

Environment Assumptions for Synthesis

Krishnendu Chatterjee², Thomas A. Henzinger¹, and Barbara Jobstmann¹

¹ EPFL, Lausanne

² University of California, Santa Cruz

August 11, 2008

Abstract. The synthesis problem asks to construct a reactive finite-state system from an ω -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 φ by computing an environment assumption ψ such that the new specification $\psi \rightarrow \varphi$ is realizable. Our aim is to construct an assumption ψ 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 a successful verification technique in hardware and software design. Although the method is automated, the success of a verification process highly depends on the quality of the specification. 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, 6]. We follow a different path by leaving the design 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 φ , if φ is not realizable, it cannot be complete and has to be weakened by introducing assumptions on the environment of the system. We do this by computing an assumption ψ such that the new specification $\psi \rightarrow \varphi$ is realizable. Our aim is to construct a condition ψ that does not constrain the system and is as weak as possible. The notion that ψ must constrain only the environment can be captured by requiring that ψ itself is realizable for the environment—i.e., there exists an environment that satisfies ψ without constraining the outputs of the system. The notion that ψ 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 ψ , and in other situations, it is NP-hard to compute such an assumption.

Example. During a typical effort of formally specifying hardware designs [5], some specifications were initially not realizable. One specification that was particularly difficult to analyze can be simplified to the following example. Consider a system with two input signals r and c , and one output signal g . The specification requires that (i) every request is eventually granted starting from the next time step, written in linear temporal logic as $\Box(r \rightarrow \bigcirc \Diamond g)$; and (ii) whenever c or g are high, then g has to stay low in the next time step, written $\Box((c \vee g) \rightarrow \bigcirc \neg g)$. This specification is not realizable because the environment can force, by sending c all the time, that g has to stay low forever (Part (ii)). Thus requests cannot be answered, and Part (i) is violated.

One assumption that makes this specification realizable is $\psi_1 = \Box \neg c$. This assumption is undesirable because it forbids the environment to send c . A system synthesized with this assumption would ignore the signal c . Assumptions $\psi_2 = \Box \Diamond \neg c$ and $\psi_3 = \Box(r \rightarrow \Diamond \neg c)$ are more desirable but still not satisfactory: ψ_2 forces the environment to lower c infinitely often even when no requests are sent, and ψ_3 is not strong enough to implement a system that in each step first produces an output and then reads the input. Assume that the system starts with output $g = 0$ in time step 0, then receives the input $r = 1$ and $c = 0$, now in time step 1, it can choose between (a) $g = 1$, or (b) $g = 0$. If it chooses to set grant to high by (a), then the environment can provide the same inputs once more ($r = 1$ and $c = 0$) and can set all subsequent inputs to $r = 0$ and $c = 1$. Then the environment has satisfied ψ_3 because during the two requests in time step 0 and 1 the signal c was kept low, but the system cannot fulfill Part (i) of its specification without violating Part (ii) due to $g = 1$ in time step 1 and $c = 1$ afterwards. On the other hand, if the system decides to choose to set $g = 0$ by (b), then the environment can choose to set the inputs to $r = 0$ and $c = 1$ and the system again fails to fulfill Part (i) without violating (ii). The assumption $\psi_4 = \Box(r \rightarrow \bigcirc \Diamond \neg c)$, which is a subset of ψ_3 , is sufficient. However, there are infinitely many sufficient assumptions between ψ_3 and ψ_4 , such as $\psi'_3 = (\neg c \wedge \bigcirc \psi_3) \vee \psi_3$. The assumption $\psi_5 = \Box(r \rightarrow \bigcirc \Diamond (\neg c \vee g))$

is also weaker than ψ_3 and still sufficient, because the environment only needs to lower c eventually if a request has not been answered yet. Finally, let $\xi = r \rightarrow \bigcirc \diamond (\neg c \vee g)$ and consider the assumption $\psi_6 = \xi W(\xi \wedge (c \vee g) \wedge \bigcirc g)$, which is a sufficient assumption (where W is the *weak-until* operator of LTL). It is desirable because it states that whenever a request is sent, the environment has to eventually lower c if it has not seen the signal g , but as soon as the system violates its specification (Part (ii)) all restrictions on the environment are dropped. If we replace ξ in ψ_6 with $\xi' = r \rightarrow \diamond (\neg c \vee g)$, 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 [17]. More specifically, an ω -regular specification φ is realizable iff there exists a winning strategy in a certain parity game constructed from φ . If φ is not realizable, then we construct an environment assumption ψ 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 game 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 specifications, and in $\text{NP} \cap \text{coNP}$ 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 work. 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 nonzero-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 results of [3] show that such most general winning strategies exist only for safety games, and also present an approach to compute a strategy, called a *permissive strategy*, which subsumes behavior of all memoryless winning strategies. Our approach is different, as we attempt to construct the minimal environment assumption that makes a game winning.

