A Truly Concurrent Game Model of the Asynchronous π-Calculus

Ken Sakayori(✉) and Takeshi Tsukada

The University of Tokyo, Tokyo, Japan
{sakayori,tsukada}@kb.is.s.u-tokyo.ac.jp

Abstract. In game semantics, a computation is represented by a *play*, which is traditionally a sequence of messages exchanged by a program and an environment. Because of the sequentiality of plays, most game models for concurrent programs are a kind of interleaving semantics. Several frameworks for truly concurrent game models have been proposed, but no model has yet been applied to give a semantics of a complex concurrent calculus such as the π-calculus (with replication).

This paper proposes a truly concurrent version of the HO/N game model in which a play is not a sequence but a directed acyclic graph (DAG) with two kinds edges, justification pointers and causal edges. By using this model, we give the first truly concurrent game semantics for the asynchronous π-calculus. In order to illustrate a possible application, we propose an intersection type system for the asynchronous π-calculus by means of our game model, and discuss when a process can be completely characterised by the intersection type system.

Keywords: HO/N game model · True concurrency · Asynchronous π-calculus

1 Introduction

Game semantics succeeded to give semantics for variety of programming languages such as PCF [1,21] and Idealized Algol [2].

The idea of game semantics has been applied to give models for concurrent calculi such as CSP [23], Idealized Parallel Algol [18] and the asynchronous π-calculus [24]. However, the sequential nature of plays forces these models to be a kind of interleaving semantics; the causalities between events are obfuscated.

Hence it is natural to investigate a concurrent extension of the traditional game models. Several frameworks for concurrent game models have been proposed by several researchers [3,27,29,34], but no model has yet been applied to give a semantics of a complex concurrent calculus such as the π-calculus (with replication), as pointed out in [10]. The goal of this paper is to develop a truly concurrent game model by which the asynchronous π-calculus can be interpreted.

The starting point of our development is an observation by Melliès [27]: in the HO/N innocent game model [21,32], only a part of the sequential information

J. Esparza and A.S. Murawski (Eds.): FOSSACS 2017, LNCS 10203, pp. 389–406, 2017.
DOI: 10.1007/978-3-662-54458-7_23

Fig. 1. Idea of the desequentialization.

is really relevant. For example, the order of consecutive occurrences of O- and P-moves are indispensable, whereas that of consecutive occurrences of P- and O-moves can be safely forgotten (unless the O-move is justified by the P-move).

Now it is natural to think of a play in which the relevant order information is made explicit. Consider a traditional sequential play on the left side in Fig. 1, where • (resp. ∘) represents an O-move (resp. a P-move) and a pointer is a justification pointer. By making the relevant sequential information explicit, we obtain a representation in the middle in Fig. 1. Then because all the relevant sequential information has been explicitly indicated by edges, we can simply forget the sequential information, resulting in the right representation in Fig. 1. This is our representation of a play that we call a *DAG-based play*.

A DAG-based play generated by this way from a sequential play satisfies a certain property, which reflects the sequential nature of the target language of the innocent game model [21]. In order to model a concurrent calculus, the condition required for DAG-based plays should be weakened. This is the idea that leads us to the definition of plays in this paper.

Following this idea, we develop a DAG-based game model for the asynchronous π-calculus, guided by the sequential game model of Laird [24]. Our model is truly concurrent in the sense that it distinguishes between $a.\bar{b} \mid c.\bar{d}$ and $a.(\bar{b} \mid c.\bar{d}) + c.(a.\bar{b} \mid \bar{d})$. Laird's model can be reconstructed by lining up the nodes of DAG-based plays of our model. We prove the soundness of our model by reducing it to that of Laird's model, using this relationship.

As a possible application of our model, we give an intersection type system based on the relationship between intersection types and game semantics which has been studied in the case of λ-calculus [7,14,37]. Based on a game-semantic consideration, we characterise a class of processes that are completely described by the intersection type system.

Organisation of the paper. Section 2 defines our target language, a variant of the asynchronous π-calculus. In Sect. 3, we define our truly concurrent game model and relate it with sequential game models. A semantics of the π-calculus is given in Sect. 4. Section 5 illustrates a possible application of our game model, giving an intersection type system for a fragment of the π-calculus. Section 6 discusses related work and Sect. 7 concludes the paper.

$$\frac{}{\Gamma \vdash \mathbf{0}; \Sigma} \qquad \frac{\Gamma, \bar{x} : T \vdash P; \Sigma, y : T}{\Gamma \vdash \nu(\bar{x}, y).P; \Sigma} \qquad \frac{}{\Gamma, \bar{x} : \mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}], \bar{\boldsymbol{y}} : \boldsymbol{S} \vdash \bar{x}\langle \bar{\boldsymbol{y}}, \boldsymbol{z} \rangle; \Sigma, \boldsymbol{z} : \boldsymbol{T}}$$

$$\frac{\Gamma \vdash P; \Sigma \quad \Gamma' \vdash Q; \Sigma'}{\Gamma, \Gamma' \vdash P|Q; \Sigma, \Sigma'} \qquad \frac{\Gamma, \bar{\boldsymbol{y}} : \boldsymbol{S} \vdash P; \Sigma, \boldsymbol{z} : \boldsymbol{T}}{\Gamma \vdash x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P; \Sigma, x : \mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}]} \qquad \frac{\Gamma, \bar{\boldsymbol{y}} : \boldsymbol{S} \vdash P; \Sigma, \boldsymbol{z} : \boldsymbol{T}}{\Gamma \vdash !x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P; \Sigma, x : \mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}]}$$

Fig. 2. Typing rules. (Contraction and exchange rules are omitted.)

2 Simply-Typed Asynchronous π-Calculus

We define the target language of the paper: the simply-typed asynchronous polyadic π-calculus with distinction between input and output channels. This is the calculus studied in the previous work of Laird [24], in which he gave an interleaving (or sequential) game model.

We assume countably infinite sets of input names and of output names. Unlike the standard π-calculus in which an input name a is *a priori* connected to the output name $\bar{a}$, we do not assume any relationship between input and output names but a connection is established by ν constructor. This design choice significantly simplifies the denotational semantics.

The *processes* are defined by the following grammar: $P, Q ::= \mathbf{0} \mid \bar{x}\langle \bar{\boldsymbol{y}}, \boldsymbol{z} \rangle \mid x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P \mid P|Q \mid !x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P \mid \nu(\bar{x}, y).P$. Here x (resp. $\bar{x}$) ranges over input (resp. output) names and $\boldsymbol{x}$ (resp. $\bar{\boldsymbol{x}}$) represents a (possibly empty) sequence of input (resp. output) names. Name creation ν creates a pair of input and output names. We abbreviate $\nu(\bar{x}_1, y_1). \ldots .\nu(\bar{x}_n, y_n).P$ as $\nu(\bar{x}_1 \ldots \bar{x}_n, y_1 \ldots y_n).P$.

The structural congruence $\equiv$ is defined as usual. The *one-step reduction relation* $\longrightarrow$ on processes is defined by the following rule:

$$\nu(\bar{\boldsymbol{z}}, \boldsymbol{w}).\nu(\bar{x}, y).(y(\bar{\boldsymbol{a}}, \boldsymbol{b}).P \mid \bar{x}\langle \bar{\boldsymbol{c}}, \boldsymbol{d} \rangle \mid Q) \longrightarrow \nu(\bar{\boldsymbol{z}}, \boldsymbol{w}).\nu(\bar{x}, y).(P\{\bar{\boldsymbol{c}}/\bar{\boldsymbol{a}}, \boldsymbol{d}/\boldsymbol{b}\} \mid Q)$$

It is worth emphasising here that the communication only occurs over names that are bound by ν. The *reduction relation* $\longrightarrow^*$ is the reflexive transitive closure of $(\longrightarrow \cup \equiv)$. We write $P \Downarrow_{\bar{x}}$ if $P \longrightarrow^* \nu(\bar{\boldsymbol{y}}, \boldsymbol{z}).(\bar{x}\langle \bar{\boldsymbol{y}}', \boldsymbol{z}' \rangle \mid Q)$ for some Q, where $\bar{x}$ is free. Note that we can observe only an output action.

