Sequentiality vs. Concurrency in Games and Logic

Samson Abramsky
Oxford University Computing Laboratory

March 22, 2001

Abstract

Connections between the sequentiality/concurrency distinction and the semantics of proofs are investigated, with particular reference to games and Linear Logic.

1 Introduction

We use Games and Logic as a mirror to understand an aspect of the sequentiality/concurrency distinction. We begin with the simple, intuitive notion of *polarized games* due to Blass [Bla72, Bla92], which prefigured many of the ideas in Linear Logic [Gir87], and which can be seen as a polarized version of ideas familiar from process calculi such as CCS [Mil99] (synchronization trees, prefixing, summation, the Expansion theorem). We analyze the 'shocking' fact that this very clear and intuitive idea leads to a non-associative composition; a kind of incompatibility between a purely sequential model and logic in a classical format. Two ways of addressing this issue have been found. One is to modify the syntax, by studying a 'hyper-sequentialized' version of sequent calculus, in which the current focus of attention in the proof is explicitly represented. This is the approach taken in Girard's Ludics [Gir01]. The other is to broaden the notion of game to encompass 'truly concurrent games'. In such games there is no longer a global polarization (we can have positions in which both players can move concurrently), although there are still local polarizations (each local decision is made by just one of the players). This idea of concurrent games was used by Abramsky and Melliès [AM99] to give a fully complete model for Linear Logic in its original form (i.e. not 'hyper-sequentialized'), and indeed the correspondence is with proof-nets, the 'parallel syntax for proof theory' in Girard's phrase [Gir95a]. In this way, the distinction between sequentiality and concurrency is reflected at a fundamental level, in the analysis of the 'space of proofs'.

The main aim of the present paper is to expose some of the conceptual issues underlying recent technical work on games and logic. The presentation is deliberately elementary in style, in the hope of making the discussion accessible both to concurrency theorists, and to those interested in the semantics of proofs—and of exhibiting a significant point of contact betwen these two fields.

The analysis of Blass games and the problem of non-associativity of composition, and the connection of this issue to the interleaving/true concurrency distinction, were first presented by the author in a lecture given at the Isaac Newton Institute for the Mathematical Sciences at Cambridge in 1995, during the programme on Semantics of Computation.

For the reader's convenience, some material on Linear Logic (specifically, the sequent calculus presentation of propositional Multiplicative-Additive Linear Logic (MALL)) is recalled in an appendix.

2 Blass Games as Polarized Processes

In this section we will view games as *polarized processes*. More precisely, we will develop a correspondence between certain 2-person games of perfect information and polarized deterministic synchronization trees.

2.1 Review of notions from process theory

We can describe (well-founded) deterministic synchronization trees inductively, as given by expressions of the form

$$\sum_{i \in I} a_i.P_i \qquad (i \neq j \;\Rightarrow\; a_i \neq a_j)$$

i.e. as the least set closed under the operation of *disjointly guarded summation* [Mil89]. It is understood that the summation, as in CCS, is associative and commutative (idempotence does not arise because of the disjointness condition). The basic case of the inductive definition is given by the empty summation, written 0 (the 'NIL' process of CCS). There are labelled transitions

$$\sum_{i \in I} a_i.P_i \xrightarrow{a_i} P_i$$

for each $i \in I$, giving the arcs from the root of the synchronization tree to its immediate sub-trees.

We could accomodate infinite branches in such trees by using a coinductive rather than an inductive definition. This issue is not important for our purposes here, for which it will be quite sufficient to consider only *finite* trees.

We can define interleaving (non-communicating) parallel composition on the synchronization trees thus: if $P = \sum_{i \in I} a_i.P_i$, and $Q = \sum_{j \in J} b_j.Q_j$, then

$$P \| Q \;=\; \sum_{i \in I} a_i.(P_i \| Q) + \sum_{j \in J} b_j.(P \| Q_j).$$

(In order to preserve the disjointness property in the summation, we require that the sorts of P and Q are disjoint [Mil89]—this will be tacitly assumed in the sequel.)

This is the Expansion Theorem [Mil89] in an appropriate version. Note that it can be taken as an (inductive or coinductive) *definition* of parallel composition as an operation on synchronization trees. It shows how to *eliminate* parallel composition in favour of purely sequential constructs. As such, it expresses the essence of the interleaving view of concurrency. Note that, in Milner's classification of the operations of process algebra [Mil89], guarded summation is built from the *dynamic* operations, while parallel composition is the key *static* operation. So the interleaving view gives primacy to the dynamic operations.

It will also be useful to recall the *left merge* operation, which is extensively used in Process Algebra [BW90]. This has the defining equation

$$P \mathbin{\lfloor\!\lfloor} Q \;=\; \sum_{i \in I} a_i.(P_i \| Q)$$

where $P = \sum_{i \in I} a_i.P_i$, and we can then express the Expansion Theorem by

$$P \| Q \;=\; P \mathbin{\lfloor\!\lfloor} Q + Q \mathbin{\lfloor\!\lfloor} P.$$

This 'biassed scheduling' will also turn out to have its counterpart in Games and Linear Logic.

2.2 Game Trees

We now consider game trees for 2-person games of perfect information. We will represent such games by *polarized* deterministic synchronization trees, i.e. by trees with one 'bit' of information added at each node to say which of the two player's turn it is to play at the game position corresponding to that node. We follow a standard practice of referring to the two players as P ('Proponent') and O ('Opponent'). In process terms, we can think of P as representing the System currently under consideration, and O as its Environment (see [AJ92b] for a more detailed account of the correspondence between notions of game theory and process theory).