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 Sections 4 and 5, we compute, respectively, safety and liveness assumptions, which are then combined in Section 6. A full version with detailed proofs can be found in [7].

2 Preliminaries

Words, languages, safety, and liveness. Given a finite alphabet Σ 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 $i+1$. Given a word $w \in \Sigma^\omega$, we write $\text{odd}(w)$ for the subsequence of w consisting of the odd positions ($\forall i \geq 0 : \text{odd}(w)_i = w_{2i+1}$). Given a set $L \subseteq \Sigma^\omega$ of infinite words, we define the set of finite prefixes by $\text{pref}(L) = \{v \in \Sigma^* \mid \exists w \in L, i \geq 0 : v = w^i\}$. Given a set $L \subseteq \Sigma^*$ of finite words, we define the set of infinite limits by $\text{safe}(L) = \{w \in \Sigma^\omega \mid \forall i \geq 0 : w^i \in L\}$. A language $L \subseteq \Sigma^\omega$ is a *safety* language if $L = \text{safe}(\text{pref}(L))$. A language $L \subseteq \Sigma^\omega$ is a *liveness* language if $\text{pref}(L) = \Sigma^*$. Every ω -regular language $L \subseteq \Sigma^\omega$ can be presented as the intersection of the safety language $L_S = \text{safe}(\text{pref}(L))$ 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 and a set I of input propositions. We use the 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 $\kappa: Q \rightarrow \mathcal{O}$ is a state labeling function. 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 transition function Δ to finite words $w \in \mathcal{I}^*$ inductively by $\Delta(q, w) = \Delta(\Delta(q, w^{|w|-1}), w_{|w|})$ for $|w| > 0$. Given a 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 π 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 case of Mealy transducers. The *language* of \mathcal{T} is the set $L(\mathcal{T}) = \{\mathcal{T}(w) \mid w \in \mathcal{I}^\omega\}$ of all generated infinite words.

Specifications and realizability. A *specification* of a reactive system is an ω -regular language $L \subseteq \Sigma^\omega$. We use Linear Temporal Logic (LTL) formulae over the atomic proposition P , as well as ω -automata with transition labels from Σ , to define specifications. Given an LTL formula (resp. ω -automaton) φ , we write $L(\varphi) \subseteq \Sigma^\omega$ for the set of infinite words that satisfy (resp. are accepted by) φ . A transducer \mathcal{T} *satisfies* a specification $L(\varphi)$, written $\mathcal{T} \models \varphi$, if $L(\mathcal{T}) \subseteq L(\varphi)$. Given an LTL formula (resp. ω -automaton) φ , the *realizability problem* asks if there exists a transducer \mathcal{T} with input alphabet \mathcal{I} and output alphabet \mathcal{O} such that $\mathcal{T} \models \varphi$. The specification $L(\varphi)$ 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 $L(\varphi)$ is Mealy realizable iff $L(\varphi')$ is Moore realizable, where the LTL formula φ' is obtained from φ by replacing all occurrences of $o \in O$ by $\bigcirc o$. 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 δ . 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 set $E' \subseteq E$ 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' . We write $E_1 = E \cap (S_1 \times S)$ and $E_2 = E \cap (S_2 \times S)$ for the sets of player-1 and player-2 edges. *Deterministic game graphs* are the special case of the probabilistic game graphs with $S_P = \emptyset$, that is, the state space is partitioned into player-1 and player-2 states. In such cases we omit S_P and δ in the definition of the game graph.

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 Π 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, but all our results hold even if strategies were probabilistic. We denote by \mathcal{A} and \mathcal{B} the sets of strategies for player 1 and player 2, respectively. A strategy α 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 sets of memoryless strategies for player 1 and player 2, respectively.

Once a start 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 well-defined, where an *event* $\mathcal{E} \subseteq \Pi$ is a measurable set of plays. Given strategies α for player 1 and β 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 write $\text{Outcome}(s, \alpha, \beta) \subseteq \Pi_s$ for 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 $\text{Pr}_s^{\alpha, \beta}(\mathcal{E})$ 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 α and β .

Objectives. An *objective* for a player is a set $\Phi \subseteq \Pi$ of winning plays. We consider ω -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 π . We consider safety, Büchi, and parity objectives. Given a set $F \subseteq S$ of states, the *safety objective* $\text{Safe}(F) = \{s_0 s_1 s_2 \dots \in II \mid \forall k \geq 0 : s_k \in F\}$ requires that only states in F be visited. The *Büchi objective* $\text{Buchi}(F) = \{\pi \in II \mid \text{Inf}(\pi) \cap F \neq \emptyset\}$ requires that some state in F be visited infinitely often. 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 $\text{Parity}(p) = \{\pi \in II \mid \min\{p(\text{Inf}(\pi))\} \text{ is even}\}$. Büchi objectives are special cases of parity objectives with two priorities.

