

Asynchronous Games 4

A Fully Complete Model of Propositional Linear Logic

Paul-André Mellies *

Abstract

We construct a denotational model of propositional linear logic based on asynchronous games and winning uniform innocent strategies. Every formula A is interpreted as an asynchronous game $[A]$ and every proof π of A is interpreted as a winning uniform innocent strategy $[\pi]$ of the game $[A]$. We show that the resulting model is fully complete: every winning uniform innocent strategy σ of the asynchronous game $[A]$ is the denotation $[\pi]$ of a proof π of the formula A .

1 Introduction

By promoting a dynamic and interactive view on proofs and programs, game semantics has modified our basic understanding of both logical systems and programming languages. The semantic analysis has revealed that formulas and types describe games, and that proofs and programs describe strategies. Hence, cut-eliminating a proof against its refutation, or evaluating a program against its environment, amounts to playing a strategy σ against a counter-strategy τ . The play $\langle \sigma | \tau \rangle$ resulting from the evaluation of σ against τ defines a “symbolic trajectory” which captures the essence as well as the syntactic details of the usual cut-elimination and evaluation procedures. Besides, the game paradigm was shown to apply to a large variety of programming languages, starting from PCF [4, 17, 31] and Idealized ALGOL [5].

The remarkable success of game semantics in the analysis of proofs and programs should not hide the fact that, after less than fifteen years of active research, game semantics remains a young and experimental subject, full of promises but still a long way from maturity. Most notably, game semantics bumps since its origins against three extremely puzzling facts, without offering a proper solution to any of them. Taken separately, each fact is generally accepted as a minor defect in a beautiful and fruitful theory. Taken to-

gether, they indicate that something fundamental remains to be understood about games. This is the starting point of this work. After reviewing the three facts, we explain how we managed to resolve them together in [29] with the notion of *asynchronous game* developed in [25, 27, 28]. This discussion conveys us to the main contribution of the paper: the formulation of a *fully complete* model of propositional linear logic, in the sense of [3].

Game semantics describes fragments of linear logic — not linear logic itself. Game semantics really emerged in the early 1990s when people realized after Blass [10] that the formulas of linear logic describe sequential games. Abramsky and Jagadeesan notice in [3] that Blass model does not provide a satisfactory interpretation of proofs as deterministic strategies, because the model does not quite define a category: strategies may be composed as in a category, but composition is not associative. Much work has been devoted subsequently to construct proper categories of sequential games and deterministic strategies. Despite these efforts, only *fragments* of linear logic could be interpreted in this way, most notably:

- multiplicative linear logic (MLL) has the tensor product and the duality of linear logic, but no additives and no exponentials. The first game models of MLL are given by Abramsky, Jagadeesan, Hyland and Ong [3, 16] who refine the category of Conway games constructed by Joyal [21] in order to obtain precise (that is, fully complete) models.
- intuitionistic linear logic (ILL) has the tensor product, the linear implication, the cartesian product, and the exponential modality of linear logic — but no duality. The first game models of ILL are given by Lammarche [22, 11] who linearizes in this way the sequential algorithm model of Berry and Curien [9]; and by McCusker [24] who combines the work by Hyland and Ong on arena games [17] and by Abramsky, Jagadeesan, Malacaria on token games [4]. Another interesting model of ILL appears in Hyland’s survey on game semantics [15].
- polarized linear logic (LLP) has all the connectives of

*This work has been supported by the FNS ACI Géométrie du Calcul (GEOCAL). Postal address: Equipe PPS, Université Paris VII, 2 place Jussieu, Case 7014, 75251 Paris Cedex 05, FRANCE. Email address: mellies@pps.jussieu.fr

linear logic — but formulas are restricted to Continuation Passing Style formulas (CPS formulas) by a polarity constraint. Girard introduces the idea of polarity in his work on LC and classical logic [13]. Laurent defines LLP and adapts to the logic the arena game model of Intuitionistic Logic [23].

Other less conventional game models have been designed, most notably a concurrent model of the multiplicative additive fragment [7], a non deterministic model of the multiplicative exponential fragment [8], and two slightly mysterious game models of propositional linear logic [18, 19]. However, it seemed hardly possible to formulate a sequential game model of propositional linear logic in the style of Blass games — until the asynchronous game model of propositional linear logic came out in [29]. The task of the present paper is precisely to refine this particular model in order to obtain a full completeness result.

Game semantics is affine — not linear. Another puzzling fact about game semantics is that the unit 1 of the tensor product and the unit \top of the cartesian product are generally identified in game models of ILL. Typically, the two units are interpreted as the *empty* game, see [22, 11, 24, 15]. From a logical point of view, the equality $1 = \top$ implies that the weakening rule may be applied on any formula A . From a categorical point of view, the unit \top is the terminal object of the underlying category of sequential games. Every strategy σ of a game B may be seen alternatively as a morphism $1 \rightarrow B$ of the category. The equality $1 = \top$ enables to compose this morphism to the canonical morphism $A \rightarrow \top$ associated to any object A . The resulting strategy

$$A \rightarrow 1 \rightarrow B$$

is *affine* in the sense that it does not need to interact on the game A in order to complete the interaction on the game B . This departs severely from linear logic, in which every proof of the linear implication

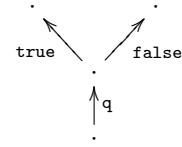
$$A \multimap B$$

uses its hypothesis exactly once. So, linearity in the precise sense of linear logic is not captured by usual game semantics, or only marginally by the notion of *strict* morphism [1]. Interestingly, linearity (identified to strictness) is the key ingredient of the asynchronous game model of linear logic formulated in [29] and will be one of the many elements of the present work.

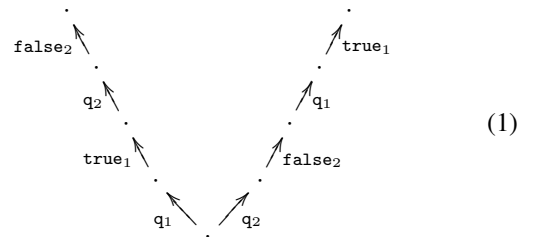
Game semantics is sequential — not positional. The most puzzling fact about mainstream game semantics (eg. arena games [17, 6] or token games [4]) is certainly that the whole theory is developed without ever mentioning the notion of *position* of a game. A game is understood simply as a

set of *plays*, where a play is defined as a (possibly justified) sequence of *moves*. Consequently, mainstream game semantics is purely “sequential” (that is, based on *sequences*) instead of being also positional. This lack of positionality conceals a fundamental aspect of games: different plays may reach the same position (or state) of the game; this position is *what* the plays compute, each play describing one particular trajectory to reach it.