We require that processes are well-typed. The syntax of *types* is given by $S, T ::= \mathbf{ch}[S_1 \ldots S_m, T_1 \ldots T_n]$. We write $x : \mathbf{ch}[S_1 \ldots S_m, T_1 \ldots T_n]$ to mean that x is an input name by which one receives m output names and n input names at once. Similarly for $\bar{y} : \mathbf{ch}[S_1 \ldots S_m, T_1 \ldots T_n]$. A sequence $S_1 \ldots S_m$ of types is often written as $\boldsymbol{S}$ and the empty sequence is written as $\_$. The type $\mathbf{ch}[\_, \_]$ is abbreviated as $\mathbf{ch}[]$. An *input type environment* is a finite sequence of type bindings of the form $x : T$ and an *output type environment* is that of the form $\bar{y} : S$. A *type judgement* is of the form $\Gamma \vdash P; \Sigma$, where Γ and Σ are input and output type environments, respectively. Typing rules are listed in Fig. 2.

Remark 1. (1) A calculus with *a priori* connection between an input name x and an output name $\bar{x}$ can be simulated by passing/receiving a pair $(x, \bar{x})$ of input and

output names. Via this translation our game semantics is applicable to a calculus with *a priori* connection because the translation reflects may-testing equivalence. (2) The standard parallel composition, which invokes communications of the two processes, can be expressed as $\nu(\bar{\boldsymbol{a}}\bar{\boldsymbol{b}}, \boldsymbol{ab}).(P|Q|(\boldsymbol{a}' \to \bar{\boldsymbol{a}})|(\boldsymbol{b} \to \bar{\boldsymbol{b}}'))$ where $\boldsymbol{a}$ and $\bar{\boldsymbol{b}}$ are free names in P and Q, and $\boldsymbol{a}' \to \bar{\boldsymbol{a}}$ is a "forwarder", a process forwarding names received from a'_i to $\bar{a}_i$.

3 Concurrent HO/N Game Model

This section introduces a truly concurrent game model in which a play is not a sequence but a directed acyclic graph (DAG). A node of a play is labelled by a move representing an event; an edge represents either a justification pointer or causality. The key is the notion of plays (Sect. 3.2) and of interactions (Sect. 3.3). The other parts are relatively straightforward adaptation of the techniques in the standard HO/N game model (e.g. [21]) or Laird's model [24].

3.1 Arenas

The definition of arenas is (essentially) the same as the definition of arenas in the case of the sequential game model of π-calculus [24]. The differences from the standard definition (e.g. [21]) are (1) all moves are questions, and (2) the owner of moves does not have to alternate.

Definition 1 (Arena). *An* arena *is a triple* $A = (\mathcal{M}_A, \lambda_A, \vdash_A)$*, where* $\mathcal{M}_A$ *is a set of* moves*,* $\lambda_A \colon \mathcal{M}_A \to \{P, O\}$ *is an* ownership function *and* $\vdash_A \subseteq (\{\star\} + \mathcal{M}_A) \times \mathcal{M}_A$ *is an* enabling relation *that satisfies: for every* $m \in \mathcal{M}_A$*, there uniquely exists* $x \in \{\star\} + \mathcal{M}_A$ *such that* $x \vdash_A m$*.*

We say that m is a *P-move* if $\lambda_A(m) = P$; it is an *O-move* if $\lambda_A(m) = O$. Every move represents an output action: a P-move is an output action of the process and an O-move is that of the environment (see a discussion after Definition 4). Let $\lambda_A^{\perp}$ denote the negation of λ_A i.e. $\lambda_A^{\perp}(m) = O$ (resp. $\lambda_A^{\perp}(m) = P$) if $\lambda_A(m) = P$ (resp. $\lambda_A(m) = O$). A move m is *initial* if $\star \vdash_A m$. An arena is *negative* (resp. *positive*) if all initial moves are O-moves (resp. P-moves). In what follows, we shall consider only negative arenas (hence we often use arenas to mean negative arenas). The *empty arena* is defined by $I := (\emptyset, \emptyset, \emptyset)$.

Negative and positive arenas correspond to input and output type environments, respectively. Hence a judgement, which consists of a pair of input and output type environments, should be expressed as a pair of arenas.

Definition 2 (Arena pair). *An* arena pair *is a pair* (A, B) *of (negative) arenas. We write* $\mathcal{M}_{A,B}$ *for* $\mathcal{M}_A + \mathcal{M}_B$*. The ownership function is defined by* $\lambda_{A,B} = [\lambda_A^{\perp}, \lambda_B]$*. The enabling relation* $\vdash_{A,B}$ *is given by:* $m \vdash_{A,B} m'$ *if and only if* $m \vdash_A m'$ *or* $m \vdash_B m'$*. (In particular,* $\star \vdash_{A,B} m$ *iff* $\star \vdash_A m$ *or* $\star \vdash_B m'$*.)*

Note that an arena pair is not a negative arena since it has an initial P-move.

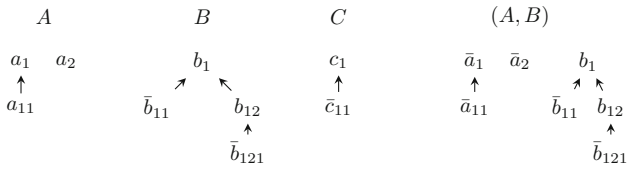


Fig. 3. Examples of arenas and an arena pair.

Example 1. Three (negative) arenas A, B and C are illustrated in Fig. 3, as well as the arena pair (A, B). Those arenas are used in examples in this paper. Nodes are labelled by moves and edges represent the enabling relation. If a name is overlined, the move is a P-move; otherwise it is an O-move. The arena pair (A, B) corresponds to the pair of the output type environment $\Gamma = \bar{a}_1 : \mathbf{ch}[\_, \mathbf{ch}[]], \bar{a}_2 : \mathbf{ch}[]$ and the input type environment $\Sigma = b_1 : \mathbf{ch}[\mathbf{ch}[], \mathbf{ch}[\mathbf{ch}[], \_]]$. (Channel names do not have to coincide with move names.)

3.2 DAG-based Plays

In the standard HO/N game model [21], a play is a sequence of moves equipped with pointers, called *justification pointers*. The justification pointers express the binder-bindee relation and the sequential structure expresses the temporal relation between the events in the sequence (e.g. in the sequence $s_1\, a\, s_2\, b\, s_3$, the event b occurs after a). The causal relation is left implicit (cf. Sect. 3.5).

In the proposed game model, we explicitly describe the causal relation as well as the justification pointers.

Definition 3 (Justified graph). *A* justified graph *over an arena pair* (A, B) *is a tuple* $s = (V_s, l_s, \overset{\frown}{s}, \overset{\rightsquigarrow}{s})$ *where:*

- V_s *is a finite set called the* vertex set
- l_s *is the* vertex labelling*, that is* $l_s \colon V_s \to \mathcal{M}_{A,B}$
- $\overset{\frown}{s} \subseteq V_s \times V_s$ *is the* justification relation
- $\overset{\rightsquigarrow}{s} \subseteq V_s \times V_s$ *is the* causality relation

such that

- $(V_s, \overset{\frown}{s} \cup \overset{\rightsquigarrow}{s})$ *is a DAG i.e. there is no cycle* $v\ (\overset{\rightsquigarrow}{s} \cup \overset{\frown}{s})^+\ v$.
- *If* $l_s(v)$ *is initial, then there is no node* v' *such that* $v \overset{\frown}{s} v'$.
- *If* $l_s(v)$ *is not initial, then there exists a unique node* v' *such that* $v \overset{\frown}{s} v'$. *Furthermore this* v' *satisfies* $l_s(v') \vdash_{A,B} l_s(v)$.