Sure and almost-sure winning. Given an objective Φ , 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 α is *almost-sure winning* for player 1 from s for Φ if for every player-2 strategy β , 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 Φ , the *sure (resp. almost-sure) winning set* $\langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi)$ (resp. $\langle\langle 1 \rangle\rangle_{\text{almost}}(\Phi)$) for player 1 is the set of states from which player 1 has a sure (resp. 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 Φ , 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], i.e., 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 Φ , 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 α and a player-2 strategy β such that $\text{Outcome}(s, \alpha, \beta) \subseteq \Phi$.

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

Theorem 2 (Probabilistic games [9]). *Given a probabilistic game graph $G = ((S, E), (S_1, S_2, S_P), \delta)$ and a parity objective Φ 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 $\langle\langle 1 \rangle\rangle_{\text{almost}}(\Phi)$ 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 or ω -automaton φ , we can construct a deterministic game graph G , a state s_I of G , and a parity objective Φ such that $L(\varphi)$ is Moore (resp. Mealy) realizable iff $s_I \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\Phi)$.*

The deterministic game graph G with parity objective Φ referred to in Theorem 3 is called the *Mealy* (resp. *Moore*) *synthesis game* for φ . Starting from an LTL formula φ , we construct the synthesis games 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 Mealy (resp. Moore) synthesis game. Both steps involve exponential blow-ups that are unavoidable: for LTL formulae φ , the realizability problem is 2EXPTIME-complete [18].

Synthesis games, by relating paths in the game graph to the specification $L(\varphi)$, have the following special form. A *Moore synthesis game* \mathcal{G} is a tuple (G, s_I, λ, Φ) , 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 Φ 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 states $s \in S_1$ (resp. S_2) and $l \in \mathcal{O}$ (resp. \mathcal{I}), 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 the Moore synthesis game \mathcal{G} for a specification formula or automaton φ (as referred to by Theorem 3), every Moore transducer $\mathcal{T} = (Q, q_I, \Delta, \kappa)$ that satisfies $L(\varphi)$ represents a winning strategy α for player 1 as follows: for all state sequences $\tau \in (S_1 \cdot S_2)^* \cdot S_1$, let w be the finite word such that $w_i = \lambda(\tau_{i+1})$ for all $0 \leq i < |\tau|$; then, if there is an edge $(\tau_{|\tau|}, s') \in E$ with $\lambda(s') = \kappa(\Delta(q_I, \text{odd}(w)))$, let $\alpha(\tau) = s'$, and else let $\alpha(\tau)$ be arbitrary. Conversely, every memoryless winning strategy α of player 1 represents a Moore transducer $\mathcal{T} = (Q, q_I, \Delta, \kappa)$ that satisfies $L(\varphi)$ as follows: let $Q = S_1$, $q_I = s_I$, $\kappa(q) = \lambda(\alpha(q))$, and $\Delta(q, l) = s'$ if $\lambda(s') = l$ and $(\alpha(q), s') \in E$. The construction of a *Mealy synthesis game* for the Mealy realizability problem is similar.

3 Assumptions

We illustrate the difficulties in defining desirable conditions on environment assumptions through several examples. W.l.o.g. we model open reactive systems as Moore transducers, and correspondingly, their environments as Mealy transducers (with inputs and outputs swapped). Given a specification formula or automaton φ that describes the desired behavior of a system \mathcal{S} (a Moore transducer), we search for an assumption on the environment of \mathcal{S} which is sufficient to ensure that \mathcal{S} exists and satisfies $L(\varphi)$. Formally, a language $K \subseteq \Sigma^\omega$ is a *sufficient assumption* for a specification $L \subseteq \Sigma^\omega$ if $(\Sigma^\omega \setminus K) \cup L$ is Moore realizable. In other words, if the specification is given by an LTL formula φ , and the environment assumption by another LTL formula ψ , then ψ is sufficient for φ iff $L(\psi \rightarrow \varphi)$ is realizable. In this case, we can view the formula $\psi \rightarrow \varphi$ as defining a corrected specification.

Example 1. Consider the specification $\varphi = \text{b U a}$, where U is the *until* operator of LTL. No system \mathcal{S} with input a and output b can implement φ , because \mathcal{S} does not control a, and φ is satisfied only if a eventually is true. We have to weaken the specification

to make it realizable. A candidate for the assumption ψ is $\diamond a$, because it forces the environment to assert the signal a eventually. Further candidates are false , which makes the specification trivially realizable, $\diamond b$, and $\diamond \neg b$, which lead to corrected specifications such as $\varphi' = \diamond b \rightarrow \varphi = (\Box \neg b) \vee \varphi$. The system can implement φ' independent of φ simply by keeping b low all the time.

