Yale University
Department of Computer Science

Games I/O Automata Play

Nick Reingold Da-Wei Wang Lenore D. Zuck

YALEU/DCS/TR-857
November 1991

This work was supported in part by the National Science Foundation under grants CCR-8910289, IRI-9015570, and CCR-8958528, by the Air Force under grant AFOSR-890382(G), and by an IBM graduate fellowship.

Games I/O Automata Play

Nick Reingold Da-Wei Wang* Lenore D. Zuck
Department of Computer Science
Yale University

Abstract

We introduce a *game* approach for specifying reactive systems. In particular, we define a simple two-player game between *System* and *Environment*, and consider the outcomes of such a game as a specification of a reactive system. We introduce six classes of game languages. We then show that the class of languages generated by I/O automata equals one of our game classes. An immediate corollary to the proof is that the fairness condition of I/O automata, which is defined as an extrinsic property by Lynch and Tuttle, can be incorporated as an intrinsic part of the automata. We also show closure properties of the six game classes. For example, we show that the class of languages defined by I/O automata is closed under union and hiding but not under intersection or complementation. The closure results are obtained by reasoning directly about games, thus demonstrating the advantage of the game-based approach.

1 Introduction

Designing correct reactive systems has been a central research topic for several years. While there is some agreement as to the importance of being able to specify reactive systems, there is not much agreement on the best way to go about it. Indeed, many different models for specification of reactive systems have been studied, for example, I/O automata ([LT87, LT89]), statecharts ([Har87]), knowledge-based protocols ([HF88]), and functional approach [Bro89, Hoa85, Mil89, BK89, Hen88], and many have been applied successfully to problems of distributed computing (e.g., [Lam83, FLS87, Har87, HZ87]). A natural question is how these models compare to one another. An obvious way to compare various formalisms is in terms of their

This work was supported in part by the National Science Foundation under grants CCR-8910289, IRI-9015570, and CCR-8958528, by the Air Force under grant AFOSR-890382(G), and by an IBM graduate fellowship.

*Current address: Department of Computer Science, University of Delaware

expressive power. Unfortunately, to date, little research has been done in comparing the expressive power of models for reactive systems.

Unlike "normal" sequential systems, whose specification is easily expressed as a relation between the input domain and the output domain, reactive systems are specified by their interaction with the environment. For example, consider a simple first-in-first-out buffer system **buf** over a domain D. The set of *actions* of **buf** is $\{read(d) : d \in D\} \cup \{write(d) : d \in D\}$, where $read(d)$ denotes that the value d is read by the buffer, and $write(d)$ denotes that the value d is written by the buffer. A specification of **buf** can be given by the set of first-in-first-out sequences over D.

Generally, a reactive system (**buf**) is associated with two types of actions, the *input* actions (e.g., $read(d)$), which are imposed on the system by the environment, and *output* actions (e.g., $write(d)$) which are imposed on the environment by the system. We adopt the trace semantics and consider the specification of a reactive system to consist of the set of sequences over the input and output actions that are allowed to occur when the system interacts with its environment.

We introduce another formalism for specifying reactive systems which we believe is quite natural and of interest in its own right. Our formalism involves viewing a reactive system as a game between the system and the environment. One advantage of this formalism is its simplicity. Another advantage is that it gives us tools to characterize the expressive power of I/O automata([LT87, LT89]) and allows us to give simple proofs of various closure and non-closure properties for systems characterized by I/O automata

The games we consider are two-player games between *System* and *Environment*. Starting with an empty sequence, the players take alternate moves and add elements to the sequence; the Environment adds elements from Σ_E^*, and the System adds elements from $\Sigma_S \cup \{\lambda\}$. A pair of strategies, one for System and one for Environment, defines an infinite sequence of alternating elements from Σ_E^* and $\Sigma_S \cup \{\lambda\}$. By concatenating the elements of this sequence we obtain its *behavior*, a sequence over the actions in $\Sigma_E \cup \Sigma_S$. Since System models the reactive system whose strategy is assumed to be known, and since Environment models the environment whose strategy is unknown, we are interested in the set of behaviors obtained when System plays its strategy against any Environment strategy. Hence, a strategy for System defines a set of sequences over the actions in Σ. We say that a set of sequences L over Σ is *deterministic game realizable*, denoted by **game**(L), if and only if L is definable by some system strategy. We denote the class of sets L which are deterministic game realizable by **game**.

We extend our notion of games to nondeterministic strategies for System and to unions of deterministic strategies for System in the obvious way. The resulting sets of sequences over Σ are termed *nondeterministic game realizable* and *union game realizable* respectively, and the resulting classes of sets of sequences are denoted **Ngame** and **Ugame** respectively. We also consider restricting strategies for System to ones which depend only on the observable behavior of the play. This leads us to three

additional classes of games, $\mathbf{game_b}$, $\mathbf{Ngame_b}$, and $\mathbf{Ugame_b}$, which are the behavior counterparts of the classes described above. These *behavior games* were also studied by Broy *et al.* [BDDW91].

The *I/O automaton* model (cf. [LT87, LT89]), which offers an elegant and appealing approach to reasoning about asynchronous concurrent computation, was used for specification, verification, and upper and lower bound proofs in many papers ([LT89, FLMW, Blo87, WLL88, LMF88, Her88, WZ91]). I/O automata are state machines that have three types of *actions*: *input actions*, which are generated by the environment and imposed on the I/O automaton, *output actions*, which are generated by the I/O automaton and imposed on the environment, and *internal actions*. Each action is a (possibly nondeterministic) state transformer. Each I/O automaton has an associated (weak) fairness condition. An *execution* of an I/O automaton is a possibly infinite sequence of alternating states and actions. The *behavior* of an execution is its restriction on the observable (input and output) actions. The specification of an I/O automaton is the set of fair behaviors it generates. Given a set of sequences L over the input and output actions, we say that L can be *generated by an automaton*, denoted by $\mathbf{loa}(L)$, if there exists an I/O automaton, whose set of fair behaviors is exactly L. We denote by **loa** the class of sets that can be generated by some I/O automaton.

We first show that **loa**, the class of languages (sets of sequences over Σ) generated by I/O automata, where the input actions are Σ_E and the output actions are Σ_S, equals **Ugame**. Since our notion of game has no explicit fairness condition, an immediate corollary to the proof is that the fairness condition of I/O automata, which is defined as an extrinsic property in [LT87], can be incorporated as an intrinsic part of the automaton.

We then compare the expressive power of the six game classes. We show that no inclusions hold among them, apart from the obvious inclusions. This demonstrates the direct correlation between the power given to the System, and the expressive power of the resulting reactive system.

Next, we consider closure properties for the classes we define. For example, we show that **loa** is closed under union and hiding but not under intersection or complementation. The proofs for closure under union and hiding are obtained by reasoning directly about games, thus demonstrating the advantage of the game-based approach.

Finally, we compare the present work with that of Broy *et al.* [BDDW91], which considered only behavior games.

2 Games, I/O Automata, and Languages

We describe reactive systems by sequences of *actions*. Actions can be internal actions of the system, which have no effect on the environment, or external actions, which describe interactions with the environment. In describing these interactions, it is helpful to classify the actions involving the system as "input" and "output" actions.

Input actions originate in the environment and are imposed on the system, while output actions are generated by the system and imposed by it on the environment.

In this section, we define two models of reactive systems. The first is our *game model*, and the second is the *I/O automaton model* ([LT87, LT89]). Both these models are described as interactions between a system and an environment. Let Σ_E and Σ_S be disjoint alphabets, and let $\Sigma = \Sigma_E \cup \Sigma_S$. Σ_E denotes the set of the environment's input actions, and Σ_S denotes the set of the system's output actions.