By way of illustration, the boolean game \mathbb{B} implements a rudimentary computing device consisting of exactly *one* memory cell. The memory cell contains a boolean value, which may be either V (for “Vrai”, the French word for “True”) or F (for “False”). The game \mathbb{B} is played in two steps. The user (Opponent) asks the value of the cell by playing the move q (for “question”). Then, the device (Player) answers by playing either the move true when the value is V or the move false when the value is F . The game is summarized in the decision tree below.

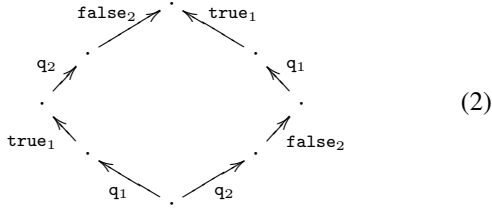


The tensor product of linear logic enables to combine two boolean games \mathbb{B}_1 and \mathbb{B}_2 in order to implement the computing device consisting of *two* memory cells (we use the superscripts 1 and 2 only to distinguish the two boolean games). The resulting game $\mathbb{B}_1 \otimes \mathbb{B}_2$ is played as follows. Suppose that the first cell has value V and the second cell has value F in the current state of the device. The user (Opponent) may start by asking the value of the first cell. In that case, she plays the move q_1 and the device (Player) reacts by playing the move true_1 . The user may then ask the value of the second cell by playing the move q_2 , and the device will react by playing the move false_2 . The sequence of interactions constructs the left branch of the (fragment of) decision tree representing the game $\mathbb{B}_1 \otimes \mathbb{B}_2$:



Now, the user may permute his order of inquiry, and start by asking the value of the second cell, then the value of the first cell. The resulting sequence of four moves constructs the right branch of the decision tree. Since the order of inquiry is somewhat irrelevant, it is natural to think that the two branches should reach the same *position* of the game. This

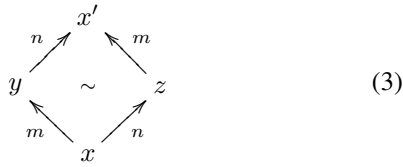
is represented in diagrams by bending the two branches of picture (1) and by drawing the decision tree as a graph:



This new picture is more informative than the previous one because it does not only indicate *how* Player and Opponent interact, but also *what* they compute: in that case, the state of the computing device, telling that the first cell has value V and the second cell has value F .

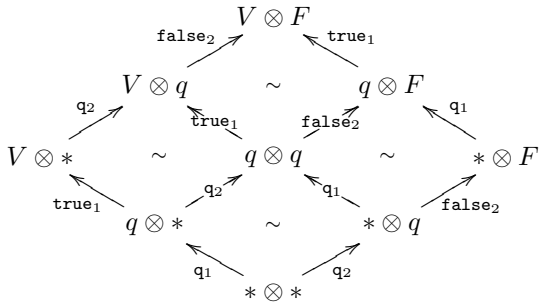
Surprisingly, the positional intuition is never really exploited in mainstream game semantics. Nonetheless, the author discovered a few years ago that positionality is just there, hidden in the core of the theory, in the notions of arena game and innocent strategy. This unexpected discovery was the starting point of asynchronous games.

Asynchronous games. Asynchronous games are games played on *asynchronous transition systems* generated by *event structures* [32, 36]. An asynchronous transition system is a directed graph equipped with 2-dimensional tiles of the shape of a 1×1 square:



Every such tile with edges $m \cdot n$ and $n \cdot m$ indicates that the two moves m and n may be permuted in a play, what one writes $m \cdot n \sim n \cdot m$. From this follows a *homotopy* relation \sim on plays, relating plays equal modulo permutation of moves. The term *homotopy* should be understood mathematically as (directed) homotopy in the topological presentation of asynchronous transition systems as a n -dimensional *cubical sets* [14].

By way of illustration, this enables to refine picture (2) as the asynchronous game represented below:



In this 2-dimensional picture, the two plays of pictures (1) and (2) appear to be homotopic after a series of four permutations. Note also that every position is given a name (like $q \otimes *$ or $V \otimes F$) which reflects the current state of the computation.

Positionality. We explain in [25, 27] how to formulate arena games and innocent strategies in the language of asynchronous games. This shift from arena games to asynchronous games is prerequisite if one takes positionality seriously. For instance, the naive idea that the current position x of a justified sequence s in an arena game may be simply defined as the set of moves (and pointers) which have been played... does not really work, for subtle reasons related to the interpretation of the exponential modality, and further discussed in [25].

A strategy of asynchronous game is defined in the usual “sequential” way, as a set of plays satisfying the usual properties of alternacy, determinism, etc. A strategy is innocent when it satisfies moreover two local consistency diagrams recalled in the Appendix (Figures 2 and 3). The two diagrams replace the condition originally formulated in arena games that an innocent strategy plays according to the current *Player view*, see [17, 31, 6] for details.

By definition, a strategy σ plays a position x when there exists a play $s \in \sigma$ with target x in the asynchronous game. The set of positions played by a strategy σ is denoted σ^\bullet . The main result of [27] is that every innocent strategy σ may be reconstructed from the set of positions σ^\bullet . In that sense, an innocent strategy is *positional* as well as *sequential*.

Moreover, once understood as positional strategies, innocent strategies compose just as *relations*: given two innocent strategies $\sigma : A \multimap B$ and $\tau : B \multimap C$, their composite $\sigma; \tau : A \multimap C$ is the innocent strategy characterized by the equation:

$$(\sigma; \tau)^\bullet = \{x \multimap z \mid \exists y, x \multimap y \in \sigma^\bullet \wedge y \multimap z \in \tau^\bullet\}. \quad (4)$$

Here, we write $x \multimap z$ for the position of $A \multimap C$ projecting on the position x in A and on the position z in C , and similarly for $x \multimap y$ and $y \multimap z$.

