Full abstraction without synchronization primitives

Andrzej S. Murawski[1]

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, UK

Abstract

Using game semantics, we prove a full abstraction result (with respect to the may-testing preorder) for Idealized Algol augmented with parallel composition ($\mathsf{IA}^{||}$). Although it is common knowledge that semaphores can be implemented using shared memory, we find that semaphores do not extend $\mathsf{IA}^{||}$ conservatively. We explain the reasons for the mismatch.

Keywords: Shared-Variable Concurrency, Mutual Exclusion, Full Abstraction, Game Semantics

1 Introduction

The mutual exclusion problem asks one to find sections of code that will allow two threads to share a single-use resource without conflict. It turns out that shared memory (with atomic reads and writes) can be used to solve it without any additional synchronization primitives. A typical solution consists of two sections of code (called *entry* and *exit* protocols respectively) that each of the two processes can use to enter and exit their designated criticial sections respectively.

> *Quite a collection of trial solutions have been shown to be incorrect and at some moment people that had played with the problem started to doubt whether it could be solved at all.*

So writes Dijkstra [3] about early attempts to attack the problem. He credits Dekker with the first correct solution, which was later simplified by several

[1] Supported by an EPSRC Advanced Research Fellowship (EP/C539753/1).

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

other authors. Peterson's tie-breaker algorithm [10], reproduced below, was particularly elegant.

```
/* Entry Code 1 */
Q[1] := 1;
turn := 1;
while (Q[2] and (turn = 1)) do skip;

/* Exit Code 1 */
Q[1] := 0;
```

```
/* Entry Code 2 */
Q[2] := 1;
turn := 2;
while (Q[1] and (turn = 2)) do skip;

/* Exit Code 2 */
Q[2] := 0;
```

Solutions to the two-process case were subsequently generalized to n processes (Lamport's bakery algorithm [7] is one of the simplest). Although the results demonstrated that, from a theoretical point of view, the sharing of memory was sufficient to enforce mutual exclusion, they were considered unsatisfactory from the conceptual and implementation-oriented points of view. The intricacy of interactions generated by the code was judged to obscure the purpose it was supposed to serve and the "busy-waiting" involved looked wasteful. This motivated the introduction of semaphores [3], a synchronization construct on a higher level than memory reads and writes.

In this paper we would like to focus on the expressive power of semaphores in the setting of shared-variable higher-order concurrency and contextual testing. We consider a variant IA of Reynolds' Idealized Algol [11] augmented with parallel composition, referred to as $\mathsf{IA}^{||}$, and prove an inequational full abstraction result for the induced notion of contextual *may-testing*. The result is obtained using game semantics by uncovering a preorder on strategies, founded on a notion reminiscent of racing computations.

Contrary to what the various mutual-exclusion algorithms might suggest, we find that there are strategies corresponding to programs with semaphores, which do not correspond to any $\mathsf{IA}^{||}$-terms. What is more, we can identify a game-semantic closure property enjoyed by all strategies corresponding to $\mathsf{IA}^{||}$-terms, which may fail in the presence of semaphores. This makes it possible to apply our model to the semantic detection of the "need for semaphores". As for contextual may-approximation and may-equivalence, we show that $\mathsf{IA}^{||}$ extended with semaphores does *not* constitute a conservative extension of $\mathsf{IA}^{||}$. We conclude by relating the apparent mismatch to non-uniformity of mutual-exclusion algorithms based on shared memory alone.

From the game-semantic perspective, our results demonstrate that a language without semaphores is considerably more difficult to handle than one incorporating them. So, the addition of communication primitives to a language can lead to cleaner mathematical structure.

Types

$$\beta ::= \mathbf{com} \mid \mathbf{exp} \mid \mathbf{var} \qquad \theta ::= \beta \mid \theta \to \theta$$

Terms

$$\frac{}{\Gamma \vdash \mathbf{skip} : \mathbf{com}} \qquad \frac{i \in \mathbb{N}}{\Gamma \vdash i : \mathbf{exp}} \qquad \frac{}{\Gamma, x : \theta \vdash x : \theta}$$

$$\frac{\Gamma \vdash M_1 : \mathbf{com} \qquad \Gamma \vdash M_2 : \beta}{\Gamma \vdash M_1 ; M_2 : \beta} \qquad \frac{\Gamma \vdash M_1 : \mathbf{exp} \qquad \Gamma \vdash M_2 : \mathbf{exp}}{\Gamma \vdash M_1 \oplus M_2 : \mathbf{exp}}$$

$$\frac{\Gamma \vdash M : \mathbf{exp} \quad \Gamma \vdash N_0 : \theta \quad \Gamma \vdash N_1 : \theta}{\Gamma \vdash \mathbf{if}\, M \,\mathbf{then}\, N_1 \,\mathbf{else}\, N_0 : \theta}$$

$$\frac{\Gamma \vdash M : \mathbf{var}}{\Gamma \vdash !M : \mathbf{exp}} \qquad \frac{\Gamma \vdash M : \mathbf{var} \quad \Gamma \vdash N : \mathbf{exp}}{\Gamma \vdash M := N : \mathbf{com}} \qquad \frac{\Gamma, x : \mathbf{var} \vdash M : \beta}{\Gamma \vdash \mathbf{newvar}\, x \,\mathbf{in}\, M : \beta}$$

$$\frac{\Gamma \vdash M : \mathbf{exp} \to \mathbf{com} \quad \Gamma \vdash N : \mathbf{exp}}{\Gamma \vdash \mathbf{mkvar}(M, N) : \mathbf{var}}$$

$$\frac{\Gamma \vdash M : \theta \to \theta' \quad \Gamma \vdash N : \theta}{\Gamma \vdash MN : \theta'} \qquad \frac{\Gamma, x : \theta \vdash M : \theta'}{\Gamma \vdash \lambda x^{\theta}.M : \theta \to \theta'} \qquad \frac{\Gamma, x : \theta \vdash M : \theta}{\Gamma \vdash \mu x^{\theta}.M : \theta}$$

Fig. 1. Syntax of IA

2 Idealized Algols

We shall be concerned with parallel extensions of Reynolds' Idealized Algol [11], which has become the canonical blueprint for synthesizing imperative and functional programming. The particular variant of Idealized Algol, presented in Figure 1 and henceforth referrred to as IA, is known in the literature as Idealized Algol with active expressions [1]. This paper is primarily devoted to IA extended with the parallel composition operator $||$. It enters the syntax through the following typing rule.

$$\frac{\Gamma \vdash M_1 : \mathbf{com} \qquad \Gamma \vdash M_2 : \mathbf{com}}{\Gamma \vdash M_1 \,||\, M_2 : \mathbf{com}}$$

We shall write $\mathsf{IA}^{||}$ to denote the extended language. Our main goal will be to arrive at a fully abstract model for contextual approximation and equivalence induced by $\mathsf{IA}^{||}$. In particular, we would like to understand how the addition of semaphores to $\mathsf{IA}^{||}$ affects the two. To that end, we consider yet another prototypical language, called PA, which is $\mathsf{IA}^{||}$ extended with semaphores. We give the syntax of PA in Figure 2. We assume that semaphores and variables are initialized to "available" and 0 respectively.

TYPES

$$\beta ::= \textbf{com} \mid \textbf{exp} \mid \textbf{var} \mid \textbf{sem} \qquad\qquad \theta ::= \beta \mid \theta \to \theta$$

TERMS

All rules defining $\mathsf{IA}^{||}$ plus the following ones

$$\frac{\Gamma \vdash M : \textbf{sem}}{\Gamma \vdash \textbf{grab}(M) : \textbf{com}} \qquad \frac{\Gamma \vdash M : \textbf{sem}}{\Gamma \vdash \textbf{release}(M) : \textbf{com}}$$

$$\frac{\Gamma, x : \textbf{sem} \vdash M : \beta}{\Gamma \vdash \textbf{newsem}\, x \,\textbf{in}\, M : \beta} \qquad \frac{\Gamma \vdash M : \textbf{com} \quad \Gamma \vdash N : \textbf{com}}{\Gamma \vdash \textbf{mksem}(M, N) : \textbf{sem}}$$

Fig. 2. Syntax of PA

Remark 2.1 PA *was introduced in [4]. It is closely related to Brookes' Parallel Algol [2], which, in contrast to* PA, *represents the coarse-grained approach to enforcing atomicity. Parallel Algol contains the* **await** M **then** N *construct which executes the guard* M *as an atomic action and, if the guard is true,* N *is run immediately afterwards, also as an indivisible operation.* PA *and Parallel Algol appear equi-expressive. Clearly, semaphores can be implemented using* **await - then** - *and ordinary variables. A translation in the other direction is also possible, for example, in the style of the encoding of Parallel Algol into the* π*-calculus [12]. We use* PA *because the game semantics we rely on is better suited to modelling fine-grained concurrency and* **await** *would have had to be interpreted indirectly by translation.*

For a closed IA-, $\mathsf{IA}^{||}$-, PA-term $\vdash M : \textbf{com}$ we shall write $M \Downarrow$ iff *there exists* a terminating run of M (the reduction rules are routine and can be found, for example, in [4]). Note that our notion of termination is *angelic*. Accordingly, the notions of contextual approximation and equivalence considered here will be consistent with *may-testing* and will not take the possibility of deadlock/divergence into account.