2.1 Games

Informally, our games are two-player games between System and Environment, where, starting with an empty "play", the players take turns making moves by appending strings to the "play". System can append a single element of Σ_S, and Environment can append any finite (possibly empty) string of elements in Σ_E. The players alternate turns, with Environment making the first move.

Formally, a *move* is an element of $\Sigma_E^* \cup \Sigma_S$. A move is a *move for Environment* if it is in Σ_E^*, and a *move for System* if it is in $\Sigma_S \cup \{\lambda\}$, where λ denotes the empty string. Note that λ can be a move for either player.

A *partial play* is either the empty sequence, $\langle\rangle$, or a sequence of moves which alternates between moves for System and moves for Environment. We do not concatenate together the elements of the sequence. Intuitively, the reason for avoiding concatenation of moves is that information could be lost, since each player is allowed to take λ-moves. Since λ can be a move for either player, a partial play should also specify the player of the first move. We will usually ignore this, as the identity of the first player will be clear from context.

For any finite partial play $\alpha = \langle a_0, a_1, \ldots, a_i\rangle$, and any partial play $\beta = \langle b_0, b_1, \ldots\rangle$ we define the *concatenation* of α and β, denoted by $\alpha \cdot \beta$ by

$$\alpha \cdot \beta = \begin{cases} \langle a_0, \ldots, a_i, b_0, \ldots\rangle & a_i \text{ and } b_0 \text{ are moves for different players} \\ \langle a_0, \ldots, a_i, \lambda, b_0, \ldots\rangle & \text{otherwise} \end{cases}$$

Let G_{odd} be the set of odd length partial plays which start with a move for Environment, let G_{even} be the set of even length partial plays which start with a move for Environment, let $G_f = G_{\text{even}} \cup G_{\text{odd}}$, let G_ω be the set of all infinite partial plays which start with a move for Environment, and let $G = G_f \cup G_\omega$. The elements of G_ω are called *plays*.

Players usually make moves according to their "strategies", which are functions from partial plays to moves. A *deterministic System strategy* is a function S mapping elements of G_{odd} to $\Sigma_S \cup \{\lambda\}$. A *deterministic Environment strategy* is a function E mapping elements of G_{even} to Σ_E^*. For a System strategy S and Environment strategy E, we define $play(S, E)$ to be the play $\langle a_0, b_0, a_1, \ldots\rangle$, where $a_0 = E(\langle\rangle)$, $b_0 = S(\langle a_0\rangle)$, $a_1 = E(\langle a_0, b_0\rangle)$, etc.

Let $\mathcal{S}$ denote the set of deterministic System strategies and $\mathcal{E}$ denote the set of deterministic Environment strategies. For a deterministic System strategy S, let $play(S, \mathcal{E})$ denote the set $\cup_{E \in \mathcal{E}} play(S, E)$, i.e., $play(S, \mathcal{E})$ is the set of plays of System playing S and Environment playing any deterministic strategy. For a set $\mathcal{S}' \subseteq \mathcal{S}$ of deterministic System strategies, let $play(\mathcal{S}', \mathcal{E})$ denote the set $\cup_{S \in \mathcal{S}'} play(S, \mathcal{E})$.

A *nondeterministic System strategy* is a function $\hat{S}$ mapping elements of G_{odd} to $2^{\Sigma_S \cup \{\lambda\}}$. For a nondeterministic System strategy $\hat{S}$, we define $D(\hat{S})$ to be the set of all deterministic System strategies S such that for all $\alpha \in G_{\text{odd}}$, $S(\alpha) \in \hat{S}(\alpha)$. For an Environment strategy E, we define $play(\hat{S}, E)$ to be $play(D(\hat{S}), E)$. That is, each $S \in D(\hat{S})$ corresponds to "fixing" choices that $\hat{S}$ might make. We say that a play α is *consistent* with a nondeterministic System strategy $\hat{S}$ if $\alpha \in play(\hat{S}, \mathcal{E})$.

For a partial play α define the *behavior* of α, written $beh(\alpha)$, as the sequence over Σ which is obtained by concatenating together all the elements of α. *Behavior games* are the games resulting when both System and Environment have to base their moves according to the behavior of partial plays rather than the plays themselves. These games are considered in [BDDW91].

For behavior games, the notions of moves, plays, and partial plays are defined as before. A *deterministic System behavior strategy* is a function S mapping Σ^* to $\Sigma_S \cup \{\lambda\}$. A *deterministic Environment behavior strategy* is a function E mapping Σ^* to Σ_E^*. For a System behavior strategy S and Environment behavior strategy E, we define $play(S, E)$ to be the play $\langle a_0, b_0, a_1, \ldots \rangle$, where $a_0 = E(\langle\rangle)$, $b_0 = S(a_0)$, $a_1 = E(a_0 b_0)$, etc. We also define a *nondeterministic System behavior strategy* and plays of such behavior strategies in the obvious way.

Let $\mathcal{S}_{\text{b}}$ denote the set of deterministic System behavior strategies and $\mathcal{E}_{\text{b}}$ denote the set of deterministic Environment behavior strategies. For a deterministic System behavior strategy S, let $play(S, \mathcal{E}_{\text{b}})$ denote the set $\cup_{E \in \mathcal{E}_{\text{b}}} play(S, E)$, i.e., $play(S, \mathcal{E}_{\text{b}})$ is the set of plays of System playing S and Environment playing any deterministic behavior strategy. For a set $\mathcal{S}' \subseteq \mathcal{S}_{\text{b}}$ of deterministic System behavior strategies, let $play(\mathcal{S}', \mathcal{E}_{\text{b}})$ denote the set $\cup_{S \in \mathcal{S}'} play(S, \mathcal{E}_{\text{b}})$.

2.2 I/O Automata

The I/O automaton model was first defined in [LT87]. See [LT87, LT89] for a complete description of the model. Here, we provide a brief summary of those parts of the model used in this paper.

I/O automata are state machines whose state to state transitions are caused by actions. Actions can be internal actions, which have no effect on the environment, or Σ actions, which describe interactions with the environment. Formally, an *I/O automaton A* (which we often call simply an *automaton*) is described by:

1. Three mutually disjoint sets of actions: $in(A) = \Sigma_E$, $out(A) = \Sigma_S$, and $int(A)$. We denote $acts(A) = \Sigma \cup int(A)$, and $loc(A) = int(A) \cup \Sigma_S$, i.e., $acts(A)$ is the set of A's actions, and $loc(A)$ is the set of A's locally controlled actions.

2. A set $states(A)$ of A's *states* and a set $initial(A) \subseteq states(A)$ of A's *initial states.*
3. A transition relation, $trans(A) \subseteq states(A) \times acts(A) \times states(A)$, that is *input enabled*, i.e., for every input action π and state s, there exists some state s' such that $(s, \pi, s') \in trans(A)$. (In general, an action π is *enabled* from a state s if for some s', $(s, \pi, s') \in trans(A)$. We denote by $enabled(s)$ the set of actions that are enabled from state s.)
4. A fairness condition, $fair(A)$, described as a partition on A's local actions with countably many equivalence classes. If $fair(A) = \{loc(A)\}$ then we say that A has the *trivial fairness partition.*

A *partial execution* η of an automaton A is a (possibly infinite) sequence of the form:

$$s_0 \pi_1 s_1 \pi_2 \ldots$$

where for every $i \geq 0$, $s_i \in states(A)$, $\pi_{i+1} \in acts(A)$, and $(s_i, \pi_{i+1}, s_{i+1})$ is a transition of A. If η is finite then it terminates in a state. We refer to s_0 as the *initial state* of η.

A partial execution is called an *execution* if its initial state is an initial state of A. A finite execution is *fair* if no local action is enabled from its last state. An infinite execution η is *fair* if for every set of local actions $\Pi \in fair(A)$, either actions from Π are taken infinitely many times in η (i.e., for infinitely many i, $\pi_i \in \Pi$), or actions from Π are not enabled infinitely many times in η (i.e., for infinitely many i, $enabled(s_i) \cap \Pi = \emptyset$).