Formally, we take games to be inductively (or, if preferred, coinductively) defined to be either a *product of games* or a *coproduct of games*

$$\prod_{i \in I} G_i \quad \text{or} \quad \coprod_{i \in I} G_i$$

where

$$\begin{aligned} \prod_{i \in I} G_i &= (\sum_{i \in I} i.G_i, O) \\ \coprod_{i \in I} G_i &= (\sum_{i \in I} i.G_i, P) \end{aligned}$$

(Since the identities of the 'actions' in the synchronization trees are irrelevant—they just label the possible moves in each position of the game—we use the indices in the summations themselves as the action names.)

There are now two versions of the empty game tree: the empty product $(0, O)$, and the empty coproduct $(0, P)$.

Clearly, we can define a formal 'game negation' which inverts the polarization at each node of the tree:

$$(\sum_{i \in I} i.G_i, \pi)^{\perp} \;=\; (\sum_{i \in I} i.G_i^{\perp}, \bar{\pi})$$

where $\bar{\mathrm{P}} = \mathrm{O}$, $\bar{\mathrm{O}} = \mathrm{P}$. This immediately yields the equations

$$
\begin{aligned}
(\coprod_{i\in I} G_i)^{\perp} &= \prod_{i\in I} G_i^{\perp} \\
(\prod_{i\in I} G_i)^{\perp} &= \coprod_{i\in I} G_i^{\perp} \\
G^{\perp\perp} &= G.
\end{aligned}
$$

In Linear Logic, the product and coproduct are the *additive* level of connectives, with the product written as & (With) and the coproduct as $\oplus$ (Plus). We see that in terms of the game trees, these connectives are accounted for by the *dynamic operations.* What about the static operations, in particular parallel composition? This has two polarized versions, corresponding to the *multiplicative* connectives $\otimes$ (Tensor) and $\invamp$ (Par) of Linear Logic.

2.3 Tensor

To define each of these binary connectives, we must consider four cases, corresponding to the possible polarizations of their arguments.

For the Tensor, three of the four cases are forced by the requirement that Tensor should distribute over coproduct. This can be taken as a desirable logical distributivity; more mathematically, to support a good notion of implication, Tensor with either argument fixed should be a left adjoint, giving rise e.g. to the adjunction

$$
\frac{A \otimes B \longrightarrow C}{A \longrightarrow B \multimap C}
$$

and hence must preserve all colimits. In game terms, thinking of Tensor as a form of conjunction, the onus is on P to defend both conjuncts, and hence P should move whenever it is his term to move in either of the two "sub-games". Thus we obtain the following 'equations' for the tensor:

$$
\begin{aligned}
\coprod_{i\in I} G_i \otimes H &= \coprod_{i\in I} (G_i \otimes H) \\
G \otimes \coprod_{j\in J} H_j &= \coprod_{j\in J} (G \otimes H_j) \\
\coprod_{i\in I} G_i \otimes \coprod_{j\in J} H_j &= \coprod_{i\in I,\, j\in J} (G_i \otimes H_j)
\end{aligned}
$$

In process terms we can think of these cases as 'biassed scheduling', of the form of the *left merge* operator studied in Process Algebra.

The final case to be considered is when the polarities of both the arguments to the Tensor are O: and here we simply apply the exact analogue of the Expansion Theorem for synchronization trees. If $G = \prod_{i\in I} G_i$, and $H = \prod_{j\in J} H_j$, then:

$$
G \otimes H = \prod_{i\in I} (G_i \otimes H) \times \prod_{j\in J} (G \otimes H_j).
$$

In game terms, this says that O has the freedom to move in either sub-game, which may then restore the situation to one in which P will be forced to move; this is analogous (and indeed, formally related) to the 'switching conditions' introduced in [AJ92b].

2.4 Par and Linear Implication

Given that we wish to have the De Morgan Duality

$$G \parr H \;=\; (G^{\perp} \otimes H^{\perp})^{\perp}$$

and the equivalence

$$G \multimap H \;=\; G^{\perp} \parr H$$

then the definitions of the other multiplicative connectives are forced by that of the tensor:

$$\prod_{i \in I} G_i \parr H \;=\; \prod_{i \in I} (G_i \parr H)$$

$$G \parr \prod_{j \in J} H_j \;=\; \prod_{j \in J} (G \parr H_j)$$

$$\prod_{i \in I} G_i \parr \prod_{j \in J} H_j \;=\; \prod_{i \in I, j \in J} (G_i \parr H_j)$$

and if $G = \coprod_{i \in I} G_i$, and $H = \coprod_{j \in J} H_j$, then:

$$G \parr H \;=\; \coprod_{i \in I} (G_i \parr H) + \coprod_{j \in J} (G \parr H_j).$$

The reader can similarly write down the definition for the Linear Implication.

This notion of game tree, and the interpretations of the multiplicative and additive connectives of Linear Logic, correspond exactly to the game semantics of Andreas Blass [Bla92] (in the 'relaxed form' in his terminology). Note that the additives provide the operations by which all such game trees can be constructed, and the multiplicatives are 'eliminated' by the polarized versions of the Expansion Theorem given above. Hence the additives have primacy in this form of game semantics, in exactly the same sense that the dynamic operations have primacy in the interleaving view of process algebra.

These definitions are very clear and natural; indeed, they seem forced, given that the interleaving point of view is being taken—as it is, implicitly at least, in the great bulk of the tradition in Game Theory, which concerns game trees in which play proceeds in a purely sequential fashion.