Linearity. Shifting from a sequential to a positional point of view enables to resolve the points raised earlier in a very natural way. To every position x , one assigns an integer $\kappa(x) \in \mathbb{Z}$ called its *payoff*. Every strategy σ is then required to be a *winning* strategy, playing only on positions of *positive* payoff. Now, a typical arena game (without bracketing policy) is translated as an asynchronous game B in which all the positions are of null payoff, except for the initial position $*_B$ which is of payoff $+1$. The game starts by a move m by Opponent, followed by a move n by Player:

$$*_B \xrightarrow{m} x \xrightarrow{n} y.$$

Suppose given another asynchronous game A of the same kind. In the definition of $A \multimap B$ appearing in [29] as well as in the present paper, Opponent starts the game by playing *simultaneously* a hidden move reaching the initial position $*_A$ of component A , and one initial move m reaching a position x in component B :

$$*_A \multimap B \xrightarrow{m} (*_A \multimap x). \quad (5)$$

A winning strategy of $A \multimap B$ cannot react to Opponent's move (5) by playing a move n in B . The resulting play

$$*_A \multimap B \xrightarrow{m} (*_A \multimap x) \xrightarrow{n} (*_A \multimap y) \quad (6)$$

would indeed reach the position $*_A \multimap y$, whose payoff happens to be -1 . We will not explain here how this payoff is calculated from the payoff $+1$ of $*_A$ in component A , and the null payoff of y in component B . This calculation is elementary, and will be explained at a later stage of the paper. What matters here is that the negative payoff of position $*_A \multimap y$ compels every winning strategy of $A \multimap B$ to be *strict*: if the strategy reacts to Opponent's first move, it has to react by playing a move in component A .

Propositional linear logic. Hence, assigning payoffs to positions treats the second puzzling fact: linearity. But what about the first one? Unexpectedly, the positional point of view happens to be also the key to full propositional linear logic. We explain how after describing briefly the problem.

The existing sequential game models of intuitionistic linear logic (ILL) define symmetric monoidal closed categories. In every symmetric monoidal closed category, there exists for every two objects A and \perp a canonical morphism:

$$\partial_A : A \longrightarrow (A \multimap \perp) \multimap \perp.$$

An object \perp is called *dualizing* when this morphism is an isomorphism for every object A . A symmetric monoidal closed category with a dualizing object \perp is called a $*$ -autonomous category. Besides, a model of propositional linear logic is simply a model of ILL in which the underlying category happens to be $*$ -autonomous, see [34, 26].

From that point of view, it should be noted that the existing symmetric monoidal closed categories of sequential games are *nearly* $*$ -autonomous. There exists indeed a tentative *dualizing* object in these categories: the sequential game \perp with a unique move played by Opponent. Of course, the game \perp is not *exactly* dualizing. In fact, the game $(A \multimap \perp) \multimap \perp$ is obtained by lifting the game A twice: once by a Player move, then by an Opponent move. The resulting game is not isomorphic to the original game A ... but not far from it! There exists indeed a strategy

$$\rho_A : (A \multimap \perp) \multimap \perp \longrightarrow A.$$

which defines with ∂_A a *retraction* between the two games. Of course, the composite $\rho_A; \partial_A$ does not coincide with the identity of $(A \multimap \perp) \multimap \perp$:

$$\rho_A; \partial_A \neq \text{id}_{(A \multimap \perp) \multimap \perp} \quad (7)$$

and this is all the problem! If the two strategies were coinciding, the strategy ∂_A would be an isomorphism, and the category would be $*$ -autonomous. This situation is interesting, but not particular to categories of games. Selinger [35] notices that it occurs in fact in any control category.

Again, the positional point of view leads to a very natural and elegant solution. Because the two strategies ∂_A and ρ_A are innocent, equation (7) may be rewritten in the positional fashion:

$$(\rho_A; \partial_A)^\bullet \neq (\text{id}_{(A \multimap \perp) \multimap \perp})^\bullet. \quad (8)$$

It appears after inspection that the two sets of positions in (8) differ only on the very "early" positions of the game — which all have positive payoffs $+1$ or more. In particular, the two sets of positions coincide *exactly* on the positions with null payoff. This motivates to focus on the set of positions with null payoff of a strategy σ :

$$\sigma^\circ = \sigma^\bullet \cap \{x \mid \kappa_A(x) = 0\}.$$

The equality follows:

$$(\rho_A; \partial_A)^\circ = (\text{id}_{(A \multimap \perp) \multimap \perp})^\circ. \quad (9)$$

This equality prompts us to call *external* any position of null payoff, and to identify any two innocent strategies σ and τ which play the same external positions — what we write $\sigma \simeq \tau$. The intuition is that every innocent strategy σ *realizes* the set of external positions σ° . A position is called *internal* when it is not external. This distinction between external and internal position revisits traditional realizability, by integrating the realizer and the object it realizes in the same computational space, instead of treating them as separate and heterogeneous entities.

Remarkably, the fact expressed by equation (4) that composition is relational remains valid when one considers only the external positions of two winning innocent strategies:

$$(\sigma; \tau)^\circ = \{x \multimap z \mid \exists y, x \multimap y \in \sigma^\circ \wedge y \multimap z \in \tau^\circ\}.$$

Consequently, the equivalence relation \simeq is preserved by composition: given two pairs of winning innocent strategies $\sigma, \sigma' : A \multimap B$ and $\tau, \tau' : B \multimap C$,

$$\sigma \simeq \sigma' \text{ and } \tau \simeq \tau' \Rightarrow \sigma; \tau \simeq \sigma'; \tau'.$$

We obtain in this way what we were looking for:

Proposition 1 *The category of asynchronous games and winning innocent strategies (modulo \simeq) defines a $*$ -autonomous category.*

Not only that: the category is cartesian, and leads to a model of propositional linear logic. The only difficulty is to interpret the exponential modality. As explained in [25], this is done by indexing copies in a way inspired by Geometry of Interaction [12] and token games [4]. Every asynchronous game is then equipped with a left and right group action on the indices of these copies. A group-theoretic notion of *uniform* strategy replaces the notion of *self-equivalent* strategy designed by Abramsky, Jagadeesan and Malacaria in token games [4]. This point is detailed in [25]. One obtains:

Proposition 2 *The category of asynchronous games and winning uniform innocent strategies (modulo \simeq) defines a model of propositional linear logic.*

Full completeness. In the model, every formula A of propositional linear logic is interpreted as an asynchronous game $[A]$ and every proof π of the formula A is interpreted as a winning uniform innocent strategy $[\pi]$ of the game $[A]$. The model is *fully complete* when the converse property holds: every winning uniform innocent strategy σ of the game $[A]$ is the denotation $\sigma \simeq [\pi]$ of a proof π of the formula A . The model formulated in [29] is not fully complete, because it identifies the two formulas

$$(A \& B) \otimes \top \quad \text{and} \quad (A \otimes \top) \& (B \otimes \top) \quad (10)$$

for any given formulas A and B . Note that the same problem occurs in the usual game models of intuitionistic linear logic, because the unit \top coincides with the unit 1.