Let B be some set and let B' be some subset of B. For every sequence α over B, we denote by $\alpha|B'$ the *restriction* of α on B', that is, the sequence obtained from α when all non-B' elements are deleted. Similarly, for a set of sequences A over B, $A|B'$ is defined in the obvious way.

The *behavior* of a partial execution η of A (and more generally, of any sequence of actions and states of A), $beh(\eta)$, is defined to be the sequence $\alpha|\Sigma$. A *fair behavior* of an automaton A is any sequence of the form $beh(\eta)$, where η is a fair execution of A. Note that our definition of behavior differs from the one in [LT87] where a behavior is defined as the restriction of execution to the actions and not only to the observable actions.

2.3 Languages

We now define the language (over Σ) described by games and by I/O automata. Denote $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. For a language $L \subseteq \Sigma^\infty$, let $pref(L)$ denote the set of prefixes for sequences in L, i.e.,

$$pref(L) = \{\sigma' \in \Sigma^* : \exists \tau \in \Sigma^\infty \quad \sigma'\tau \in L\}.$$

2.3.1 Languages Defined by Games

Recall that for a partial play α, $beh(\alpha)$ is the sequence over Σ which is obtained by concatenating together all the elements of α. For a set of plays, $\mathcal{A}$, we define $beh(\mathcal{A})$ to be $\bigcup_{\alpha \in \mathcal{A}} beh(\alpha)$. Define the *linearization* of α, written $lin(\alpha)$, as the sequence of actions contained within α where λ-moves of Environment are replaced by a new symbol, λ_E. Note that $lin(\alpha)$ is a sequence over $\Sigma \cup \{\lambda, \lambda_E\}$.

Let $L \subseteq \Sigma^\infty$. We say that L is *game realizable*, or simply $\mathbf{game}(L)$, if L is the set of behaviors of play between some System strategy S and every Environment strategy E, i.e.,

$$\mathbf{game}(L) \quad \text{iff} \quad beh(play(S, \mathcal{E})) = L \quad \text{for some } S \in \mathcal{S}.$$

We say that L is *union-game realizable*, or simply $\mathbf{Ugame}(L)$, if there exists a set of System strategies $\mathcal{S}' \subseteq \mathcal{S}$ such that $L = beh(play(\mathcal{S}', \mathcal{E}))$.

We denote by **Ugame** (resp. **game**) the set of languages L over Σ such that $\mathbf{Ugame}(L)$ (resp. $\mathbf{game}(L)$).

For a language $L \subseteq \Sigma^\infty$, we say that L is *nondeterministically game realizable*, written $\mathbf{Ngame}(L)$, if there is a nondeterministic System strategy $\hat{S}$ such that $L = beh(play(\hat{S}, \mathcal{E}))$. Since we always take the union over all possible environment strategies, we need not explicitly consider nondeterministic Environment strategies.

Let $L \subseteq \Sigma^\infty$. We say that L is *behavior game realizable*, or simply $\mathbf{game_b}(L)$, if L is the set of behaviors of plays in a behavior game between some System behavior strategy S and every Environment behavior strategy E, i.e.,

$$\mathbf{Ugame_b}(L) \quad \text{iff} \quad beh(play(S, \mathcal{E}_\mathrm{b})) = L \quad \text{for some } S \in \mathcal{S}_\mathrm{b}.$$

We say that L is *union behavior game realizable*, or simply $\mathbf{Ugame_b}(L)$, if there exists a set of System behavior strategies $\mathcal{S}' \subseteq \mathcal{S}_\mathrm{b}$ such that $L = beh(play(\mathcal{S}', \mathcal{E}_\mathrm{b}))$.

We say that L is *nondeterministically behavior game realizable* if there is a nondeterministic System behavior strategy S such that $L = beh(play(S, \mathcal{E}_\mathrm{b}))$.

The six game classes are summarized in Table 1.

	Behavior Game	General Game
single deterministic strategy	$\mathbf{game_b}$	**game**
single nondeterministic strategy	$\mathbf{Ngame_b}$	**Ngame**
union of deterministic strategies	$\mathbf{Ugame_b}$	**Ugame**

Table 1: Game classes

2.3.2 Languages Defined by I/O Automata

Let A be an automaton. The *language generated by* A, or simply $L(A)$, is the set of A's fair behaviors. A language $L \subseteq \Sigma^\infty$ is *automaton realizable*, or simply $\mathbf{loa}(L)$, if $L = L(A)$ for some automaton A. We denote by **loa** the set $\{L \subseteq \Sigma^\infty : \mathbf{loa}(L)\}$.

A special class of automaton realizable languages is the class of languages realizable by automata with the trivial fairness partition. If L is realizable by such an automaton, we denote it by $\mathsf{loat}(L)$. Similarly, we denote by loat the class of all languages L such that $\mathsf{loat}(L)$.

2.3.3 Examples of Game Languages

We now give examples of non-trivial game languages.

Example 1:
Assume $\Sigma_S = \{0,1\}$, and let h be the homomorphism on Σ defined by:

$$h(a) = \begin{cases} \lambda & a \in \Sigma_E \\ a & a \in \Sigma_S \end{cases}$$

(So, for any $\sigma \in \Sigma^\infty$, $h(\sigma) = \sigma|\Sigma_S$.) For any language L, $h^{-1}(L)$ is all strings obtained by arbitrarily inserting elements of Σ_E into sequences from L.

Consider the nondeterministic strategy $\hat{S}$ of System such that for every $\alpha \in G_{\mathsf{even}}$,

- if $beh(\alpha)$ is empty then $\hat{S}(\alpha) = \{\lambda, 0, 1\}$.
- if $beh(\alpha)$ has some 1's, then $\hat{S}(\alpha) = \{1\}$.
- in all other cases, let n_α be the number of λ-moves for Environment in α before the first non-λ move (of either Environment or System). If the number of 0-moves in α is less than n_α, then $\hat{S}(\alpha) = \{0\}$, else $\hat{S}(\alpha) = \{1\}$.

Clearly, $L_1 = beh(play(\hat{S}, \mathcal{E})) = h^{-1}(\{0^n1^\omega : n \geq 0\} \cup \{\lambda\})$. Hence L_1 is in Ngame.

Example 2:
Let Σ_S and h be as in Example 1. Let $L_2 = h^{-1}(\{0^n1^\omega : n \geq 0\})$. Note that while every prefix of 0^ω can be extended to be in L_2, 0^ω is not in L_2. In Section 4 we show that $L_2 \notin \mathsf{Ngame}$. However, L_2 is in $\mathsf{Ugame}_\mathsf{b}$: For every n, let S_n be the strategy whose first n moves are 0 and whose subsequent moves are all 1's. Then $L_2 = beh(\bigcup_n (play(S_n, \mathcal{E}_\mathsf{b})))$ thus $L_2 \in \mathsf{Ugame}_\mathsf{b}$.