2.5 The second level: strategies

As we have seen, games add one crucial bit of structure to processes, namely the information about whose turn it is to play at each position. This opens the way to allowing a great deal of additional structure to be articulated explicitly. The usual process models are rather amorphous;

they allow a great variety of behaviours to be expressed easily, but the structural characteristics of various classes of behaviour are hard to capture.

For example, processes are used to model both specifications and implementations: both types, and the inhabitants of those types. In the world of games, one has the basic distinction between games and strategies. If games are represented as trees, then one can naturally represent strategies for P or O for a game G as certain kinds of subtree of G. From the point of view of process theory, both games and strategies are processes, but we can use the distinction offered by the game setting to build a more structured theory.

We exemplify this by defining O- and P-strategies for Blass games. For simplicity we shall consider only deterministic, total strategies for well-founded games, which have a defined response in every position which can arise. For $\pi \in \{P, O\}$ we define $S^{\pi}(G)$, the strategies for π on the game G, inductively by:

$$\begin{array}{lcl} S^{\pi}(\sum_{i \in I} i.G_i, \pi) & = & \{\bar{i}.\sigma_i \mid i \in I \wedge \sigma_i \in S^{\pi}(G_i)\} \quad (I \neq \varnothing) \\ S^{\pi}(0, \pi) & = & \{0\} \\ S^{\pi}(\sum_{i \in I} i.G_i, \bar{\pi}) & = & \{\sum_{i \in I} i.\sigma_i \mid \forall i \in I.\, \sigma_i \in S^{\pi}(G_i)\}. \end{array}$$

Thus a strategy for player π on a game G is a sub-synchronization tree of G which makes a deterministic choice of move at each position where π is to play in G, and has a response for every possible move by $\bar{\pi}$ where it is $\bar{\pi}$'s turn. Note that, following CCS [Mil89], we represent the player π's own moves by *output actions* $\bar{i}$, and the moves that π 'observes' $\bar{\pi}$ to make by *input actions* i.

Given $\sigma \in S^P(G)$, $\tau \in S^O(G)$, we can define the result of playing σ off against τ to be

$$(\sigma \mid \tau) \setminus S$$

using the CCS parallel composition and restriction operators [Mil89], where S contains all the actions appearing in G. This will result in a uniquely determined silent computation, which for well-founded G will end in an empty node $(0, \pi)$, where player π is to play but no move is possible. We decree that player π loses that play; and can then proceed to define the notion of winning strategy in the usual fashion. For non-well-founded game trees, infinite plays are possible, and we must add information to say who wins these. For more on this, see [Abr97], and for further development of the connection between processes, games and Linear Logic, see [Abr00]. We have already developed these ideas more than enough for our purposes in this paper.

3 Composition

We now turn to the crucial issue of *composition* of strategies. It seems that, with our simple but highly suggestive refinement from processes into Blass games, we have all the necessary ingredients to make a full-blown model of Multiplicative-Additive Linear Logic (MALL), at the level, not just of formulas, but of *proofs*. More precisely, we can see a clear idea for how to define a category $\mathcal{G}$ based on games and strategies which can serve as a model for MALL. The objects of $\mathcal{G}$ will be Blass games; a morphism from A to B will be a strategy $\sigma \in S^P(A \multimap B) = S^P(A^{\perp} \parr B)$.

We have defined interpretations of tensor product ($A \otimes B$), par ($A ⅋ B$), negation ($A^{\perp}$), products ($\prod_{i \in I} A_i$) and coproducts ($\coprod_{i \in I} A_i$) in this category. This is all so natural that one might take it for granted that the details must all work out.

However, there is a snag, of a rather fundamental kind. Moreover, this problem arises precisely from the polarization structure we have imposed on the Linear connectives, and hence from the interleaving view of concurrency on which, implicitly, it is based.

The problem concerns *associativity of composition*. It was known to Andreas Blass [Bla], although not mentioned in his papers on Game Semantics. It was noticed independently and discussed in [AJ92b].
To analyze associativity, we should consider strategies

$$\sigma : A \to B \qquad \tau : B \to C \qquad \theta : C \to D$$

which we can view as

$$\sigma : A^{\perp} ⅋ B \qquad \tau : B^{\perp} ⅋ C \qquad \theta : C^{\perp} ⅋ D.$$

Each of the games A, B, C, D can have either P or O polarity at the root, yielding 16 possibilities. Of these, *just one is problematic*, namely when:

- A, C have polarity O
- B, D have polarity P.

In this case, $A^{\perp}$ has polarity P, so in the game $A^{\perp} ⅋ B$ we are in the case of the Expansion Theorem for Par, and P has the choice of playing in either $A^{\perp}$ or B. Suppose that σ in fact chooses to play in $A^{\perp}$. Similarly, $C^{\perp}$ has polarity P, so in the game $C^{\perp} ⅋ D$, P has the choice of playing in either $C^{\perp}$ or D. Suppose that θ in fact chooses to play in D. Now in $B^{\perp} ⅋ C$, O must play first. Suppose that we first form the composition $\sigma ; \tau : A^{\perp} ⅋ C$. Then again O is to play first in this game. If we now form the composition

$$(\sigma ; \tau) ; \theta : A^{\perp} ⅋ D,$$

then since P is to play first, the only possibility is that the first move is that of θ, which is made in D. Now reasoning in an entirely similarly fashion, if we first form the composition $\tau ; \theta$, this must first wait for O to move, while $\sigma ; (\tau ; \theta)$ must begin with a move by P, which must then be made by σ, and hence must be made in $A^{\perp}$. We therefore conclude that:

- $(\sigma ; \tau) ; \theta$ plays initially in D
- $\sigma ; (\tau ; \theta)$ plays initially in $A^{\perp}$

and hence $(\sigma ; \tau) ; \theta \neq \sigma ; (\tau ; \theta)$.