Note that $\overset{\frown}{s}$ *and* $\overset{\rightsquigarrow}{s}$ *do not have to be disjoint. We define* $\vec{s} := (\overset{\frown}{s} \cup \overset{\rightsquigarrow}{s})$. *The set of justified graphs over an arena pair* (A, B) *is denoted by* $J_{A,B}$.

In what follows, we shall identify isomorphic justified graphs.

Given a justified graph s over (A, B), a *P-node* (resp. an *O-node*) is a node $v \in V_s$ whose label is a P-move (resp. an O-move). We write V_s^P for the set of P-nodes and V_s^O for the set of O-nodes (e.g. $V_s^P := \{v \in V_s \mid \lambda_{A,B}(l_s(v)) = P\}$).

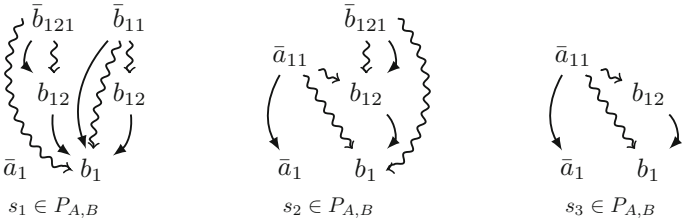


Fig. 4. Examples of plays over the arena pair (A, B) in Fig. 3.

Definition 4 (Play). *Let $s = (V_s, l_s, \curvearrowright_s, \not\rightarrow_s)$ be a justified graph over (A, B). It is a* play *if it satisfies the following conditions:*

(P1) *for every $v, v' \in V_s$, $v \not\rightarrow_s v'$ implies $v \in V_s^P$ and $v' \in V_s^O$,*
(P2) *for every $v_p \in V_s^P$ and $v_o \in V_s^O$, if $v_p \rightarrow_s^+ v_o$, then $v_p \not\rightarrow_s v_o$, and*
(P3) *for every $v_o \in V_s^o$, there exists $v_p \in V_s^P$ such that $v_p \not\rightarrow_s v_o$.*

We write $P_{A,B}$ for the set of plays over (A, B).

Condition (P1) reflects the asynchronous nature of the target language. Recall that a P-move corresponds to an output action of a process and an O-move to an output action of the environment. No P-node should be causally related to P-nodes since an output action of the process cannot cause any other output of the process. Similarly no O-node should be causally related to O-nodes since an output action of the environment cannot cause any other output of the environment (provided that the environment is also described by the asynchronous π-calculus). An output action of a process may cause an output action of the environment; however it is a matter of the environment and a play describes the behaviour of a process, not the environment. Hence $\not\rightarrow_s \subseteq V_s^P \times V_s^O$.

Condition (P2) comes from a purely technical requirement. (We need this condition to establish Lemma 2, as well as a proposition stating the copycat strategy is the identity.)

Condition (P3) is the counterpart of the even-length condition. Here we regard the even-length condition for sequential plays as the requirement that every O-move in the sequence should be responded by a P-move.

Example 2. Figure 4 shows three different plays over the arena pair (A, B) in Fig. 3. The solid arrows represent justification pointers, and squiggly arrows represent causalities. Nodes are labelled by moves and different nodes may be labelled by a same move. Note that plays may have a join point, i.e. a node that is linked to two "incomparable" nodes, like the node labelled by $\bar{a}_{11}$ in s_2.

Remark 2. A play can be seen as a process, e.g. the play s_2 in Fig. 4 corresponds to the process $\nu(\bar{a}_{11}, a_{11}).(b_1(\_, b_{12}).b_{12}(\bar{b}_{121}, \_).(\bar{a}_{11} \mid \bar{b}_{121}) \mid \bar{a}_1\langle \_, a_{11}\rangle)$ (whose type differs from that described by the arena pair). The formal description of the connection to the linear internal π-calculus is left for the future work.

3.3 Strategies and Composition

Strategy. In most variants of sequential game models, a strategy σ is a collection of plays that is *(even-length) prefix closed*: if $sm_Om_P \in \sigma$, then $s \in \sigma$. The set of strategies in our game model is defined by the same way, though the notion of prefix should be adapted to our setting.

Definition 5 (Prefix). *Let $s = (V_s, l_s, \curvearrowright_s, \rightsquigarrow_s)$ be a play. Let $U \subseteq V_s$ be a subset that satisfies* (1) *$v \in U$ and $v \xrightarrow{s} v'$ implies $v' \in U$ and* (2) *for all $v_o \in U^O$ there exists $v_p \in U^P$ such that $v_p \rightsquigarrow_s v_o$. The* prefix $s[U] := (U, l, \curvearrowright, \rightsquigarrow)$ *of s induced by U is the restriction of s to U, i.e.,*

$$l(v) := l_s(v) \quad \curvearrowright := (\curvearrowright_s) \cap (U \times U) \quad \rightsquigarrow := (\rightsquigarrow_s) \cap (U \times U).$$

We write $s' \sqsubseteq s$ if s' is a prefix of s. A prefix of a play is a play.

Example 3. In Fig. 4, the play s_3 is a prefix of s_2 induced by the set of nodes labelled by $m \in \{\bar{a}_1, \bar{a}_{11}, b_1, b_{12}\}$.

Definition 6 (Strategy). *Let (A, B) be an arena pair. A set $\sigma \subseteq P_{A,B}$ of plays over (A, B) is a* strategy *of (A, B), written as $\sigma \colon A \to B$, if it satisfies* prefix-closedness *(S1):*

(S1) *If $s \in \sigma$ and $s' \sqsubseteq s$, then $s' \in \sigma$.*

Composition. The composition of strategies is defined by using the notion of interactions. Since plays are not sequences but graphs, an interaction should also be represented by a graph that we call an *interaction graph.*

Definition 7. *Let (A, B, C) be a triple of arenas. The set $\mathcal{M}_{A,B,C}$ of moves of (A, B, C) is the disjoint union $\mathcal{M}_A + \mathcal{M}_B + \mathcal{M}_C$. The enabling relation $\vdash_{A,B,C}$ is defined by: $x \vdash_{A,B,C} m$ if $x \vdash_X m$ for some $X \in \{A, B, C\}$. The ownership function is defined by: $\lambda_{A,B,C} := [\lambda_A, \lambda_B, \lambda_C]$. The set $J_{A,B,C}$ of* justified graphs of (A, B, C) *is defined by the same way as in Definition 3.*

For $X \in \{A, B, C, (A, B), (B, C), (A, C)\}$, we write V_X for the set of nodes restricted to the component X and V_X^P and V_X^O for the sets of nodes labelled by P-moves and by O-moves in the component X. For example, $v \in V_{A,B}^P$ means either (1) $l_u(v) \in \mathcal{M}_B$ and $\lambda_B(l_u(v)) = P$, or (2) $l_u(v) \in \mathcal{M}_A$ and $\lambda_A(l_u(v)) = O$.

