Asynchronous games: innocence without alternation

Paul-André Melliès, Samuel Mimram

Asynchronous games: innocence without alternation

Paul-André Melliès and Samuel Mimram*

Abstract. The notion of innocent strategy was introduced by Hyland and Ong in order to capture the interactive behaviour of λ-terms and PCF programs. An innocent strategy is defined as an alternating strategy with partial memory, in which the strategy plays according to its view. Extending the definition to non-alternating strategies is problematic, because the traditional definition of views is based on the hypothesis that Opponent and Proponent alternate during the interaction. Here, we take advantage of the diagrammatic reformulation of alternating innocence in asynchronous games, in order to provide a tentative definition of innocence in non-alternating games. The task is interesting, and far from easy. It requires the combination of true concurrency and game semantics in a clean and organic way, clarifying the relationship between asynchronous games and concurrent games in the sense of Abramsky and Melliès. It also requires an interactive reformulation of the usual acyclicity criterion of linear logic, as well as a directed variant, as a scheduling criterion.

1 Introduction

The alternating origins of game semantics. Game semantics was invented (or reinvented) at the beginning of the 1990s in order to describe the dynamics of proofs and programs. It proceeds according to the principles of trace semantics in concurrency theory: every program and proof is interpreted by the sequences of interactions, called *plays*, that it can have with its environment. The novelty of game semantics is that this set of plays defines a *strategy* which reflects the interactive behaviour of the program inside the *game* specified by the type of the program.

Game semantics was originally influenced by a pioneering work by Joyal [16] building a category of games (called Conway games) and alternating strategies. In this setting, a game is defined as a decision tree (or more precisely, a dag) in which every edge, called *move*, has a polarity indicating whether it is played by the program, called Proponent, or by the environment, called Opponent. A play is alternating when Proponent and Opponent alternate strictly – that is, when neither of them plays two moves in a row. A strategy is alternating when it contains only alternating plays.

The category of alternating strategies introduced by Joyal was later refined by Abramsky and Jagadeesan [2] in order to characterize the dynamic behaviour of proofs in (multiplicative) linear logic. The key idea is that the tensor product of linear logic, noted $\otimes$, may be distinguished from its dual, noted $\parr$, by enforcing a *switching policy* on plays – ensuring for instance that a strategy of $A \otimes B$ reacts to an Opponent move played in the subgame A by playing a Proponent move in the same subgame A.

* This work has been supported by the ANR Invariants algébriques des systèmes informatiques (INVAL). Physical address: Équipe PPS, CNRS and Université Paris 7, 2 place Jussieu, case 7017, 75251 Paris cedex 05, France. Email addresses: `mellies@pps.jussieu.fr` and `smimram@pps.jussieu.fr`.

The notion of *pointer game* was then introduced by Hyland and Ong, and independently by Nickau, in order to characterize the dynamic behaviour of programs in the programming language PCF – a simply-typed λ-calculus extended with recursion, conditional branching and arithmetical constants. The programs of PCF are characterized dynamically as particular kinds of strategy with partial memory – called *innocent* because they react to Opponent moves according to their own *view* of the play. This view is itself a play, extracted from the current play by removing all its "invisible" or "inessential" moves. This extraction is performed by induction on the length of the play, using the pointer structure of the play, and the hypothesis that Proponent and Opponent alternate strictly.

This seminal work on pointer games led to the first generation of game semantics for programming languages. The research programme – mainly guided by Abramsky and his collaborators – was extraordinarily successful: by relaxing in various ways the innocence constraint on strategies, it suddenly became possible to characterize the interactive behaviour of programs written in PCF (or in a call-by-value variant) extended with imperative features like states, references, etc. However, because Proponent and Opponent strictly alternate in the original definition of pointer games, these game semantics focus on sequential languages like Algol or ML, rather than on concurrent languages.

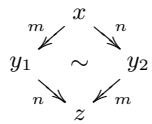
Concurrent games. This convinced a little community of researchers to work on the foundations of non-alternating games – where Proponent and Opponent are thus allowed to play several moves in a row at any point of the interaction. Abramsky and Melliès [3] introduced a new kind of game semantics to that purpose, based on *concurrent games* – see also [1]. In that setting, games are defined as partial orders (or more precisely, complete lattices) of *positions*, and strategies as *closure operators* on these partial orders. Recall that a closure operator σ on a partial order D is a function $\sigma : D \longrightarrow D$ satisfying the following properties:

$$\begin{array}{lll} \text{(1)}\ \sigma \text{ is increasing:} & \forall x \in D, & x \leq \sigma(x), \\ \text{(2)}\ \sigma \text{ is idempotent:} & \forall x \in D, & \sigma(x) = \sigma(\sigma(x)), \\ \text{(3)}\ \sigma \text{ is monotone:} & \forall x, y \in D, & x \leq y \Rightarrow \sigma(x) \leq \sigma(y). \end{array}$$

The order on positions $x \leq y$ reflects the intuition that the position y contains more information than the position x. Typically, one should think of a position x as a set of moves in a game, and $x \leq y$ as set inclusion $x \subseteq y$. Now, Property (1) expresses that a strategy σ which transports the position x to the position $\sigma(x)$ increases the amount of information. Property (2) reflects the intuition that the strategy σ delivers all its information when it transports the position x to the position $\sigma(x)$, and thus transports the position $\sigma(x)$ to itself. Property (3) is both fundamental and intuitively right, but also more subtle to justify. Note that the interaction induced by such a strategy σ is possibly non-alternating, since the strategy transports the position x to the position $\sigma(x)$ by "playing in one go" all the moves appearing in $\sigma(x)$ but not in x.

Asynchronous transition systems. Every closure operator σ is characterized by the set $\mathrm{fix}(\sigma)$ of its fixpoints, that is, the positions x satisfying $x = \sigma(x)$. So, a strategy is expressed alternatively as a set of positions (the set of fixpoints of the closure operator) in concurrent games, and as a set of alternating plays in pointer games. In order to understand how the two formulations of strategies are related, one should start from an obvious analogy with concurrency theory: pointer games define an *interleaving semantics* (based on sequences of transitions) whereas concurrent games define a *truly concurrent semantics* (based on sets of positions, or states) of proofs and programs. Now, Mazurkiewicz taught us this important lesson: a truly concurrent semantics may be regarded as an interleaving semantics (typically a transition system) equipped with *asynchronous*

tiles – represented diagrammatically as 2-dimensional tiles

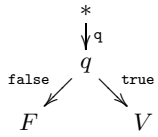


(1)

expressing that the two transitions m and n from the state x are *independent*, and consequently, that their scheduling does not matter from a truly concurrent point of view. This additional structure induces an equivalence relation on transition paths, called *homotopy*, defined as the smallest congruence relation $\sim$ identifying the two schedulings $m{\cdot}n$ and $n{\cdot}m$ for every tile of the form (1). The word *homotopy* should be understood mathematically as (directed) homotopy in the topological presentation of asynchronous transition systems as n*-cubical sets* [15]. This 2-dimensional refinement of usual 1-dimensional transition systems enables to express simultaneously the interleaving semantics of a program as the set of transition paths it generates, and its truly concurrent semantics, as the homotopy classes of these transition paths. When the underlying 2-dimensional transition system is contractible in a suitable sense, explained later, these homotopy classes coincide in fact with the positions of the transition system.