Definition 2.2 Let $\Gamma \vdash M_1, M_2 : \theta$ be $\mathsf{IA}^{||}$-terms. $\Gamma \vdash M_1 : \theta$ is said to ***contextually approximate*** $\Gamma \vdash M_2 : \theta$ (written $\Gamma \vdash M_1 \sqsubseteq_{\mathsf{IA}^{||}} M_2 : \theta$), if, and only if, for any $\mathsf{IA}^{||}$-context $C[-]$ such that $\vdash C[M_i] : \textbf{com}$ $(i = 1, 2)$, $C[M_1] \Downarrow$ implies $C[M_2] \Downarrow$. Further, $\Gamma \vdash M_1 : \theta$ and $\Gamma \vdash M_2 : \theta$ are ***contextually equivalent*** (written $\Gamma \vdash M_1 \cong_{\mathsf{IA}^{||}} M_2$) if each contextually approximates the other.

Analogously, one can define contextual approximation (resp. equivalence) using terms and contexts of IA or PA. We shall write $\sqsubseteq_{\mathsf{IA}}$ (respectively $\cong_{\mathsf{IA}}$) or $\sqsubseteq_{\mathsf{PA}}$ (resp. $\cong_{\mathsf{PA}}$ when referring to them). For example, it can be readily seen that $\sqsubseteq_{\mathsf{IA}^{||}}$ is not a conservative extension of IA.

Example 2.3 The two IA-terms

$$\lambda f^{\mathbf{exp}\to\mathbf{com}\to\mathbf{com}}.\mathbf{newvar}\,x\,\mathbf{in}\,f(!x)(x:=!x+2)$$
$$\lambda f^{\mathbf{exp}\to\mathbf{com}\to\mathbf{com}}.\mathbf{newvar}\,x\,\mathbf{in}\,f(!x)(x:=!x+1;x:=!x+1)$$

are IA-equivalent, but are not PA-equivalent.

The main result of our paper is an explicit characterization of $\sqsubseteq_{\mathsf{IA}^{||}}$ (Theorem 4.5) in terms of a preorder on strategies. It will allow us to demonstrate that PA is *not* a conservative extension of $\mathsf{IA}^{||}$.

Example 2.4 In view of the results given below, the simplest example illustrating the non-conservativity of PA with respect to $\mathsf{IA}^{||}$ (as far as contextual approximation is concerned) are the terms x and $x||x$, where x is a free identifier of type **com**. We shall have $x \sqsubseteq_{\mathsf{IA}^{||}} x||x$ and $x \not\sqsubseteq_{\mathsf{PA}} x||x$.

Informally, x approximates $x||x$ in $\mathsf{IA}^{||}$, because any successful run of $C[x]$ can be closely followed by that of $C[x||x]$ in which each atomic action of the second x takes place right after the corresponding action of the first x (one keeps on racing the other). In contrast, in PA, x might be instantiated with code that will try to acquire a semaphore, in which case $x||x$ will not terminate (take, for example, $C[\,] \equiv \mathbf{newsem}\,S\,\mathbf{in}\,((\lambda x^{\mathbf{com}}.[\,])\,\mathbf{grab}(S))$).

Similar terms demonstrate that contextual equivalence is not preserved either. We have $(x\,\mathbf{or}\,(x||x)) \cong_{\mathsf{IA}^{||}} (x||x)$, but $(x\,\mathbf{or}\,(x||x)) \not\cong_{\mathsf{PA}} (x||x)$, where $M\,\mathbf{or}\,N$ stands for

$$\mathbf{newvar}\,X\,\mathbf{in}\,((X:=0\,||\,X:=1);\mathbf{if}\,!X\,\mathbf{then}\,M\,\mathbf{else}\,N).$$

3 Game semantics

IA and PA have already been studied using game semantics, in [1] and [4] respectively. The full abstraction results presented therein are particularly elegant, as they characterize $\sqsubseteq_{\mathsf{IA}}$ and $\sqsubseteq_{\mathsf{PA}}$ via (complete-)play containment. Next we shall review the game model of PA (originally presented in [4]), as our full abstraction result for $\mathsf{IA}^{||}$ will be phrased in terms of strategies from that model. More precisely, we are going to exhibit a preorder, different from inclusion, that will turn out to capture contextual approximation in $\mathsf{IA}^{||}$. The induced equivalence relation, characterizing $\cong_{\mathsf{IA}^{||}}$, is also different from play equivalence.

Game semantics uses arenas to interpret types.

Definition 3.1 *An* arena A *is a triple* $\langle M_A, \lambda_A, \vdash_A\rangle$*, where*

- M_A *is a set of* moves*;*

- $\lambda_A : M_A \to \{O, P\} \times \{Q, A\}$ *is a function determining whether* $m \in M_A$ *is an* **Opponent** *or a* **Proponent** *move, a* **question** *or an* **answer***; we write* $\lambda_A^{OP}, \lambda_A^{QA}$ *for the composite of* λ_A *with respectively the first and second projections;*
- $\vdash_A$ *is a binary relation on* M_A*, called* **enabling***, such that* $m \vdash_A n$ *implies* $\lambda_A^{QA}(m) = Q$ *and* $\lambda_A^{OP}(m) \neq \lambda_A^{OP}(n)$*. Moreover, if* $n \in M_A$ *is such that* $m \not\vdash_A n$ *for any* $m \in M_A$ *then* $\lambda_A(n) = (O, Q)$*.*

If $m \vdash_A n$ we say that m *enables* n. We shall write I_A for the set of all moves of A which have no enabler; such moves are called *initial*. Note that an initial move must be an Opponent question.

In arenas used to interpret base types all questions are initial and all P-moves are answers enabled by initial moves as detailed in the table below, where $m \in \mathbb{N}$.

Arena	O-question	P-answers	Arena	O-question	P-answers
$[\![\mathbf{com}]\!]$	*run*	*done*	$[\![\mathbf{exp}]\!]$	q	m
$[\![\mathbf{var}]\!]$	*read* *write*(m)	m *ok*	$[\![\mathbf{sem}]\!]$	*grab* *release*	ok^g ok^r

Contexts and function types are modelled with the help of additional constructions on arenas:

$$\begin{array}{ll} M_{A\times B} = M_A + M_B & M_{A\Rightarrow B} = M_A + M_B \\ \lambda_{A\times B} = [\lambda_A, \lambda_B] & \lambda_{A\Rightarrow B} = [\langle \lambda_A^{PO}, \lambda_A^{QA} \rangle, \lambda_B] \\ \vdash_{A\times B} = \vdash_A + \vdash_B & \vdash_{A\Rightarrow B} = \vdash_A + \vdash_B + \{ (b, a) \mid b \in I_B \text{ and } a \in I_A \} \end{array}$$

The function $\lambda_A^{PO} : M_A \to \{O, P\}$ is defined by $\lambda_A^{PO}(m) = O$ iff $\lambda_A^{OP}(m) = P$.

Arenas provide all the details necessary to specify the allowable exchanges of moves. Formally, they will be justified sequences satisfying some extra properties. A *justified sequence* in arena A is a finite sequence of moves of A equipped with pointers. The first move is initial and has no pointer, but each subsequent move n must have a unique pointer to an earlier occurrence of a move m such that $m \vdash_A n$. We say that n is (explicitly) ***justified*** by m or, when n is an answer, that n ***answers*** m. If a question does not have an answer in a justified sequence, we say that it is *pending* (or *open*) in that sequence. In what follows we use the letters q and a to refer to question- and answer-moves respectively, o and p to stand for O- and P-moves, and m to denote arbitrary moves.

Not all justified sequences will be regarded as valid. In order to constitute a legal play, a justified sequence must satisfy a well-formedness condition, which reflects the "static" style of concurrency in PA: any process starting sub-processes must wait for the children to terminate in order to continue. In game terms: if a question is answered then all questions justified by it must have been answered earlier (exactly once). This is made precise in the following definition.

Definition 3.2 *The set P_A of* **plays** *over A consists of justified sequences s over A satisfying the two conditions below.*

FORK *In any prefix $s' = \cdots\ q \cdots m$ of s, the question q must be pending before m is played.*

WAIT *In any prefix $s' = \cdots\ q \cdots a$ of s, all questions justified by q must be answered.*

Note that interleavings of justified sequences are not justified sequences; instead we shall call them *shuffled sequences*. For two shuffled sequences s_1 and s_2, $s_1 \amalg s_2$ denotes the set of all interleavings of s_1 and s_2. For two sets of shuffled sequences S_1 and S_2, $S_1 \amalg S_2 = \bigcup_{s_1 \in S_1, s_2 \in S_2} s_1 \amalg s_2$. Given a set X of shuffled sequences, we define $X^0 = X$, $X^{i+1} = X^i \amalg X$. Then $X^{\circledast}$, called *iterated shuffle* of X, is defined to be $\bigcup_{i \in \mathbb{N}} X^i$.

We say that a subset σ of P_A is *O-complete* if $s \in \sigma$ and $so \in P_A$ entail $so \in \sigma$.

Definition 3.3 *A* strategy *σ on A (written $\sigma : A$) is a prefix-closed and O-complete subset of P_A.*

Strategies $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ are composed in the standard way, by considering all possible interactions of plays from τ with shuffled sequences of $\sigma^{\circledast}$ in the shared arena B, and then hiding the B moves.