From the proof search point of view, the introduction rule of the additive unit \top :

$$\overline{\vdash \Gamma, \top}$$

plays the role of a *garbage collector*: once the rule has collected the context Γ , the proof search succeeds. Proving one of the two formulas (10) in a context Γ consists in applying the garbage collector on a piece Δ of the context before proving the formulas A and B . The collected piece is the same for the two formulas A and B in the case of the formula $(A \& B) \otimes \top$, whereas it is selected independently for each formula A and B in the case of the formula $(A \otimes \top) \& (B \otimes \top)$. Consequently, the linear implication

$$(A \& B) \otimes \top \multimap (A \otimes \top) \& (B \otimes \top).$$

holds in (intuitionistic) linear logic, but not its converse.

Bracketing revisited. In order to obtain a fully complete model of propositional linear logic, we thus need to refine the notion of winning strategy given in [29]. We achieve this by revisiting the well-bracketing condition of arena games [17, 31, 6]. We observe first that usual well-bracketing may be reformulated by assigning a payoff to

every *path* of the game, indicating roughly the number of pending questions appearing in the path; then by requiring that every *paths* $x \longrightarrow y$ between two positions $x, y \in \sigma^\bullet$ has *positive* payoff. However, the resulting condition is still not sufficient to obtain full completeness. We thus go further and require that every *walk* $x \longleftarrow y$ followed by the strategy σ between two positions $x, y \in \sigma^\bullet$, is of positive payoff, see Section 2 for a definition of walk. This generalized form of well-bracketing happens to be sufficient to establish full completeness. The reader may check for instance that it separates the two formulas (10).

Contributions of this work. We solve in this paper a famous and longstanding problem of linear logic. The solution is far from straightforward. It results from the stubborn and meticulous analysis of asynchronous games reported in [25, 27, 29]. Many ideas and techniques are imported from these three papers. But the key to the full completeness theorem is formulated here for the first time: the extension of the payoff function from finite positions to finite walks. The proof of full completeness is also new, and extremely instructive on the very nature of linear logic.

Related works. Much of the related work has been already mentioned. The interested reader will also find sequential games played on graphs in the reformulation by Hyland and Schalk of the sequential algorithm model [18].

Synopsis. The rest of the paper is structured in the following way. After the necessary preliminaries on event structures (Section 2) we define formally the notions of *asynchronous arena* and *winning uniform innocent* strategy (Section 3). We then construct the model of propositional linear logic (Section 4) and give an outline of the proof of full completeness (Section 5) before concluding (Section 6). We can only give an outline of the proof in this extended abstract; a detailed account is given in a draft full paper [30].

2 Event structures

Event structures. An *event structure* $(M, \leq, \#)$ is a partial order (M, \leq) whose elements are called *events*, equipped with a binary symmetric irreflexive relation $\#$, satisfying:

- the set $m \downarrow = \{n \in M \mid n \leq m\}$ is *finite* for every event m ,
- $m \# n \leq p$ implies $m \# p$ for every events m, n, p .

Two events $m, n \in M$ are called *incompatible* when $m \# n$, and *compatible* otherwise. Two events m and n are called *independent* when they are compatible: $\neg(m \# n)$, and *incomparable*: $\neg(m \leq n)$ and $\neg(n \leq m)$. In that case, we write:

$$m \text{ I } n.$$

An event m is *initial* when $m \downarrow = \{m\}$ and *second* when $m \downarrow$ contains only initial events besides the event m itself.

Positions. A *position* is a downward closed subset of (M, \leq) consisting of pairwise compatible events. The set of (possibly infinite) positions defines a domain (complete pointed partial order) denoted \mathcal{D} . The set of finite (resp. infinite) positions of this domain is denoted \mathcal{D}^{fin} (resp. \mathcal{D}^∞).

The positional graph. Every event structure A induces a graph \mathcal{G} whose nodes are the *finite* positions $x, y \in \mathcal{D}^{\text{fin}}$, and whose edges

$$m : x \rightarrow y$$

are the events verifying $y = x \uplus \{m\}$. Here, we write \uplus to mean that $y = x \cup \{m\}$ and that the move m is not element of x . Note that the graph \mathcal{G} is rooted, with root the empty position noted $*$.

Tiles. The positional graph is also equipped with 2-dimensional *tiles* expressing that two independent events m and n starting from the same position x may be permuted. We use the notation \sim to indicate a tile in our diagrams.

Walks. A walk w is a sequence

$$w = (x_0, m_1^{\epsilon_1}, x_1, \dots, x_{k-1}, m_k^{\epsilon_k}, x_k) \quad (11)$$

in which every x_i is a finite position, every m_i is a move, every ϵ_i is a signature in $\{+1, -1\}$, and

$$\begin{aligned} m_i &: x_{i-1} \rightarrow x_i \text{ when } \epsilon_i = +1, \\ m_i &: x_i \rightarrow x_{i-1} \text{ when } \epsilon_i = -1. \end{aligned}$$

In that case, we write $w : x_0 \longleftrightarrow x_k$ and say that w *crosses* the positions x_0, \dots, x_k . So, a walk is just a path in the non oriented graph associated to the graph \mathcal{G} . The concatenation of two walks $w_1 : x \longleftrightarrow y$ and $w_2 : y \longleftrightarrow z$ defines a walk noted $w_1 \cdot w_2 : x \longleftrightarrow z$. We write \mathcal{W} for the set of walks of the event structure.

3 Asynchronous games and strategies

3.1 The arenas

Asynchronous arenas. An *asynchronous arena* $A = (M_A, \leq_A, \#_A, \lambda_A, \kappa_A, \kappa_A^\infty)$ is an event structure whose events $m \in M_A$ are called the *moves* of the arena, equipped with:

- a *polarity* function $\lambda_A : M_A \rightarrow \{-1, +1\}$ on moves,
- a *payoff* function $\kappa_A : \mathcal{D}_A^{\text{fin}} + \mathcal{W}_A \rightarrow \mathbb{Z}$ on finite positions and walks,
- a *payoff* function $\kappa_A^\infty : \mathcal{D}_A^\infty \rightarrow \{-\infty, +\infty\}$ on infinite positions.

A move m with polarity $\lambda_A(m) = +1$ (resp. $\lambda_A(m) = -1$) is called a *Player* (resp. *Opponent*) move. Every asynchronous arena is required to satisfy the additional properties:

- every two initial moves m and n are incompatible: $m \#_A n$.
- the payoff of the initial position $*_A$ is either -1 or $+1$,
- the polarity of every initial move m is the *opposite* of the payoff of the initial position: $\lambda_A(m) = -\kappa_A(*_A)$,
- the polarity of every second move n is the payoff of the initial position: $\lambda_A(n) = \kappa_A(*_A)$,
- the payoff of an empty walk $w : x \longleftrightarrow_A x$ is null,
- the payoff of a non-empty walk $w : x \longleftrightarrow_A y$ which crosses the initial position $*_A$ is equal to the payoff $\kappa_A(y)$ of its target y .