2.3.4 Closure Properties of Languages

Let $\mathcal{L}$ be some class of languages over an alphabet $\Sigma = \Sigma_E \cup \Sigma_S$. We define the following closure properties of $\mathcal{L}$:

union: $\mathcal{L}$ is closed under *finite union* (resp. *countable union*, *arbitrary union*) if the union of any finite (resp. countable, arbitrary) collection of sets in $\mathcal{L}$ is in $\mathcal{L}$.

intersection: $\mathcal{L}$ is closed under *intersection* if the intersection of any finite collection of sets in $\mathcal{L}$ is in $\mathcal{L}$.

complementation: $\mathcal{L}$ is closed under *complementation* if the complement of every set in $\mathcal{L}$ (i.e., $\Sigma^\infty - L$) is also in $\mathcal{L}$.

hiding: $\mathcal{L}$ is closed under *hiding* if for every L in $\mathcal{L}$ and $a \in \Sigma$, $L \setminus a = L|(\Sigma - \{a\})$ is in $\mathcal{L}$.

renaming: $\mathcal{L}$ is closed under *renaming* if for every $\Sigma_1, \Sigma_2 \subseteq \Sigma$, every bijection $g: \Sigma_1 \to \Sigma_2$, and every set $L \subseteq \Sigma_1^\infty$ in $\mathcal{L}$, $g(L)$ is in $\mathcal{L}$ set ($g(L)$ is obtained by applying g to every sequence in L, and applying g to a sequence means applying it to each element of the sequences).

parallel composition: $\mathcal{L}$ is closed under *parallel composition* if for every $\Sigma_1, \Sigma_2 \subseteq \Sigma$ and every $L_1 \subseteq \Sigma_1^\infty$ and $L_2 \subseteq \Sigma_2^\infty$ in $\mathcal{L}$,

$$L_1 \parallel L_2 = \{\sigma \in \Sigma^\infty : \sigma|\Sigma_1 \in L_1 \text{ and } \sigma|\Sigma_2 \in L_2\}$$

is also in $\mathcal{L}$.

3 Games and I/O Automata

We now establish that a language L is generated by an automaton if and only if it is generated by an automaton with the trivial fairness partition if and only if it is union-game realizable.

In the following proofs, we use often refer to the "play" defined by a partial execution of an automaton, which is a play whose behavior equals the behavior of the partial execution. There are several ways to define such a play and the one given here (and used in the subsequent proofs) is not unique: let η be a partial execution of some automaton, and assume $\beta = beh(\eta) = a_1 a_2 \ldots$. Then $play(\eta)$ is β where a λ is inserted in between every two consecutive Σ_E or Σ_S actions, and appending an infinite tail of λ's when β is finite.

The following theorem establishes that Ugame $\subseteq$ loat.

Theorem 3.1 *For ever language* $L \subseteq \Sigma^\infty$, *if* Ugame$(L)$ *then* loat(L).

Proof Assume that Ugame(L). Fix $\mathcal{S}' \subseteq \mathcal{S}$ such that $L = beh(play(\mathcal{S}', \mathcal{E}))$. Let A be an automaton with the trivial fairness partition such that $int(A) = \{\lambda\}$, $states(A) = \mathcal{S}' \times G_f$, $initial(A) = \mathcal{S}' \times \{\langle\rangle\}$, and $trans(A)$ consists of

- $((S, \sigma), a, (S, \sigma \cdot \langle a \rangle))$, if either $a \in \Sigma_E$ or $S(\sigma) = a$,
- $((S, \sigma), a, (S, \sigma \cdot \langle \lambda, a \rangle))$, if $a = S(\sigma \cdot \langle \lambda \rangle)$.

We next show that $L = L(A)$. In one direction, let η be a fair execution of A. By the construction of A, η must be infinite. Assume that η's initial state is $(S, \langle\rangle)$. The second components of the states occurring in η are the prefixes of some infinite play $\langle a_0, b_0, \ldots\rangle$. To show that $beh(\eta) \in L$ we construct a strategy E of Environment such that $beh(play(S, E)) = beh(\eta)$. E is defined inductively as follows:

- $E(\langle\rangle) = a_0$.
- For every $i > 1$, $E(\langle a_0, b_0, \ldots, a_{i-1}, b_{i-1}\rangle) = a_i$.
- For every sequence $\sigma \in G_{\mathbf{even}}$ such that σ is not a prefix of $play(\eta)$, $E(\sigma)$ is defined arbitrarily.

Obviously, $beh(\eta) = beh(play(S, E)) \in L$. Consequently, $L(A) \subseteq L$.

In the other direction, let α be a play in L. Since $\mathbf{Ugame}(L)$, there exists some $S \in \mathcal{S}'$ and $E \in \mathcal{E}$ such that $\alpha = play(S, E)$. Assume that $lin(\alpha) = \langle \rho_1, \rho_2, \ldots\rangle$. Our goal is to construct a fair execution $\eta_\alpha = s_0 \pi_1 s_1 \pi_2 \ldots$ of A such that $beh(\alpha) = beh(\eta_\alpha)$.

For every $i \geq 0$, s_i is of the form (S, σ_i) where $\sigma_i \in G_f$. We define the σ_i's, $i \geq 0$, and the π_i's, $i \geq 1$, inductively. Obviously, $\sigma_0 = \langle\rangle$. Ideally, we would like π_i to be ρ_i and σ_i to be $\sigma_{i-1} \cdot \langle \pi_i \rangle$ for every $i \geq 1$. However, some ρ_i's could be equal to λ_E, i.e., λ-moves for E which should not be included as events of η_α, but should be included (as λ's) in the σ_i's. Consequently, when defining the π_i's, we "skip" those ρ_i's which equal to λ_E and append λ to the appropriate σ_i.

Formally, we define, for every $i \geq 1$, an integer ℓ_i which points to the element in $lin(\alpha)$ which should be first considered when constructing π_i. Let $\ell_1 = 1$. For every $i \geq 1$, we define π_i, σ_i, and ℓ_{i+1} inductively by:

- If $\rho_{\ell_i} = \lambda_E$ then $\pi_i = \rho_{\ell_i+1}$, $\sigma_i = \sigma_{i-1} \cdot \langle \lambda, \pi_i \rangle$ and $\ell_{i+1} = \ell_i + 2$.
- If $\rho_{\ell_i} \neq \lambda_E$ then $\pi_i = \rho_{\ell_i}$, $\sigma_i = \sigma_{i-1} \cdot \langle \pi_i \rangle$ and $\ell_{i+1} = \ell_i + 1$.

Since in α Environment takes infinitely many steps it follows from the construction that η_α is a fair execution of A and $play(\eta_\alpha) = \alpha$. ■

The following theorem establishes that $\mathbf{loa} \subseteq \mathbf{Ugame}$.

Theorem 3.2 *For every language $L \subseteq \Sigma^\infty$, if* $\mathbf{loa}(L)$ *then* $\mathbf{Ugame}(L)$.

Proof Let A be an I/O automaton which generates L and assume that $fair(A) = \{\Pi_1, \ldots\}$. Let $dove = dove(0), dove(1), \ldots$ be an infinite sequence over the Π_i's such that every Π_i in $fair(A)$ appears in $dove$ infinitely many times. Let $\perp$ be a symbol not in $acts(A)$.

Fix a function

$$f\colon states(A) \times 2^{acts(A)} \rightarrow (acts(A) \cup \{\perp\}) \times states(A)$$

such that for every state $s \in states(A)$ and every set $acts' \subseteq loc(A)$ the following all hold:

1. if $acts' \cap enabled(s) \neq \emptyset$ then $f(s, acts') = (a, s')$ such that $a \in acts'$ and $(s, a, s') \in trans(A)$.

2. if $acts' \cap enabled(s) = \emptyset$ but $enabled(s) \cap loc(A) \neq \emptyset$ then $f(s, acts') = (a, s')$ such that $a \in loc(A)$ and $(s, a, s') \in trans(A)$.

3. if $enabled(s) \cap loc(A) = \emptyset$ then $f(s, acts') = (\perp, s)$.

The existence of f is guaranteed by the Axiom of Choice.

Fix a function g from $states(A) \times \Sigma_E^*$ to alternating sequences of states and actions of A which start with actions, such that if $g(s, \sigma) = \eta'$ then $s\eta'$ is a finite partial execution of A and $\eta'|acts(A) = \sigma$. Again, the existence of g is guaranteed by the Axiom of Choice.