Example 1 shows that there may be several different sufficient assumptions for a given specification $L \subseteq \Sigma^\omega$, but not all of them are satisfactory. For instance, the assumption false does not provide the desired information, and the assumption that $\diamond b$ cannot be satisfied by any environment that controls only a . Environment assumptions that are unsatisfiable or falsifiable by the system correspond to a corrected specification $\psi \rightarrow \varphi$ that can be satisfied vacuously [2, 14] by the system. In order to exclude such assumptions, we require that an environment assumption $K \subseteq \Sigma^\omega$ for L fulfill the following condition.

- (1) *Realizability by the environment:* There exists an implementation of the environment that satisfies K . Formally, we require that the language K be Mealy realizable with input alphabet \mathcal{O} and output alphabet \mathcal{I} .

Note that Condition 1 implies that the specification L has to be nonempty for a suitable assumption K to exist. If a formula φ is not satisfiable, then there exists only the trivial solution $\psi = \text{false}$. We assume from now on that specifications are nonempty (i.e., satisfiable). Apart from Condition 1, we aim to restrict the environment “as little as possible.” For this purpose, we need to order different assumptions. An obvious candidate for this order is language inclusion.

- (2) *Minimality:* There exists no other sufficient assumption that is realizable by the environment and strictly weaker than K . Formally, there is no language $K' \subseteq \Sigma^\omega$ such that $K \subset K'$ and K' is both a sufficient assumption for L and realizable by the environment.

The following example shows we cannot ask for a *unique* minimal assumption.

Example 2. Consider the specification $\varphi = (b \cup a_1) \vee (\neg b \cup a_2)$, where a_1 and a_2 are inputs and b is an output. Again, φ is not realizable. Consider the assumptions $\psi_1 = \diamond a_1$ and $\psi_2 = \diamond a_2$. Both are sufficient because, assuming ψ_1 , the system can keep the signal b constantly high, and assuming ψ_2 , it can keep b constantly low. Both the assumptions are also realizable by the environment. However, if we assume the disjunction $\psi = \psi_1 \vee \psi_2$, then the system does not know which of the two signals a_1 and a_2 the environment is going to assert eventually. Since a unique minimal assumption has to subsume all other sufficient assumptions and ψ is not sufficient, it follows that there exists no unique minimal assumption that is sufficient.

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

Example 3. Consider the specification $\varphi = \Box(a \rightarrow \bigcirc b) \wedge \Box(b \rightarrow \bigcirc \neg b)$ with input a and output b . The specification is not realizable because whenever a is set to true in two consecutive steps, the system cannot produce a value for b such that φ is satisfied. One

natural assumption is $\psi = \Box(a \rightarrow \bigcirc \neg a)$. Another assumption is $\psi' = \psi \vee \Diamond(\neg a \wedge \bigcirc b)$, which is weaker than ψ w.r.t. language inclusion and still sufficient and realizable by the environment. Looking at the resulting corrected system specification $\psi' \rightarrow \varphi = (\psi \vee \Diamond(\neg a \wedge \bigcirc b)) \rightarrow \varphi = \psi \rightarrow (\Box(\neg a \rightarrow \bigcirc \neg b) \wedge \varphi)$, we see that ψ' restricts the system instead of the environment.

Intuitively, using language inclusion as ordering criterion 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 ψ should refer only to input signals. Let us consider the specification of Example 3 once more. Another sufficient assumption is $\psi'' = (a \rightarrow \bigcirc \neg a) W(b \wedge \bigcirc b)$, which is weaker than ψ . This assumption requires that the environment guarantees $a \rightarrow \bigcirc \neg a$ as long as the system does not make a mistake (by setting b to true in two consecutive steps), which clearly meets 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 ψ and ψ'' are safety assumptions, while the assumptions in Example 2 are liveness assumptions. In general, every ω -regular language can be decomposed into a safety and a liveness component. We use this separation to provide a way to compute environment assumptions in two steps. In both steps, we restrict the environment strategies of synthesis games to find sufficient environment assumptions. More precisely, we put restrictions on the player-2 edges, which represent decisions made by the environment. If the given specification is satisfiable, then these restrictions lead to assumptions that are realizable by the environment.

4 Safety Assumptions

We first compute assumptions that restrict the safety behavior of the environment.