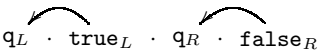
Asynchronous games. Guided by these intuitions, Melliès introduced the notion of *asynchronous game*, which unifies in a surprisingly conceptual way the two heterogeneous notions of pointer game and concurrent game. Asynchronous games are played on asynchronous (2-dimensional) transition systems, where every transition (or move) is equipped with a *polarity*, expressing whether it is played by Proponent or by Opponent. A *play* is defined as a path starting from the root (noted $*$) of the game, and a *strategy* is defined as a well-behaved set of alternating plays, in accordance with the familiar principles of pointer games. Now, the difficulty is to understand how (and when) a strategy defined as a set of plays may be reformulated as a set of positions, in the spirit of concurrent games.

The first step in the inquiry is to observe that the asynchronous tiles (1) offer an alternative way to describe *justification pointers* between moves. For illustration, consider the boolean game $\mathbb{B}$, where Opponent starts by asking a question $\mathtt{q}$, and Proponent answers by playing either $\mathtt{true}$ or $\mathtt{false}$. The game is represented by the decision tree



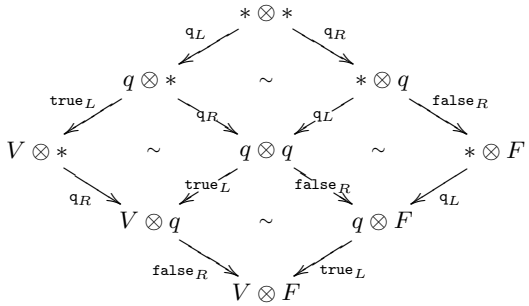
(2)

where $*$ is the root of the game, and the three remaining positions are called q, F and V (V for "Vrai" in French). At this point, since there is no concurrency involved, the game may be seen either as an asynchronous game, or as a pointer game. Now, the game $\mathbb{B} \otimes \mathbb{B}$ is constructed by taking two boolean games "in parallel." It simulates a very simple computation device, containing two boolean memory cells. In a typical interaction, Opponent starts by asking with $\mathtt{q}_L$ the value of the left memory cell, and Proponent answers $\mathtt{true}_L$; then, Opponent asks with $\mathtt{q}_R$ the value of the right memory cell, and Proponent answers $\mathtt{false}_R$. The play is represented as follows in pointer games:



The play contains two justification pointers, each one represented by an arrow starting from a move and leading to a previous move. Typically, the justification pointer from the move $\mathtt{true}_L$

to the move $\mathtt{q}_L$ indicates that the answer $\mathtt{true}_L$ is necessarily played after the question $\mathtt{q}_L$. The same situation is described using 2-dimensional tiles in the asynchronous game $\mathbb{B} \otimes \mathbb{B}$ below:



(3)

The justification pointer between the answer $\mathtt{true}_L$ and its question $\mathtt{q}_L$ is replaced here by a *dependency* relation between the two moves, ensuring that the move $\mathtt{true}_L$ cannot be permuted before the move $\mathtt{q}_L$. The dependency itself is expressed by a "topological" obstruction: the lack of a 2-dimensional tile permuting the transition $\mathtt{true}_L$ before the transition $\mathtt{q}_L$ in the asynchronous game $\mathbb{B} \otimes \mathbb{B}$.

This basic correspondence between justification pointers and asynchronous tiles allows a reformulation of the original definition of *innocent strategy* in pointer games (based on views) in the language of asynchronous games. Surprisingly, the reformulation leads to a purely local and diagrammatic definition of innocence in asynchronous games, which does not mention the notion of view any more. This diagrammatic reformulation leads then to the important discovery that innocent strategies are *positional* in the following sense. Suppose that two alternating plays $s, t : * \longrightarrow x$ with the same target position x are elements of an innocent strategy σ, and that m is an Opponent move from position x. Suppose moreover that the two plays s and t are equivalent modulo homotopy. Then, the innocent strategy σ extends the play $s \cdot m$ with a Proponent move n if and only if it extends the play $t \cdot m$ with the same Proponent move n. Formally:

$$s \cdot m \cdot n \in \sigma \quad \text{and} \quad s \sim t \quad \text{and} \quad t \in \sigma \quad \text{implies} \quad t \cdot m \cdot n \in \sigma. \tag{4}$$

From this follows that every innocent strategy σ is characterized by the set of *positions* (understood here as homotopy classes of plays) reached in the asynchronous game. This set of positions defines a closure operator, and thus a strategy in the sense of concurrent games. Asynchronous games offer in this way an elegant and unifying point of view on pointer games and concurrent games.

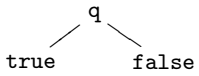
Concurrency in game semantics. There is little doubt that a new generation of game semantics is currently emerging along this foundational work on concurrent games. We see at least three converging lines of research. First, authors trained in game semantics – Ghica, Laird and Murawski – were able to characterize the interactive behaviour of various concurrent programming languages like Parallel Algol [12] or an asynchronous variant of the π-calculus [17] using directly (and possibly too directly) the language of pointer games. Then, authors trained in proof theory and game semantics – Curien and Faggian – relaxed the sequentiality constraints required by Girard on *designs* in ludics, leading to the notion of L-net [9] which lives at the junction of syntax (expressed as proof nets) and game semantics (played on event structures). Finally, and more recently, authors trained in process calculi, true concurrency and game semantics – Varacca and Yoshida – were able to extend Winskel's truly concurrent semantics of CCS, based on event structures, to a significant fragment of the π-calculus, uncovering along the way a series of nice conceptual properties of *confusion-free* event structures [25].