This argument is very robust, and assumes very little about the specifics of how strategies are defined. It is based on the way that polarization is enforced in the definition of the multiplicative connectives in Blass games. As we have seen, that enforcement of polarization is rather directly related to the interleaving interpretation of concurrency.

	A	B	C	D	initial move by
0	O	O	O	O	O
1	O	O	O	P	θ
2	O	O	P	O	O
3	O	O	P	P	τ
4	O	P	O	O	O
5	O	P	O	P	O
6	O	P	P	O	?
7	O	P	P	P	σ
8	P	O	O	O	O
9	P	O	O	P	O
10	P	O	P	O	O
11	P	O	P	P	O
12	P	P	O	O	O
13	P	P	O	P	O
14	P	P	P	O	O
15	P	P	P	P	O

Table 1: How polarities determine the initial move

What of the other 15 possible polarizations of A, B, C, D? We can see fairly easily that none of them give rise to an analogous problem. Indeed, in 12 of these cases, O is to move first in $A^{\perp} ⅋ D$; while in the three remaining cases, one of the three strategies σ, τ or θ must unambiguously move first as forced by the polarization, whichever way the composition is bracketed. We list the possibilities explicitly in Table 1.

4 Syntactic Perspective

To show how easily the problem we described in the previous section can arise in MALL, we shall give a simple example of a proof in MALL whose interpretation in Blass games gives rise to exactly this situation.

Firstly, let α be a propositional atom. We define A, B, C and D to be the following formulas:

$$A = C = \alpha \& \alpha, \qquad B = D = \alpha \oplus \alpha.$$

Note that these formulas have exactly the polarities of the problematic case discussed in the previous section (remembering that & is product and $\oplus$ is coproduct). We define three proofs.

$$\Pi_1 : \qquad \dfrac{\dfrac{\dfrac{}{\vdash \alpha^{\perp}, \alpha}\,\mathsf{Id}}{\vdash \alpha^{\perp}, \alpha \oplus \alpha}\,\oplus\mathsf{R}}{\vdash \alpha^{\perp} \oplus \alpha^{\perp}, \alpha \oplus \alpha}\,\oplus\mathsf{R}$$

$$\Pi_2 : \quad \dfrac{\dfrac{\dfrac{}{\vdash \alpha^\perp, \alpha}\,\mathsf{Id} \qquad \dfrac{}{\vdash \alpha^\perp, \alpha}\,\mathsf{Id}}{\vdash \alpha^\perp \& \alpha^\perp, \alpha}\,\& \qquad \dfrac{\dfrac{}{\vdash \alpha^\perp, \alpha}\,\mathsf{Id} \qquad \dfrac{}{\vdash \alpha^\perp, \alpha}\,\mathsf{Id}}{\vdash \alpha^\perp \& \alpha^\perp, \alpha}\,\&}{\vdash \alpha^\perp \& \alpha^\perp, \alpha \& \alpha}\,\&$$

$$\Pi_3 : \quad \dfrac{\dfrac{\dfrac{}{\vdash \alpha^\perp, \alpha}\,\mathsf{Id}}{\vdash \alpha^\perp \oplus \alpha^\perp, \alpha}\,\oplus\mathsf{R}}{\vdash \alpha^\perp \oplus \alpha^\perp, \alpha \oplus \alpha}\,\oplus\mathsf{R}$$

The interpretation of these proofs as strategies σ, τ, θ gives rise to exactly the problematic situation described in the previous section. The relationship between the form of the sequent calculus proofs and the temporal properties of strategies (where do we move first?) is that

the ***last rule*** *of a proof corresponds to the* ***first move*** *in the associated strategy.*

Thus the first proof above, Π_1, indeed 'moves' first in $A^\perp$, while the third proof Π_3 moves first in D. On the other hand, rules introducing connectives of negative polarity, such as &, correspond to O-moves; so we see that in the second proof, Π_2, O is to move first.

Now we can consider the two ways of composing these proofs using cuts:

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \vdots\\ \vdash A^\perp, B\end{array} \qquad \begin{array}{c}\Pi_2\\ \vdots\\ \vdash B^\perp, C\end{array}}{\vdash A^\perp, C}\,\mathsf{Cut} \qquad \begin{array}{c}\Pi_3\\ \vdots\\ \vdash C^\perp, D\end{array}}{\vdash A^\perp, D}\,\mathsf{Cut}$$

$$\dfrac{\begin{array}{c}\Pi_1\\ \vdots\\ \vdash A^\perp, B\end{array} \qquad \dfrac{\begin{array}{c}\Pi_2\\ \vdots\\ \vdash B^\perp, C\end{array} \qquad \begin{array}{c}\Pi_3\\ \vdots\\ \vdash C^\perp, D\end{array}}{\vdash B^\perp, D}\,\mathsf{Cut}}{\vdash A^\perp, D}\,\mathsf{Cut}$$

corresponding to $(\sigma;\tau);\theta$ and $\sigma;(\tau;\theta)$ respectively. Performing cut-elimination on these two proofs leads to Π_3 and Π_1 repectively. These two proofs can be considered equal *up to permutation of rules.* The 'rigid' sequential interpretation of proofs as strategies does not satisfy this equality, and hence distinguishes the two proofs.