Nonrestrictive safety assumptions on games. Given a deterministic game graph $G = ((S, E), (S_1, S_2))$, a *safety assumption* is a set $E_S \subseteq E_2$ of player-2 edges requiring that player 2 chooses only edges *not* in E_S . 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. For a given player-1 objective Φ , a safety assumption refers to the safety component of the objective, namely, $\Phi_S = \text{Safe}(\langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi))$. Let $\text{AssumeSafe}(E_S, \Phi) = \{\pi = s_0 s_1 s_2 \dots \mid \text{either (i) there exists } i \geq 0 \text{ such that } (s_i, s_{i+1}) \in E_S, \text{ or (ii) } \pi \in \Phi_S\}$ be the set of all plays in which either one of the edges in E_S is chosen, or that satisfy the safety component of Φ . The safety assumption E_S is *safe-sufficient* for a state $s \in S$ and player-1 objective Φ if player 1 has a winning strategy from s for the modified objective $\text{AssumeSafe}(E_S, \Phi)$. A synthesis game $\mathcal{G} = (G, s_I, \lambda, \Phi)$ with a safety assumption E_S specifies the environment assumption $K(E_S)$ defined 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 $(\pi_i, \pi_{i+1}) \notin E_S$ for all $i \geq 0$.

Theorem 4. *Let $\mathcal{G}_\varphi = (G, s_I, \lambda, \Phi)$ be the Moore synthesis game for an LTL formula (or ω -automaton) φ , and let E_S be a safety assumption. If E_S is safe-sufficient for the state s_I and objective Φ , then $K(E_S)$ is a sufficient assumption for the specification $\text{safe}(\text{pref}(L(\varphi)))$.*

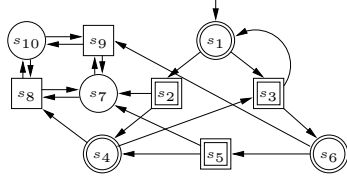


Fig. 1. Game with two equally small safe-sufficient safety assumptions for s_1 : $E_S = \{(s_3, s_1)\}$ and $E_{S'} = \{(s_5, s_7)\}$.

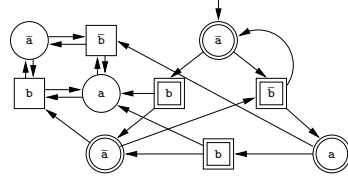


Fig. 2. Synthesis game for $\Box(a \rightarrow \bigcirc b) \wedge \Box(b \rightarrow \bigcirc \neg b)$.

The following example shows that there exist safety games without a unique minimal safety assumption that is safe-sufficient.

Example 4. Consider the game shown in Figure 1. Circles denote states of player 1; boxes denote states of player 2. The objective for player 1 is to stay inside the set $\{s_1, \dots, s_6\}$ of states marked by double lines. Player 1 has no winning strategy from 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 s_1 . If we consider a specification where the corresponding synthesis game has this structure, then neither of these assumptions are satisfactory. Figure 2 shows such a synthesis game, for the specification $\Box(a \rightarrow \bigcirc b) \wedge \Box(b \rightarrow \bigcirc \neg b)$ with input a and output b (cf. Example 3). Using the safety assumption E_S , the corrected specification would allow only the implementation that keeps b constantly low. The other safety assumption $E_{S'}$ leads to a corrected specification that additionally enforces $\Box(\neg a \rightarrow \bigcirc \neg b)$.

Therefore, besides safe-sufficiency, we look for a safety assumption that does not restrict player 1. This condition can be formalized as follows. Given a deterministic game graph $G = ((S, E), (S_1, S_2))$, a safety assumption E_S is *restrictive* for a state $s \in S$ and a player-1 objective Φ if there exist strategies $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$ for the two players such that the play $\text{Outcome}(s, \alpha, \beta)$ contains an edge from E_S and is in Φ_S . Intuitively, a nonrestrictive safety assumption allows all edges that do not lead to an immediate violation of the safety component of the objective for player 1.

Theorem 5. *Given a deterministic game graph $G = ((S, E), (S_1, S_2))$, an objective Φ 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 nonrestrictive and safe-sufficient for s and Φ . Moreover, if $s \in \langle\langle 1, 2 \rangle\rangle_{\text{sure}}(\Phi)$ and E_S is the minimal safety assumption for s and Φ , then player 2 has a winning strategy from s for the objective to avoid all edges in E_S .*

Applying Theorem 5 to environment assumptions, we obtain Theorem 6.