Definition 8 (Restriction). *Let $u = (V, l, \curvearrowright, \rightsquigarrow)$ be a justified graph over (A, B, C) and $X \in \{(A, B), (B, C), (A, C)\}$. The* restriction $u\restriction_X$ *of u to X is defined by $u\restriction_X := (V\restriction_X,\ l\restriction_X,\ \curvearrowright\restriction_X,\ \rightsquigarrow\restriction_X)$, where*

$$V\restriction_X := V_X, \quad l\restriction_X(v) := l(v), \quad \curvearrowright\restriction_X := (\curvearrowright) \cap (V_X \times V_X).$$

The definition of $\rightsquigarrow\restriction_X$ needs some care. If $X \in \{(A, B), (B, C)\}$, then $\rightsquigarrow\restriction_X$ is just the restriction of the original causal relation, i.e. $\rightsquigarrow\restriction_X := \{(v, v') \in V_X^P \times V_X^O \mid v \rightsquigarrow v'\}$ (cf. Condition (P1)). If $X = (A, C)$, then $\rightsquigarrow\restriction_{A,C}$ relates moves linked through the intermediate component B, i.e. $\rightsquigarrow\restriction_{A,C} := \{(v, v') \in V_{A,C}^P \times V_{A,C}^O \mid \exists n \geq 0.\ \exists v_1, \ldots, v_n \in V_B.\ v \rightsquigarrow v_1 \rightsquigarrow \cdots \rightsquigarrow v_n \rightsquigarrow v'\}$.

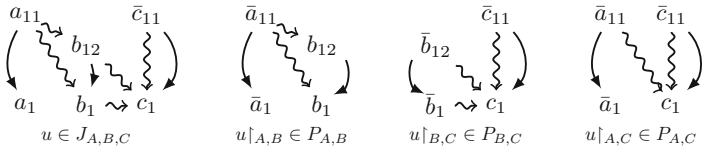


Fig. 5. Example of a justified graph and restrictions.

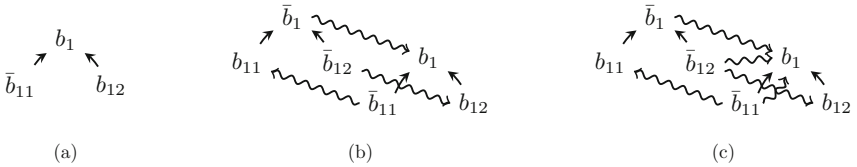


Fig. 6. Construction of a copycat play.

Example 4. Figure 5 shows a justified graph u over the triple (A, B, C) (in Fig. 3) and its restrictions to components (A, B), (B, C) and (A, C).

Note that although $\bar{a}_{11} \not\twoheadrightarrow_u c_1$, we have $\bar{a}_{11} \twoheadrightarrow_u u\restriction_{A,C} c_1$ because $\bar{a}_{11} \twoheadrightarrow_u b_1 \twoheadrightarrow_u c_1$.

Definition 9 (Interaction graph). *Let $u \in J_{A,B,C}$ be a justified graph over (A, B, C) and V be the set of nodes of u. We say that u is an* interaction graph *if it satisfies the following conditions.*

(I1) *If $v \twoheadrightarrow_u v'$, then $(v, v') \in V_X^P \times V_X^O$ for some $X \in \{(A, B), (B, C)\}$.*
(I2) *Both $u\restriction_{A,B}$ and $u\restriction_{B,C}$ are plays.*

Condition (I1) is a variant of the switching condition. *The set of interaction graphs over (A, B, C) is denoted as* $\mathrm{Int}(A, B, C)$.

In fact u in Example 4 is an interaction graph.

Definition 10 (Composition). *Let $\sigma\colon A \to B$ and $\tau\colon B \to C$ be strategies. The* composition *of σ and τ is defined by*

$$\tau \circ \sigma := \{u\restriction_{A,C} \mid u \in \mathrm{Int}(A, B, C),\ \ u\restriction_{A,B} \in \sigma,\ \ u\restriction_{B,C} \in \tau\}.$$

Note that the definition of composition is applicable to sets of plays that are not necessarily strategies. By abuse of notation, we shall write $\tau \circ \sigma$ even if σ and τ are not strategies but just sets of plays.

Theorem 1. *The composite of strategies is a strategy. The composition is associative.*

Category. We define the category $\mathcal{P}$ of negative arenas and strategies: an object of $\mathcal{P}$ is a (negative) arena and a morphism from A to B is a strategy $\sigma\colon A \to B$. The composite of $\sigma\colon A \to B$ and $\tau\colon B \to C$ is given by the composition $\tau \circ \sigma$ of strategies defined above. Given an arena A, the identity morphism $\mathrm{id}_A\colon A \to A$ is the "copycat strategy": when the environment makes a move m in one component, then it responds by making a copy of m in the other component. It is the set of *copycat plays*, whose construction is illustrated in Fig. 6: (*a*) take a "justified graph without causality" of the arena (in this example, the arena is B in Fig. 3); (*b*) make positive and negative copies and connect the corresponding nodes by a causal edge $\rightsquigarrow$ in the appropriate direction; and (*c*) add causal edges so as to satisfy Condition (P2), resulting in a play over (B, B).

3.4 Distributive-Closed Freyd Category

In this section, we define the categorical structures of $\mathcal{P}$, which is used in Sect. 4 to give an interpretation of the π-calculus. A category with the structures below is called a *distributive-closed Freyd category* [24]. The definitions in this section are adapted from the interleaving game model for the π-calculus [24].

Monoidal product. Let $A = (\mathcal{M}_A, \lambda_A, \vdash_A)$ and $B = (\mathcal{M}_B, \lambda_B, \vdash_B)$ be arenas. The arena $A \odot B$ is defined as $(\mathcal{M}_A + \mathcal{M}_B, [\lambda_A, \lambda_B], \vdash_{A,B})$, where $\vdash_{A,B}$ is the enabling relation defined in Definition 2. Given strategies $\sigma\colon A \to B$ and $\tau\colon C \to D$, the strategy $\sigma \odot \tau\colon A \odot C \to B \odot D$ is defined by the juxtaposition of plays in σ and τ, namely $\sigma \odot \tau := \{s \uplus t \mid s \in \sigma, t \in \tau\}$ where $s \uplus t$ is the juxtaposition of plays. Then the triple $(\mathcal{P}, \odot, I)$ is a symmetrical monoidal category.

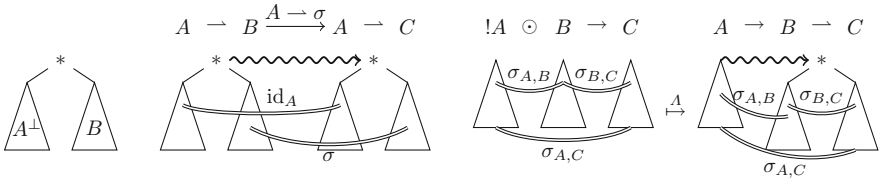
Closed Freyd structure. An input prefixing $a(\bar{\boldsymbol{x}}, \boldsymbol{y}).P$ should be interpreted by using a kind of closed structure (intuitively because the input prefix bounds variables in P like λ-abstraction). Laird [24] used *closed Freyd categories* [33].

A Freyd category consists of a symmetric (pre)monoidal category $\mathcal{P}$, a cartesian category $\mathcal{A}$ and an identity-on-object strict (pre)monoidal functor $!\colon \mathcal{A} \to \mathcal{P}$. Intuitively $\mathcal{P}$ is that of types and "terms" whereas $\mathcal{A}$ is the category of types and "values"; the functor ! gives us a way to regard a "value" as a "term". In our context, "terms" are processes and "values" are processes of the form $\sum_i a_i(\bar{\boldsymbol{x}}_i, \boldsymbol{y}_i).P_i$, where P_i has no free input channel except for those in $\boldsymbol{y}_i$.

We define the game-semantic counterpart of the processes of the this form.

Definition 11 (Well-opened play, strategy). *A play s is* well-opened *if it contains precisely one initial O-node v_0 to which all other nodes are connected (i.e. $v \stackrel{s}{\Rightarrow}{}^{*} v_0$ for every $v \in V_s$). We write $W_{A,B}$ for the set of well-opened plays over (A, B). A* well-opened strategy *from arena A to arena B, written as $\sigma\colon A \stackrel{\bullet}{\to} B$, is a set σ of well-opened plays that is prefix-closed (S1).*

Then we define an operator !, a mapping from well-opened strategies to strategies and the composition of well-opened strategies by using !.