The analogy we wish to make is again with concurrency theory. The two proofs differ in the order in which they introduce the two occurences of $\oplus$ in the sequent $\vdash \alpha^\perp \oplus \alpha^\perp, \alpha \oplus \alpha$. Computationally, each of these introduction rules amounts to specifying one bit of information (are we in the left disjunct or the right disjunct?), so that we can see the rôle of these two introduction rules in the proof as like performing two 'actions', say:

- a: set the boolean variable B_1
- b: set the boolean variable B_2.

From the sequential, interleaving point of view, we must distinguish between the computation histories ab and ba, taking account of the order in which the 'events' a and b occur. In a true concurrency model, if the events a and b are causally idependent of each other, we can identify these two histories [WN95].

The purpose of proof nets [Gir87, Gir95a] and Geometry of Interaction [Gir89, Gir90, Gir95, DR93, DR95] was to find a more intrinsic representation of proofs in which the spurious ordering of rules imposed by sequent calculus was factored out. This is directly akin to the issue of representing true concurrency; and we can indeed see proof nets as a 'parallel syntax for proof theory' [Gir95a].

5 Discussion

It is useful to compare the issue we have raised in connection with Classical Linear Logic with a familiar issue in the proof theory of Classical Logic. There is a well known incompatibility between unconstrained Classical Logic proof theory and *confluence*, as shown very clearly and simply by a well-known example due to Yves Lafont [GLT89]. In our view, the example we have discussed shows equally clearly the incompatibility between unconstrained Classical Linear Logic proof theory and a purely sequential, interleaving view of the semantics of proofs.

There seem to be two approaches to the computational interpretation of proofs in Classical Logic.

1. 'Tame' the syntax, by restricting the permitted reductions, or by adding information to resolve the non-deterministic choices, and hence restoring confluence. This approach is typified by systems such as Girard's LC [Gir91], the systems LKT and LKQ of Danos, Joinet and Schellinx [DJS96], and the $\lambda\mu$-calculus of Parigot [Par92].

2. Study the full, 'untamed' non-confluent calculus, and find the computational structure which is there. This is typified by work such as [BB96, Urb00, Lai01].

In a similar fashion, there seem to be two possibilities for studying Linear Logic proofs.

1. Taming the syntax, in such a way as to control the scheduling, and to avoid the 'bad' combination of polarities which led to the problem with composition. Most game semantics applies to the *Intuitionistic* version of Linear Logic—indeed much of it applies only to the negative fragment [Abr97]. In game terms, this means that one only considers games in which Opponent starts, and in which the players strictly alternate. Under these constraints, no problems with composition arise. What is lost is any direct access to the natural game interpretation of classical (Linear) negation, by interchange of rôles between the players. However, it should be noted that these kind of games are quite sufficient to yield good models of the $\lambda\mu$-calculus, as shown e.g. in [Ong96, Lai99].

But this is not the only option. One can go beyond the scope of Intuitionistic Linear Logic, and study a more expressive, yet still 'tamed' system of Linear Logic. This is Linear Logic with focussing, as introduced by Andreoli [AP91], and used in the current work by Girard on Ludics [Gir01]. We shall study this approach in the next section.

2. Alternatively, we can extend the semantics to embrace true concurrency, and hence overcome the problems with composition while still interpreting the full, unconstrained original syntax of Linear Logic. This is what is done in the concurrent games model, which we shall discuss in Section 7.

6 The sequential way out: Focussing

We shall consider the focussing version of MALL used in [Gir01], and based on the system introduced in [AP91]. We shall refer to this system as $\mathrm{MALL}_{\mathsf{foc}}$. We shall only use the propositional part of the system, which we recapitulate here for the reader's convenience.

Firstly, the formulas of the system are *positive formulas* P, Q ..., built from propositional atoms α, β, ... by the grammar

$$P \ ::= \ \alpha \mid {\downarrow}P^{\perp} \mid P \oplus P \mid P \otimes P.$$

Here ${\downarrow}P^{\perp}$ is the 'shifted negation' of P, in which the shift is used to restore positive polarity. One can think of this as a kind of double negation interpretation: in game/process terms, it is *polarized prefixing*, in which the polarity is reversed at each node of the game for P (as in Linear negation), but then positive polarity is restored by prefixing with a 'dummy' P-move.

A sequent in $\mathrm{MALL}_{\mathsf{foc}}$ is an expression $\Gamma \vdash \Delta;\Sigma$, where Γ, Δ, Σ are finite multisets of formulas, with the following *stoup constraint*:

If the stoup Σ is non-empty, then it contains exactly one formula; moreover, in this case Γ consists only of propositional atoms.

Negative propositions are handled implicitly through the left hand sides of sequents.
The rules of $\mathrm{MALL}_{\mathsf{foc}}$ are as follows.

Cut

$$\frac{\Gamma \vdash \Delta; P \quad \Gamma', P \vdash \Delta';}{\Gamma, \Gamma' \vdash \Delta, \Delta';}\ ;\mathsf{Cut} \qquad \frac{\Gamma \vdash \Delta, P; \quad \Gamma', P \vdash \Delta';}{\Gamma, \Gamma' \vdash \Delta, \Delta';}\ \mathsf{Cut};$$

Identity

$$\frac{}{\alpha \vdash ; \alpha}\ \mathsf{Id}$$

Focussing

$$\frac{\Gamma \vdash \Delta; P}{\Gamma \vdash \Delta, P;}\ \mathsf{Foc}$$

Shift

$$\frac{\Gamma, P \vdash \Delta;}{\Gamma \vdash \Delta; {\downarrow}P^{\perp}}\ {\vdash}{\downarrow} \qquad \frac{\Gamma \vdash \Delta, P;}{\Gamma, {\downarrow}P^{\perp} \vdash \Delta;}\ {\downarrow}{\vdash}$$