For modelling concurrent programs, one considers a special class of so-called saturated strategies, which contain all possible (sequential) observations of the relevant (parallel) interactions. Consequently, actions of the environment (O-moves) can always be observed earlier (as soon as they have been enabled), actions of the program can always be observed later (but not later than moves that they justify). To formalize this, for any arena A, one defines a preorder $\preceq$ on P_A as the least reflexive and transitive relation satisfying $s_0 s_1 o s_2 \preceq s_0 o s_1 s_2$ and $s_0 p s_1 s_2 \preceq s_0 s_1 p s_2$ for all s_0, s_1, s_2. In the above-mentioned pairs of plays, moves on the left-hand side of $\preceq$ are meant to have the same justifiers as on the right-hand side. The two saturation conditions, in various formulations, have a long history in the semantics of concurrency [13,5,6].

Definition 3.4 *A strategy σ is* saturated *iff $s \in \sigma$ and $s \preceq s'$ imply $s' \in \sigma$.*

Arenas and saturated strategies form a Cartesian closed category $\mathcal{G}_{\mathrm{sat}}$, in which $\mathcal{G}_{\mathrm{sat}}(A, B)$ consists of saturated strategies on $A \Rightarrow B$. The identity strategy is defined by "saturating" the alternating plays $s \in P_{A_1 \Rightarrow A_2}$ in which P "copies" O-moves to the other A-component (formally, for any even-length prefix t of s we have $t \upharpoonright A_1 = t \upharpoonright A_2$). We used A_1 and A_2 to distinguish the two copies of A in the arena $A \Rightarrow A$).

PA-terms $x_1 : \theta_1, \cdots, x_n : \theta_n \vdash M : \theta$ can be interpreted in $\mathcal{G}_{\mathrm{sat}}$ as strategies in the arena $[\![\theta_1]\!] \times \cdots \times [\![\theta_n]\!] \Rightarrow [\![\theta]\!]$. The identity strategies are used to interpret free identifiers. Other elements of the syntax are interpreted by composition with designated strategies. Below we give plays defining some of them (as the least *saturated* strategies containing the plays). We use subscripts to indicate the subarena a move comes from.

	Arena	Generators
;	$[\![\mathbf{com}]\!]_0 \times [\![\beta]\!]_1 \Rightarrow [\![\beta]\!]_2$	$q_2\, run_0\, done_0\, q_1\, a_1\, a_2$
$\vert\vert$	$[\![\mathbf{com}]\!]_0 \times [\![\mathbf{com}]\!]_1 \Rightarrow [\![\mathbf{com}]\!]_2$	$run_2\, run_0\, run_1\, done_0\, done_1\, done_2$
:=	$[\![\mathbf{var}]\!]_0 \times [\![\mathbf{exp}]\!]_1 \Rightarrow [\![\mathbf{com}]\!]_2$	$run_2\, q_1\, m_1\, write(m)_0\, ok_0\, done_2$
!	$[\![\mathbf{var}]\!]_0 \Rightarrow [\![\mathbf{exp}]\!]_1$	$q_1\, read_0\, m_0\, m_1$
grab	$[\![\mathbf{sem}]\!]_0 \Rightarrow [\![\mathbf{com}]\!]_1$	$run_1\, grab_0\, ok_0^g\, done_1$
release	$[\![\mathbf{sem}]\!]_0 \Rightarrow [\![\mathbf{com}]\!]_1$	$run_1\, release_0\, ok_0^r\, done_1$
newvar	$q_2\, q_1\, (read_0\, 0_0)^*\, (\sum_{i\in\mathbb{N}}(write(i)_0\, ok_0\, (read_0\, i_0)^*))^*\, a_1\, a_2$	
newsem	$q_2\, q_1\, (grab_0\, ok_0^g\, release_0\, ok_0^r)^*\, (grab_0\, ok_0^g + \epsilon)\, a_1\, a_2$	

The strategies for variable- and semaphore-binding are for playing in arenas $([\![\mathbf{var}]\!]_0 \Rightarrow [\![\beta]\!]_1) \Rightarrow [\![\beta]\!]_2$ and $([\![\mathbf{sem}]\!]_0 \Rightarrow [\![\beta]\!]_1) \Rightarrow [\![\beta]\!]_2$ respectively.

As shown in [4], the interpretation of PA sketched above yields a fully abstract model as detailed in Theorem 3.5. A play is called ***complete*** if it does not contain unanswered questions. We write $\mathsf{comp}(\sigma)$ to denote the set of non-empty *complete* plays of the strategy σ.

Theorem 3.5 *[4] For any* PA*-terms* $\Gamma \vdash M_1 : \theta$ *and* $\Gamma \vdash M_2 : \theta$*,* $\Gamma \vdash M_1 \sqsubseteq_{\mathsf{PA}} M_2$ *if, and only if,* $\mathsf{comp}([\![\Gamma \vdash M_1]\!]) \subseteq \mathsf{comp}([\![\Gamma \vdash M_2]\!])$*. Hence,* $\Gamma \vdash M_1 \cong_{\mathsf{PA}} M_2$ *if, and only if,* $\mathsf{comp}([\![\Gamma \vdash M_1]\!]) = \mathsf{comp}([\![\Gamma \vdash M_2]\!])$*.*

We are going to prove an analogous result for $\mathsf{IA}^{||}$, though the preorder involved will be much more complicated.

4 Cloning

A shuffled sequence which is an interleaving of plays will be called a shuffled play. A shuffled play will be called *complete* if it is an interleaving of complete plays. In order to capture contextual approximation in $\mathsf{IA}^{||}$, it turns out useful to introduce an auxiliary operation on complete shuffled plays. The operation will clone part of the sequence, namely, a selected question along with all the moves that it justifies.

Formally, let s be a complete shuffled play and let q be an occurrence of a question in s. Suppose $m_1, \cdots, m_k$ are all the moves hereditarily justified by q in s and, in particular, that m_k is the answer justified by q. For convenience we write m_0 for q, so that $s = s_0 m_0 s_1 m_1 \cdots s_k m_k s_{k+1}$, where each s_i $(0 \leq i \leq k+1)$ is a possibly empty *sequence* of moves. Let us now define another sequence s_q to be s in which each m_i $(0 \leq i \leq k)$ is followed by its fresh copy m'_i, i.e.

$$s_q = s_0 m_0 m'_0 s_1 m_1 m'_1 \cdots s_k m_k m'_k s_{k+1},$$

m_0 and m'_0 are justified by the same move (from s_0, if any) and m'_i justifies m'_j $(i < j)$ if, and only if, m_i justifies m_j. We shall call m'_0 and m'_k the ***anchor points***. Intuitively, s_q can be thought of as s in which part of the play is being "shadowed", as in a racing computation. Note that if s is a complete play and q is chosen to be the initial question, then the whole of s will be cloned and s_q will become a complete shuffled play.

Definition 4.1 Given two complete shuffled plays $s, t \in P_A$, we shall write $s \ll t$ provided s contains an occurrence of a question q such that $t = s_q$. If we want to stress that q is an X-question $(X \in \{\mathrm{O}, \mathrm{P}\})$, we write $s \ll_X t$. In what follows, we shall often consider the transitive closure of the above relations, which will be denoted by $\ll^*$, $\ll^*_O$ and $\ll^*_P$ respectively.

Example 4.2 (i) Consider the following two plays in $((\llbracket\mathbf{com}\rrbracket_3 \Rightarrow \llbracket\mathbf{com}\rrbracket_2) \times \llbracket\mathbf{exp}\rrbracket_1) \Rightarrow \llbracket\mathbf{com}\rrbracket_0$.

$$\begin{array}{ccccccccc} s_1 = & r_0 & r_2 & r_3 & q_1 & 0_1 & d_3 & d_2 & d_0 \\ & O & P & O & P & O & P & O & P \end{array} \qquad \begin{array}{ccccccccccc} s_2 = & r_0 & r_2 & r_3 & r_3 & q_1 & 0_1 & d_3 & d_3 & d_2 & d_0 \\ & O & P & O & O & P & O & P & P & O & P \end{array}$$

We have omitted some pointers for the sake of clarity: in both plays r_0 justifies r_2, q_1, d_0; r_2 justifies d_2; r_0 justifies d_0, and q_1 justifies 0_1. Then $s_1 \ll_O s_2$.

(ii) Consider the following two plays in $\llbracket\mathbf{com}\rrbracket_1 \Rightarrow \llbracket\mathbf{com}\rrbracket_0$.

$$s_1 = r_0 \; r_1 \; d_1 \; d_0 \qquad s_2 = r_0 \; r_1 \; r_1 \; d_1 \; d_1 \; d_0$$

Note that $s_1 \ll_P s_2$.

Definition 4.3 Let $\sigma_1, \sigma_2 : A$. We define $\sigma_1 \leqslant \sigma_2$ to hold when for any $s_1 \in \mathsf{comp}(\sigma_1)$ there exists $s_2 \in \mathsf{comp}(\sigma_2)$ such that $s_1 \ll^*_P s_2$.

Example 4.4 $\llbracket x : \mathbf{com} \vdash x : \mathbf{com}\rrbracket \leqslant \llbracket x : \mathbf{com} \vdash x || x : \mathbf{com}\rrbracket$

$\leqslant$ underpins our full abstraction result. The remainder of the paper will be devoted to its proof.