So, a new generation of game semantics for concurrent programming languages is currently emerging... but their various computational models are still poorly connected. We would like a regulating theory here, playing the role of Hyland and Ong pointer games in traditional (that is, alternating) game semantics. Asynchronous games are certainly a good candidate, because they combine interleaving semantics and causal semantics in a harmonious way. Unfortunately, they were limited until now to alternating strategies [20]. The key contribution of this paper is thus to extend the asynchronous framework to non-alternating strategies in a smooth way.

Asynchronous games without alternation. One particularly simple recipe to construct an asynchronous game is to start from a *partial order* of events where, in addition, every event has a polarity, indicating whether it is played by Proponent or Opponent. This partial order $(M, \preceq)$ is then equipped with a *compatibility relation* satisfying a series of suitable properties – defining what Winskel calls an *event structure*. A *position* x of the asynchronous game is defined as a set of *compatible* events (or moves) closed under the "causality" order:

$$\forall m, n \in M, \qquad m \preceq n \quad \text{and} \quad n \in x \quad \text{implies} \quad m \in x.$$

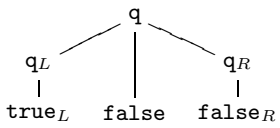
Typically, the boolean game $\mathbb{B}$ described in (2) is generated by the event structure



where $\mathtt{q}$ is an Opponent move, and $\mathtt{false}$ and $\mathtt{true}$ are two *incompatible* Proponent moves, with the positions q, V, F defined as $q = \{\mathtt{q}\}$, $V = \{\mathtt{q}, \mathtt{true}\}$ and $F = \{\mathtt{q}, \mathtt{false}\}$. The tensor product $\mathbb{B} \otimes \mathbb{B}$ of two boolean games is then generated by putting side by side the two event structures, in the expected way. The resulting asynchronous game looks like a flower with four petals, one of them described in (3). More generally, every formula of linear logic defines an event structure – which generates in turn the asynchronous game associated to the formula. For instance, the event structure induced by the formula

$$(\mathbb{B} \otimes \mathbb{B}) \multimap \mathbb{B} \tag{5}$$

contains the following partial order of compatible events:



(6)

which may be seen alternatively as a (maximal) position in the asynchronous game associated to the formula.

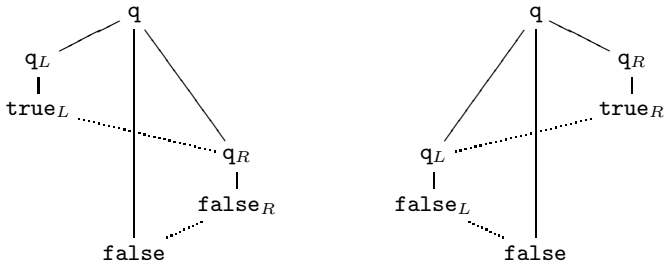
This game implements the interaction between a boolean function (Proponent) of type (5) and its two arguments (Opponent). In a typical play, Opponent starts by playing the move $\mathtt{q}$ asking the value of the boolean output; Proponent reacts by asking with $\mathtt{q}_L$ the value of the left input, and Opponent answers $\mathtt{true}_L$; then, Proponent asks with $\mathtt{q}_R$ the value of the right input, and Opponent answers $\mathtt{false}_R$; at this point only, using the knowledge of its two arguments, Proponent answers $\mathtt{false}$ to the initial question:

$$\mathtt{q} \cdot \mathtt{q}_L \cdot \mathtt{true}_L \cdot \mathtt{q}_R \cdot \mathtt{false}_R \cdot \mathtt{false} \tag{7}$$

Of course, Proponent could have explored its two arguments in the other order, from right to left, this inducing the play

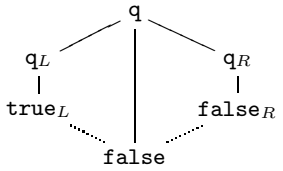
$$\mathtt{q} \cdot \mathtt{q}_R \cdot \mathtt{false}_R \cdot \mathtt{q}_L \cdot \mathtt{true}_L \cdot \mathtt{false} \tag{8}$$

The two plays start from the empty position $*$ and reach the same position of the asynchronous game. They may be seen as different linearizations (in the sense of order theory) of the partial order (6) provided by the game. Each of these linearizations may be represented by adding causality (dotted) edges between moves to the original partial order (6), in the following way:



(9)

The play (7) is an element of the strategy representing the *left* implementation of the *strict conjunction*, whereas the play (8) is an element of the strategy representing its *right* implementation. Both of these strategies are alternating. Now, there is also a *parallel* implementation, where the conjunction asks the value of its two arguments at the same time. The associated strategy is not alternating anymore: it contains the play (7) and the play (8), and moreover, all the (possibly non-alternating) linearizations of the following partial order:

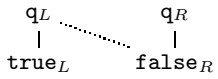


(10)

This illustrates an interesting phenomenon, of a purely concurrent nature: every play s of a concurrent strategy σ coexists with other plays t in the strategy, having the same target position x – and in fact, equivalent modulo homotopy. It is possible to reconstruct from this set of plays a *partial order* on the events of x, refining the partial order on events provided by the game. This partial order describes entirely the strategy σ under the position x: more precisely, the set of plays in σ reaching the position x coincides with the set of the linearizations of the partial order.

Our definition of innocent strategy will ensure the existence of such an underlying "causality order" for every position x reached by the strategy. Every innocent strategy will then define an event structure, obtained by putting together all the induced partial orders. The construction requires refined tools from the theory of asynchronous transition systems, and in particular the fundamental, but perhaps too confidential, notion of *cube property*.

Ingenuous strategies. We introduce in Section 4 the notion of *ingenuous* strategy, defined as a strategy regulated by an underlying "causality order" on moves for every reached position, and satisfying a series of suitable diagrammatic properties. One difficulty is that ingenuous strategies do not compose properly. Consider for instance the ingenuous strategy σ of type $\mathbb{B} \otimes \mathbb{B}$ generated by the partial order:



(11)

The strategy answers $\mathtt{true}_L$ to the question q_L, but answers $\mathtt{false}_R$ to the question q_R only if the question q_L has been already asked. Composing the strategy σ with the *right* implementation

of the strict conjunction pictured on the right handside of (9) induces a play $\mathtt{q} \cdot \mathtt{q}_R$ stopped by a *deadlock* at the position $\{\mathtt{q}, \mathtt{q}_R\}$. On the other hand, composing the strategy with the *left* or the *parallel* implementation is fine, and leads to a complete interaction.