Tensor

$$\frac{\Gamma \vdash \Delta; P \quad \Gamma' \vdash \Delta'; Q}{\Gamma, \Gamma' \vdash \Delta, \Delta'; P \otimes Q}\vdash\otimes \qquad \frac{\Gamma, P, Q \vdash \Delta;}{\Gamma, P \otimes Q \vdash \Delta;}\otimes\vdash$$

Plus

$$\frac{\Gamma \vdash \Delta; P}{\Gamma \vdash \Delta; P \oplus Q}\vdash\oplus l \qquad \frac{\Gamma \vdash \Delta; Q}{\Gamma \vdash \Delta; P \oplus Q}\vdash\oplus r$$

$$\frac{\Gamma, P \vdash \Delta; \quad \Gamma, Q \vdash \Delta;}{\Gamma, P \oplus Q \vdash \Delta;}\oplus\vdash$$

We now ask: to what extent can the problematic situation in standard MALL involving the proofs Π_1, Π_2, Π_3 which we discussed in section 4 be replicated in $\mathrm{MALL}_{\mathsf{foc}}$?

If we begin with Π_1, we can certainly find a reasonable analogue Π_1':

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\alpha \vdash ; \alpha}\mathsf{Id}}{\alpha \vdash ; \alpha \oplus \alpha}\vdash\oplus r}{\alpha \vdash \alpha \oplus \alpha ;}\mathsf{Foc}}{\vdash \alpha \oplus \alpha ; {\downarrow}\alpha^{\perp}}\vdash{\downarrow}}{\vdash \alpha \oplus \alpha ; {\downarrow}\alpha^{\perp} \oplus \alpha}\vdash\oplus r}{\vdash {\downarrow}\alpha^{\perp} \oplus \alpha, \alpha \oplus \alpha ;}\mathsf{Foc}$$

This has the essential feature that the last positive rule is applied to the left hand formula in the final sequent, which will not participate in the Cut with (the analogue of) Π_2.

Now we must find an analogue of Π_2. This is harder work, and we must use the 'double negation' style representation of $\alpha \& \alpha$ as ${\downarrow}({\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp})^{\perp}$. We then obtain Π_2':

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\alpha \vdash ; \alpha}\mathsf{Id}}{\alpha \vdash \alpha ;}\mathsf{Foc}}{\alpha, {\downarrow}\alpha^{\perp} \vdash ;}{\downarrow}\vdash \quad \dfrac{\dfrac{\dfrac{}{\alpha \vdash ; \alpha}\mathsf{Id}}{\alpha \vdash \alpha ;}\mathsf{Foc}}{\alpha, {\downarrow}\alpha^{\perp} \vdash ;}{\downarrow}\vdash}{\alpha, {\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp} \vdash ;}\oplus\vdash}{\alpha \vdash ; {\downarrow}({\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp})^{\perp}}\vdash{\downarrow}}{\alpha \vdash {\downarrow}({\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp})^{\perp} ;}\mathsf{Foc} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\alpha \vdash ; \alpha}\mathsf{Id}}{\alpha \vdash \alpha ;}\mathsf{Foc}}{\alpha, {\downarrow}\alpha^{\perp} \vdash ;}{\downarrow}\vdash \quad \dfrac{\dfrac{\dfrac{}{\alpha \vdash ; \alpha}\mathsf{Id}}{\alpha \vdash \alpha ;}\mathsf{Foc}}{\alpha, {\downarrow}\alpha^{\perp} \vdash ;}{\downarrow}\vdash}{\alpha, {\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp} \vdash ;}\oplus\vdash}{\alpha \vdash ; {\downarrow}({\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp})^{\perp}}\vdash{\downarrow}}{\alpha \vdash {\downarrow}({\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp})^{\perp} ;}\mathsf{Foc}}{\alpha \oplus \alpha \vdash {\downarrow}({\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp})^{\perp} ;}\oplus\vdash$$

Note that Π_2 introduces first the left-hand &, then the right-hand &. Π_2 is clearly equivalent by permutation of rules to the MALL proof which introduces the two &'s in the opposite order. There is an analogous version of Π_2' which introduces the two $\oplus$'s in the opposite order. However, whereas Π_2 is essentially completely symmetric between the two &'s, in the case of Π_2' there is an evident asymmetry between the left-hand &, which is represented implicitly by the $\oplus$ on the left-hand side of the sequent, and the right-hand &, which is represented in 'double-negation' form as ${\downarrow}({\downarrow}\alpha^{\perp} \oplus {\downarrow}\alpha^{\perp})^{\perp}$ on the right-hand side of the sequent. This asymmetry makes a telling difference as regards the possibilities for forming cuts, as we shall now see.

We now need an analogue of Π_3. This should be a proof of a sequent

$$\downarrow(\downarrow\alpha^\perp \oplus \downarrow\alpha^\perp)^\perp \vdash A \oplus B;$$