Theorem 6. *Let $\mathcal{G} = (G, s_I, \lambda, \Phi)$ be the Moore synthesis game for a satisfiable LTL formula (or ω -automaton) φ . Then there exists a unique minimal safety assumption E_S that is nonrestrictive and safe-sufficient for the state s_I and objective Φ . Moreover, the corresponding assumption $K(E_S)$ is realizable by the environment.*

Computing nonrestrictive safety assumptions. Given a deterministic game graph G and player-1 objective Φ , we compute the unique minimal nonrestrictive and safe-sufficient safety assumption E_S as follows. First, we compute the set $\langle\langle 1, 2 \rangle\rangle_{sure}(\Phi)$ of states. Note that for this set the players cooperate. We can compute $\langle\langle 1, 2 \rangle\rangle_{sure}(\Phi)$ in polynomial time for all objectives we consider. In particular, if Φ is a parity objective, then $\langle\langle 1, 2 \rangle\rangle_{sure}(\Phi)$ can be computed by reduction to Büchi automata [13]. Then the safety assumption E_S is the set of all player-2 edges $(s, t) \in E_2$ such that $s \in \langle\langle 1, 2 \rangle\rangle_{sure}(\Phi)$ and $t \notin \langle\langle 1, 2 \rangle\rangle_{sure}(\Phi)$.

Theorem 7. *For every deterministic game graph G and player-1 objective Φ , the edge set $E_S = \{(s, t) \in E_2 \mid s \in \langle\langle 1, 2 \rangle\rangle_{sure}(\Phi) \text{ and } t \notin \langle\langle 1, 2 \rangle\rangle_{sure}(\Phi)\}$ is the unique minimal safety assumption that is nonrestrictive and safe-sufficient for all states $s \in \langle\langle 1, 2 \rangle\rangle_{sure}(\Phi)$. The set E_S can be computed in polynomial time for parity objectives Φ .*

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, the set E_S defines the environment assumption $\psi_{E_S} = (\neg a \vee \neg b) W((\neg a \vee \neg b) \wedge a \wedge (\bigcirc \neg b) \wedge b \wedge \bigcirc b)$. This safety assumption meets our intuition of a minimal environment assumption, because it states that the environment has to ensure that either a or b is low as long as the system makes no obvious fault by either violating $\Box(a \rightarrow \bigcirc b)$ or $\Box(b \rightarrow \bigcirc \neg b)$.

5 Liveness Assumptions

In a second step, we now put liveness assumptions on the environment.

Strongly fair assumptions on games. Given a deterministic game graph $G = ((S, E), (S_1, S_2))$ and a player-1 objective Φ , a *strongly fair assumption* is a set $E_L \subseteq E_2$ of player-2 edges requiring that player 2 plays such that if a state $s \in \text{Source}(E_L)$ is visited infinitely often, then for all states $t \in S$ such that $(s, t) \in E_L$, the edge (s, t) is chosen infinitely often. Let $\text{AssumeFair}(E_L, \Phi)$ be the set of plays π such that either (i) there is a state $s \in \text{Source}(E_L)$ that appears infinitely often in π and there is an edge $(s, t) \in E_L$ that appears only finitely often in π , or (ii) π belongs to the objective Φ . Formally, $\text{AssumeFair}(E_L, \Phi) = \{\pi = s_0 s_1 s_2 \dots \mid \text{either (i) } \exists (s, t) \in E_L \text{ such that } s_k = s \text{ for infinitely many } k\text{'s and there are only finitely many } j\text{'s such that } s_j = s \text{ and } s_{j+1} = t, \text{ or (ii) } \pi \in \Phi\}$. The strongly fair assumption $E_L \subseteq E_2$ is *live-sufficient* for a state $s \in S$ and player-1 objective Φ if player 1 has a winning strategy from s for the modified objective $\text{AssumeFair}(E_L, \Phi)$. A state $s \in S$ is *live for player 1* if player 1 has a winning strategy from s for the objective $\text{Safe}(\langle\langle 1, 2 \rangle\rangle_{sure}(\Phi))$.

Theorem 8. *Given a deterministic game graph $G = ((S, E), (S_1, S_2))$ and a safety or Büchi objective Φ , for every state $s \in S$ that is live for player 1, there exists a strongly fair assumption E_L that is live-sufficient for s and Φ .*

A synthesis game $\mathcal{G} = (G, s_I, \lambda, \Phi)$ with a strongly fair assumption E_L specifies the environment assumption $K(E_L)$ defined 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 $i \geq 0$ such that for all $j > i$ we have $\pi_i \neq s$, 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 $K(E_L)$ is realizable by the environment. These definitions together with Theorem 3 and 8 lead to the following theorem.

Theorem 9. *Let $\mathcal{G} = (G, s_I, \lambda, \Phi)$ be a Moore synthesis game for an LTL formula (or ω -automaton) φ , and let E_L be a strongly fair assumption. If E_L is live-sufficient for the state s_I and objective Φ , then $K(E_L)$ is a sufficient assumption for the specification $L(\varphi)$. Moreover, the assumption $K(E_L)$ is realizable by the environment. Conversely, if Φ is a safety or Büchi objective, if s_I is live for player 1, and if there exists some sufficient assumption $K \neq \emptyset$ for the specification $L(\varphi)$, then there exists a strongly fair assumption that is live-sufficient.*

Computing strongly fair assumptions. We now focus on solution of deterministic player games with objectives $\text{AssumeFair}(E_L, \Phi)$, where Φ is a parity objective. Given a deterministic game graph G , an objective Φ , 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 Φ . Hence given Φ as a Büchi 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., an NP upper bound), even for the case when Φ is a Büchi 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 Φ , and if Φ is a Büchi objective, then the solution is achieved in polynomial time.