The opposite of the payoff $\kappa_A(*_A)$ is called the *polarity* of the arena, denoted π_A . Thus, every initial move is *Opponent* in a negative game and *Player* in a positive game. A finite position x or a walk w is declared *winning* when its payoff is positive or null. An infinite position x is declared *winning* when its payoff is $+\infty$. A finite position x is *external* when its payoff $\kappa_A(x)$ is null, and *internal* otherwise.

3.2 The strategies

Path. A walk $w : x_0 \longleftrightarrow x_k$ as formulated in (11) is called a *path* from x_0 to x_k when $\epsilon_i = +1$ for every index i . We write $w : x_0 \longrightarrow x_k$ in that case.

A path is *alternating* when $\lambda(m_{i+1}) = -\lambda(m_i)$ for every index $i \in [1, \dots, k-1]$.

Play. A *play* is a path starting from the empty position $*_A$.

Strategy. A *strategy* σ is a set of alternating plays such that, for every positions x, y, z, z_1, z_2 :

1. the empty play $(*)$ is element of σ ,
2. every play $s \in \sigma$ starts by an *Opponent* move, and ends by a *Player* move,
3. for every play $s : * \longrightarrow x$, for every *Opponent* move $m : x \rightarrow y$ and *Player* move $n : y \rightarrow z$,
$$s \cdot m \cdot n \in \sigma \Rightarrow s \in \sigma.$$
4. for every play $s : * \longrightarrow x$, for every *Opponent* move $m : x \rightarrow y$ and *Player* moves $n_1 : y \rightarrow z_1$ and $n_2 : y \rightarrow z_2$,
$$s \cdot m \cdot n_1 \in \sigma \text{ and } s \cdot m \cdot n_2 \in \sigma \Rightarrow n_1 = n_2.$$

Thus, a strategy is a non empty set (Clause 1) of even-length plays (Clause 2) closed under even-length prefix (Clause 3) and deterministic (Clause 4).

3.3 The innocent strategies

The notion of *innocent* strategy is introduced in [27] in order to reformulate the notion of innocence formulated originally in the language of arena games. A strategy σ is called *innocent* when it is *backward consistent* and *forward consistent* in the following sense.

Backward consistency. A strategy σ is *backward consistent* (see Figure 2 in Appendix) when for every play s_1 , for every path s_2 , for every moves $m_1, n_1, m_2, n_2 \in M_A$, it follows from $s_1 \cdot m_1 \cdot n_1 \cdot m_2 \cdot n_2 \cdot s_2 \in \sigma$ and $n_1 \perp m_2$ and $m_1 \perp m_2$ that $n_1 \perp n_2$ and $m_1 \perp n_2$ and $s_1 \cdot m_2 \cdot n_2 \cdot m_1 \cdot n_1 \cdot s_2 \in \sigma$.

Forward consistency. A strategy σ is *forward consistent* (see Figure 3 in Appendix) when for every play s_1 and for every moves $m_1, n_1, m_2, n_2 \in M_A$, it follows from $s_1 \cdot m_1 \cdot n_1 \in \sigma$ and $s_1 \cdot m_2 \cdot n_2 \in \sigma$ and $m_1 \perp m_2$ and $m_2 \perp n_1$ that $m_1 \perp n_2$ and $n_1 \perp n_2$ and $s_1 \cdot m_1 \cdot n_1 \cdot m_2 \cdot n_2 \in \sigma$.

3.4 Positions and walks on a strategy

Finite position of a strategy. The set of finite positions played by a strategy σ is defined as

$$\sigma^\bullet = \{x \in \mathcal{D}_A \mid \exists s \in \sigma, s : *A \longrightarrow x\}.$$

Infinite position of a strategy. An infinite play s is defined as a sequence of moves:

$$*A \xrightarrow{m_1} x_1 \xrightarrow{m_2} x_2 \cdots x_{k-1} \xrightarrow{m_k} x_k \xrightarrow{m_{k+1}} x_{k+1} \cdots \quad (12)$$

The target position x of such an infinite play s is defined as:

$$x = \bigcup_{i \in \mathbb{N}} x_i = \{m_i \mid i \in \mathbb{N}\}.$$

In that case, we write $s : *A \xrightarrow{\infty} x$. Given a strategy σ , the set σ^∞ denotes the set of infinite plays (12) such that

$$*A \xrightarrow{m_1} x_1 \xrightarrow{m_2} \cdots \xrightarrow{m_{2k}} x_{2k}$$

is element of the strategy σ , for every index $k \in \mathbb{N}$. Then,

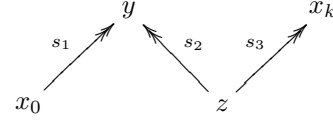
$$\sigma^{\bullet\bullet} = \{x \in \mathcal{D}_A \mid \exists s \in \sigma^\infty, s : *A \xrightarrow{\infty} x\}.$$

Alternating walks. A walk is *alternating* when

$$\lambda_A(m_{i+1}) \times \epsilon_{i+1} = -\lambda_A(m_i) \times \epsilon_i$$

for every index $i \in [1, \dots, k-1]$. Here, $(- \times -)$ denotes the usual product of two signatures in the ring \mathbb{Z} . So, at each step of an alternating walk w , either the direction ϵ_i or the polarity $\lambda(m_i)$ of the move changes. This generalizes

the previous notion of alternating play. A typical alternating walk $w : x_0 \longleftrightarrow x_k$ looks like this:



where s_1, s_2 and s_3 are alternating plays, the last moves of s_1 and s_2 are of the same polarity, and the first moves of s_2 and s_3 are of the same polarity.

Walk on a strategy. A walk w on a strategy σ is defined as an alternating walk (11) satisfying:

- $\lambda_A(m_1) \times \epsilon_1 = -1$ and $\lambda_A(m_k) \times \epsilon_k = +1$,
- every position x_{2i} is a position of the strategy σ , for $i \in [0, \dots, \frac{k}{2}]$.

Note that the length k of the walk is an even number by the first clause, and by the definition of an alternating walk. This justifies the use of the fraction $\frac{k}{2}$ as a natural number in the last clause.