Theorem 4.5 (Full Abstraction) *Let $\Gamma \vdash M_1, M_2 : \theta$ be $\mathsf{IA}^{||}$-terms. Then $\Gamma \vdash M_1 \sqsubseteq_{\mathsf{IA}^{||}} M_2$ if, and only if, $\llbracket\Gamma \vdash M_1\rrbracket \leqslant \llbracket\Gamma \vdash M_2\rrbracket$.*

5 Definability

First we proceed to establish the left-to-right implication of Theorem 4.5, for which we need to prove a definability result. Recall from [4] that, for any complete play s, it is possible to construct a PA-term such that the corresponding strategy is the least saturated strategy containing s. This property no longer holds for $\mathsf{IA}^{||}$-terms (this will follow from the next section in which we identify a closure property of strategies corresponding to $\mathsf{IA}^{||}$-terms). Instead we shall prove a weakened result for $\mathsf{IA}^{||}$ (Lemma 5.2).

Example 5.1 Let us write $[cond]$ for $\mathbf{if}\ cond\ \mathbf{then\,skip\,else}\ \Omega_{\mathbf{com}}$. Consider the play $s = r_0 r_1 r_2 d_2 d_1 d_0$ from $[\![(\mathbf{com}_2 \rightarrow \mathbf{com}_1) \rightarrow \mathbf{com}_0]\!]$ and the term

$$\lambda f^{\mathbf{com}\rightarrow\mathbf{com}}.\mathbf{newvar}\ X\ \mathbf{in}\ \mathbf{newsem}\ S\ \mathbf{in}\ f(\mathbf{grab}(S); X := 1); [!X = 1],$$

which is actually interpreted by the least saturated strategy containing s. When semaphores are no longer available, the"best" one can do to make sure that the assignment $X := 1$ is executed once is to protect it with the guard $[!X = 0]$ instead of $\mathbf{grab}(S)$. However, this will not prevent multiple assignments from taking place if f runs several copies of its argument in parallel (so that each can pass the test $!X = 0$ before X is set to 1). Accordingly, the strategy corresponding to

$$\lambda f^{\mathbf{com}\rightarrow\mathbf{com}}.\mathbf{newvar}\ X\ \mathbf{in}\ f([!X = 0]; X := 1); [!X = 1],$$

will contain, among others, the complete play $r_0 r_1 r_2 r_2 d_2 d_2 d_1 d_0$. In fact, the strategy contains all complete plays t such that $s \ll_O^* t$. This observation admits the following generalization.

Lemma 5.2 *Suppose Θ is an $\mathsf{IA}^{||}$-type and $s \in \mathsf{comp}(P_{[\![\Theta]\!]})$. There there exists an $\mathsf{IA}^{||}$-term $\vdash M_s : \Theta$ such that*

$$\mathsf{comp}([\![\vdash M_s]\!]) = \{\, u \mid \exists t \in P_{[\![\Theta]\!]}.\ (s \ll_O^* t \text{ and } t \preceq u)\,\}.$$

The technical details behind the construction of M_s are presented in Appendix A. Here we describe some of the underlying ideas. First of all, it is worth noting that, since saturated strategies are involved, s determines dependencies of P-moves on preceding O-moves. To enforce that order, we arrange for O-moves to generate global side-effects ($G_i := 1$) so that P-moves can only take place if the side-effects corresponding to preceding O-moves occurred.

The example above shows that with shared memory alone we are unable to control the exact number of O-moves in complete plays. However, we can make sure that whenever copies of O-moves from the original play are played, they are globally synchronized. To this end, before the corresponding flag variable G_i is set to 1, we arrange for a test $[!G_i = 0]$. This creates a "window of opportunity" for the racing O-moves, into which they have to fit if a complete play is to be reached (late arrivals will fail the test and cause divergence).

Having synchronized racing on O-moves, we also need to make sure that the "races" are consistent with s. The global side effects are not enough for that purpose as they only signal that in *one* of the races the requisite moves have been made. To ensure consistency with s in cloned subplays (i.e. to ensure that all relevant moves from s are cloned) we introduce local flags L_i, each of which is set at the same time as G_i, except that there is a *local* test whether L_i has indeed been set. It suffices to use this mechanism for O-questions only, as the presence of O-answers follows from the fact that a complete play is to be reached in the end.

Example 5.3 Consider $\theta \equiv (((\mathbf{com}_4 \to \mathbf{com}_3) \to \mathbf{com}_2) \to \mathbf{com}_1) \to \mathbf{com}_0$ and the following play $s \in P_{[\![\theta]\!]}$, in which we suppressed pointers from questions to answers.

$$\begin{array}{cccccccccc} r_0 \frown & r_1 \frown & r_2 \frown & r_3 \frown & r_4 & d_4 & d_3 & d_2 & d_1 & d_0 \\ O & P & O & P & O & P & O & P & O & P \end{array}$$

The term M_s below satisfies Lemma 5.2. Note how the presence of L_4 ensures that in any complete play from $[\![\vdash M_s]\!]$ containing two occurrences of r_2 we must also have at least one occurrence of r_4 hereditarily justified by r_2. G_4 alone would not suffice for this purpose. M_s is an optimized version of the term derived from our proof, which is shown in Appendix B. The optimizations were possible because of the particular shape of s.

$$\begin{array}{l} \lambda f.\ \mathbf{newvar}\, G_2, G_4\, \mathbf{in} \\ \qquad\qquad f(\lambda g.\ [!G_2 = 0]; G_2 := 1; \\ \qquad\qquad\qquad \mathbf{newvar}\, L_4\, \mathbf{in} \\ \qquad\qquad\qquad\qquad g([!G_4 = 0]; G_4 := 1; L_4 := 1); \\ \qquad\qquad\qquad\qquad [!L_4 = 1]); \\ \qquad [\bigwedge_{j \in \{2,4\}} (!G_j = 1)] \end{array}$$

With the definability result in place, we obtain the following corollary (its proof is available in Appendix 5.4).

Corollary 5.4 *For any* $\mathsf{IA}^{||}$*-terms* $\vdash M_1, M_2 : \theta$*, if* $\vdash M_1 \sqsubseteq_{\mathsf{IA}^{||}} M_2 : \theta$ *then* $[\![\vdash M_1 : \theta]\!] \leqslant [\![\vdash M_2 : \theta]\!]$.

6 Soundness

In this section we identify a technical property satisfied by strategies corresponding to $\mathsf{IA}^{||}$-terms. In addition to helping us complete the proof of our full abstraction result, it provides us with a tool for checking whether a given strategy might originate from an $\mathsf{IA}^{||}$-term.

Lemma 6.1 *Let $\Gamma \vdash M$ be an $\mathsf{IA}^{||}$-term, $\sigma = [\![\Gamma \vdash M]\!]$ and $s \in \mathsf{comp}(\sigma)$. Then, for any play t such that $s \ll_O^* t$, there exists $u \in \mathsf{comp}(\sigma)$ such that $t \ll_P^* u$.*

Intuitively, the Lemma asserts that, for each successful interaction between the environment and the system, the environment can always trigger others, which closely follow (race) the original blueprint. Its logical structure resembles the conditions used to characterize **mkvar**-free computation in the game semantics literature [8,9].

Before discussing the proof, let us consider a number of examples.

Example 6.2 (i) Lemma 6.1 fails for the strategy σ used to interpret semaphore-binding, generated by plays of the form

$$q_2\, q_1\, (grab_0\, ok_0^g\, release_0\, ok_0^r)^*\, (grab_0\, ok_0^g + \epsilon)\, a_1\, a_2.$$

Observe that $s = q_2\, q_1\, grab_0\, ok_0^g\, a_1\, a_2 \in \sigma$ and consider $t = q_2\, q_1\, grab_0\, grab_0\, ok_0^g\, ok_0^g\, a_1\, a_2$. Clearly $s \ll_O t$. However, note that $t \ll_P^* u$, where $u \in \mathsf{comp}(\sigma)$, must imply $t = u$ (the only P-move that can possibly be taken to support $t \ll_P^+ u$ is q_1, but plays in σ can only contain one occurrence of q_1). $t = u \in \mathsf{comp}(\sigma)$ is impossible, though, because any play from σ that contains two occurrences of ok_0^g must contain at least one occurrence of $release_0$. Consequently, the "semaphore strategy" does not satisfy Lemma 6.1.

(ii) The reasoning above does *not* apply to the strategy τ responsible for memory management. For instance, for $s = q_2\, q_1\, write(3)\, ok\, a_1\, a_2$, $t = q_2\, q_1\, write(3)\, write(3)\, ok\, ok\, a_1\, a_2$ we do have $t \in \mathsf{comp}(\tau)$, because one of the defining plays is $q_2\, q_1\, write(3)\, ok\, write(3)\, ok\, a_1\, a_2$.

(iii) The identity strategy is easily seen to satisfy Lemma 6.1.

(iv) All the other strategies corresponding to the syntax of $\mathsf{IA}^{||}$ satisfy the Lemma vacuously, because $s \in \mathsf{comp}(\sigma)$ and $s \ll_O^* t$, where t is a play, imply $s = t$.