This dynamic phenomenon is better understood by introducing two new binary connectives $\oslash$ and $\ominus$ called "before" and "after", describing sequential composition in asynchronous games. The game $A \oslash B$ is defined as the 2-dimensional restriction of the game $A \otimes B$ to the plays s such that every move played before a move in A is also in A; or equivalently, every move played after a move B is also in B. The game $A \ominus B$ is simply defined as the game $B \oslash A$, where the component B thus starts.

Now, the ingenuous strategy σ in $\mathbb{B} \otimes \mathbb{B}$ specializes to a strategy in the subgame $\mathbb{B} \oslash \mathbb{B}$, which reflects it, in the sense that every play $s \in \sigma$ is equivalent modulo homotopy to a play $t \in \sigma$ in the subgame $\mathbb{B} \oslash \mathbb{B}$. This is not true anymore when one specializes the strategy σ to the subgame $\mathbb{B} \ominus \mathbb{B}$, because the play $\mathtt{q}_L \cdot \mathtt{true}_L \cdot \mathtt{q}_R \cdot \mathtt{false}_R$ is an element of σ which is not equivalent modulo homotopy to any play $t \in \sigma$ in the subgame $\mathbb{B} \ominus \mathbb{B}$. For that reason, we declare that the strategy σ is innocent in the game $\mathbb{B} \oslash \mathbb{B}$ but *not* in the game $\mathbb{B} \otimes \mathbb{B}$.

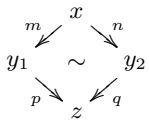
Innocent strategies. This leads to an interactive criterion which tests dynamically whether an ingenuous strategy σ is *innocent* for a given formula of linear logic. The criterion is based on *scheduling conditions* which recast, in the framework of non-alternating games, the *switching conditions* formulated by Abramsky and Jagadeesan for alternating games [2]. The idea is to *switch* every tensor product $\otimes$ of the formula as $\oslash$ or $\ominus$ and to test whether every play s in the strategy σ is equivalent modulo homotopy to a play $t \in \sigma$ in the induced subgame. Every such switching reflects a choice of scheduling by the counter-strategy: an innocent strategy is thus a strategy flexible enough to adapt to *every* scheduling of the tensor products by Opponent. An ingenuous strategy satisfies the scheduling criterion if and only if the underlying proof-structure satisfies a directed (and more liberal) variant of the acyclicity criterion introduced by Girard [13] and reformulated by Danos and Regnier [10]. A refinement based on the notion of synchronized clusters of moves enables then to strengthen the scheduling criterion, and to make it coincide with the usual non-directed acyclicity criterion.

We will establish in Section 4 that every ingenuous strategy may be seen alternatively as a closure operator, whose fixpoints are precisely the *halting positions* of the strategy. This connects (non alternating) asynchronous games to concurrent games. However, there is a subtle mismatch between the interaction of two ingenuous strategies seen as sets of plays, and seen as sets of positions. Typically, the right implementation of the strict conjunction in (9) composed to the strategy σ in (11) induces two different fixpoints in the concurrent game model: the deadlock position $\{\mathtt{q}, \mathtt{q}_R\}$ reached during the asynchronous interaction, and the complete position $\{\mathtt{q}, \mathtt{q}_L, \mathtt{true}_L, \mathtt{q}_R, \mathtt{true}_R, \mathtt{false}\}$ which is never reached interactively. The innocence assumption is precisely what ensures that this will never occur: the fixpoint computed in the concurrent game model is unique, and coincides with the position eventually reached in the asynchronous game model. In particular, innocent strategies compose properly.

Plan of the paper. We did our best to give in this introduction an informal but detailed overview of this demanding work, which combines together ideas from several fields: game semantics, concurrency theory, linear logic, etc. We focus now on the conceptual properties of innocent strategies, expressed in the diagrammatic language of asynchronous transition systems. The cube property is recalled in Section 2. The diagrammatic properties of positionality are studied in Section 3. The notion of ingenuous strategy is introduced in Section 4, and reformulated as a class of well-behaved closure operators in Section 5. Finally, the notion of innocent strategy is defined in Section 6, by strengthening the notion of ingenuous strategy with a scheduling criterion capturing the essence of the acyclicity criterion of linear logic.

2 The cube property

The *cube property* expresses a fundamental causality principle in the diagrammatic language of asynchronous transition systems [5, 24, 26]. The property is related to stability in the sense of Berry [6]. It was first noticed by Nielsen, Plotkin and Winskel in [21], then reappeared in [22] and [14, 18] and was studied thoroughly by Kuske in his PhD thesis; see [11] for a survey. The most natural way to express the property is to start from what we call an *asynchronous graph*. Recall that a *graph* $G = (V, E, \partial_0, \partial_1)$ consists of a set V of vertices (or *positions*), a set E of edges (or *transitions*), and two functions $\partial_0, \partial_1 : E \to V$ called respectively source and target functions. An *asynchronous graph* $G = (G, \diamond)$ is a graph G together with a relation $\diamond$ on coinitial and cofinal transition paths of length 2. Every relation $s \diamond t$ is represented diagrammatically as a 2-dimensional tile



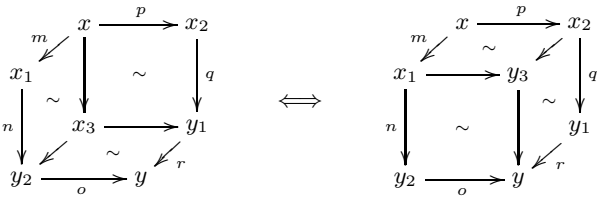
(12)

where $s = m \cdot p$ and $t = n \cdot q$. In this diagram, the transition q is intuitively the *residual* of the transition m after the transition n. One requires the two following properties for every asynchronous tile:

1. $m \neq n$ and $p \neq q$,
2. the pair of transitions (n, q) is uniquely determined by the pair of transitions (m, p), and conversely.

The main difference with the asynchronous tile (1) occurring in the asynchronous transition systems defined in [26, 23] is that the transitions are not labelled by events: so, the 2-dimensional structure is purely "geometric" and not deduced from an independence relation on events. What matters is that the 2-dimensional structure enables one to define a homotopy relation $\sim$ on paths in exactly the same way. Moreover, every homotopy class of a path $s = m_1 \cdots m_k$ coincides with the set of linearizations of a partial order on its transitions if, and only if, the asynchronous graph satisfies the following *cube property*:

> *Cube property:* a hexagonal diagram induced by two coinitial and cofinal paths $m \cdot n \cdot o$: $x \longrightarrow\!\!\!\!\!\rightarrow y$ and $p \cdot q \cdot r : x \longrightarrow\!\!\!\!\!\rightarrow y$ is filled by 2-dimensional tiles as pictured in the lefthand side of the diagram below, if and only it is filled by 2-dimensional tiles as pictured in the righthand side of the diagram:



The cube property is for instance satisfied by every asynchronous transition system and every transition system with independence in the sense of [26, 23]. The correspondence between homotopy classes and sets of linearizations of a partial order adapts, in our setting, a standard result on pomsets and asynchronous transition systems with dynamic independence due to Bracho, Droste and Kuske [7].