in which the last positive rule is the one introducing the $\oplus$ in the right-hand side of the sequent. This leads to the following proof Π'_3, in which we take $A = \downarrow(\downarrow\alpha^\perp)^\perp$—this 'double negation' is essential.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\alpha \vdash ; \alpha}\ \mathsf{Id}}{\alpha \vdash \alpha ;}\ \mathsf{Foc}}{\vdash \alpha ; \downarrow\alpha^\perp}\ \vdash\downarrow}{\vdash \alpha ; \downarrow\alpha^\perp \oplus \downarrow\alpha^\perp}\ \vdash\oplus l}{\vdash \alpha, \downarrow\alpha^\perp \oplus \downarrow\alpha^\perp ;}\ \mathsf{Foc}}{\downarrow\alpha^\perp \vdash \downarrow\alpha^\perp \oplus \downarrow\alpha^\perp ;}\ \downarrow\vdash}{\vdash \downarrow\alpha^\perp \oplus \downarrow\alpha^\perp ; \downarrow(\downarrow\alpha^\perp)^\perp}\ \vdash\downarrow}{\vdash \downarrow\alpha^\perp \oplus \downarrow\alpha^\perp ; \downarrow(\downarrow\alpha^\perp)^\perp \oplus B}\ \vdash\oplus l}{\vdash \downarrow\alpha^\perp \oplus \downarrow\alpha^\perp , \downarrow(\downarrow\alpha^\perp)^\perp \oplus B ;}\ \mathsf{Foc}}{\downarrow(\downarrow\alpha^\perp \oplus \downarrow\alpha^\perp)^\perp \vdash \downarrow(\downarrow\alpha^\perp)^\perp \oplus B ;}\ \downarrow\vdash$$

Note the necessity to double-negate α in order to allow the $\oplus$ which will end up on the left-hand side of the sequent to be introduced. Note also that, crucially, because of the stoup constraint *we cannot perform the $\oplus$-introduction on the right hand side of the sequent as the last rule in the proof.* This means that what we actually obtain combining Π'_1, Π'_2, Π'_3 is the analogous situation to case 7 in Table 1. If we worked in symmetric fashion from right to left, we would obtain a situation analogous to case 1. However, *no analogue of the problematic case 6 can be found in* $\mathrm{MALL}_{\mathsf{foc}}$.

Although we have only considered a particular example, it should illustrate the way in which $\mathrm{MALL}_{\mathsf{foc}}$ tames MALL by putting sufficient constraints on the control flow so that it admits a deterministic, sequential scheduling.

It is precisely for this reason that $\mathrm{MALL}_{\mathsf{foc}}$ is used in the work on Ludics. While Ludics is a much richer and more subtle setting than Blass games, it does have certain salient features in common. These include the primacy of the additives, and also the fact that interaction in Ludics is strictly polarized; in fact, the two players strictly alternate, although either player can start. In Ludics, control is retained over the scheduling by explicit use of the shift operators (which in process terms can be seen as polarized versions of prefixing). So Ludics as a semantic universe corresponds to (and was in some sense inspired by) the focussing version of MALL, and it is for $\mathrm{MALL}_{\mathsf{foc}}$ that a full completeness result is proved for Ludics in [Gir01].

7 The concurrent way out: Concurrent Games

When confronted with a mismatch between syntax and semantics, one may:

- modify the syntax, or

- change the model.

Thus in the case of PCF, confronted with the non full abstraction of the standard denotational model, one may extend PCF with parallel constructs, or look for a more refined model which captures sequentiality. In the case of Linear Logic, the movement is rather in the opposite direction: starting from a sequential model, we can either constrain the syntax to enforce sequentiality, or enrich the model to allow concurrency to be represented directly. We now turn to this latter possibility.

Neither game theory nor concurrency theory offers ready made solutions to this problem. While some suggestive remarks connecting games of imperfect information with concurrent 'teams' of players have been made [HS95], no precise, let alone tractable formalization has been proposed so far as we are aware.

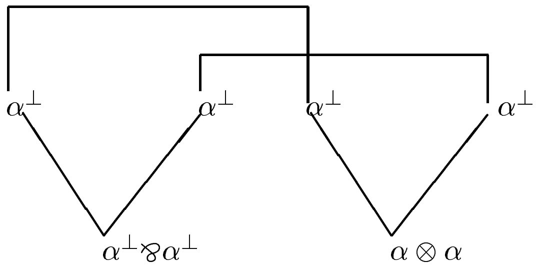
As we have seen, Blass games give primacy to the additive connectives of Linear Logic. They give a direct and natural representation of the idea that (occurrences of) $\&$ and $\oplus$ represent points in the computation at which *choices* have to be made. In the case of $\&$, the choice is made by O, in the case of $\oplus$, by P. However, the account given of the multiplicative connectives in these games is less satisfactory, as we have seen: the insistence on retaining a global polarization, so that in each position it is exactly one player's turn to move, leads to over-specification of the sequential order of events, with bad formal consequences.

By contrast, we find an 'authentic' semantic account of the multiplicatives (and, to some extent, of the exponentials) in the Geometry of Interaction [Gir89, Gir90, Gir95, DR93, DR95]. Here the 'local, asynchronous' character of the multiplicatives, suggested by the intrinsic, geometrical representation of multiplicative proofs as proof nets, is turned into a concrete form of symbolic dynamics. The basic idea of Geometry of Interaction is that multiplicative proofs are represented by permutations acting on structures of some kind.

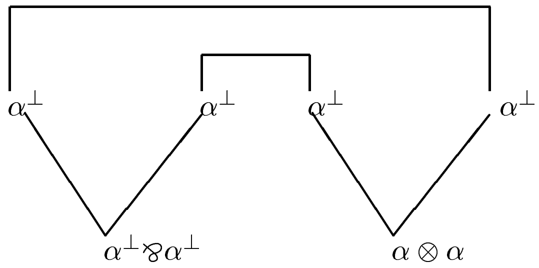
7.1 Background

The original version of the Geometry of Interaction was developed by Girard for the multiplicative fragment [Gir88]. This is still the best setting in which to explain the basic ideas on which the interpretation is based.