We show that for every fair execution η of A whose initial state is s_0, there exists a deterministic strategy S_η for System such that (1) $beh(play(S_\eta, \mathcal{E})) \subseteq L$ and (2) there exists some $E_\eta \in \mathcal{E}$ such that $play(\eta) = play(S_\eta, E_\eta)$.

Intuitively, S_η follows η when possible. If Environment makes a move which makes it impossible to follow η then S_η uses *dove* to ensure that the play corresponds to some fair execution of A.

Formally, for every $\gamma \in G_{\text{odd}}$ we define $S_\eta(\gamma)$ by induction on the length of γ. We also define $exec(\gamma)$, which is an partial execution of A, starting in s_0, such that $play(exec(\gamma)) = \gamma \cdot \langle S_\eta(\gamma) \rangle$. We consider three cases:

$\gamma = play(\eta)$:

Let $S_\eta(\gamma)$ be λ and $exec(\gamma)$ be the shortest prefix of η such that $play(exec(\gamma)) = \gamma$.

$\gamma \in pref(play(\eta))$ and $\gamma \neq play(\eta)$:

Assume $\gamma = \langle a_0, b_0, \ldots, a_i \rangle$ and $play(\eta) = \langle a_0, b_0, \ldots, a_i, b_i, \ldots \rangle$. Let $S_\eta(\gamma)$ be b_i and $exec(\gamma)$ be the shortest prefix of η such that $play(exec(\gamma)) = \gamma \cdot \langle b_i \rangle$.

$\gamma \notin pref(play(\eta))$:

Assume that $\gamma = \langle a_0, b_0, \ldots, a_i \rangle$ and that $exec(\gamma')$ is defined for every proper (odd length) prefix γ' of γ. Let $\gamma' = \langle a_0, b_0, \ldots, a_{i-1} \rangle$ (if $\gamma = \langle a_0 \rangle$ then $\gamma' = \langle\rangle$, and $exec(\gamma') = s_0$). Let s be the last state in $exec(\gamma')$, let s' be the last state in $g(s, a_i)$, and let $(b, s'') = f(s', dove(i))$. If $b \in out(A)$ then define $S_\eta(\gamma) = b$, else define $S_\eta(\gamma) = \lambda$. If $b \neq \perp$ then define $exec(\gamma) = exec(\gamma')g(s, a_i)bs''$, else define $exec(\gamma) = exec(\gamma')g(s, a_i)$.

■

Corollary 3.3 *For every language* $L \subseteq \Sigma^\infty$,

$$\mathsf{loa}(L) \iff \mathsf{Ugame}(L) \iff \mathsf{loat}(L).$$

Proof The claim follows immediately from Theorem 3.2, Theorem 3.1, and the observation that every automaton with the trivial fairness partition is an automaton. ■

4 Inclusion Relations Among Game Classes

It is easy to see that each general game class contains the corresponding behavior game class since a behavior strategy is a strategy. It is also easy to see that for either general or behavior games, a single deterministic strategy can be considered a nondeterministic strategy, and that a nondeterministic strategy can be considered a union of deterministic strategies. We therefore have the following inclusions of the classes:

$$\begin{array}{ccccc} \mathsf{game_b} & \subseteq & \mathsf{Ngame_b} & \subseteq & \mathsf{Ugame_b} \\ \cap\!\!| & & \cap\!\!| & & \cap\!\!| \\ \mathsf{game} & \subseteq & \mathsf{Ngame} & \subseteq & \mathsf{Ugame} \end{array} \tag{1}$$

We will show that all the $\subseteq$ relations in the above are strict, and no other inclusion relations hold other than those implied by transitivity.

We begin with a lemma which is useful in constructing languages which are not in **Ngame**.

Lemma 4.1 *Suppose Σ_s is finite, $L \in$ **Ngame**, and $L \cap \Sigma_s^* = \emptyset$. Then $L \cap \Sigma_s^\omega$ is a closed subset of Σ_s^ω.*

Proof Since L is in **Ngame**, there exists a nondeterministic System strategy $\hat{S}$ such that $L = beh(play(\hat{S}, \mathcal{E}))$. Let $E_\lambda \in \mathcal{E}$ be such that $E_\lambda(\sigma) = \lambda$ for every $\sigma \in G_{\mathbf{even}}$. Let $\beta \in \Sigma_s^\omega$ be such $pref(\beta) \subseteq pref(L \cap \Sigma_s^\omega)$. We have to show that $\beta \in L$.

Since every $\sigma \in pref(L \cap \Sigma_s^\omega)$ is in $pref(beh(play(\hat{S}, E_\lambda)))$, every prefix of β is in $pref(beh(play(\hat{S}, E_\lambda)))$. Let $\hat{\alpha}$ be the set of partial plays of $\hat{S}$ and E_λ whose behavior is a prefix of β, i.e.,

$$\hat{\alpha} = \{\alpha \in G_f : beh(\alpha) \in pref(\beta) \text{ and } \alpha \cdot \alpha' \in play(\hat{S}, E_\lambda) \text{ for some } \alpha' \in G_\omega.\}$$

Since for every prefix β' of β there is some element $\alpha \in \hat{\alpha}$ such that $beh(\alpha) = \beta'$, $\hat{\alpha}$ is infinite. The elements of $\hat{\alpha}$ define a tree, $T(\hat{\alpha})$ whose root is $\langle\rangle$ and for every α_1 and α_2 in $\hat{\alpha}$, α_1 is a child of α_2 in $T(\hat{\alpha})$ if and only if $\alpha_1 = \alpha_2 \cdot \langle a \rangle$ for some $a \in \Sigma_s \cup \{\lambda\}$. Since Σ_s is finite, $T(\hat{\alpha})$ has a finite $(|\Sigma_s| + 1)$ branching. Since $\hat{\alpha}$ is infinite, there are infinitely many nodes in $T(\hat{\alpha})$. It now follows from Kōnig's Infinity Lemma that $T(\hat{\alpha})$ has some infinite path, which yields an infinite play α. From the construction it follows that every prefix of $beh(\alpha)$ is a prefix of β. Since α is an infinite play whose behavior is in L, and $L \cap \Sigma_s^* = \emptyset$, $beh(\alpha)$ is infinite. Consequently, $beh(\alpha) = \beta$. ■

Lemma 4.2 *There is a language which is in* $\mathsf{Ugame_b}$ *but not in* **Ngame**.

Proof Consider the language L_2 of Example 2 from section 2.3.3; $L_2 = h^{-1}(\{0^n1^\omega : n \geq 0\})$, where h is the homomorphism which projects onto Σ_s. As argued in section 2.3.3, L_2 is in $\mathbf{Ugame}_b$. That L_2 is not in **Ngame** follows from the previous lemma.

■

Recall the language L_1 of Example 1, section 2.3.3; $L_1 = h^{-1}(\{0^n1^\omega : n \geq 0\} \cup \{\lambda\})$. As argued in section 2.3.3, L_1 is in **Ngame**. Notice that $L_1 \cap \Sigma_s^* = \{\lambda\}$, while $L_2 \cap \Sigma_s^* = \emptyset$. Thus, the hypothesis in Lemma 4.1 that $L \cap \Sigma_s^*$ be empty is necessary.

The following observation, which is an immediate consequence of the definition of deterministic System strategies, is used to construct languages which are not in **game**.

Observation 4.3 *For every language $L \in$* **game**, *and for every $a_1, a_2 \in \Sigma_s$,*

$$a_1, a_2 \in pref(L) \qquad \textit{iff} \qquad a_1 = a_2.$$

Lemma 4.4 *There is a language which is in* **Ngame** *but not in* **game**.

Proof Suppose that $\{0,1\} \subseteq \Sigma_s$, and let $L_3 = beh(play(\hat{S}, \mathcal{E}))$, where $\hat{S}$ is the nondeterministic System strategy defined by:

$$\hat{S}(\alpha) = \{0, 1, \lambda\} \qquad \text{for every } \alpha \in G_{\text{odd}}.$$

Obviously, $L_3 \in$ **Ngame**. That L_3 is not in **game** follows from the observation above.