Every asynchronous graph G equipped with a distinguished initial position (noted $*$) induces an asynchronous graph $[G]$ whose positions are the homotopy classes of paths starting from the position $*$, and whose edges $m : [s] \longrightarrow [t]$ between the homotopy classes of the paths $s : * \longrightarrow\!\!\!\!\!\rightarrow x$ and $t : * \longrightarrow\!\!\!\!\!\rightarrow y$ are the edges $m : x \longrightarrow y$ such that $s \cdot m \sim t$. When the original asynchronous graph G satisfies the cube property, the resulting asynchronous graph $[G]$ is "contractible" in the sense that every two coinitial and cofinal paths are equivalent modulo homotopy.

So, we will suppose from now on that all our asynchronous graphs satisfy the cube property and are therefore contractible. The resulting framework is very similar to the domain of configurations of an event structure. Indeed, every contractible asynchronous graph defines a partial order on its set of positions, defined by reachability: $x \leq y$ when $x \longrightarrow\!\!\!\!\!\rightarrow y$. Moreover, this order specializes to a finite distributive lattice under every position x, rephrasing – by Birkhoff representation theorem – the fact already mentioned that the homotopy class of a path $* \longrightarrow\!\!\!\!\!\rightarrow x$ coincides with the linearizations of a partial order on its transitions. Finally, every transition may be labelled by an "event" representing the transition modulo a "zig-zag" relation, identifying the moves m and q in every asynchronous tile (12). The idea of "zig-zag" is folklore: it appears for instance in [23] in order to translate a transition system with independence into a labelled event structure.

3 Positionality in asynchronous games

Before considering 2-Player games, we express the notion of a positional strategy in 1-Player games. A 1-Player game $(G, *)$ is simply defined as an asynchronous graph G together with a distinguished initial position $*$. A play is defined as a path starting from $*$, and a strategy is defined as a set of plays of the 1-Player game, closed under prefix. A strategy σ is called *positional* when for every three paths $s, t : * \longrightarrow\!\!\!\!\!\rightarrow x$ and $u : x \longrightarrow\!\!\!\!\!\rightarrow y$, we have

$$s \cdot u \in \sigma \quad \text{and} \quad s \sim t \quad \text{and} \quad t \in \sigma \quad \text{implies} \quad t \cdot u \in \sigma. \tag{13}$$

This adapts the definition of (4) to a non-polarized setting and extends to the non-alternating setting. Note that a positional strategy is the same thing as a subgraph of the 1-Player game, where every position is reachable from $*$ inside the subgraph. This subgraph inherits a 2-dimensional structure from the underlying 1-Player game; it thus defines an asynchronous graph, denoted G_σ.

The advantage of considering asynchronous graphs instead of event structures appears at this point: the "event" associated to a transition is deduced from the 2-dimensional geometry of the graph. So, the "event" associated to a transition m in the graph G_σ describes the "causality cascade" leading the strategy to play the transition m ; whereas the "event" associated to the same transition m in the 1-Player game G is simply the move of the game. This subtle difference is precisely what underlies the distinction between the formula (5) and the various strategies (9) and (10). For instance, there are three "events" associated to the output move `false` in the parallel implementation of the strict conjunction, each one corresponding to a particular pair of inputs $(\texttt{true}, \texttt{false})$, $(\texttt{false}, \texttt{false})$, and $(\texttt{false}, \texttt{true})$. This phenomenon is an avatar of Berry stability, already noticed in [19].

From now on, we only consider positional strategies satisfying two additional properties:

1. forward compatibility preservation: every asynchronous tile of the shape (12) in the 1-Player game G belongs to the subgraph G_σ of the strategy σ when its two coinitial transitions $m : x \longrightarrow y_1$ and $n : x \longrightarrow y_2$ are transitions in the subgraph G_σ. Diagrammatically,

where the dotted edges indicate edges in G.

2. backward compatibility preservation: dually, every asynchronous tile of the shape (12) in the 1-Player game G belongs to the subgraph G_σ of the strategy σ when its two cofinal transitions $p : y_1 \longrightarrow z$ and $q : y_2 \longrightarrow z$ are transitions in the subgraph G_σ.

These two properties ensure that the asynchronous graph G_σ is contractible and satisfies the cube property. Contractibility means that every two cofinal plays $s, t : * \longrightarrow\!\!\!\!\!\rightarrow x$ of the strategy σ are equivalent modulo homotopy *inside* the asynchronous graph G_σ – that is, every intermediate play in the homotopy relation is an element of σ. Moreover, there is a simple reformulation as a set of plays of a positional strategy satisfying the two preservation properties: it is (essentially) a set of plays satisfying (1) a suitable cube property and (2) that $s \cdot m \in \sigma$ and $s \cdot n \in \sigma$ implies $s \cdot m \cdot p \in \sigma$ and $s \cdot n \cdot q \in \sigma$ when m and n are the coinitial moves of a tile (12). This characterization enables us to regard a positional strategy either as a set of plays, or as an asynchronous subgraph of the game.

4 Ingenuous strategies in asynchronous games

A 2-Player game $(G, *, \lambda)$ is defined as an asynchronous graph $G = (V, E, \diamond)$ together with a distinguished initial position $*$, and a function $\lambda : E \to \{-1, +1\}$ which associates a *polarity* to every transition (or move) of the graph. Moreover, the equalities $\lambda(m) = \lambda(q)$ and $\lambda(n) = \lambda(p)$ are required to hold in every asynchronous tile (12) of the asynchronous graph G. The convention is that a move m is played by *Proponent* when $\lambda(m) = +1$ and by *Opponent* when $\lambda(m) = -1$.

A strategy σ is called *ingenuous* when it satisfies the following properties:

1. it is *positional*, and satisfies the backward and forward compatibility preservation properties of Section 3,
2. it is *deterministic*, in the following concurrent sense: every pair of coinitial moves $m : x \longrightarrow y_1$ and $n : x \longrightarrow y_2$ in the strategy σ where the move m is played by Proponent, induces an asynchronous tile (12) in the strategy σ. Diagrammatically,

3. it is *courteous*, in the following sense: every asynchronous tile (12) where the two moves $m : x \longrightarrow y_1$ and $p : y_1 \longrightarrow z$ are in the strategy σ, and where m is a Proponent move, is an asynchronous tile in the strategy σ. Diagrammatically,

when $\lambda(m) = +1$.

