E_l$  with at most  $k$  edges ( $|E_l| \leq k$ ) that is sufficient for  $s$  and  $\Phi$  is NP-hard.*

**Computing locally-minimal strongly fair assumptions.** Since finding the minimal set of edges is NP-hard, we focus on computing a *locally* minimal set of edges. Given a deterministic game graph  $G$ , a state  $s \in S$ , and a parity objective  $\Phi$ , we call a set  $E_l \subseteq E_2$  of player-2 edges locally-minimal strongly fair assumption if  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l, \Phi))$  and for all proper subsets  $E_l'$  of  $E_l$  we have  $s \notin \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l', \Phi))$ . We now show that a locally-minimal strongly fair assumption set  $E_l^*$  of edges can be computed by polynomial calls to a procedure that checks given a set  $E_l$  of edges whether  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l, \Phi))$ . The procedure is as follows:

1. *Iteration 0.* Let the initial set of assumption edges be all player-2 edges, i.e., let  $E^0 = E_2$ ; if  $s \notin \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E^0, \Phi))$ , then there is no subset  $E_l$  of edges such that  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l, \Phi))$ . If  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E^0, \Phi))$ , then we proceed to the next iterative step.
2. *Iteration  $i$ .* Let the current set of assumption edges be  $E^i$  such that we have  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E^i, \Phi))$ . If there exists  $e \in E^i$ , such that  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E^i \setminus \{e\}, \Phi))$ , then let  $E^{i+1} = E^i \setminus \{e\}$ , and proceed to iteration  $i + 1$ . Else if no such  $e$  exists, then  $E^* = E^i$  is a locally-minimal strongly fair assumption set of edges.

The claim that the set of edges obtained above is a locally-minimal strongly fair assumption set can be proved as follows: for a set  $E_l$  of player-2 edges, if  $s \notin \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l, \Phi))$ , then for all subsets  $E_l'$  of  $E_l$  we have  $s \notin \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_l', \Phi))$ . It follows from above that for the set  $E^*$  of player-2 edges obtained by the procedure satisfies that  $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E^*, \Phi))$ ,

and for all proper subsets  $E'$  of  $E^*$  we have  $s \notin \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E', \Phi))$ . The desired result follows.

**Theorem 12.** *The computed set  $E^*$  of edges is a locally-minimal strongly fair assumption.*

## 6 Combining Safety and Liveness Assumptions

Let  $\varphi$  be a specification and let  $\mathcal{G}_\varphi = (G, s_I, \lambda)$  be the corresponding synthesis game with winning objective  $\Phi$ . We first compute a non-restrictively safety assumption  $E_s$  as described in Section 4. If  $\varphi$  is satisfiable, it follows from Theorem 6 and 7 that  $E_s$  exists and that the corresponding environment assumption is realizable for the environment. Then, we modify the winning objective of player 1 with the computed safety assumption: we extend the set of winning plays of player 1 with all plays, in which player 2 follows one of the edges in  $E_s$ . Since  $E_s$  is safe-sufficient, it follows that  $s_I$  is live for player 1 in the modified game. On the modified game, we compute a locally-minimal strongly fair assumption as described in Section 5 (Theorem 12). Finally, using Theorem 8 and 9, we conclude the following.

**Theorem 13.** *Given a specification  $\varphi$ , if the assumption  $\psi = \psi_{E_s} \cap \psi_{E_l} \neq \emptyset$ , where  $E_s$  and  $E_l$  are computed as shown before, then  $\psi$  is a sufficient assumption for  $\varphi$  that is realizable for the environment. If  $\varphi$  has a corresponding synthesis game  $\mathcal{G}_\varphi$  with a safety, reachability, or Büchi objective for player 1, then if there exists a sufficient environment assumption  $\psi \neq \emptyset$ , then the assumption  $\psi = \psi_{E_s} \cap \psi_{E_l}$ , where  $E_s$  and  $E_l$  are computed as shown before, is not empty.*