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

1. *State space.* $\tilde{S} = S \cup \{\tilde{s} \mid s \in \text{Source}(E_L) \text{ and } E(s) \setminus E_L \neq \emptyset\}$.
2. *State space partition.* $\tilde{S}_1 = S_1$, $\tilde{S}_P = \text{Source}(E_L)$, and $\tilde{S}_2 = \tilde{S} \setminus (\tilde{S}_1 \cup \tilde{S}_P)$.
3. *Edges and transition.* We explain edges for the three different kind of states.
 - (a) For a state $s \in \tilde{S}_1$ we have $\tilde{E}(s) = E(s)$.
 - (b) For a state $s \in \tilde{S}_2$ if $s \in S_2$, then $\tilde{E}(s) = E(s)$; else $s = \tilde{s}'$ and $s' \in \text{Source}(E_L)$ and we have $\tilde{E}(s) = \{(s, t) \mid (s', t) \in E\}$.
 - (c) For a state $s \in \tilde{S}_P$, if $E(s) \subseteq E_L$, then $\tilde{E}(s) = E(s)$ else $\tilde{E}(s) = (E(s) \cap E_L) \cup \{(s, \tilde{s})\}$. In both case the transition function is uniform over its successors.
4. *Objective.* For all states $s \in S$, we have that $\tilde{p}(s) = p(s)$, and for a state $\tilde{s} \in \tilde{S} \setminus S$, let \tilde{s} be the copy of s , then $\tilde{p}(\tilde{s}) = p(s)$.

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 $E(s)$.

We refer to the above reduction as the edge assumption reduction and denote it by AssRed , i.e., $(\tilde{G}, \tilde{p}) = \text{AssRed}(G, E_L, p)$. The following theorem states the connection

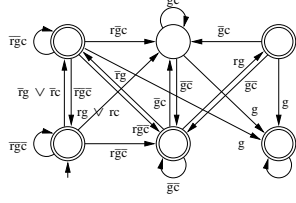


Fig. 3. Constructed environment assumption for $\Box(r \rightarrow \Diamond g) \wedge \Box(c \rightarrow \bigcirc \neg g)$.

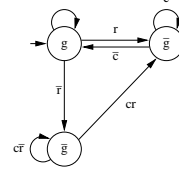


Fig. 4. System constructed with assumption shown in Figure 3.

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 μ over a finite set A that assigns positive probability to each element in A , if the probability distribution μ 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, and let Φ be a parity objective defined by a priority function p . Let E_L be a set 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 Φ , a set E_L of player-2 edges, and a state s of G , whether $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_L, \Phi))$ can be decided in quadratic time if Φ is a Büchi objective, and in $NP \cap coNP$ if Φ is a parity objective.*

Complexity of computing a minimal strongly fair assumption. We consider the problem of finding a minimal set of edges on which a strong fair assumption is sufficient. Due to space limitation, we present here only the theorem, the proof can be found in [7].

Theorem 11. *Given a deterministic game graph G , a Büchi objective Φ , a number $k \in \mathbb{N}$, and a state s of G , the problem of deciding if there is a strongly fair assumption E_L with at most k edges (i.e., $|E_L| \leq k$) which is live-sufficient for s and Φ , is NP-hard.*

Computing locally minimal strongly fair assumptions. Since finding a 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 , and a player-1 objective Φ , a set

$E_L \subseteq E_2$ of player-2 edges is a *locally-minimal strongly fair assumption* for s and Φ if $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_L, \Phi))$ and for all proper subsets $E_L' \subset E_L$, we have $s \notin \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_L', \Phi))$. A locally-minimal strongly fair assumption E_L^* can be computed by a polynomial number of calls to a procedure that checks, given a set E_L of player-2 edges, whether $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_L, \Phi))$. The computation proceeds as follows. Initially all player-2 edges are in E_L^* . As long as $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_L^*, \Phi))$, we search for an edge e such that $s \in \langle\langle 1 \rangle\rangle_{\text{sure}}(\text{AssumeFair}(E_L^* \setminus \{e\}, \Phi))$. If such an e exists, then we remove e from E_L^* and proceed; otherwise we stop and return E_L^* . In the worst case, we have at most $\frac{m \cdot (m+1)}{2}$ procedure calls, where m is the number of player-2 edges.