Note that, for simplicity, we express this series of conditions on strategies seen as asynchronous subgraphs. However, the conditions may be reformulated in a straightforward fashion on strategies defined as sets of plays. The forward and backward compatibility preservation properties of Section 3 ensure that the set of plays of the strategy σ reaching the same position x is regulated

by a "causality order" on the moves occurring in these plays – which refines the "justification order" on moves (in the sense of pointer games) provided by the asynchronous game.

Our concurrent notion of determinism is not exactly the same as the usual notion of determinism in sequential games: in particular, a strategy may play several Proponent moves from a given position, as long as it converges later. Courtesy ensures that a strategy σ which accepts an Opponent move n *after* playing an independent Proponent move m, is ready to delay its own action, and to accept the move n *before* playing the move m. Together with the receptivity property introduced in Section 6, this ensures that the "causality order" on moves induced by such a strategy refines the underlying "justification order" of the game, by adding only order dependencies $m \preceq n$ where m is an Opponent move and n is a Proponent move. This adapts to the non-alternating setting the fact that, in alternating games, the causality order $p \preceq q$ provided by the view of an innocent strategy coincides with the justification order when p is Proponent and q is Opponent.

The experienced reader will notice that an ingenuous (and not necessarily receptive) strategy may add order dependencies $m \preceq n$ between two Opponent moves m and n. In the next Section, we will see that this reflects an unexpected property of concurrent strategies expressed as closure operators.

5 Ingenuous strategies in concurrent games

In this section, we reformulate ingenuous strategies in asynchronous games as strategies in concurrent games. The assumption that our 2-Player games are played on contractible asynchronous graphs induces a partial order on the set of positions, defined by reachability: $x \leq y$ when $x \longrightarrow\!\!\!\!\!\rightarrow y$. The concurrent game associated to an asynchronous game G is defined as the ideal completion of the partial order of positions reachable (in a finite number of steps) from the root $*$. So, the positions in the concurrent game are either *finite* when they are reachable from the root, or *infinite* when they are defined as an infinite directed subset of finite positions. The complete lattice D is then obtained by adding a top element $\top$ to the ideal completion. Considering infinite as well as finite positions introduces technicalities that we must confront in order to cope with infinite interactions, and to establish the functoriality property at the end of Section 6.

Now, we will reformulate ingenuous strategies in the asynchronous game G as *continuous* closure operators on the complete lattice D. By continuous, we mean that the closure operator preserves joins of directed subsets. Every closure operator σ on a complete lattice D induces a set of fixpoints:

$$\mathrm{fix}(\sigma) \quad = \quad \{\ x \in D \quad | \quad \sigma(x) = x\ \} \tag{14}$$

closed under arbitrary meets. Moreover, when the closure operator σ is continuous, the set $\mathrm{fix}(\sigma)$ is closed under joins of directed subsets. Conversely, every subset X of the complete lattice D closed under arbitrary meets defines a closure operator

$$\sigma \quad : \quad x \ \mapsto \ \bigwedge \{\ y \in X \quad | \quad x \leq y\ \} \tag{15}$$

which is continuous when the subset X is closed under joins of directed subsets. Moreover, the two translations (14) and (15) are inverse operations.

Now, every ingenuous strategy σ defines a set $\mathrm{halting}(\sigma)$ of *halting positions*. We say that a *finite* position x is *halting* when (1) the position is reached by the strategy σ and (2) there is no Proponent move $m : x \longrightarrow y$ in the strategy σ. The definition of a halting position can be extended to infinite positions by ideal completion: infinite positions are thus defined as downward-closed directed subsets $\hat{x}$ of finite positions. We do not provide the details here for lack of space.

It can be shown that the set of halting positions of an ingenuous strategy σ is closed under arbitrary meets, and under joins of directed subsets. It thus defines a continuous closure operator, noted σ°, defined by (15). The closure operator σ° satisfies a series of additional properties:

1. The domain $\mathrm{dom}(\sigma^\circ)$ is closed under (arbitrary) compatible joins,
2. For every pair of positions $x, y \in \mathrm{dom}(\sigma^\circ)$ such that $x \leq y$, either $\sigma^\circ(x) = \sigma^\circ(y)$ or there exists an Opponent move $m : \sigma^\circ(x) \longrightarrow z$ such that $z \leq_P \sigma^\circ(z)$ and $\sigma^\circ(z) \leq \sigma^\circ(y)$.

Here, the *domain* $\mathrm{dom}(\sigma^\circ)$ of the closure operator σ° is defined as the set of positions $x \in D$ such that $\sigma^\circ(x) \neq \top$; and the Proponent reachability order $\leq_P$ refines the reachability order $\leq$ by declaring that $x \leq_P y$ means, for two finite positions x and y, that there exists a path $x \longrightarrow\!\!\!\!\!\rightarrow y$ containing only Proponent moves; and then, by extending the definition of $\leq_P$ to all positions (either finite or infinite) in D by ideal completion.

Conversely, every continuous closure operator τ which satisfies the two additional properties mentioned above induces an ingenuous strategy σ in the following way. The *dynamic domain* of the closure operator τ is defined as the set of positions $x \in \mathrm{dom}(\tau)$ such that $x \leq_P \tau(x)$. So, a position x is in the dynamic domain of τ when the closure operator increases it in the proper way, that is, without using any Opponent move. The ingenuous strategy σ induced by the closure operator τ is then defined as the set of plays whose intermediate positions are all in the dynamic domain of τ. This defines an inverse to the operation $\sigma \mapsto \sigma^\circ$ from ingenuous strategies to continuous closure operators satisfying the additional properties 1. and 2. This pair of constructions thus provides a one-to-one correspondence between the ingenuous strategies and continuous closure operators satisfying the additional properties 1. and 2.