(a) $A \multimap B$. (b) $A \multimap \sigma$. (c) $\Lambda(\sigma)$.

Fig. 7. The action of $A \multimap (-)$ and Λ.

Definition 12. *Let $\sigma\colon A \xrightarrow{\bullet} B$ be a well-opened strategy. The strategy $!\sigma\colon !A \to B$ is defined by $!\sigma := \{s_1 \uplus \cdots \uplus s_n \mid n \geq 0,\ \forall i \leq n.\, s_i \in \sigma\}$ where $s_1 \uplus \cdots \uplus s_n$ is the juxtaposition of plays $s_1, \ldots, s_n$.*

Definition 13 (Composition of well-opened strategies). *Let $\sigma\colon A \xrightarrow{\bullet} B$ and $\tau\colon B \xrightarrow{\bullet} C$ be well-opened strategies. We define $\tau \circ_{\mathcal{A}} \sigma := \tau \circ !\sigma$.*

Lemma 1. *The composite of well-opened strategies with respect to $\circ_{\mathcal{A}}$ is a well-opened strategy. The composition $\circ_{\mathcal{A}}$ of well-opened strategies is associative.*

The *category $\mathcal{A}$ of negative arenas and well-opened strategies* is defined by the following data: an object is a negative arena, a morphism from A to B is a well-opened strategy $\sigma\colon A \xrightarrow{\bullet} B$, the composition is given by $\circ_{\mathcal{A}}$. The identity morphism is $\mathrm{id}_A \cap W_{A,A}$, where id_A is the copycat strategy. The category $\mathcal{A}$ is cartesian: the cartesian product of A and B is $A \odot B$.

By defining $!A := A$ for objects, the operation $!$ becomes a functor $!\colon \mathcal{A} \to \mathcal{P}$. This is identity on objects and strict symmetric monoidal functor and thus $(\mathcal{A}, \mathcal{P}, !)$ is a Freyd category.

Lemma 2. *The Freyd category $(\mathcal{A}, \mathcal{P}, !)$ is closed, i.e. for every arena A, the functor $!(-) \odot A\colon \mathcal{A} \to \mathcal{P}$ has the right-adjoint $A \multimap (-)\colon \mathcal{P} \to \mathcal{A}$.*

The action of $A \multimap (-)$ on objects and on morphisms is illustrated in Fig. 7. We write Λ for the bijective map $\mathcal{P}(!A \odot B, C) \to \mathcal{A}(A, B \multimap C)$ and $\mathbf{app}_{A,B}\colon !(A \multimap B) \odot A \to B$ for the counit. The bijection $\mathcal{P}(!A \odot B, C) \cong \mathcal{A}(A, B \multimap C)$ induced by the adjunction intuitively corresponds to the following bijection of the π-calculus processes: $\bar{\boldsymbol{x}} : \boldsymbol{S}, \bar{\boldsymbol{y}} : \boldsymbol{T} \vdash P; \boldsymbol{z} : \boldsymbol{U} \quad \longleftrightarrow \quad \bar{\boldsymbol{x}} : \boldsymbol{S} \vdash a(\bar{\boldsymbol{y}}, \boldsymbol{z}).P; a : \mathbf{ch}[\boldsymbol{T}, \boldsymbol{U}]$.

Distributive law. The process obtained by (the π-term representation of) the above adjunction has the input prefix $a(\bar{\boldsymbol{y}}, \boldsymbol{z})$ as expected but it has only one free input channel. We use the *distributive law* of the distributive-closed Freyd category to model a process with multiple free input channel. By using the syntax of the π-calculus, the distribution law can be seen as the following map:

$$\bar{\boldsymbol{x}}\colon \boldsymbol{S} \vdash a(\bar{\boldsymbol{y}}, \boldsymbol{z}\boldsymbol{z}').P;\ a\colon \mathbf{ch}[\boldsymbol{T}, \boldsymbol{U}\boldsymbol{U}'] \ \longrightarrow\ \bar{\boldsymbol{x}}\colon \boldsymbol{S} \vdash a(\bar{\boldsymbol{y}}, \boldsymbol{z}).P;\ a\colon \mathbf{ch}[\boldsymbol{T}, \boldsymbol{U}], \boldsymbol{z}'\colon \boldsymbol{U}'.$$

Definition 14 (Distributive-closed Freyd category [24]). *A closed Freyd category* $!\colon \mathcal{A} \to \mathcal{P}$ *is* distributive-closed *if there is a family of morphisms* $\varrho_A\colon !(A \multimap (B \odot C)) \longrightarrow B \odot !(A \multimap C)$ *in* $\mathcal{P}$*, natural in* B *and* C *which makes certain diagrams commute.*

Theorem 2. *The game model* $!\colon \mathcal{A} \to \mathcal{P}$ *is distributive-closed.*

Trace. The operator $\nu(\bar{x}, y).P$ is interpreted as a trace operator. We define $Tr^B_{A,C}(f) := \mathbf{app}_{B,C} \circ \mathbf{symm}_{B,B\multimap C} \circ \varrho_{B,B,C} \circ !\Lambda(\mathbf{symm}_{B,C} \circ f)$, given a morphism $f\colon A\odot B \to C\odot B$ in $\mathcal{P}$. Then Tr is the trace operator for the symmetrical monoidal category $\mathcal{P}$ [24].

Additional structures. Some additional structures are required to interpret the π-calculus: the *minimum strategy* $\bot_{A,B}$ (with respect to the set-inclusion), the *diagonal* $\Delta_A\colon A \xrightarrow{\bullet} A \odot A$, the *codiagonal* $\nabla_A\colon A \odot A \xrightarrow{\bullet} A$ (defined by $\nabla_A := \pi_1 \cup \pi_2$ where $\pi_i\colon A\odot A \xrightarrow{\bullet} A$ is the projection), and the *dereliction* $der_A\colon A \to A$ (defined as $\mathrm{id}_A \cap W_{A,A}$).

3.5 Relation to Sequential Game Models

Laird's interleaving game model. Our model can be seen as a truly concurrent version of the interleaving game model $\mathcal{P}_{\mathrm{L}}$ of Laird [24]. The idea is to relate a (concurrent) play $s = (V_s, l_s, \curvearrowright_s, \rightarrow_s)$ to an interleaving play by lining up the nodes in V_s in such a way that if $v_1 \rightarrow_s v_2$, then v_2 appears before v_1. We write $|s|$ for the set of sequential plays obtained by this way.

Example 5. Let s_2 be the play in Fig. 4. Then $|s_2|$ is given as:

$$\left\{ \bar{a}_1 b_1 b_{12} \bar{a}_{12} \bar{b}_{121},\ \bar{a}_1 b_1 b_{12} \bar{b}_{121} \bar{a}_{12},\ b_1 \bar{a}_1 b_{12} \bar{a}_{12} \bar{b}_{121},\ b_1 \bar{a}_1 b_{12} \bar{b}_{121} \bar{a}_{12},\ b_1 b_{12} \bar{a}_1 \bar{a}_{12} \bar{b}_{121},\ b_1 b_{12} \bar{a}_1 \bar{b}_{121} \bar{a}_{12},\ b_1 b_{12} \bar{b}_{121} \bar{a}_1 \bar{a}_{12} \right\}$$

Theorem 3. $|-|$ *induces an identity-on-object functor from* $\mathcal{P}$ *to* $\mathcal{P}_{\mathrm{L}}$*, which preserves the structure of distributed-closed Freyd categories (and the additional structures). Furthermore* $|\sigma|$ *is the minimum strategy if and only if so is* σ*.*

Sequential HO/N game model. The standard sequential HO/N game model [21] is a subcategory of our concurrent model. Since our game model only have question moves, we compare our model with the HO/N game model without *answer* (and thus without *well-bracketing*).