3.5 The winning strategies

A strategy is *winning* when the four conditions below are satisfied:

1. **totality:** given any play $s \cdot m$ in which $s \in \sigma$ and m is an Opponent move, there exists a Player move n such that $s \cdot m \cdot n \in \sigma$,
2. **finite positions:** the payoff of every finite position $x \in \sigma^\bullet$ is positive or null,
3. **infinite positions:** the payoff of every infinite position $x \in \sigma^{\bullet\bullet}$ is positive or null,
4. **walks:** the payoff of every alternating walk on the strategy σ is positive or null.

This definition extends the definition of winning strategy in [29] which required only *totality* and positive payoff on *finite positions*. The payoff condition 3. on *infinite positions* reformulates the usual winning condition on strategies [15]. The payoff condition 4. on walks strengthens the usual *well-bracketing* condition of arena games — in order to achieve full completeness.

3.6 Asynchronous games and uniform strategies

Asynchronous games. An asynchronous game A is an asynchronous arena equipped with a left and right group actions on moves:

$$G_A \times M_A \times H_A \longrightarrow M_A$$

satisfying four *coherence axioms* given in [25]. These axioms ensure that the actions on moves extend to actions on plays, in a pointwise manner. The action of two elements $g \in G_A$ and $h \in H_A$ on a play $s = m_1 \cdots m_k$ is thus defined as follows:

$$g \cdot s \cdot h = (g \cdot m_1 \cdot h) \cdots (g \cdot m_k \cdot h).$$

Uniformity. The definition appears in [25]. A strategy σ is *uniform* when for every play $s \in \sigma$ and every $h \in H_A$, there exists $g \in G_A$ such that $g \cdot s \cdot h \in \sigma$. Uniformity reformulates the notion of *self-equivalent* strategy in [4].

4 The model of propositional linear logic

We explicate below our asynchronous game model of propositional linear logic. For lack of space, we will not describe here the group actions associated to each game; we refer the reader to [25] for a detailed account.

Dual. The dual A^\perp of an asynchronous game A is obtained by reversing the polarity of moves and the payoffs of positions and walks. Note that the operation is involutive: $A^{\perp\perp} = A$; and that the dual of a positive game is a negative game, and conversely.

Lifting. The *positive lifting* (with payoff p) of a negative asynchronous game A is the positive game $\downarrow^p A$ (often written $\downarrow A$ when no confusion is possible) defined below:

- $M_{\downarrow A} = M_A + \{\mathbf{new}\}$ with $m \#_{\downarrow A} n \iff m \#_A n$, and

$$m \leq_{\downarrow A} n \iff \begin{cases} m = \mathbf{new}, \\ \text{or } m \leq_A n, \end{cases}$$
- $\kappa_{\downarrow A}(*_{\downarrow A}) = -1$; and $\kappa_{\downarrow^p A}(\{\mathbf{new}\}) = p$; $\kappa_{\downarrow A}(y) = \kappa_A(x)$ when $y = x \uplus \{\mathbf{new}\}$,
- $\kappa_{\downarrow A}(w) = \kappa_A(w)$ when the walk w does not contain the initial position $*_{\downarrow A}$; $\kappa_{\downarrow A}(w) = 0$ when w is the empty walk on the initial position $*_{\downarrow A}$; $\kappa_{\downarrow A}(w) = \kappa_{\downarrow A}(y)$ when the walk $w : x \ll\!\!\!\rightarrow \downarrow A y$ is non empty and contains the initial position $*_{\downarrow A}$;
- $\kappa_{\downarrow A}^\infty = \kappa_A^\infty$.

This defines by duality a *negative lifting* $\uparrow^p A$ on positive games.

Affine vs. linear lifting. Two lifted games $\downarrow^p A$ and $\downarrow^q A$ differ only on the payoff of the position $\{\mathbf{new}\}$. This enables to distinguish the *linear* lifting and the *affine* lifting of a negative game A , defined respectively as $\downarrow^{+1} A$ and $\downarrow^0 A$.

Sum. The sum of two positive games A and B is the positive game $A \oplus B$ defined below:

- $M_{A \oplus B} = M_A + M_B$, $\leq_{A \oplus B} = \leq_A + \leq_B$, and

$$m \#_{A \oplus B} n \iff \begin{cases} m \#_A n, \\ \text{or } m \#_B n, \\ \text{or } m \in M_A \text{ and } n \in M_B, \\ \text{or } m \in M_B \text{ and } n \in M_A, \end{cases}$$

- $\kappa_{A \oplus B}(y) = \kappa_A(x)$ when $y = \mathbf{inl}(x)$ and $\kappa_{A \oplus B}(y) = \kappa_B(x)$ when $y = \mathbf{inr}(x)$, where \mathbf{inl} and \mathbf{inr} are the left and right injections $\mathcal{D}_A \rightarrow \mathcal{D}_{A \oplus B} \leftarrow \mathcal{D}_B$,
- $\kappa_{A \oplus B}(w) = \kappa_A(w)$ when w is a walk in the component A ; $\kappa_{A \oplus B}(w) = \kappa_B(w)$ when w is a walk in the component B ; $\kappa_{A \oplus B}(w) = \kappa_{A \oplus B}(y)$ when $w : x \ll\!\!\!\rightarrow_{A \oplus B} y$ is a walk on the two components A and B ,
- $\kappa_{A \oplus B}^\infty = \kappa_A^\infty + \kappa_B^\infty$.

This definition extends easily to the sum $\bigoplus_{i \in I} A_i$ of a family $(A_i)_{i \in I}$ of positive games.

Tensor product. Every positive game P may be seen as a sum $\bigoplus_{i \in I} \downarrow^{p_i} L_i$ of a family of lifted negative games L_i . The tensor product of two positive games

$$P = \bigoplus_{i \in I} \downarrow^{p_i} K_i \quad \text{and} \quad Q = \bigoplus_{j \in J} \downarrow^{q_j} L_j$$

is defined as the positive game

$$P \otimes Q = \bigoplus_{(i,j) \in I \times J} (\downarrow^{p_i} K_i \otimes \downarrow^{q_j} L_j)$$

where $\downarrow^{p_i} K_i \otimes \downarrow^{q_j} L_j$ is defined as the lifting

$$\downarrow^{p_i} K_i \otimes \downarrow^{q_j} L_j = \downarrow^{p_i \otimes q_j} N_{(i,j)}$$

of the negative game $N_{(i,j)}$ defined in the following way:

$$M_{N_{(i,j)}} = M_{K_i} + M_{L_j} \quad \leq_{N_{(i,j)}} = \leq_{K_i} + \leq_{L_j}$$

and

$$m \#_{N_{(i,j)}} n \iff \begin{cases} m \#_{K_i} n, \\ \text{or } m \#_{L_j} n. \end{cases}$$

The payoffs κ_x and κ_w and κ_x^∞ are computed by projecting the position x , and the walk w on each component K_i and L_j , and then by applying the calculation table of Figure 1 — in which the payoffs in K_i and L_j appear in the first row and first column. Note that the table is symmetric in K_i and L_j because the tensor product is commutative.