6 Innocent strategies

Despite the one-to-one correspondence between ingenuous strategies and concurrent strategies described in Section 5, there is a subtle mismatch between the two notions – which fortunately disappears when ingenuity is refined into innocence. On the one hand, it is possible to construct a category $\mathcal{G}$ of asynchronous games and ingenuous strategies, where composition is defined by "parallel composition+hiding" on strategies seen as sets of plays. On the other hand, it is possible to construct a category $\mathcal{C}$ of concurrent games and concurrent strategies, defined in [3], where composition coincides with relational composition on the sets of fixpoints of closure operators. Unfortunately – and here comes the mismatch – the translation $\mathcal{G} \to \mathcal{C}$ described in Section 5 is not functorial, in the sense that it does not preserve composition of strategies. This is nicely illustrated by the example of the ingenuous strategy (11) whose composition with the right implementation of the strict conjunction (9) induces a deadlock. More conceptually, this phenomenon comes from the fact that the category $\mathcal{G}$ is compact closed: the tensor product $\otimes$ is identified with its dual $\parr$.

This motivates a strengthening ingenuous strategies by a *scheduling criterion* which distinguishes the tensor product from its dual, and plays the role, in the non-alternating setting, of the *switching conditions* introduced by Abramsky and Jagadeesan for alternating games [2]. The criterion is sufficient to ensure that strategies do not deadlock during composition. In order to explain it here, we limit ourselves, for simplicity, to formulas of multiplicative linear logic (thus constructed using $\otimes$ and $\parr$ and their units 1 and $\bot$) extended with the two lifting modalities $\uparrow$ and $\downarrow$. The tensor product, as well as its dual, are interpreted by the expected "asynchronous product" of asynchronous graphs: in particular, every play s of $A \otimes B$ may be seen as a pair of plays (s_A, s_B) of A and B modulo homotopy. The two connectives $\otimes$ and $\parr$ are then distinguished by attaching a label $\otimes$ or $\parr$ to every asynchronous tile (12) appearing in the game. Typically, an asynchronous tile (12) between a move m in A and a move n in B is labelled

by $\otimes$ in the asynchronous game $A \otimes B$ and labelled by Γ in the asynchronous game $A \,\Gamma\, B$. The lifting modality $\uparrow$ (resp. $\downarrow$) is then interpreted as the operation of "lifting" a game with an initial Opponent (resp. Proponent) move. Note that there is a one-to-one relationship between the lifting modalities $\uparrow$ and $\downarrow$ appearing in the formula, and the moves of the asynchronous game G which denotes the formula. A nice aspect of our asynchronous approach is that we are able to formulate our scheduling criterion in two alternative but equivalent ways, each of them capturing a particular point of view on the correctness of proofs and strategies:

1. **a scheduling criterion** based on a switching as "before" $\otimes\!\!\!\!\!\!\bigcirc$ or "after" $\otimes\!\!\!\!\!\!\bigcirc$ of every tensor product $\otimes$ in the underlying formula of linear logic. As explained in the introduction, the scheduling criterion requires that every path s in the strategy σ is equivalent modulo homotopy to a path t in the strategy σ which respects the scheduling indicated by the switching. This is captured diagrammatically by orienting every $\otimes$-tile as $\otimes\!\!\!\!\!\!\bigcirc$ or $\otimes\!\!\!\!\!\!\bigcirc$ according to the switching, and by requiring that every play $s \in \sigma$ *normalizes* to a 2-dimensional normal form $t \in \sigma$ w.r.t. these semi-commutations [8] or standardization tiles [18].
2. **a directed acyclicity criterion** which reformulates the previous scheduling criterion along the lines of Girard's long trip criterion [13] and Danos-Regnier's acyclicity criterion [10]. Every position x reached by an ingenuous strategy σ induces a partial order $\preceq$ on the moves appearing in the position x. Every relation $m \preceq n$ of the partial order induces a *jump* between the lifting modalities associated to the moves m and n in the formula. The acyclicity criterion then requires that every switching of the Γ connectives as "Left" or "Right" (that is, in the sense of Girard) induces a graph with no *directed* cycles.

The scheduling criterion ensures that the operation $\sigma \mapsto \sigma^\circ$ defines a *lax* functor, in the sense that every fixpoint of $\sigma^\circ ; \tau^\circ$ is also a fixpoint of $(\sigma ; \tau)^\circ$. Now, an ingenuous strategy σ is called *asynchronous* when it additionally satisfies the following *receptivity* property: for every play $s : * \longrightarrow\!\!\!\!\!\rightarrow x$ and for every move $m : x \longrightarrow y$,

$$s \in \sigma \quad \text{and} \quad \lambda(m) = -1 \quad \text{implies} \quad s \cdot m \in \sigma.$$

The category $\mathcal{A}$ is then defined as follows: its objects are the asynchronous games equipped with $\otimes$-tiles and Γ-tiles, and its morphisms $A \to B$ are the asynchronous strategies of $A \multimap B$, defined as $A^* \,\Gamma\, B$, satisfying the scheduling criterion – where A^* is the asynchronous game A with Opponent and Proponent interchanged. The scheduling criterion ensures that the operation $\sigma \mapsto \sigma^\circ$ defines a strong monoidal functor $\mathcal{A} \to \mathcal{C}$ from the category $\mathcal{A}$ to the category $\mathcal{C}$ of concurrent games and concurrent strategies – thus extending the programme of [4, 20] in the non-alternating setting.

The notion of asynchronous strategy is too liberal to capture the notion of innocent strategy, at least because there exist asynchronous strategies which are not definable as the interpretation of a proof of MLL extended with lifting modalities $\uparrow$ and $\downarrow$. The reason is that the scheduling criterion tests only for *directed* cycles, instead of the usual non-directed cycles. On the other hand, it should be noted that the directed acyclicity criterion coincides with the usual non-directed acyclicity criterion in the situation treated in [2] – that is, when the formula is purely multiplicative (i.e. contains no lifting modality), every variable X and $X^\perp$ is interpreted as a game with a Proponent and an Opponent move, and every axiom link is interpreted as a "bidirectional" copycat strategy. The full completeness result in [3] uses a similar directed acyclicity criterion for MALL. Hence, directed acyclicity is a fundamental, but somewhat hidden, concept of game semantics.

On the other hand, we would like to mirror the usual non-directed acyclicity criterion in our asynchronous and interactive framework. This leads us to another stronger scheduling criterion based on the idea that in every play s played by an asynchronous strategy σ, an Opponent move m and a Proponent move n appearing in the play, and directly related by the causality order $m \preceq n$

induced by σ can be played synchronously. We write $m \leftrightharpoons n$ in that case, and say that two moves are synchronized in s when they lie in the same equivalence class generated by $\leftrightharpoons$. A cluster of moves t in the play $s = m_1 \cdots m_k$ is then defined as a path $t = m_i \cdots m_{i+j}$ such that all the moves appearing in t are synchronized. Every play s in the strategy σ can be reorganized as a sequence of maximal clusters, using a standardisation mechanism [18, 19]. The resulting clustered play is unique, modulo permutation of clusters, noted $\sim_{OP}$. This relation generalizes to the non-alternating case the relation $\sim_{OP}$ introduced in [20]. This leads to

3. **a clustered scheduling criterion** based, just as previously, on a switching as "before" $\oslash$ or "after" $\ominus$ of every tensor product $\otimes$ in the underlying formula of linear logic. The difference is that we ask that every clustered play s in the strategy σ may be reorganized modulo $\sim_{OP}$ as a clustered play which respects the scheduling indicated by the switching.