■

Lemma 4.5 *There is a language which is in* $\mathbf{Ngame}_b$ *but not in* $\mathbf{game}_b$.

Proof Observe that for every language L in $\mathbf{game}_b$ and for every $a_1, a_2, b \in \Sigma_s$, if both βa_1 and βa_2 are in $pref(L)$ for some $\beta \in \Sigma^*$, then $a_1 = a_2$.

Assume that $0, 1 \in \Sigma_s$ and let $L_4 = beh(play(\hat{S}, \mathcal{E}_b))$, where $\hat{S}$ is the nondeterministic System behavior strategy such that $\hat{S}(\langle\lambda\rangle) = \{0, 1\}$. Obviously, both 0 and 1 are in $pref(beh(play(\hat{S}, \mathcal{E}_b)))$, hence L_4 violates the observation above. Consequently, $L_4 \notin \mathbf{game}_b$.

■

The following lemma about languages in $\mathbf{Ugame}_b$ will be used to show that the language L_5, defined below, is *not* in $\mathbf{Ugame}_b$.

Lemma 4.6 *Let $L \subseteq \Sigma^\infty$ be such that $L \in \mathbf{Ugame}_b$ and assume that $i \in \Sigma_E$. If $i^\omega \in L$ then $i^n \in L$ for infinitely many n's.*

Proof Assume that $i^\omega \in L$. Then there exists a System behavior strategy S and an Environment strategy E such that $play(S,E) = \langle i^{m_1}, \lambda, i^{m_2}, \lambda, \ldots \rangle$. For every $n \geq 1$ let E_n be the environment strategy which plays i^{m_k} on its k^{th} move for $k \leq n$, and plays λ thereafter. Then $beh(play(S,E_n)) = i^{(\sum_{k=1}^{n} m_k)}$. Since $\sum_k m_k = \infty$ and each m_k is finite, the claim follows. ■

Lemma 4.7 *There is a language which is in* **game** *but not in* $\mathsf{Ugame_b}$.

Proof Assume $0 \in \Sigma_s$, and let $L_5 = beh(play(S,\mathcal{E}))$ where S is the deterministic System strategy which plays 0 after every λ move of Environment, and λ after every non-λ move of Environment. Obviously, no play of S and $E \in \mathcal{E}$ has two consecutive λ moves, hence there are no finite sequences in L_5, and, in particular, $i^n \notin L_5$ for every n. However, let E be the Environmentstrategy which always plays i. It is easy to see that $beh(play(S,E)) = i^\omega$. The previous lemma shows that L_5 cannot be in $\mathsf{Ugame_b}$.

■

Theorem 4.8 *The inclusion relations of Equation (1) are all strict. Apart from inclusions which follow from (1) by transitivity, no other inclusion relations hold among the six game classes.*

Proof That $\mathsf{game_b}$ is strictly contained in $\mathsf{Ngame_b}$ follows from Lemma 4.5. If $\mathsf{Ngame_b}$ were equal to $\mathsf{Ugame_b}$ then it would follow that **Ngame** was contained in $\mathsf{Ugame_b}$, which would contradict Lemma 4.2. Therefore $\mathsf{Ngame_b}$ is strictly contained in $\mathsf{Ugame_b}$.

Lemma 4.4 shows that **game** is strictly contained in **Ngame**. If **Ngame** were equal to **Ugame** then **Ngame** would be contained in $\mathsf{Ugame_b}$, which would contradict Lemma 4.2. Therefore **Ngame** is strictly contained in **Ugame**.

If any behavior game class (i.e., a class from the top row of Equation (1)) were equal to the corresponding general game class (i.e., the game class directly below it in Equation (1)) then it would follow that **game** was contained in $\mathsf{Ugame_b}$, which would contradict Lemma 4.7.

This shows that all the inclusions of Equation (1) are strict. The second assertion of the lemma follows from Lemma 4.7, which shows that **game** cannot be contained in $\mathsf{Ugame_b}$, and Lemma 4.2, which shows that $\mathsf{Ugame_b}$ cannot be contained in **Ngame**.

■

We summarize the theorem in (2) below where $\cap$ and $\subset$ indicate proper subclass relations.

$$\begin{array}{ccccc} \mathsf{game_b} & \subset & \mathsf{Ngame_b} & \subset & \mathsf{Ugame_b} \\ \cap & & \cap & & \cap \\ \mathsf{game} & \subset & \mathsf{Ngame} & \subset & \mathsf{Ugame} \end{array} \qquad (2)$$

5 Closure Properties

	union	hiding	intersection/ complementation / parallel composition	renaming
$\mathsf{game_b}$/**game**	no	no	no	yes
$\mathsf{Ngame_b}$/**Ngame**	finite	no	no	yes
$\mathsf{Ugame_b}$	arbitrary	no	no	yes
Ugame	arbitrary	yes	no	yes

Table 2: Closure Properties of Game Classes

In this section we discuss the closure properties of the six game classes, **game**, $\mathsf{game_b}$ **Ngame**, $\mathsf{Ngame_b}$ **Ugame**, and $\mathsf{Ugame_b}$. Our results are summarized in Table 2.

We first consider renaming.

Theorem 5.1 *All six game classes are closed under renaming.*

Proof We prove the claim only for the case of **game**; the other cases are similar. Let $L = beh(play(S, \mathcal{E}))$ for some $S \in \mathcal{S}$, and assume $L \subseteq \Sigma_1^\infty$. Let $\Sigma_1 \subseteq \Sigma_2$, and let g be a bijection from Σ_1 to Σ_2. It suffices to show that $g(L)$ is in **game**. Define a strategy $S' \in \mathcal{S}$ as follows:

$$S'(\eta) = g(S(g(\eta))) \qquad \text{for every } \eta \in G_{\text{even}}.$$

It is easy to see that $g(L) = beh(play(S', \mathcal{E}))$. ■

Next we consider closure under union. Here, the "amount" of closure depends on whether System is allowed to play a deterministic strategy, a nondeterministic strategy, or a union of strategies, but does not depend on whether the strategies are required to be behavior strategies.

Lemma 5.2 Ugame *and* $\mathsf{Ugame_b}$ *are closed under arbitrary unions.*

Proof Immediate from the definition of as a union of games. ■

Theorem 5.3 loa *is closed under arbitrary unions.*

Proof This follows from Theorem 3.3 and the lemma above. ■

The main obstacle in proving the above result by reasoning directly about I/O automata is the possible need to deal with automata having different fairness partitions. After seeing the present work, Alan Fekete pointed out to us that the proof that $\mathsf{loa} = \mathsf{loat}$ could be rewritten without reference to games, yielding a purely automata based proof. This implies a purely automata based proof of Theorem 5.3.

Theorem 5.4 **Ngame**, **Ngame$_b$** *are closed under finite unions but are not closed under countable unions.*

Proof We prove the claim only for the case of **Ngame**. The case of **Ngame$_b$** is similar and left to the reader.