Recall the example from the introduction with the signals `req`, `cancel`, and `grant` and the specification  $G(\text{req} \rightarrow XF\text{grant}) \wedge G((\text{cancel} \vee \text{grant}) \rightarrow X\neg\text{grant})$ . Applying our algorithm we get the environment assumption  $\hat{\psi}$  shown in Figure 4 (double lines indicate Büchi states). We could not describe the language using an LTL formula, therefore we give its relation to the assumptions proposed in the introduction. Our assumption  $\hat{\psi}$  includes  $\psi_1 = G(\neg\text{cancel})$  and  $\psi_2 = G(F(\neg\text{cancel}))$ , is a strict subset of  $\psi_6 = \xi W(\xi \wedge (\text{cancel} \vee \text{grant}) \wedge X\text{grant})$  with  $\xi = \text{req} \rightarrow XF(\neg\text{cancel} \vee \text{grant})$ , and is incomparable to all other sufficient assumptions. Even though, the constructed assumption is not the weakest w.r.t. language inclusion, it still serves its purpose: Figure 5 shows a system synthesized with a modified version of [12] using the assumption  $\hat{\psi}$ .

## 7 Acknowledgements

The authors would like to thank Rupak Majumdar for stimulating and fruitful discussions.

## References

1. Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.
2. I. Beer, S. Ben-David, U. Eisner, and Y. Rodeh. Efficient detection of vacuity in ACTL formulas. In *Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 279–290, 1997.
3. Julien Bernet, David Janin, and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *Theoretical Informatics and Applications (RAIRO)*, 36:261–275, 2002.
4. Dirk Beyer, Thomas A. Henzinger, and Vasu Singh. Algorithms for interface synthesis. In *International Conference on Computer Aided Verification (CAV'07)*, pages 4–19, 2007.
5. R. Bloem, S. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Automatic hardware synthesis from specifications: A case study. In *DATE*, pages 1188–1193, 2007.
6. R. Bloem, S. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Specify, compile, run: Hardware from PSL. In *6th International Workshop on Compiler Optimization Meets Compiler Verification*, pages 3–16, 2007.
7. Mihaela Gheorghiu Bobaru, Corina Pasareanu, and Dimitra Giannakopoulou. Automated assume-guarantee reasoning by abstraction refinement. In *International Conference on Computer Aided Verification (CAV'08)*, 2008. Accepted for publication.
8. K. Chatterjee, T.A. Henzinger, and M. Jurdziński. Games with secure equilibria. *Theoretical Computer Science*, 365:67–82, 2006.
9. K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Simple stochastic parity games. In *CSL'03*, volume 2803 of *LNCS*, pages 100–113. Springer, 2003.
10. E.A. Emerson and C. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS'91*, pages 368–377. IEEE, 1991.
11. Marco Faella. Games you cannot win. In *Workshop on Games and Automata for Synthesis and Validation*, Lausanne, Switzerland, 2007.
12. B. Jobstmann and R. Bloem. Optimizations for LTL synthesis. In *6th Conference on Formal Methods in Computer Aided Design (FMCAD'06)*, pages 117–124, 2006.
13. V. King, O. Kupferman, and M. Y. Vardi. On the complexity of parity word automata. In *Foundations of Software Science and Computation Structures*, pages 276–286, 2001. LNCS 2030.
14. O. Kupferman and M. Y. Vardi. Vacuity detection in temporal model checking. In *Correct Hardware Design and Verification Methods (CHARME'99)*, pages 82–96, Berlin, September 1999. Springer-Verlag. LNCS 1703.
15. D.A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.
16. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *21st Symposium on Logic in Computer Science (LICS'06)*, pages 255–264, Seattle, WA, aug 2006.
17. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. Symposium on Principles of Programming Languages (POPL '89)*, pages 179–190, 1989.
18. R. Rosner. *Modular Synthesis of Reactive Systems*. PhD thesis, Weizmann Institute of Science, 1992.
19. Specman elite - testbench automation. <http://www.verisity.com/products/specman.html>.
20. M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.
21. Vera - testbench automation. <http://www.synopsys.com/products/vera/vera.html>.