An asynchronous strategy is called innocent when it satisfies this stricter scheduling criterion. Although this tentative definition of innocence is fine conceptually, we believe that it has to be supported by further proof-theoretic investigations. An interesting aspect of our scheduling criteria is that they may be formulated in a purely diagrammatic and 2-dimensional way: in particular, the switching conditions are expressed here using the underlying logic MLL with lifting modalities $\uparrow$ and $\downarrow$ for clarity only, and may be easily reformulated diagrammatically.

7 Conclusion

Extending the framework of asynchronous games to non-alternating strategies requires an exploration of the fine-grained structure of causality, using classical concepts of concurrency theory like the cube property. Interestingly, it appears that enforcing good causality properties on strategies is not sufficient to combine game semantics and concurrency theory in a harmonious way. Indeed, we uncover a subtle and unexpected mismatch between composition performed in asynchronous games and composition performed in concurrent games. The mismatch is resolved by strengthening the purely causal notion of *ingenuous strategy* into the more contextual notion of *innocent strategy* by imposing a *scheduling criterion* which reformulates in a purely interactive and diagrammatic fashion the usual acyclicity criterion of linear logic. The criterion is sufficient to ensure the existence of a strong monoidal functor from the category of asynchronous games to the category of concurrent games.

In the near future, we plan to investigate the relationship between asynchronous games and L-nets. The scheduling criterion should help, since L-nets are themselves based on an acyclicity criterion [9]. We also plan to investigate the relationship between asynchronous games and the recent work by Varacca and Yoshida on confusion-free event structures and the π-calculus [25]. Our point of view that every strategy σ is an asynchronous graph G_σ embedded in its asynchronous game G is certainly a propitious starting point, since it enables a study of the formal properties of these embeddings, in the spirit of Nielsen and Winskel [26], which precisely underlies the construction in [25]. These connections would support our claim that asynchronous games provide indeed a valuable foundation for concurrency in game semantics.

Acknowledgments. We would like to thank Martin Hyland together with Pierre-Louis Curien, Claudia Faggian, Russ Harmer, Daniele Varacca, and Nobuko Yoshida for spontaneous and lively blackboard discussions.

References

1. S. Abramsky. Sequentiality vs. concurrency in games and logic. *Mathematical Structures in Computer Science*, 13(04):531–565, 2003.

2. S. Abramsky and R. Jagadeesan. Games and Full Completeness for Multiplicative Linear Logic. *The Journal of Symbolic Logic*, 59(2):543–574, 1994.
3. S. Abramsky and P.-A. Melliès. Concurrent games and full completeness. In *LICS*, volume 99, pages 431–442, 1999.
4. P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Timeless Games. In *CSL*, pages 56–77, 1997.
5. M.A. Bednarczyk. *Categories of asynchronous systems*. PhD thesis, University of Sussex, 1988.
6. G. Berry. Modèles complètement adéquats et stables des lambda-calculs typés. *Thèse de Doctorat d'État, Université Paris VII*, 1979.
7. F. Bracho, M. Droste, and D. Kuske. Representation of computations in concurrent automata by dependence orders. *Theoretical Computer Science*, 174(1-2):67–96, 1997.
8. M. Clerbout, M. Latteux, and Y. Roos. *The book of traces*, chapter 12, Semi-commutations, pages 487–552. World Scientific, 1995.
9. P.L. Curien and C. Faggian. L-Nets, Strategies and Proof-Nets. In *CSL*, pages 167–183. Springer, 2005.
10. V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28(3):181–203, 1989.
11. M. Droste and D. Kuske. Automata with concurrency relations – a survey. *Advances in Logic, Artificial Intelligence and Robotics*, pages 152–172, 2002.
12. D.R. Ghica and A.S. Murawski. Angelic Semantics of Fine-Grained Concurrency. In *FoSSaCS*, pages 211–225, 2004.
13. J.-Y. Girard. Linear logic. *TCS*, 50:1–102, 1987.
14. G. Gonthier, J.-J. Lévy, and P.-A. Melliès. An abstract standardisation theorem. In *Proc. of the 8th Annual Symposium on Logic in Computer Science*, pages 72–81, 1992.
15. É. Goubault. Geometry and Concurrency: A User's Guide. *Mathematical Structures in Computer Science*, 10(4):411–425, 2000.
16. A. Joyal. Remarques sur la theorie des jeux à deux personnes. *Gazette des Sciences Mathématiques du Quebec*, 1(4):46–52, 1977.
17. J. Laird. A game semantics of the asynchronous π-calculus. *Proceedings of 16th CONCUR*, pages 51–65, 2005.
18. P.-A. Melliès. Axiomatic Rewriting 1: A diagrammatic standardization theorem. *Lecture Notes in Computer Science*, pages 554–638.
19. P.-A. Melliès. Axiomatic rewriting 4: A stability theorem in rewriting theory. *Logic in Computer Science*, pages 287–298, 1998.
20. P.-A. Melliès. Asynchronous games 2: the true concurrency of innocence. In *Proceedings of the 15th CONCUR*, number 3170 in LNCS, pages 448–465. Springer Verlag, 2004.
21. M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part I. *Theoretical Computer Science*, 13:85–108, 1981.
22. P. Panangaden, V. Shanbhogue, and E.W. Stark. Stability and sequentiality in data flow networks. In *International Conference on Automates, Languages and Programming*, volume 443 of *LNCS*, pages 253–264. Springer Verlag, 1990.
23. V. Sassone, M. Nielsen, and G. Winskel. Models for concurrency: Towards a classification. *Theoretical Computer Science*, 170(1):297–348, 1996.
24. M.W. Shields. Concurrent Machines. *The Computer Journal*, 28(5):449–465, 1985.
25. D. Varacca and N. Yoshida. Typed Event Structures and the π-calculus. MFPS, pages 373–398. Elsevier, 2006.
26. G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of Logic in Computer Science*, volume 3, pages 1–148. Oxford University Press, 1995.