Let $\hat{S}_1, \hat{S}_2$ be two nondeterministic System strategies. Consider the System strategy $\hat{S}$ defined as follows. For every $\alpha \in G_{\mathbf{odd}}$,

$$\hat{S}(\alpha) = \begin{cases} \hat{S}_1(\alpha) \cup \hat{S}_2(\alpha) & \alpha \text{ is consistent with both } \hat{S}_1 \text{ and } \hat{S}_2 \\ \hat{S}_1(\alpha) & \alpha \text{ is consistent with } \hat{S}_1 \text{ but not with } \hat{S}_2 \\ \hat{S}_2(\alpha) & \alpha \text{ is consistent with } \hat{S}_2 \text{ but not with } \hat{S}_1 \\ \{\lambda\} & \alpha \text{ is not consistent with either } \hat{S}_1 \text{ or } \hat{S}_2. \end{cases}$$

It is easy to see that every behavior that is realizable by $\hat{S}_1$ or $\hat{S}_2$ is also realizable by $\hat{S}$. To prove that every behavior realizable by $\hat{S}$ is also realizable by $\hat{S}_1$ or $\hat{S}_2$, observe that each move performed by $\hat{S}$ can only make the play inconsistent with at most one strategy. If a play is inconsistent with, say $\hat{S}_1$, then the following moves performed by $\hat{S}$ are the same as those performed by $\hat{S}_2$. Therefore every play generated by $\hat{S}$ is consistent with either $\hat{S}_1$ or $\hat{S}_2$. Consequently, **Ngame** is closed under finite unions.

To see that **Ngame** is not closed under countable unions, consider the language L_2 of Example 2 (Section 2), which, as we showed in Section 4, is not in **Ngame**. Yet, it is easy to see that L_2 is a countable union of languages in **game**, thus L_2 is a countable union of languages in **Ngame**. Consequently, **Ngame** is not closed under countable unions. ■

Theorem 5.5 **game** *and* **game$_b$** *are not closed under finite union.*

Proof Assume $\Sigma_S = \{0,1\}$. For every $a \in \{0,1\}$, let S_a be the behavior strategy for System that always plays a, and let L_a be $beh(play(S_a, \mathcal{E}_b)) = (\Sigma_E^* a)^\omega$. From Observation 4.3 it follows that $L_0 \cup L_1$ is not in **game**. ■

We now show that no game class is closed under intersection, complementation, or parallel composition. That **loa** is not closed under intersection or complementation was previously shown in [WZR90].

Theorem 5.6 *No game class is closed under intersection, complementation, or parallel composition.*

Proof To show that none of the game classes is closed under intersection, note that for L_0 and L_1 as in the proof of Theorem 5.5, $L_0 \cap L_1 = \emptyset$ and $\emptyset$ is not in any game class.

By De Morgan's laws, **Ugame**, **Ngame**, **Ugame$_b$**, and **Ngame$_b$** are not closed under complementation. It is easy to see that **game**, **game$_b$** are not closed under complementation.

As for parallel composition, note that if $L_1, L_2 \subseteq \Sigma'$, then $L_1 \parallel L_2 = L_1 \cap L_2$. Therefore, since no game class is closed under intersection, none is closed under parallel composition.

■

In the remainder of the section, we consider closure under hiding. While it is easy to show that each of the game classes (and therefore **loa**) is closed under hiding of output actions, no game class except **Ugame** (and hence **loa**) is closed under hiding of input actions. Hence, no game class except **Ugame** is closed under hiding. This is shown in:

Theorem 5.7 *None of* $\textbf{Ugame}_\textbf{b}$, **Ngame**, **game**, $\textbf{Ngame}_\textbf{b}$ *and* $\textbf{game}_\textbf{b}$ *is closed under hiding of input actions.*

Proof Assume $\Sigma_E = \{a, b\}$ and $\Sigma_S = \{0, 1\}$. Consider the deterministic behavior strategy S of System such that for every $\alpha \in G_{\textbf{even}}$,

$$S(\alpha) = \begin{cases} 0 & \text{number of 0's in } \alpha < \text{number of } B\text{'s in the first element of } \alpha \\ 1 & \text{otherwise} \end{cases}$$

Let $L = beh(play(S, \mathcal{E}))$. It is easy to see that $L \setminus a$ is L_2 of Example 2 (Section 2). Hence, $L \setminus a$ is not in **Ngame**. Consequently, none of **Ngame**, **game**, $\textbf{Ngame}_\textbf{b}$, and $\textbf{game}_\textbf{b}$ is closed under hiding.

To show that $\textbf{Ugame}_\textbf{b}$ is not closed under hiding, consider the behavior system strategy S defined by:

$$S(\alpha) = \begin{cases} \lambda & \text{last play of Environment is } ab \\ 0 & \text{otherwise} \end{cases}$$

for every $\alpha \in G_{\textbf{even}}$.

It is easy to see that $a^\omega \in L \setminus b$. However, $a^n \notin L \setminus a$ for every $n > 0$. It therefore follows that $L \setminus a$ is not in $\textbf{Ugame}_\textbf{b}$. ■

Now we consider **Ugame**. Lemma 5.8 establishes that **game** (and therefore **Ugame**) is closed under hiding of output actions. Lemma 5.10 establishes that hiding an input action from a **game** language results in a **Ugame** language. Consequently, **Ugame** (and therefore **loa**) is also closed under hiding of input actions. The proofs of both lemmas are based on reasoning about strategies. We are not aware of any proof that **loa** is closed under hiding of input actions which reasons directly about automata.

Lemma 5.8 *Let* L *be in* **game** *and* a *be in* Σ_S. *Then* $L \setminus a$ *is in* **game**.

Proof Let $S \in \mathcal{S}$ be such that $L = beh(play(S, \mathcal{E}))$. We construct a system strategy $S' \in \mathcal{S}$ that replaces any a-move of S by a λ-move such that $beh(play(S', \mathcal{E})) = L \setminus a$.

For every partial play η over $\Sigma - \{a\}$, we say that η is *extendible* if there exists a prefix η' of $play(S, E)$, such that η' is obtained from η by replacing some λ-moves of

system by a. If such an η' exists, then we say that η' is an *extension* of η. Note that if η is extendible then it has a unique extension.

For every $\eta \in G_{\text{odd}}$, define

$$S'(\alpha) = \begin{cases} b & \eta \text{ is extendible and } b = S(\eta') \neq a \text{ where } \eta' \text{ is the extension of } \eta \\ \lambda & \text{otherwise} \end{cases}$$

We leave it to the reader to check that $L \setminus a = beh(play(S', \mathcal{E})$. ■

Corollary 5.9 *Let L be in* Ugame *and a be in Σ_S. Then $L \setminus a$ is in* Ugame.

Proof The proof follows immediately since any language in **Ugame** is a union of **game** sets and since, from Lemma 5.8, **game** sets are closed under hiding of output actions. ■

As shown in Theorem 5.7, **game** is not closed under hiding of input actions. Hence, the proof that **Ugame** is closed under hiding of input actions is along different lines than the previous proof. We first show that the language obtained by hiding of an input action from a **game** language is in **Ugame**.

Lemma 5.10 *Let L be in* **game** *and a be in Σ_E. Then $L \setminus a$ is in* Ugame.

Proof Let $S \in \mathcal{S}$ be such that $L = beh(play(S, \mathcal{E}))$. For every $E \in \mathcal{E}$, let $\sigma(E)$ denote the sequence $beh(play(S, E))$, so that L is the union of $\sigma(E)$ over all $E \in \mathcal{E}$. For every Environment strategy $E \in \mathcal{E}$, we construct a system strategy $S(E) \in \mathcal{S}$ such that $beh(play(S(E), \mathcal{E})$ includes $\sigma(E) \setminus a$ and is a subset of $L \setminus a$. Hence, if we let $\mathcal{S}'$ be the union (over all $E \in \mathcal{E}$) of $S(E)$, then $beh(play(\mathcal{S}', \mathcal{E})$ is exactly $L \setminus a$. It remains to show how to construct $S(E)$ for a given Environment strategy E.

Similarly to the proof of Lemma 5.8, for every partial play η over $\Sigma - \{a\}$, we say that η is *extendible* if there exists a prefix η' of $\mathbf{game}(S, E)$, such that η' is obtained from η by replacing some λ-moves of Environment with elements of a^+. If such an η' exists, then we say that η' is an *extension* of η. Note that if η is extendible then it has a unique extension.