Units. The unit 0 is the positive game with no move. The unit 1 is the affine lifting $\downarrow^0 \top$ of the dual \top of the unit 0.

\otimes	$-\infty$	$-p$	0	$+p$	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-q$	$-\infty$	$-p - q$	$-q$	$-q$	$+\infty$
0	$-\infty$	$-p$	0	p	$+\infty$
$+q$	$-\infty$	$-p$	q	$p + q$	$+\infty$
$+\infty$	$-\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

Figure 1. The table for calculating the payoff $p \otimes q$ from the payoffs p and q .

Linear logic. The tensor product and the sum are called *positive* connectives because they are naturally defined on *positive* games, see [13, 23]. We use the linear lifting \downarrow^{+1} and \uparrow^{-1} in order to extend the definition to any (positive or negative) asynchronous game. Typically, the sum $A \oplus B$ and the tensor product $A \otimes B$ of two asynchronous games A and B is defined by lifting first the negative games (if any), then applying the earlier construction on positive games. The tensor product $A \otimes B$ is thus defined as:

$$\begin{aligned} A &\otimes \downarrow^{+1} B && \text{when } A \text{ is positive and } B \text{ negative,} \\ \downarrow^{+1} A &\otimes B && \text{when } A \text{ is negative and } B \text{ positive,} \\ \downarrow^{+1} A &\otimes \downarrow^{+1} B && \text{when } A \text{ and } B \text{ are negative,} \end{aligned}$$

and similarly for the sum construction. Identically, the positive lifting \downarrow^k of a positive game A is defined as:

$$\downarrow^k A = \downarrow^k (\uparrow^{-1} A).$$

De Morgan duality. The two *negative* connectives of linear logic are then deduced by De Morgan duality:

$$A \& B = (A^\perp \oplus B^\perp)^\perp \quad A \wp B = (A^\perp \otimes B^\perp)^\perp.$$

Exponentials. The exponential modality $!$ is defined by decomposing it as the *affine* lifting \downarrow^0 followed by an infinite tensor product:

$$!A = \bigotimes_{i \in \mathbb{N}} \downarrow^0 A \quad (13)$$

This decomposition is one of the two decompositions studied by Jacobs in [20].

As already explained in the introduction, we identify any two strategies σ and τ playing the same positions of null payoff, what we write $\sigma \simeq \tau$. We then construct a category with asynchronous games A, B as objects, and uniform winning innocent strategies of $A \multimap B = A^\perp \wp B$ as morphisms $A \rightarrow B$. We obtain:

Proposition 3 *The resulting category defines a model of propositional linear logic.*

Note that the three asynchronous games $A, \downarrow^p A$ and $\uparrow^p A$ are isomorphic in this category, when $p \neq 0$.

5 Full completeness

Our proof of full completeness is based on a *directed* proof search. We start from a uniform winning innocent strategy σ of the asynchronous game $[A]$, and we search a proof π of the formula A , such that $[\pi] = \sigma$. This search is *driven* by the strategy σ . The distinctive difficulty compared to other similar proofs of full abstraction for PCF [4, 17, 31] or of full completeness for LLP [23] is that the contexts involved are *not* affine anymore. This requires to deal explicitly with the context. The proof is done in three stages.

Stage 1. We establish full completeness for the multiplicative additive fragment enriched with the lifting modalities \downarrow^p and \uparrow^p . A good part of the difficulties are already there! When it is Player’s turn to play, the MALL sequent may be written as

$$\vdash \Gamma, \uparrow^p (\downarrow^{q_1} M \otimes \downarrow^{q_2} N)$$

where $\Gamma = \uparrow^{p_1} P_1, \dots, \uparrow^{p_k} P_k$ is a sequence of lifted *positive* formulas (we use the language of LLP here). Suppose that the strategy σ plays the move m which “plays” the tensor $\downarrow^{q_1} M \otimes \downarrow^{q_2} N$. Then, one may deduce from the diagrammatic properties of innocence that the strategy σ *splits* the context in three parts: Γ_1 is the part “connected” to the formula M , Γ_2 is the part “connected” to the formula N , and Γ_3 is the part not connected to anything. The payoff condition on walks then ensures that the strategy σ itself “splits” in two winning innocent strategies σ_1 and σ_2 of the sequents:

$$\vdash \Gamma_1, M \quad \vdash \Gamma_2, N$$

The proof is then concluded by induction on the size of σ .

Stage 2. The full completeness result established at stage 1 is extended to a *non uniform* variant of linear logic, in which a proof π of the formula $!A$ is defined as a *family* of proofs $(\pi_k)_{k \in \mathbb{N}}$ of the formula A . Every winning innocent strategy σ is shown to be the interpretation of a proof π .

Stage 3. Every *uniform* winning innocent strategy σ is shown to be the interpretation of a proof π of usual linear logic. This completes the proof:

Theorem 4 (Full Completeness) *Asynchronous games and uniform winning innocent strategies define a fully complete model of propositional linear logic.*

6 Conclusion and future works

We formulate a fully complete model of propositional linear logic for the first time. The construction requires to introduce a rich body of techniques: asynchronous games, internal vs. external positions, payoffs on walks, non uniform linear logic, etc. These techniques generalize and

clarify the usual notions of mainstream game semantics: arena games, well-bracketing, uniform strategies, etc. The next step will be to see how these new ideas clarify the current game models of references in programming languages [5, 2]. By this line of research, we expect to see an asynchronous *geometry* of programming languages and interactive behaviours emerge slowly — at the crossing point of the Curry-Howard isomorphism, the object-based semantics of states advocated by Reddy in [33] and the geometry of concurrency described in [14]. The full completeness theorem established in the present paper is certainly an important and encouraging step in that direction.