Theorem 12. *Given a deterministic game graph G , a state s of G , and a parity objective Φ , the computed set E_L^* is a locally-minimal strongly fair assumption for s and Φ . If Φ is a Büchi objective, then we can compute a locally-minimal strongly fair assumption E_L^* for s and Φ in polynomial time.*

6 Combining Safety and Liveness Assumptions

Now we put everything together. Let φ be an LTL formula (or ω -automaton) and let $\mathcal{G} = (G, s_I, \lambda, \Phi)$ be the corresponding Moore synthesis game. We first compute a nonrestrictive safety assumption E_S as described in Section 4 (Theorem 7). If φ is satisfiable, then it follows from Theorem 6 that E_S exists and that the corresponding environment assumption $K(E_S)$ is realizable by the environment. Then, we modify the player-1 objective with the computed safety assumption: we extend the set of winning plays for player 1 with all plays in which player 2 follows one of the edges in E_S . Since E_S is safe-sufficient for s_I and Φ , 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 E_L^* as described in Section 5 (Theorem 12). Finally, using Theorems 8 and 9, we conclude the following.

Theorem 13. *Given an LTL formula (or ω -automaton) φ , let $\hat{K} = K(E_S) \cap K(E_L^*)$, where E_S and E_L^* are computed as described in Theorems 7 and 12. If $K \neq \emptyset$, then K is a sufficient assumption for the specification $L(\varphi)$ which is realizable by the environment. Conversely, if the Moore synthesis game for φ has a safety or Büchi objective, and if there exists a sufficient assumption $K \neq \emptyset$ for the specification $L(\varphi)$, then the computed assumption \hat{K} is nonempty.*

Recall the example from the introduction with the signals r , c , and g , and the specification $\Box(r \rightarrow \bigcirc \Diamond g) \wedge \Box((c \vee g) \rightarrow \bigcirc \neg g)$. Our algorithm computes the environment assumption $\hat{\psi}$ shown in Figure 3 (double lines indicate Büchi states). Since it is not straightforward to describe the language using an LTL formula, we give its relation to the assumptions proposed in the introduction. The computed assumption $\hat{\psi}$ includes $\psi_1 = \Box \neg c$ and $\psi_2 = \Box \Diamond \neg c$, is a strict subset of $\psi_6 = \xi W(\xi \wedge (c \vee g) \wedge \bigcirc g)$ with $\xi = r \rightarrow \bigcirc \Diamond (\neg c \vee g)$, and is incomparable to all other sufficient assumptions. Even though the computed assumption is not the weakest w.r.t. language inclusion, it still

serves its purpose: Figure 4 shows a system synthesized from the corrected specification of [12] using the environment assumption $\hat{\psi}$.

Acknowledgements. The authors thank Rupak Majumdar for stimulating and fruitful discussions. This research was supported in part by the NSF grants CCR-0132780, CNS-0720884, and CCR-0225610, by the Swiss National Science Foundation (Indo-Swiss Research Program and NCCR MICS), and by the European COMBEST project.

References

1. B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2:117–126, 1987.
2. I. Beer, S. Ben-David, U. Eisner, and Y. Rodeh. Efficient detection of vacuity in ACTL formulas. In *CAV'97*, pages 279–290. Springer, 1997.
3. J. Bernet, D. Janin, and I. Walukiewicz. Permissive strategies: from parity games to safety games. *Theoretical Informatics and Applications*, pages 261–275, 2002.
4. D. Beyer, T. A. Henzinger, and V. Singh. Algorithms for interface synthesis. In *CAV'07*, pages 4–19. Springer, 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'07*, pages 1188–1193. ACM, 2007.
6. M. G. Bobaru, C. Pasareanu, and D. Giannakopoulou. Automated assume-guarantee reasoning by abstraction refinement. In *CAV'08*. Springer, 2008. Accepted for publication.
7. K. Chatterjee, T.A. Henzinger, and B. Jobstmann. Environment assumptions for synthesis. *CoRR*, abs/0805.4167, 2008.
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*, 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. M. 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 *FMCAD'06*, pages 117–124. IEEE, 2006.
13. V. King, O. Kupferman, and M. Y. Vardi. On the complexity of parity word automata. In *FoSSaCS'06*, pages 276–286. Springer, 2001.
14. O. Kupferman and M. Y. Vardi. Vacuity detection in temporal model checking. In *CHARME'99*, pages 82–96. Springer, 1999.
15. D.A. Martin. Borel determinacy. *Annals of Mathematics*, pages 363–371, 1975.
16. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *LICS'06*, pages 255–264. IEEE, 2006.
17. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL'89*, pages 179–190. ACM, 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*, pages 1–37, 1994.
21. Vera - testbench automation. <http://www.synopsys.com/products/vera/vera.html>.