To prove the Lemma it suffices to show that the property involved is preserved by composition. The natural approach would be to try to apply the property to the two strategies alternately with the hope of deriving it for the composite. However, given the current formulation, this alternation might seemingly have no end! To recover, we shall make the property more precise by relating the operations witnessing $t \ll_P^* u$ to those fulfulling the same task for $s \ll_O^* t$. Intuitively, we want to express the fact that each of the clonings underlying $t \ll_P^* u$ is embedded into a cloning underpinning $s \ll_O^* t$. To make the intuition precise, let us assign a fresh colour to the two anchor points involved in each step of $s \ll_O^* t$ (the colours are to stay with the moves as additional moves are being added). Then we shall say that $t \ll_P^* u$ ***occurs***

within $s \ll_O^* t$ iff for each pair of anchor points generated during the passage from t to u (according to $t \ll_P^* u$), both are between moves of the same colour.

An immediate consequence of the new requirement will be that the maximum distance (calculated in a way to be introduced) between anchor points involved in $s \ll_O^* t$ will be strictly larger than the maximum distance between anchor points generated by $t \ll_P^* u$ [2]. This is not necessarily the case for the obvious notion of distance (number of moves in-between), because $\ll$-steps add moves to plays.

Definition 6.3 Given a sequence of moves s, we define the ***alternating length*** of s to be the number of times the ownership of moves changes as we scan the sequence from left to right. The empty sequence is assumed to have alternating length 0.

For instance, $o_1 o_2 o_3$ is of (alternating) length 0, $o_1 o_2 p_2 p_3$ has length 1 and $o_1 p_1 o_2 p_3$ is of length 3. From now on, the distance between anchor points will be defined to be the *alternating length* of the segment between them (without the points). Given $s_1 \ll_X^* s_2$ we shall say that the associated ***weight*** is the largest of the distances between anchor points involved in the transitions from s_1 to s_2. Note that if $s \ll t$ then s and t have the same alternating length. Because of that, if $t \ll_P^* u$ occurs within $s \ll_O^* t$, the weight of $t \ll_P^* u$ must be strictly smaller than that of $s \ll_O^* t$.

Another consequence of "occurring within", crucial for establishing compositionality, is the fact that during composition of σ with τ, due to the embeddings, local decreases in weight effected by σ imply that the corresponding weight calculated for $\sigma^{\circledast}$ also decreases. Moreover, the decreases caused by σ and τ can be meaningfully combined. As a consequence, we can show a strengthened version of Lemma 6.1 (details in Appendix D).

Lemma 6.4 *Let $\Gamma \vdash M$ be an $\mathsf{IA}^{||}$-term, $\sigma = [\![\Gamma \vdash M]\!]$ and $s \in \mathsf{comp}(\sigma)$. Then, for any play t such that $s \ll_O^* t$, there exists $u \in \mathsf{comp}(\sigma)$ such that $t \ll_P^* u$, and $t \ll_P^* u$ occurs within $s \ll_O^* t$.*

Example 6.5 The closure property spelt out in Lemma 6.4 shows that the problems identified in Example 5.1 are unavoidable: there can be no $\mathsf{IA}^{||}$-term $\vdash M : ((\mathbf{com}_2 \rightarrow \mathbf{com}_1) \rightarrow \mathbf{com}_0$ such that $\mathsf{comp}([\![\vdash M]\!]) = \{\, r_0 r_1 r_2 d_2 d_1 d_0 \,\}$. In contrast, the PA-term

$$\lambda f^{\mathbf{com}\rightarrow\mathbf{com}}.\mathbf{newvar}\, X\, \mathbf{in}\, \mathbf{newsem}\, S\, \mathbf{in}\, f(\mathbf{grab}(S); X := 1); [!X = 1]$$

does satisfy the equation. Consequently, in PA, semaphores (in fact, even a single occurrence of **grab**) cannot be replaced with shared memory up to observational equivalence.

[2] Abusing the notation somewhat, here we regard $s \ll_O^* t$ as shorthand for a concrete sequence demonstrating that $s \ll_O^* t$.

Example 6.6 The test-and-set instruction ($\mathbf{test\text{-}set}(X)$) sets the value of the given variable to 1 and returns the old value as a single atomic (non-interruptible) operation. Observe that, if we added $\mathbf{test\text{-}set}(X)$ to $\mathsf{IA}^{||}$, we could replace

$$\lambda f^{\mathbf{com}\to\mathbf{com}}.\mathbf{newvar}\, X\, \mathbf{in}\, \mathbf{newsem}\, S\, \mathbf{in}\, f(\mathbf{grab}(S); X := 1); [!X = 1]$$

with

$$\lambda f^{\mathbf{com}\to\mathbf{com}}.\mathbf{newvar}\, X\, \mathbf{in}\, f([\mathbf{test\text{-}set}(X) = 0]); [!X = 1].$$

Since the definability argument for PA [4] only relies on "grabs" of this kind, it carries over to $\mathsf{IA}^{||} + \mathbf{test\text{-}set}$. Consequently, $\mathsf{IA}^{||} + \mathbf{test\text{-}set}$ has the same discriminating power as PA.

Using Lemma 6.4 we can eventually complete the proof of Theorem 4.5 (details in Appendix E) by showing :

Corollary 6.7 *For any* $\mathsf{IA}^{||}$*-terms* $\vdash M_1 : \theta$ *and* $\vdash M_2 : \theta$*, if* $[\![\vdash M_1 : \theta]\!] \leqslant [\![\vdash M_2 : \theta]\!]$ *then* $\vdash M_1 \sqsubseteq_{\mathsf{IA}^{||}} M_2 : \theta$*.*

7 Conclusion

We have constructed an inequationally fully abstract model of $\mathsf{IA}^{||}$ inside an existing model of PA and given an explicit characterization of contextual approximation in PA in terms of a preorder on complete plays. We have also identified a closure property that all $\mathsf{IA}^{||}$-terms satisfy and some PA-terms do not. Consequently, we can conclude that semaphores cannot be programmed in $\mathsf{IA}^{||}$ if the translation is to preserve observational equivalence (of PA-terms). So, why do the solutions to the mutual exclusion problem not apply?

The reason is that semaphores offer a *uniform* solution to the mutual exclusion problem. Whenever different processes intend to use a critical section, they can run identical entry and exit protocols ($\mathbf{grab}(S)$ and $\mathbf{release}(s)$ respectively). In contrast, existing solutions based on shared memory are not uniform, even though they are often "symmetric", in that the code run by each process depends only on its identifier. For instance, in Peterson's algorithm the codes for the two processes are the same up to the permutation that swaps 1 and 2. Such solutions will not help us to mimic the effect of $\mathbf{grab}(S)$ in, say, $f(\mathbf{grab}(S))$, because f can also make its argument run in parallel with itself, a scenario which does not arise in the framework of cooperating sequential processes.

Furthermore, our results demonstrate that PA is not a conservative extension of $\mathsf{IA}^{||}$ with respect to observational equivalence (and hence also observational approximation). Here are the simplest instances of that failure, now easily verifiable, thanks to Theorems 3.5 and 4.5.

$$\text{(i)} \quad \begin{array}{l} x : \mathbf{com} \vdash\ x\ \sqsubseteq_{\mathsf{IA}^{||}}\ (x||x) : \mathbf{com} \\ x : \mathbf{com} \vdash\ x\ \not\sqsubseteq_{\mathsf{PA}}\ (x||x) : \mathbf{com} \end{array}$$

$$\text{(ii)} \quad \begin{array}{l} x : \mathbf{com} \vdash\ (x\ \mathbf{or}\ (x||x))\ \cong_{\mathsf{IA}^{||}}\ (x||x) : \mathbf{com} \\ x : \mathbf{com} \vdash\ (x\ \mathbf{or}\ (x||x))\ \not\cong_{\mathsf{PA}}\ (x||x) : \mathbf{com} \end{array}$$

Acknowledgement

I am grateful to Samson Abramsky for discussions on the topic of this paper.

References

[1] Abramsky, S. and G. McCusker, *Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions*, in: P. W. O'Hearn and R. D. Tennent, editors, *Algol-like languages*, Birkhaüser, 1997 pp. 297–329.

[2] Brookes, S. D., *The essence of Parallel Algol*, Information and Computation **179** (2002), pp. 118–149.

[3] Dijkstra, E. W., *Cooperating sequential processes*, in: F. Genuys, editor, *Programming Languages: NATO Advanced Study Institute*, Academic Press, 1968 pp. 43–112.

[4] Ghica, D. R. and A. S. Murawski, *Angelic semantics of fine-grained concurrency*, Annals of Pure and Applied Logic **151(2-3)** (2008), pp. 89–114.

[5] Jifeng, H., M. B. Josephs and C. A. R. Hoare, *A theory of synchrony and asynchrony*, in: *Programming Concepts and Methods*, Elsevier, 1990 pp. 459–473.

[6] Laird, J., *A game semantics of Idealized CSP*, in: *Proceedings of MFPS'01*, Elsevier, 2001 pp. 1–26, ENTCS, Vol. 45.

[7] Lamport, L., *A new solution of Dijkstra's concurrent programming problem*, Communications of the ACM **17** (1974), pp. 453–455.

[8] McCusker, G., *On the semantics of Idealized Algol without the bad-variable constructor.*, in: *Proceedings of MFPS*, Electronic Notes in Theoretical Computer Science **83** (2003).

[9] Murawski, A. S., *Bad variables under control*, in: *Proceedings of CSL*, Lecture Notes in Computer Science **4646**, Springer, 2007 pp. 558–572.