We next define a function g from partial plays over $\Sigma - \{a\}$ to partial plays over Σ. For every extendible η, $g(\eta)$ is to be the extension of η. For an η which is not extendible, let η' be the longest extendible prefix of η and assume that $\eta = \eta' \cdot \eta''$, and let $g(\eta)$ be $g(\eta') \cdot \eta''$. For every $\eta \in G_{\text{odd}}$, define $S(E)(\eta) = S(g(\eta))$.

We leave it to the reader to check that $beh(play(S(E), \mathcal{E})$ includes $\sigma(E) \setminus a$ and that it is a subset of $L \setminus a$. ■

Corollary 5.11 *Let L be in* Ugame *and a be in Σ_E. Then $L \setminus a$ is in* Ugame.

Proof The proof is similar to that of Corollary 5.9. ■

From Corollary 5.9 and Corollary 5.11 we obtain:

Theorem 5.12 Ugame *is closed under hiding.*

6 Comparison with [BDDW91]

Broy, Dederichs, Dendorfer, and Rainer [BDDW91] considered various formalisms for describing reactive systems. In this subsection we compare our results to those of [BDDW91]. The classes defined there are *input enabled and input free.* Formally, A language $L \subseteq \Sigma^\infty$ is *input enabled and input free* if for every $\alpha \in pref(L)$:

1. for every $i \in \Sigma_E$, $\alpha i \in pref(L)$, and
2. there exists some $\beta \in \Sigma_S^\infty$ such that $\alpha\beta \in L$.

(1) means that every prefix of a string in L can be followed by any element in Σ_E to yield a prefix of a string in L, and (2) means that every prefix of a string in L can be extended by a string over Σ_S to yield a string in L.

The class of input enabled and input free languages is termed IEF in [WZR90], and *Local-SL* in [BDDW91].

Broy *et al.* define a class of languages which are generated by automata with a *strong fairness* condition. More specifically, the authors define I/O automata similar to those we define in Section 2, with the following differences:

1. the fairness partition is finite.
2. an infinite execution η is fair if actions from each fairness class that is enabled infinitely many times are taken infinitely many times. Note that we only require that, in fair executions, only actions from fairness classes that are eventually permanently enabled are infinitely many times taken.

The class of languages that are generated by automata with strong fairness is denoted *Automatic* in [BDDW91]. The class of languages that are generated by the standard automata is denoted *Automatic-WF.* Broy *et al.* also consider certain games, which are exactly our behavior games. They use *Strategic*, *Strategic-ND*, and *Fully-realizable* to denote our $\mathbf{game_b}$, $\mathbf{Ngame_b}$, and $\mathbf{Ugame_b}$ respectively. They claim that

$$\mathbf{game_b} \subset \mathbf{Ngame_b} \subset \mathbf{Ugame_b} \subset \textit{Local-SL}$$

(where $\subset$ stands for proper inclusion), and conjecture that

$$\mathbf{Ugame_b} = \mathbf{loa}.$$

However, from Corollary 3.3 and Equation (2) it immediately follows that $\mathbf{Ugame_b}$ is a proper subclass of loa, refuting the conjecture. Abadi had previously observed that the conjecture was false.

As to *Local-SL*, we show in [WZR90] that loa is a strict subclass of it, which coincides with the results of [BDDW91].

7 Conclusion and Future Work

We presented games between a powerful Environment and a System, where the players use strategies to obtain infinite sequences of symbols over a given alphabet. The games define finite- and infinite-string languages over Σ in a natural way. We considered different restrictions on the class of strategies allowed, therefore obtaining several classes of languages.

Our main result is the equivalence between the class of languages generated by I/O automata and one of our game classes. We also include comparisons between the different game classes, and a study of their closure properties. Finally, we compare our results to those in [BDDW91].

We believe that the game approach is a powerful tool for analyzing the expressive power of reactive systems. In fact, this work began when the authors tried to investigate the closure properties of **loa** languages, and failed to do so using the I/O automaton model. The same results were extremely easy to obtain by reasoning about **Ugame**. It would be interesting to apply the game approach to other reactive models.

Closed sets have generated much interest lately, partly because they describe *safety* properties of systems ([MP89]). Generally speaking, a language L over Σ is *closed* if for every $\sigma \in \Sigma^\omega$, $\sigma \in L$ if and only if every prefix of σ is in $pref(L)$. We currently do not know the exact relation between our six game classes and their closed counterparts. We do know, however, that the closed **loa** languages are in **Ngame** but not necessarily in **game**, and that they do not include all the $\mathbf{game}_b$ languages.

Another interesting topic is the notion of strong fairness and whether it can be incorporated as (a version of) a game. While it's obvious how to incorporate strong fairness in the I/O automaton model, there seems no natural way of doing that in the game model. If, indeed, there is no way, then it might imply that strong fairness is a state-dependent notion. Games have a much different notion of states, and it's possible that no game-like model can capture the notion of state needed to define strong fairness.

References

[BDDW91] M. Broy, F. Dederichs, C. Dendorfer, and R. Weber. Characterizing the behavior of reactive systems by trace sets (extended abstract). In *Third Workshop on Concurrency and Compositionality*, March 1991.

[BK89] J.A. Bergstra and J.W. Klop. Process theory based on bisimulation semantics. In *LNCS 345*, 1989.

[Blo87] Bard Bloom. Constructing two-writer atomic registers. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 249–259, 1987.

[Bro89] M. Broy. Towards a design methodology for distributed systems. In M. Broy, editor, *Constructive Methods in Computer Science*, pages 311–364. Springer, 1989.

[FLMW] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. Commutativity-based locking for nested transactions. *Journal of Computer and System Science.* to appear.

[FLS87] A. Fekete, N. Lynch, and L. Shrira. *A Modular Proof of Correctness for a Network Synchronizer*, volume 312 of *Lecture Notes in Computer Science*, pages 219–256. Springer-Verlag, Amsterdam, Netherlands, July 1987.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(33):231–278, 6 1987.

[Hen88] M. Hennessy. *Algebraic Theory of Process.* MIT Press, 1988.

[Her88] Maurice Herlihy. Impossibility and universality results for wait-free synchronization. In *Proc. 7th ACM Symp. on Principles of Distributed Computing*, pages 276–290, 1988.

[HF88] J. Y. Halpern and R. Fagin. Modeling knowledge and action in distributed systems. Technical Report, IBM, RJ6303, 1988.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Process.* Prentice-Hall, 1985.

[HZ87] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Simple knowledge-based derivations and correctness proofs for a family of protocols. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 269–280, 1987. To appear in JACM.

[Lam83] L. Lamport. Specifying concurrent program modules. *ACM Trans. on Programming Languages and Systems*, 5(2):190–222, 4 1983.

[LMF88] N. A. Lynch, Y. Mansour, and A. Fekete. Data link layer: Two impossibility results. In *Proc. 7th ACM Symp. on Principles of Distributed Computing*, pages 149–170, August 1988.

[LT87] N. A. Lynch and M. R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 137–151, August 1987.

[LT89] N. Lynch and M. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[Mil89] R. Milner. *Concurrency and Communication.* Prentice-Hall, 1989.

[MP89] Zohar Manna and Amir Pnueli. The anchored version of temporal logic. In *LNCS 354*, pages 201–284, 1989.

[WLL88] J. Welch, L. Lamport, and N. Lynch. A lattice-structured proof of a minimum spanning tree algorithm. In *Proc. 7th ACM Symp. on Principles of Distributed Computing*, pages 28–43, August 1988.

[WZ91] D.-W. Wang and L. D. Zuck. Real-time sequence transmission problem. In *Proc. 10th ACM Symp. on Principles of Distributed Computing*, August 1991.

[WZR90] D.-W. Wang, L. D. Zuck, and Nick Reingold. The power of I/O automata. Unpublished manuscript, December 1990.