An arena A is *alternating* if $m \vdash_A m'$ implies $\lambda_A(m) = \lambda_A^{\perp}(m')$. Let $\mathcal{G}$ be the category of negative alternating arenas and innocent strategies (we omit the definition, which is standard). We write $\ulcorner \hat{s} \urcorner$ for the *P-view* [21] of the sequential play $\hat{s}$. Given a sequential play $\hat{s} = m_1 \dots m_n$, a DAG-based play is given by

$$\|\hat{s}\| := (V_{\hat{s}}, l_{\hat{s}},\ \{(i,j) \mid \rho_{\hat{s}}(i) = j\},\ \{(i,j) \in V^P_{\hat{s}} \times V^O_{\hat{s}} \mid m_j \in \ulcorner m_1 \dots m_i \urcorner\})$$

where $V_{\hat{s}} := \{1, \ldots, n\}$, $l_{\hat{s}}(i) := m_i$ and $\rho_{\hat{s}}$ is the partial function describing the justification pointer. Note that the occurrence m_i of a P-move is causally related to an occurrence m_j of an O-move if and only if m_j appears in the P-view of m_i. This map is naturally extended to strategies, namely $\|\hat{\sigma}\| := \{\|\hat{s}\| \mid \hat{s} \in \hat{\sigma}\}$.

Theorem 4. *$\|-\|$ induces a faithful functor from $\mathcal{G}$ to $\mathcal{P}$.*

Remark 3. It is natural to ask if one can give a similar map from Laird's interleaving model. The answer seems negative: all maps that we have checked are not functorial. See [8] for a related result.

4 Game Semantics of the π-calculus

We give an interpretation of the π-calculus, following the result of Laird [24] applicable to every distributive-closed Freyd category with additional structures.

A type and a type environment are interpreted as objects of $\mathcal{P}$. The interpretation of a type $\mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}]$ and a sequence $\boldsymbol{S}$ of types are defined by:

$$[\![\mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}]]\!] := [\![\boldsymbol{S}]\!] \multimap [\![\boldsymbol{T}]\!] \quad [\![S_1 \ldots S_n]\!] := [\![S_1]\!] \odot \cdots \odot [\![S_n]\!] \quad [\![\_]\!] := I.$$

The interpretation of an input type environment is given by the tensor product of elements, e.g. $[\![x_1 : S_1, \ldots, x_n : S_n]\!] := [\![S_1]\!] \odot \cdots \odot [\![S_n]\!]$.

A process $\Gamma \vdash P; \Sigma$ is interpreted as a morphism $[\![P]\!] \colon [\![\Gamma]\!] \to [\![\Sigma]\!]$ in $\mathcal{P}$. The interpretation is defined by induction on the type derivations. The rules are listed in Fig. 8.

$$\begin{aligned}
[\![\Gamma \vdash \mathbf{0}; \Sigma]\!] &= \bot_{[\![\Gamma]\!],[\![\Sigma]\!]} \\
[\![\Gamma, \Gamma' \vdash P|Q; \Sigma, \Sigma']\!] &= [\![P]\!] \odot [\![Q]\!] \\
[\![\Gamma, \bar{x} : \mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}], \bar{\boldsymbol{y}} : \boldsymbol{S} \vdash \bar{x}\langle \bar{\boldsymbol{y}}, \boldsymbol{z}\rangle; \Sigma, \boldsymbol{z} : \boldsymbol{T}]\!] &= \bot_{[\![\Gamma]\!],[\![\Sigma]\!]} \odot \mathbf{app}_{[\![\boldsymbol{S}]\!],[\![\boldsymbol{T}]\!]} \\
[\![\Gamma \vdash x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P; \Sigma, x : \mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}]]\!] &= (\mathrm{id}_{[\![\Sigma]\!]} \odot der_{[\![(\boldsymbol{S},\boldsymbol{T})]\!]}) \circ [\![!x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P]\!] \\
[\![\Gamma \vdash !x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P; \Sigma, x : \mathbf{ch}[\boldsymbol{S}, \boldsymbol{T}]]\!] &= \varrho_{[\![\boldsymbol{S}]\!],[\![\Sigma]\!],[\![\boldsymbol{T}]\!]} \circ !\Lambda([\![P]\!]) \\
[\![\Gamma \vdash \nu(\bar{x}, y).P; \Sigma]\!] &= Tr^{[\![T]\!]}_{[\![\Gamma]\!],[\![\Sigma]\!]}([\![P]\!])
\end{aligned}$$

Fig. 8. Interpretation of processes. (Contraction and exchange rules are omitted.)

The distributive-closed Freyd structure together with additional structures (of Δ, ∇, $\bot$, *der*) gives a (weak) soundness result with respect to the reduction.

Theorem 5. *Let $\Gamma \vdash P; \Sigma$ and $\Gamma \vdash Q; \Sigma$ be processes of the same type.*

1. *If $P \equiv Q$, then $[\![\Gamma \vdash P; \Sigma]\!] = [\![\Gamma \vdash Q; \Sigma]\!]$.*
2. *If $P \longrightarrow Q$, then $[\![\Gamma \vdash P; \Sigma]\!] \supseteq [\![\Gamma \vdash Q; \Sigma]\!]$.*

The relationship to Laird's model (Theorem 3) gives a finer result, which does not follow from the general theory of the distributive-closed Freyd categories.

Lemma 3. (Adequacy). *$P \Downarrow_{\bar{x}}$ iff $[\![P]\!] \neq \bot$ for every process $\bar{x} : \mathbf{ch}[] \vdash P; \_$.*

Proof. Because of Theorem 3, we have $|[\![P]\!]| = [\![P]\!]_L$, where $[\![P]\!]_L$ is the interpretation of the process in Laird's game model [24]. Laird [24] shows that $P \Downarrow_{\bar{x}}$ if and only if $[\![P]\!]_L \neq \bot$. Since $|-|$ preserves $\bot$, we obtain the claim. □

Lemma 3 and monotonicity of the interpretation lead to the next theorem.

Theorem 6. *Let $\bar{x}$ be a testing name that does not occur in Γ. If $[\![\Gamma \vdash P; \Sigma]\!] \subseteq [\![\Gamma \vdash Q; \Sigma]\!]$, then $C[P] \Downarrow_{\bar{x}}$ implies $C[Q] \Downarrow_{\bar{x}}$ for all context $C[\,]$.*

Unlike Laird's model [24], our model is not complete since our model is truly concurrent. For example, $[\![a().\bar{b}\langle\rangle \mid c().\bar{d}\langle\rangle]\!] \neq [\![\nu(\bar{x}, x).(\bar{x}\langle\rangle \mid x().a().(\bar{b}\langle\rangle \mid c().\bar{d}\langle\rangle) \mid x().c().(a().\bar{b}\langle\rangle \mid \bar{d}\langle\rangle))]\!]$ in our model, whereas they are testing equivalent.

5 Discussion: Relationally-Describable Process

Using our game model, we study the relational interpretations of process in the form of intersection type system that describes the behaviour of processes. The intersection type system is a fully abstract model for *a class of* process which we characterise with the help of "interaction graph".