[10] Peterson, G. L., *Myths about the mutual exclusion problem*, Information Processing Letters **12** (1981), pp. 115–116.

[11] Reynolds, J. C., *The essence of Algol*, in: J. W. de Bakker and J. van Vliet, editors, *Algorithmic Languages*, North Holland, 1981 pp. 345–372.

[12] Röckl, C. and D. Sangiorgi, *A pi-calculus process semantics of Concurrent Idealised Algol*, in: *Proceedings of FoSSaCS*, Lecture Notes in Computer Science **1578** (1999), pp. 306–321.

[13] Udding, J. T., *A formal model for defining and classifying delay-insensitive circuits and systems*, Distributed Computing **1(4)** (1986), pp. 197–204.

- $\beta = \mathbf{com}$: $\lambda p_1 \cdots p_h.\ SYN_i^q; (P_1 || \cdots || P_{j_i}); WAIT_{i'}$
- $\beta = \mathbf{exp}$: $\lambda p_1 \cdots p_h.\ SYN_i^q; (P_1 || \cdots || P_{j_i}); WAIT_{i'}; m_{i'}$
- $\beta = \mathbf{var}$:
  · $m_i = \mathit{read}$: $\mathbf{mkvar}(\,\lambda x^{\mathbf{exp}}.\Omega_{\mathbf{com}},\ SYN_i^q; (P_1 \,||\, \cdots \,||\, P_m); WAIT_{i'}; m_{i'})$
  · $m_i = \mathit{write}(v)$:

$$\mathbf{mkvar}(\lambda x^{\mathbf{exp}}.[x = v]; SYN_i^q; (P_1 \,||\, \cdots \,||\, P_m); WAIT_{i'},\ \Omega_{\mathbf{exp}})$$

Fig. A.1. Stage 1

Appendix

A Proof of Lemma 5.2

Let Θ be an $\mathsf{IA}^{||}$-type and $s = m_0 \cdots m_n \in \mathsf{comp}(P_{[\![\Theta]\!]})$. If m_i is an O-question, we shall write $s \upharpoonright m_i$ for the justified sequence obtained from s by erasing all moves not justified by m_i. Note that, since m_i is an O-question and s is complete, $s \upharpoonright m_i$ is also a complete play. Next we define a procedure $\textsc{Term}(i, \theta)$, where the arguments (i, θ) are such that m_i is an O-question and $s \upharpoonright m_i \in \mathsf{comp}(P_{[\![\theta]\!]})$. $\textsc{Term}(i, \theta)$ returns a $\mathsf{IA}^{||}$-term. Whenever it is called, θ will be a syntactic subtype of Θ occurring positively.

Suppose $\theta \equiv \theta_1 \to \ldots \to \theta_h \to \beta$. Let $m_{i_1}, \cdots, m_{i_{j_i}}$ and $m_{i'}$ by respectively the P-questions and P-answer justified by m_i in s. Then $\textsc{Term}(i, \theta)$ returns one of the terms specified in Figure A.1, depending on β, where

- SYN_i^q is shorthand for $[G_i = 0]; G_i := 1; L_i := 1$,
- $WAIT_j$ stands for $[\bigwedge_{x \in O_j} (G_x = 1)]$, where O_j is the set of indices of O-moves in s strictly smaller than j.

The role of SYN_i^q is to synchronize multiple occurrences of m_i by providing a time slot (before G_i is set to 1) after which clones of m_i will trigger divergence. At the same time the flag G_i is set to 1 to signal that m_i (and potential clones) has been played. Assuming m_j is a P-move, $WAIT_j$ is to be viewed as a guard that checks whether the flags corresponding to O-moves preceding m_j have been set. Terms $P_1, \cdots, P_{j_i}$ (all of type $\mathbf{com}$) will be specified in the next step. $(P_1 || \cdots || P_{j_i})$ is meant to collapse to $\mathbf{skip}$ for $j_i = 0$.

Let $1 \le k \le j_i$. Suppose that m_{i_k} is an initial move of θ_x for some $1 \le x \le h$, and that $\theta_x \equiv \theta'_1 \to \ldots \to \theta'_h \to \beta'$. For any $1 \le y \le h'$, let $i^y_{k,1}, \cdots, i^y_{k,j^y_k}$ be all the indices of O-moves justified by m_{i_k} that come from θ'_y. Let

$$P^y_k \equiv \textsc{Term}(i^y_{k,1}, \theta'_y)\ \mathbf{or}\ \cdots\ \mathbf{or}\ \textsc{Term}(i^y_{k,j^y_k}, \theta'_y).$$

P^y_k is meant to collapse to $\Omega_{\theta'_y}$ for $j^y_k = 0$.

- $\beta' = \mathbf{com}$:

$$WAIT_{i_k}; (\mathbf{newvar}\,\{\, L_j \,\}_{j\in\mathcal{L}_{i_k}}\,\mathbf{in}\,(p_x P_k^1 \cdots P_k^{h'}); [\bigwedge_{j\in\mathcal{L}_{i_k}} (L_j = 1)]); SYN^a_{i_k}$$

- $\beta' = \mathbf{exp}$:

$$\begin{array}{l} WAIT_{i_k};\\ (\mathbf{newvar}\, z, \{\, L_j \,\}_{j\in\mathcal{L}_{i_k}}\,\mathbf{in}\,(z := p_x P_k^1 \cdots P_k^{h'}); [!z = s_{i'_k}]; [\bigwedge_{j\in\mathcal{L}_{i_k}} (L_j = 1)]);\\ SYN^a_{i_k}\end{array}$$

- $\beta' = \mathbf{var}$:
  - $s_{i_k} = \mathit{read}$:

$$\begin{array}{l} WAIT_{i_k};\\ (\mathbf{newvar}\, z, L_{\vec{\mathcal{L}_{i_k}}}\,\mathbf{in}\,(z := !(p_x P_k^1 \cdots P_k^{h'})); [!z = s_{i'_k}]; [\bigwedge_{j\in\mathcal{L}_{i_k}} (L_j = 1)]);\\ SYN^a_{i_k}\end{array}$$

  - $s_{i_k} = \mathit{write}(v)$:

$$WAIT_{i_k}; (\mathbf{newvar}\, L_{\vec{\mathcal{L}_{i_k}}}\,\mathbf{in}\,(p_x P_k^1 \cdots P_k^{h'}) := v; [\bigwedge_{j\in\mathcal{L}_{i_k}} (L_j = 1)]); SYN^a_{i_k}$$

Fig. A.2. Stage 2

P_k is then defined to be one of the terms listed in Figure A.2, chosen according to the shape of β', where

- $SYN^a_{i_k}$ is shorthand for $[G_{i'_k} = 0]; G_{i'_k} := 1$ and i'_k is the index of m_{i_k}'s answer in s,
- $\mathcal{L}_{i_k} = \bigcup_{1\le y\le h'}\{\, i^y_{k,1}, \cdots, i^y_{k,j^y_k} \,\}$.

As before, the role of $SYN^a_{i_k}$ is to create a window of opportunity for $m_{i'_k}$ (and copies thereof) before signalling that the event has already taken place. $\mathcal{L}_{i_k}$ groups indices of all the O-questions justified by m_{i_k}. It is used to range over local variables that make sure exhaustive exploration of s in any race initiated by O.

To satisfy Lemma 5.2, it now suffices to call $\textsc{Term}(0,\Theta)$ and bind the free global and local variables. Suppose $\textsc{Term}(0,\Theta) \equiv \lambda p_1^{\theta_1}\cdots p_h^{\theta_h}.M$. Observe that $\{\, G_j : \mathbf{var} \,\}_{j\in O_n}, L_0 : \mathbf{var} \vdash \textsc{Term}(0,\Theta) : \Theta$. The term $\lambda p_1^{\theta_1}\cdots p_h^{\theta_h}.\mathbf{newvar}\,\{\, G_j \,\}_{j\in O_n}, L_0\,\mathbf{in}\,M$ then satisfies Lemma 5.2.

This is a consequence of the following invariant maintained throughout the construction.

Suppose $\textsc{Term}(i,\theta)$ is called during the execution of $\textsc{Term}(0,\Theta)$ and $\{\, G_j \,\}_{j\in X}, L_i \vdash \textsc{Term}(i,\theta) : \theta$ for some $X \subseteq \{\, 0,\cdots,n \,\}$. Then

$$\mathsf{comp}([\![\vdash \lambda f^{\theta\to\mathbf{com}}.\mathbf{newvar}\,\{\, G_j \,\}_{j\in X}, L_i\,\mathbf{in}\,((f\,\textsc{Term}(i,\theta)); [!L_i = 1]) : \mathbf{com}]\!])$$

equals $\{\, run_0\, run_1\, u\, done_1\, done_0 \mid \exists t \in P^{\circledast}_{[\![\theta]\!]}.(\, s \upharpoonright m_i \ll^*_O t \text{ and } t \preceq^* u) \,\}$.