References

- [1] S. Abramsky. *Axioms for Definability and Full Completeness*, pages 55–75. MIT Press, 1999.
- [2] S. Abramsky, K. Honda, and G. McCusker. Fully abstract game semantics for general reference. In *13th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1998.
- [3] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994. Conference version appeared in FSTTCS '92.
- [4] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.
- [5] S. Abramsky and G. McCusker. Linearity, sharing and state: a fully abstract game semantics for idealized algol with active expressions, 1997.
- [6] S. Abramsky and G. McCusker. Game semantics. In *Computational logic*. Springer Verlag, 1999.
- [7] S. Abramsky and P.-A. Melliès. Concurrent games and full completeness. In *Logic in Computer Science 99*, pages 431–442, Trento, July 1999. IEEE Computer Society Press.
- [8] P. Baillot, V. Danos, T. Ehrhard, and L. Regnier. Believe it or not, ajm’s games model is a model of classical linear logic. In *LICS*, pages 68–75, 1997.
- [9] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.
- [10] A. Blass. A games semantics for linear logic. *Annals of Pure and Applied Logic*, 56:183–220, 1992.
- [11] P.-L. Curien. On the symmetry of sequentiality. In *Proceedings of Mathematical Foundations of Programming Semantics (MFPS’93)*, number 802 in LNCS. Springer Verlag, 1993.
- [12] J.-Y. Girard. Geometry of interaction I: interpretation of system F. In *Logic Colloquium 88*, pages 221–260, Amsterdam, July 1989. North Holland.
- [13] J.-Y. Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.
- [14] E. Goubault. Geometry and concurrency: A user’s guide. *Mathematical Structures in Computer Science*, 10(4), Aug. 2000.
- [15] M. Hyland. Game semantics. In *Semantics and logics of computation*, Publications of the Newton Institute. Cambridge University Press, 1997.
- [16] M. Hyland and L. Ong. Fair games and full completeness for multiplicative linear logic without the mix rule. Manuscript, 1992.
- [17] M. Hyland and L. Ong. On full abstraction for PCF: I, II and III. *Information and Computation*, 163(2):285–408, Dec. 2000.
- [18] M. Hyland and A. Schalk. Games on graphs and sequentially realizable functionals. In *Logic in Computer Science 02*, pages 257–264, Kopenhagen, July 2002. IEEE Computer Society Press.
- [19] P. Hyvernât. Synchronous games, deadlocks and linear logic. Submitted manuscript, Jan. 2005.
- [20] B. Jacobs. Semantics of weakening and contraction. *Ann. Pure Appl. Logic*, 69(1):73–106, 1994.
- [21] A. Joyal. Remarques sur la théorie des jeux à deux personnes. *Gazette des Sciences Mathématiques du Québec*, 1(4):46–52, 1977.
- [22] F. Lamarche. Sequentiality, games and linear logic. Manuscript, 1992.
- [23] O. Laurent. Polarized games (extended abstract). In *Proceedings of the seventeenth annual symposium on Logic In Computer Science*, pages 265–274, Copenhagen, July 2002. IEEE Computer Society Press.
- [24] G. McCusker. Games and full abstraction for a functional metalanguage with recursive types. Phd thesis, Imperial College, University of London, 1996. Published in Springer-Verlag’s Distinguished Dissertations in Computer Science series, 1998.
- [25] P.-A. Melliès. Asynchronous games 1: a group-theoretic formulation of uniformity. Available at <http://www.pps.jussieu.fr/~mellies/papers.html>, 2003.
- [26] P.-A. Melliès. Categorical models of linear logic revisited. Prépublication électronique PPS//03/09//n°22 (pp), Laboratoire Preuves, Programmes et Systèmes, Mar. 2003. To appear in *Theoretical Computer Science*.
- [27] P.-A. Melliès. Asynchronous games 2: the true concurrency of innocence. In P. Gardner and N. Yoshida, editors, *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR 2004)*, number 3170 in LNCS. Springer Verlag, 2004.
- [28] P.-A. Melliès. Asynchronous games 2: the true concurrency of innocence (journal version). 39 pages. Available at <http://www.pps.jussieu.fr/~mellies/papers.html>, 2004.
- [29] P.-A. Melliès. Asynchronous games 3: an innocent model of linear logic. In L. Birkedal, editor, *Category Theory in Computer Science*. Electronic Notes in Theoretical Computer Science, 2004.
- [30] P.-A. Melliès. Asynchronous games 4: a fully complete model of propositional linear logic. Draft paper, 2005.
- [31] H. Nickau. Hereditarily sequential functionals. In A. Nerode and Y. V. Matiyasevich, editors, *Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at St. Petersburg*, volume 813 of *Lecture Notes in Computer Science*, pages 253–264. Springer Verlag, 1994.

- [32] E. W. S. P. Panangaden, V. Shanbhogue. Stability and sequentiality in data flow networks. In A. Nerode and Y. V. Matiyasevich, editors, *International Conference on Automates, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 253–264. Springer Verlag, 1990.
- [33] U. Reddy. Global state considered unnecessary: Introduction to object-based semantics. In *Algol-like Languages*, volume 2 of *Progress in Theoretical Computer Science*, pages 227–295. Birkhauser, 1997.
- [34] R. Seely. Linear logic, *-autonomous categories and cofree coalgebras. In *Applications of categories in logic and computer science*, number 92. Contemporary Mathematics, 1989.
- [35] P. Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.
- [36] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1995.

Appendix: the two figures of innocence

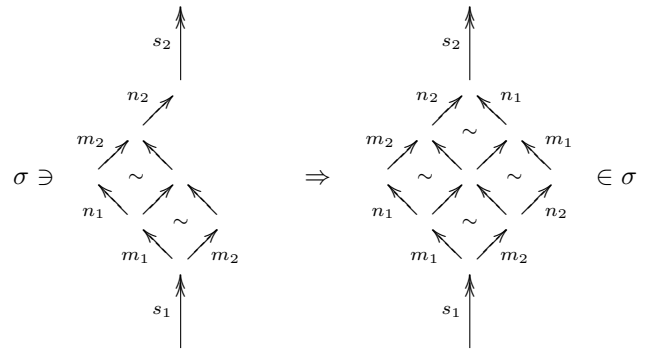


Figure 2. Local consistency (backward)

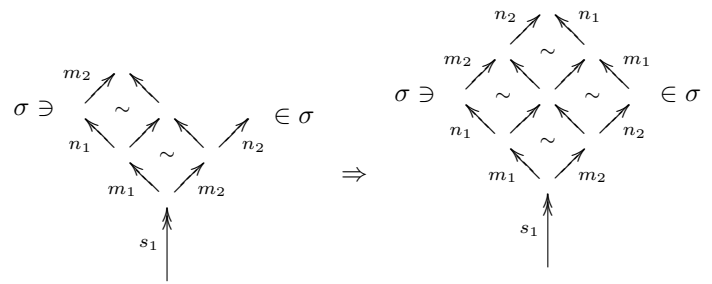


Figure 3. Local consistency (forward)