The syntax of *types* and *intersections* are defined by the following grammar:

$$\varphi, \psi ::= \mathbf{ch}[\xi_1 \dots \xi_n, \zeta_1 \dots \zeta_k] \qquad \xi, \zeta ::= \langle \varphi_1, \dots, \varphi_n \rangle$$

where $\langle \cdots \rangle$ is a finite multiset defined by an enumeration of the elements. A *type environment* is a sequence of type bindings of the form $x{:}\xi$ (or $\bar{y}{:}\zeta$). Given intersections $\xi = \langle \varphi_1, \dots, \varphi_n \rangle$ and $\zeta = \langle \psi_1, \dots, \psi_k \rangle$, we write $\xi \wedge \zeta$ for $\langle \varphi_1, \dots, \varphi_n, \psi_1, \dots, \psi_k \rangle$. This operation is extended to type environments by pointwise application. The typing rules are listed below (some rules are omitted):

$$\frac{}{\emptyset \vdash \mathbf{0};\ \emptyset} \qquad \frac{\Xi \vdash P;\ \Theta \qquad \Xi' \vdash P';\ \Theta'}{\Xi, \Xi' \vdash P|P';\ \Theta, \Theta'} \qquad \frac{}{\bar{x} : \mathbf{ch}[\boldsymbol{\xi}, \boldsymbol{\zeta}], \bar{\boldsymbol{y}} : \boldsymbol{\xi} \vdash \bar{x}\langle \bar{\boldsymbol{y}}, \boldsymbol{z} \rangle;\ \Theta, \boldsymbol{z} : \boldsymbol{\zeta}}$$

$$\frac{\Xi, \bar{\boldsymbol{y}}{:}\boldsymbol{\xi} \vdash x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P;\ \Theta, \boldsymbol{z}{:}\boldsymbol{\zeta}}{\Xi \vdash x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P;\ \Theta, x{:}\mathbf{ch}[\boldsymbol{\xi}, \boldsymbol{\zeta}]} \qquad \frac{\forall i \in I.\ \Xi_i \vdash x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P;\ \Theta_i}{\bigwedge_{i \in I} \Xi_i \vdash !x(\bar{\boldsymbol{y}}, \boldsymbol{z}).P;\ \bigwedge_{i \in I} \Theta_i} \qquad \frac{\Xi, \bar{x}{:}\xi \vdash P;\ \Theta, y{:}\xi}{\Xi \vdash \nu(\bar{x}, y).P;\ \Theta}$$

This type system is inspired by the correspondence between intersection type systems and the operation called *time forgetting map* [4], which is an operation that forgets the temporal structure of plays, in sequential game models (see, e.g., [37]). Time forgetting map is the operation that forgets the causal relation in the case of our concurrent game model.

Completeness of the type system holds for every process, but soundness does not; the reason is explained by a game-semantic consideration. We would thus like to find a class for which the relational interpretation is sound.

Let $s \in P_{A,B}$ and $t \in P_{B,C}$ be plays. We say that s and t are *composable* if $s{\restriction}_B$ coincides with $t{\restriction}_B$ except for the causal relations. Then it would be natural

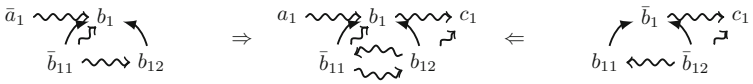


Fig. 9. Composable plays with a cycle.

to think of an "interaction graph" by composing them (see Fig. 9). Unfortunately the resulting "interaction graph" may not be acyclic and hence not be an interaction graph; in this case we say that the pair (s, t) contains a cycle.

This notion of cycle can be extended to strategies and processes. The composition of strategies $\tau \circ \sigma$ is *cycle-free* if every pair of composable plays $s \in \sigma$ and $t \in \tau$ is cycle-free. A process P is *relationally-describable* if every composition in the definition of $[\![P]\!]$ is cycle-free.

Theorem 7. *Let* $\Gamma \vdash P;\ \Sigma$ *be a relationally-describable process and let* $\tilde{x} \in \mathrm{dom}(\Gamma)$. *Then* $P \Downarrow_{\tilde{x}}$ *if and only if* $\tilde{x} : \mathbf{ch}[\boldsymbol{\xi}, \boldsymbol{\zeta}] \vdash P;\ \emptyset$ *for some* $\boldsymbol{\xi}$ *and* $\boldsymbol{\zeta}$.

This is because the operation of forgetting the causal relation commutes with cycle-free composition. Note that the notion of cycle is stronger than deadlock: $\nu(\bar{a}\bar{b}, ab).(a_1.\bar{b}_2 | b_3.\bar{a}_4 | \bar{a}_5)$ (subscripts are used to distinguish occurrences) is *not* relationally-describable because connecting a_1 to $\bar{a}_4$ and b_3 to $\bar{b}_2$ creates a cycle.

Restricting the form of processes by focusing on cycles is a reminiscent of the correctness criterion for MLL proof nets. The formal relationship between our notion of cycle in an interaction graph and the correctness criterion, and the connection between cycle (in our sense) and the type system, which gives a typed π-calculus corresponding to polarised proof-nets satisfying the correctness criterion, proposed by Honda and Laurent [19] are worth investigating.

6 Related Work

Melliès [27] studied HO/N innocent strategies from a truly concurrent point of view. Among others, he introduced the notions of *alternating homotopy* and *diagrammatic innocence*, which influence to this work. These ideas were subsequently developed by Melliès and Mimram [29,30], who introduced *asynchronous games*. They focused on the fact that some moves of a play in an innocent strategy can be exchanged, and studied games whose rules explicitly describe which moves should be commutable. Our game model is also inspired by [27] (and [28]) but we focused on a different aspect of the alternating homotopy, that is, the fact that the connection between a successive pair of O- and P-moves (in HO/N innocent strategies) are quite tight (see also [25,36]); in our game model, a strategy explicitly describes indispensable connections $\rightsquigarrow$ between events. Because of these differences, their game model differs from ours; indeed our strategy is not necessarily *positional*. Nevertheless those models seems closely related; for example, it seems worth investigating the connection between *scheduled strategies* [30] and cycle-free composition.

A related approach using a map of *event structures* has been proposed by Rideau and Winskel [34] and extensively studied recently [9,10]. In this game model, a strategy is a map from an event structure describing the internal causal relation to another event structure expressing the observable events. We think that their model should be closely related to the (pre)sheaf version [36] of our game model, although we have not established any formal relationship yet.

From a technical point of view, an important difference between above models and our model is the way to deal with duplication of moves. Our model uses HO/N-style justification pointers, whereas the above models use the idea of *thread indexing* [10,26] in the style of AJM game model [1]. Both approaches have advantages and disadvantages (for example, an advantage of the HO/N-style is that a morphism is a strategy, not an equivalence class of strategies modulo reindexing). Hence we think that it is good to have a truly concurrent model using justification pointers.

Laird [24] briefly discussed an idea of a truly concurrent version of his interleaving game model, introducing the notion of *justified pomset*. His idea is very closed to ours; indeed a play s in our game model can be seen as a pomset $(V_s, \overrightarrow{s}^*)$ ordered by (reflexive transitive closure of) the adjacent relation $\overrightarrow{s}^*$.

A DAG-based reformulation of the HO/N game model is a reminiscent of *L-nets* [13,17]. The conditions required for L-nets are essentially the same as those we require for plays, though L-nets corresponds to strategies, not to plays. An interpretation of the π-calculus using *differential nets* [15] seems to be relevant to our development.

The game-semantics study of this paper has many parallels to the syntactic study of the π-calculus. The relationship between the HO/N game model for PCF [21] and the π-calculus has originally been studied by Hyland and Ong themselves [20], who gave a translation from PCF terms to processes of the π-calculus based on the idea of their game model. The π-terms representing sequential functional computation can be characterised by a simple type system proposed by Berger, Honda and Yoshida [5], which lead to the type system of [19]. We conjecture that processes typed by the simple type system of [5,19] is related to relationally-describable processes. Boreale [6] gave an encoding from the asynchronous π-calculus to the *internal* π *-calculus* [35]. Our game model can be seen as a variant of the encoding by regarding the plays as the processes of the *linear* internal π-calculus, in which each name must be used exactly once.

There are some pieces of work based on the techniques other than games but related to this work, such as event structure semantics of several variants of the π-calculus by Crafa, Varacca and Yoshida [11,12] and Varacca and Yoshida [38], and a data-flow semantics by Jagadeesan and Jagadeesan [22].