B Term generated for Example 5.3

$$
\begin{array}{l}
\lambda f.\ \mathbf{newvar}\, G_0, G_2, G_4, G_6, G_8, L_0\, \mathbf{in} \\
\qquad\qquad [!G_0 = 0]; G_0 := 1; L_0 := 1; [!G_0 = 1]; \\
\qquad\qquad \mathbf{newvar}\, L_2\, \mathbf{in} \\
\qquad\qquad\qquad f(\lambda g.\ [!G_2 = 0]; G_2 := 1; L_2 := 1; [\bigwedge_{j\in\{0,2\}}(!G_j = 1)]; \\
\qquad\qquad\qquad\qquad \mathbf{newvar}\, L_4\, \mathbf{in} \\
\qquad\qquad\qquad\qquad\qquad g([!G_4 = 0]; G_4 := 1; L_4 := 1; [\bigwedge_{j\in\{0,2,4\}}(!G_j = 1)]); \\
\qquad\qquad\qquad\qquad\qquad [!L_4 = 1]; \\
\qquad\qquad\qquad\qquad [!G_6 = 0]; G_6 := 1; [\bigwedge_{j\in\{0,2,4,6\}}(!G_j = 1)]); \\
\qquad\qquad\qquad [!L_2 = 1]; \\
\qquad\qquad [!G_8 = 0]; G_8 := 1; [\bigwedge_{j\in\{0,2,4,6,8\}}(!G_j = 1)]
\end{array}
$$

C Proof of Corollary 5.4

Proof. Let $s \in \mathsf{comp}([\![\Gamma \vdash M_1 : \theta]\!])$. Consider $t = run\, s\, done \in P_{[\![\theta\rightarrow\mathbf{com}]\!]}$. By Lemma 5.2, there exists a $\mathsf{IA}^{||}$-term $\vdash M_t : \theta \rightarrow \mathbf{com}$ such that $\mathsf{comp}([\![\vdash M_t : \theta \rightarrow \mathbf{com}]\!]) = \{\, t'' \mid \exists t' \in P_{[\![\theta\rightarrow\mathbf{com}]\!]}.(t \ll^*_O t',\, t' \preceq t'') \,\}$.

Since $t \in \mathsf{comp}([\![\vdash M_t : \theta \rightarrow \mathbf{com}]\!])$, $[\![\vdash M_t M_1]\!] \neq \{\, \epsilon \,\}$. By the Soundness result from [4], $M_t M_1 \Downarrow$. Because $\vdash M_1 \sqsubseteq_{\mathsf{IA}^{||}} M_2 : \theta$, this implies $M_t M_2 \Downarrow$. By the Adequacy result from [4], $[\![\vdash M_t M_2]\!] \neq \{\, \epsilon \,\}$ follows.

Recall that $\mathsf{comp}([\![\vdash M_t : \theta \rightarrow \mathbf{com}]\!]) = \{\, t'' \mid \exists t' \in P_{[\![\theta\rightarrow\mathbf{com}]\!]}.(t \ll^*_O t',\, t' \preceq t'') \,\}$. Since each t' above is of the form $run\, s'\, done$ for some $s' \in P_{[\![\theta]\!]}$, we obtain $\mathsf{comp}([\![\vdash M_t : \theta \rightarrow \mathbf{com}]\!]) = \{\, run\, s''\, done \mid \exists s' \in P_{[\![\theta]\!]}.(s \ll^*_P s',\, s'' \preceq s') \,\}$ (in particular note that $t' \preceq t''$ translates into $s'' \preceq s'$). Since $[\![\vdash M_t M_2]\!] \neq \{\, \epsilon \,\}$, we can conclude that there exist s', s'' such that $s \ll^*_P s'$, $s'' \preceq s'$ and $s'' \in \mathsf{comp}([\![\vdash M_2]\!])$. Finally, because $[\![\vdash M_2]\!]$ is a strategy, we also have $s' \in \mathsf{comp}([\![\vdash M_2]\!])$, so $[\![\vdash M_1 : \theta]\!] \leqslant [\![\vdash M_2 : \theta]\!]$. □

D Proof of Lemma 6.4

It is easy to see that all the strategies corresponding to various parts of $\mathsf{IA}^{||}$ syntax satisfy the Lemma, as do identity strategies. So, it suffices to show that the property in question, referred to as $(\star)$ and restated below, is preserved by composition.

Let $s \in \mathsf{comp}(\sigma)$. For any play t such that $s \ll^*_O t$, there exists $u \in \mathsf{comp}(\sigma)$ such that $t \ll^*_P u$ and $t \ll^*_P u$ occurs within $s \ll^*_O t$.

We first prove a one-step variant of what we aim to establish.

Lemma D.1 *Suppose $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ satisfy $(\star)$. Let $s \in \mathsf{comp}(\sigma;\tau)$. Then, for any play t such that $s \ll_O t$, there exists $u \in \mathsf{comp}(\sigma;\tau)$ such that $t \ll_P^* u$ and $t \ll_P^* u$ occurs within $s \ll_O t$.*

Proof. Suppose that $s \ll_O t$ results from cloning in C (the alternative case of A can be dealt with in a similar way). We shall construct a (possibly infinite) sequence $u_0, u_1, u_2, \cdots$ of interaction sequences of σ and τ with the following properties, where we write u_i^L, u_i^R for $u_i \upharpoonright (A,B), u_i \upharpoonright (B,C)$ respectively and k ranges over $\mathbb{N}$.

- $u_0^L \in \sigma^{\circledast}$, $u_0^R \in \tau$
- $u_1^L \in \sigma^{\circledast}$, $u_0^R \ll_O u_1^R$, $u_0 \upharpoonright A = u_1 \upharpoonright A$
- $u_{2k+2}^R \in \tau$, $u_{2k+1}^L \ll_O^* u_{2k+2}^L$, $u_{2k+1}^R \ll_P^* u_{2k+2}^R$, $u_{2k+1} \upharpoonright A = u_{2k+2} \upharpoonright A$
- $u_{2k+3}^L \in \sigma^{\circledast}$, $u_{2k+2}^L \ll_P^* u_{2k+3}^L$, $u_{2k+2}^R \ll_O^* u_{2k+3}^R$, $u_{2k+2} \upharpoonright C = u_{2k+3} \upharpoonright C$
- $u_k \upharpoonright A, C \ll_P^* u_{k+1} \upharpoonright (A,C)$ $(k > 0)$

We take u_0 to be the witness for s. u_1 is generated from u_0 by applying the same changes as those required to pass from s to t (i.e. $u_1 \upharpoonright (A,C) = t$). Note that the conditions listed above for u_0 and u_1 will be met.

- Given u_{2k}, u_{2k+1} we construct u_{2k+2} as follows. Since $u_{2k}^R \in \tau$, $u_{2k}^R \ll_O^* u_{2k+1}^R$ and τ satisfies $(\star)$ there exists $v \in \tau$ such that $u_{2k+1}^R \ll_P^* v$. We then modify u_{2k+1} in the same way as u_{2k+1}^R turns into v to obtain u_{2k+2}. Note that A-moves in u_{2k+1} will be unaffected. If the passage from u_{2k+1} to u_{2k+2} does not involve copying any moves from B we stop the construction at u_{2k+2}. Note that then we will also have $u_{2k+1}^L = u_{2k+2}^L$.
- Given u_{2k+1}, u_{2k+2} we construct u_{2k+3} as follows. Since $u_{2k+1}^L \in \sigma^{\circledast}$, $u_{2k+1}^L \ll_O^* u_{2k+2}^L$ and σ satisfies $(\star)$, by applying $(\star)$ for each copy of σ involved in u_{2k+1}^L we obtain $v \in \sigma^{\circledast}$ such that $u_{2k+2}^L \ll_P^* v$. We then modify u_{2k+2} in the same way as u_{2k+2}^L turns into v to obtain u_{2k+3}. Note that C-moves in u_{2k+2} will be unaffected. If the passage from u_{2k+2} to u_{2k+3} does not involve copying any moves from B we stop the construction at u_{2k+3}. Note that then we will also have $u_{2k+2}^R = u_{2k+2}^R$.

If the construction terminates, i.e. $u_{2k+1}^L = u_{2k+2}^L$ or $u_{2k+2}^R = u_{2k+3}^R$, the conditions above imply that u_y, where $y = 2k+2$ or $y = 2k+3$ respectively, is an interaction sequence of σ and τ. Consequently, $u = u_y \upharpoonright (A,C) \in \sigma;\tau$ and $t = u_1 \upharpoonright (A,C) \ll_P^* u$, as required. The fact that $t \ll_P^* u$ occurs within $s \ll_O t$ follows by applying $(\star)$ alternately to τ and σ.

It suffices now to eliminate the possibility that the construction lasts forever. To that end we shall argue that the weight of $u_{2k+1}^R \ll_P^* u_{2k+2}^R$ is always greater than that of $u_{2k+3}^R \ll_P^* u_{2k+4}^R$.

By $(\star)$ for σ we have that the weight of $u^L_{2k+1} \ll^*_O u^L_{2k+2}$ is strictly greater than that of $u^L_{2k+2} \ll^*_P u^L_{2k+3}$. To conclude this we need to apply $(\star)$ separately to each thread of σ in u^L_{2k+1}. Because of "occurring within" the fact that weights decrease in each thread by $(\star)$ indeed implies that that of $u^L_{2k+1} \ll^*_O u^L_{2k+2}$ is greater than that of $u^L_{2k+2} \ll^*_P u^L_{2k+3}$.