7 Conclusion and Future Work

We have developed a truly concurrent version of the HO/N game model [21,32], in which a computation is represented by a DAG of messages instead of a sequence. The resulting game model has the categorical structure needed to

interpret the asynchronous π-calculus proposed by Laird [24]. By using the connection between our model and Laird's model [24], we have proved soundness of the interpretation of the processes in our concurrent game model. This is the first truly concurrent game semantics for the π-calculus.

We have several topics left for future work:

- Formal description of the connection between plays and processes mentioned in Remark 2. By this connection, our game semantics can be seen as an approximation of the processes of the π-calculus by a linear π-calculus, which is a reminiscent of the *Taylor expansion* of the λ-calculus [16] (see also [37]).
- Development of the (pre)sheaf version of the game model [36], which would be related to the game model based on [34].
- Development of a model of the synchronous π-calculus. This requires us to deal with causal edges from O-moves and/or to P-moves. To simply relax the requirements for $\rightsquigarrow$ does not seem to work: for example, the copycat strategy of this paper is no longer the identity in the relaxed version.
- Development of a model of the π-calculus with the matching primitive. We expect that a nominal game model [31] would be useful for this purpose.

Acknowledgements. We would like to thank Naoki Kobayashi and anonymous referees for useful comments. This work is partially supported by JSPS Kakenhi Grant Number 15H05706 and JSPS Kakenhi Grant Number 16K16004.

References

1. Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Inf. Comput. **163**(2), 409–470 (2000)
2. Abramsky, S., McCusker, G.: Linearity, sharing and state: a fully abstract game semantics for idealized algol with active expressions. Electr. Notes Theor. Comput. Sci. **3**, 2–14 (1996)
3. Abramsky, S., Melliès, P.-A.: Concurrent games and full completeness. In: 14th Annual IEEE Symposium on Logic in Computer Science, pp. 431–442 (1999)
4. Baillot, P., Danos, V., Ehrhard, T., Regnier, L.: Timeless games. In: 11th International Workshop on Computer Science Logic, pp. 56–77 (1997)
5. Berger, M., Honda, K., Yoshida, N.: Sequentiality and the pi-calculus. In: TLCA, pp. 29–45 (2001)
6. Boreale, M.: On the expressiveness of internal mobility in name-passing calculi. Theor. Comput. Sci. **195**(2), 205–226 (1998)
7. Boudes, P.: Thick subtrees, games and experiments. In: Curien, P.-L. (ed.) TLCA 2009. LNCS, vol. 5608, pp. 65–79. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02273-9_7
8. Castellan, S., Clairambault, P.: Causality vs. interleavings in concurrent game semantics. In: 27th International Conference on Concurrency Theory, CONCUR 2016, pp. 32:1–32:14 (2016)
9. Castellan, S., Clairambault, P., Winskel, G.: Symmetry in concurrent games. In: Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014, pp. 28:1–28:10 (2014)

10. Castellan, S., Clairambault, P., Winskel, G.: The parallel intensionally fully abstract games model of PCF. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, pp. 232–243 (2015)
11. Crafa, S., Varacca, D., Yoshida, N.: Compositional event structure semantics for the internal *pi*-calculus. In: CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, pp. 317–332 (2007)
12. Crafa, S., Varacca, D., Yoshida, N.: Event structure semantics of parallel extrusion in the pi-calculus. In: Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, pp. 225–239 (2012)
13. Curien, P.-L., Faggian, C.: An approach to innocent strategies as graphs. Inf. Comput. **214**, 119–155 (2012)
14. Di Gianantonio, P., Lenisa, M.: Innocent game semantics via intersection type assignment systems. In: Computer Science Logic 2013, CSL 2013, pp. 231–247 (2013)
15. Ehrhard, T., Laurent, O.: Interpreting a finitary pi-calculus in differential interaction nets. Inf. Comput. **208**(6), 606–633 (2010)
16. Ehrhard, T., Regnier, L.: Uniformity and the taylor expansion of ordinary lambda-terms. Theor. Comput. Sci. **403**(2–3), 347–372 (2008)
17. Faggian, C., Maurel, F.: Ludics nets, a game model of concurrent interaction. In: 20th IEEE Symposium on Logic in Computer Science (LICS 2005), pp. 376–385 (2005)
18. Ghica, D.R., Murawski, A.S.: Angelic semantics of fine-grained concurrency. Ann. Pure Appl. Logic **151**(2–3), 89–114 (2008)
19. Honda, K., Laurent, O.: An exact correspondence between a typed pi-calculus and polarised proof-nets. Theor. Comput. Sci. **411**(22–24), 2223–2238 (2010)
20. Hyland, J.M.E., Ong, C.-H.L.: Pi-calculus, dialogue games and PCF. In: Proceedings of the Seventh International Conference on Functional Programming Languages and Computer Architecture, FPCA 1995, pp. 96–107 (1995)
21. Hyland, J.M.E., Ong, C.-H.L.: On full abstraction for PCF: I, II, and III. Inf. Comput. **163**(2), 285–408 (2000)
22. Jategaonkar Jagadeesan, L., Jagadeesan, R.: Causality and true concurrency: a data-flow analysis of the pi-calculus. In: Alagar, V.S., Nivat, M. (eds.) AMAST 1995. LNCS, vol. 936, pp. 277–291. Springer, Heidelberg (1995). doi:10.1007/3-540-60043-4_59
23. Laird, J.: A game semantics of idealized CSP. Electr. Notes Theor. Comput. Sci. **45**, 232–257 (2001)
24. Laird, J.: A game semantics of the asynchronous π-calculus. In: 16th International Conference on CONCUR 2005 - Concurrency Theory, pp. 51–65 (2005)
25. Levy, P.B.: Morphisms between plays. In: Lecture Slides, GaLoP (2013)
26. Melliès, P.-A.: Asynchronous games 1: a group-theoretic formulation of uniformity (2003). (Unpublished manuscript)
27. Melliès, P.-A.: Asynchronous games 2: the true concurrency of innocence. Theor. Comput. Sci. **358**(2–3), 200–228 (2006)
28. Melliès, P.-A.: Game semantics in string diagrams. In: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, pp. 481–490 (2012)
29. Melliès, P.-A., Mimram, S.: Asynchronous games: innocence without alternation. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 395–411. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74407-8_27

30. Melliès, P.-A., Mimram, S.: From asynchronous games to concurrent games (2008). (Unpublished manuscript)
31. Murawski, A.S., Tzevelekos, N.: Nominal game semantics. Found. Trends Program. Lang. **2**(4), 191–269 (2016)
32. Nickau, H.: Hereditarily sequential functionals. In: Nerode, A., Matiyasevich, Y.V. (eds.) LFCS 1994. LNCS, vol. 813, pp. 253–264. Springer, Heidelberg (1994). doi:10.1007/3-540-58140-5_25
33. Power, J., Thielecke, H.: Closed Freyd-and kappa-categories. In: Automata, Languages and Programming. In: 26th International Colloquium, ICALP 1999, pp. 625–634 (1999)
34. Rideau, S., Winskel, G.: Concurrent strategies. In: Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, pp. 409–418 (2011)
35. Sangiorgi, D.: pi-calculus, internal mobility, and agent-passing calculi. Theor. Comput. Sci. **167**(1&2), 235–274 (1996)
36. Tsukada, T., Ong, C.-H.L.: Nondeterminism in game semantics via sheaves. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, pp. 220–231 (2015)
37. Tsukada, T., Ong, C.-H.L.: Plays as resource terms via non-idempotent intersection types. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016, New York, USA, July 5–8, 2016, pp. 237–246. ACM (2016)
38. Varacca, D., Yoshida, N.: Typed event structures and the linear pi-calculus. Theor. Comput. Sci. **411**(19), 1949–1973 (2010)