Since the clonings supporting $u^L_{2k+1} \ll^*_O u^L_{2k+2}$ originate from some of those behind $u^R_{2k+1} \ll^*_P u^R_{2k+2}$ and some of those behind $u^L_{2k+2} \ll^*_P u^L_{2k+3}$ give rise to $u^R_{2k+2} \ll^*_O u^R_{2k+3}$, we can conclude, thanks to "occurring within", that the weight of $u^R_{2k+1} \ll^*_P u^R_{2k+2}$ must be greater than that of $u^R_{2k+2} \ll^*_O u^R_{2k+3}$. Finally, $(\star)$ applied to τ implies that the weight of $u^R_{2k+2} \ll^*_O u^R_{2k+3}$ exceeds that of $u^R_{2k+3} \ll^*_P u^R_{2k+4}$. By transitivity, the weight of $u^R_{2k+1} \ll^*_P u^R_{2k+2}$ must then exceed that of $u^R_{2k+3} \ll^*_P u^R_{2k+4}$. $\square$

To lift the result to many $\ll_O$ steps it turns out useful to examine the interaction of $\ll_O$ and $\ll_P$ steps. In particular, we will be interested in possible completions of diagrams of the form

$$\begin{tikzcd} s_1 \arrow[r, "\ll_O"] \arrow[d, "\ll_P"'] & s_2 \\ s_3 & \end{tikzcd}$$

Three cases arise, as shown below, which depend on whether the cloning involves disjoint parts of s_1 (left diagram) or not (the latter two).

$$\begin{tikzcd} s_1 \arrow[r, "\ll_O"] \arrow[d, "\ll_P"'] & s_2 \arrow[d, dashed, "\ll_P"] \\ s_3 \arrow[r, dashed, "\ll_O"'] & v \end{tikzcd} \qquad \begin{tikzcd} s_1 \arrow[rr, "\ll_O"] \arrow[d, "\ll_P"'] & & s_2 \arrow[d, dashed, "\ll_P"] \\ s_3 \arrow[r, dashed, "\ll_O"'] & \bullet \arrow[r, dashed, "\ll_O"'] & v \end{tikzcd} \qquad \begin{tikzcd} s_1 \arrow[r, "\ll_O"] \arrow[dd, "\ll_P"'] & s_2 \arrow[d, dashed, "\ll_P"] \\ & \bullet \arrow[d, dashed, "\ll_P"] \\ s_3 \arrow[r, dashed, "\ll_O"'] & v \end{tikzcd}$$

In the first case the two clonings simply commute. The second and third diagrams illustrate the case when $\ll_O$ and $\ll_P$ interfere. The second one applies if the weight of $\ll_O$ is strictly smaller than that of $\ll_P$, the third one applies if the opposite is the case (the weights cannot be the same for interfering clonings). It is worth observing that the weight of $s_1 \ll_O s_2$ (resp. $s_2 \ll_P s_3$) is the same as that of $s_3 \ll^*_O v$ (resp. $s_2 \ll^*_P v$) in each case. We shall write $s \ll_{X,k} t$ if $s \ll_X t$ and the associated weight is exactly k.

Next we extend Lemma D.1 somewhat as an auxiliary step towards proving Lemma 6.4.

Lemma D.2 *Suppose σ satisfies Lemma D.1. Let $s \in \mathsf{comp}(\sigma)$. Any play t such that $s \ll^*_{O,k} t$ satisfies Lemma 6.4.*

Proof. Suppose $s \ll^n_{O,k} t$. We reason by induction on n. For $n = 1$ the result follows from Lemma D.1.

Suppose $s \ll^n_{O,k} s' \ll_{O,k} t'$. Then we obtain the diagram below as follows. u' is obtained from the inductive hypothesis. v' is obtained by applying repeatedly the third diagram from above (the third diagram applies because the weight of $s' \ll_P^* u'$ must be smaller than the weight of $s \ll_{O,k}^n s'$, which is the same as that of $s' \ll_{O,k} t'$). w' is then obtained by applying Lemma D.1 to u' and v'.

$$\begin{tikzcd}
s \arrow[r, "\ll_{O,k}^n"] & s' \arrow[r, "\ll_{O,k}"] \arrow[d, dashed, "\ll_P^*"'] & t' \arrow[d, dashed, "\ll_P^*"] \\
& u' \arrow[r, dashed, "\ll_{O,k}"'] & v' \arrow[d, dashed, "\ll_P^*"] \\
& & w'
\end{tikzcd}$$

That $t' \ll_P^* w'$ occurs within $s \ll_O^{n+1} t$ follows from the inductive hypothesis and the way the third diagram preserves nesting. □

Finally we are ready to prove Lemma 6.4 by induction.

Proof. Suppose $s \ll_O^n t$. If $n = 1$, Lemma 6.4 follows from Lemma D.1.

Suppose $s \ll_O^n t \ll_{O,k} t'$. Then we can form the diagram below in the following way. u is obtained from the inductive hypothesis. u' is obtained by repeated applications of possibly all three diagrams discussed earlier. v' is delivered by Lemma D.2.

$$\begin{tikzcd}
s \arrow[r, "\ll_O^n"] & s' \arrow[r, "\ll_{O,k}"] \arrow[d, dashed, "\ll_P^*"'] & t' \arrow[d, dashed, "\ll_P^*"] \\
& u \arrow[r, dashed, "\ll_{O,k}^*"'] & u' \arrow[d, dashed, "\ll_P^*"] \\
& & v'
\end{tikzcd}$$

That $t' \ll_P^* v'$ occurs within $s \ll_O^{n+1} t'$ follows from the inductive hypothesis and preservation of nesting by the three diagrams. □

E Proof of Corollary 6.7

Let $\sigma_1 = [\![\vdash M_1 : \theta]\!]$, $\sigma_2 = [\![\vdash M_2 : \theta]\!]$ and suppose $\sigma_1 \leqslant \sigma_2$. Take $C[-]$ such that $\vdash C[M_1] : \mathbf{com}$ and $C[M_1] \Downarrow$. We need to show that $C[M_2] \Downarrow$.

Let $\tau = [\![x : \theta \vdash C[x] : \mathbf{com}]\!]$. By Soundness [4] $\sigma_1; \tau \neq \{\epsilon\}$. Let $\mathit{run}\, s_0\, \mathit{done} \in \sigma_1^{\circledast} || \tau$ be the corresponding witness.

We shall construct a sequence (potentially infinite) $s_1, s_2, \cdots$ of complete shuffled plays such that, for any $k \geq 0$,

- $s_{2k+1} \in \sigma_2^{\circledast}$, $\mathit{run}\, s_{2k}\, \mathit{done} \ll_O^* \mathit{run}\, s_{2k+1}\, \mathit{done}$;

- $run\, s_{2k+2}\, done \in \tau$, $s_{2k+1} \ll_O^* s_{2k+2}$.

Here is how the sequence is constructed.

- Because $\sigma_1 \leqslant \sigma_2$, there exists $s_1 \in \sigma_2^{\circledast}$ such that $s_0 \ll_P^{\circledast} s_1$. Hence, $run\, s_0\, done \ll_O^* run\, s_1\, done$
- Given s_{2k}, s_{2k+1} such that $run\, s_{2k}\, done \in \tau$ and $run\, s_{2k}\, done \ll_O^* run\, s_{2k+1}\, done$, by Lemma 6.4 for τ, there exists s_{2k+2} such that $run\, s_{2k+2}\, done \in \tau$ and $run\, s_{2k+1}\, done \ll_P^* run\, s_{2k+2}\, done$. Hence, $s_{2k+1} \ll_O^* s_{2k+2}$, as required. If $s_{2k+1} = s_{2k+2}$ we terminate the construction (s_{2k+2} is the last element of the sequence).
- Given s_{2k+1}, s_{2k+2} such that $s_{2k+1} \in \sigma_2^{\circledast}$ and $s_{2k+1} \ll_O^* s_{2k+2}$, by Lemma 6.4 for each thread of σ_2 in s_{2k+1}, one can obtain $s_{2k+3} \in \sigma_2^{\circledast}$ such that $s_{2k+2} \ll_P^* s_{2k+3}$, i.e. $run\, s_{2k+2}\, done \ll_O^* s_{2k+3}$. If $s_{2k+2} = s_{2k+3}$ we terminate the construction at s_{2k+3}.

Note that if the construction stabilizes we have $s_{2k+1} = s_{2k+2}$ or $s_{2k+2} = s_{2k+3}$. By the properties of the sequence mentioned above, in both cases we then have $run\, s_{2k+2}\, done \in \sigma_2^{\circledast} || \tau$, i.e. $\sigma_2; \tau \neq \{\,\epsilon\,\}$. By Adequacy [4], it follows that $C[M_2] \Downarrow$.

To wrap up the proof we show that the construction always stabilizes. Suppose we end up with an infinite sequence $s_0, s_1, s_2 \cdots$. Let $t_i = run\, s_i\ done$. Then we have an infinite sequence of the form

$$t_0 \ll_P^* t_1 \ll_O^* t_2 \ll_P^* t_3 \ll_O^* t_4 \ll_P^* \cdots.$$

Let us write w_k for the weight of $t_k \ll_X^* t_{k+1}$. We shall argue that w_k always decreases and, hence, cannot be infinite. By Lemma 6.4 for τ and the construction we clearly have $w_k > w_{k+1}$ when k is even. For k odd, the same Lemma applied to σ_2 only tells us that the weight decreases locally (in each thread of σ_2). However, because of nesting ("occurs within"), this also implies a global decrease (calculated within the shuffled sequences s_{2k} and s_{2k+1}).