Consider then the multiplicative fragment, with the restriction that the Axiom is only used for propositional atoms, $\vdash \alpha^{\perp}, \alpha$. Now, if we look at the cut-free proofs, for example of $\alpha \otimes \alpha \multimap \alpha \otimes \alpha$, *i.e.* of the sequent $\vdash \alpha^{\perp} \invamp \alpha^{\perp}, \alpha \otimes \alpha$, there are in fact just two, corresponding to the identity and twist maps.

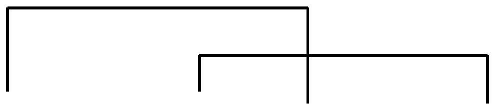


Id

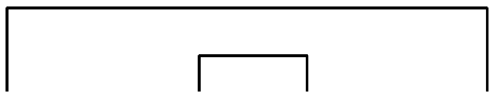


Twist

As we see from these examples, cut-free proof nets in this fragment have the structure of a set of trees, one for each formula in the conclusion, with the leaves connected up in pairs by the axiom links. Moreover, the structure of the trees is determined uniquely by the formulae in the sequent (this is where the restriction on axioms is applied). Hence, a complete invariant to distinguish the different cut-free proofs of a given sequent is given by the information as to how the leaves are joined up.

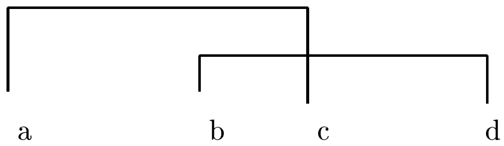


Id



Twist

We can model this information by a permutation on the set of leaves obtained as the product of the transpositions corresponding to the axiom links. Thus,

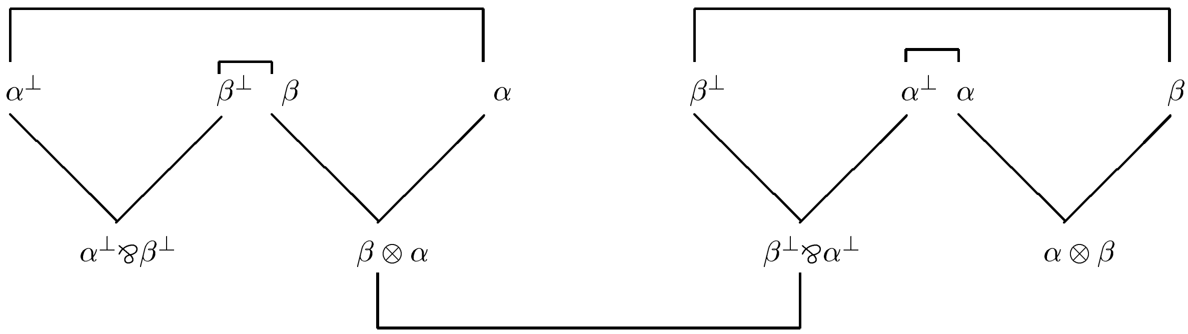


corresponds to the permutation

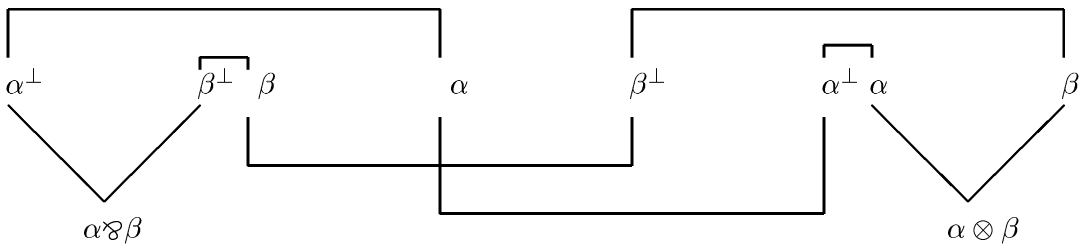
$$\begin{pmatrix} a & b & c & d \\ c & d & a & b \end{pmatrix}$$

Note that these transpositions are *disjoint*. So, a cut-free proof is represented by an *involution*, *i.e.* a self-inverse permutation. This representation of cut-free proofs can be thought of as modelling the 'information flow' between the leaves in a dynamic fashion; think of tokens travelling in both directions across the axiom links, as opposed to modelling the linkage statically by a graph. Note that we are using functions to represent this information flow, but without input-output bias, since the flow is bidirectional and symmetric.

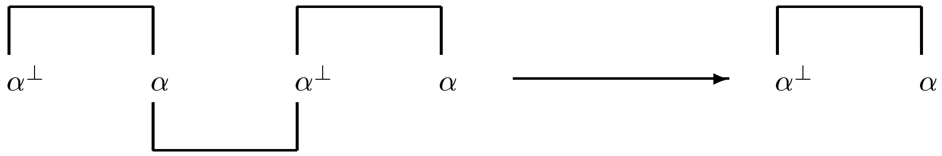
Returning to our example, consider performing cut-elimination on *twist* $\circ$ *twist*: The proof net for `twist` $\circ$ `twist` before cut elimination is:



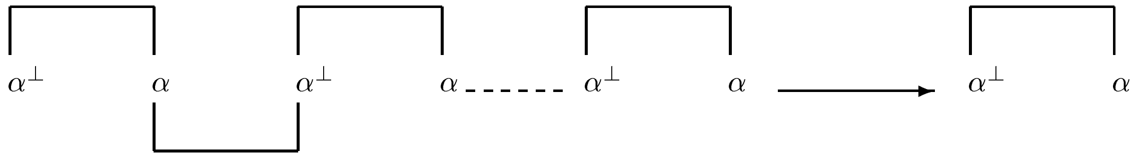
The proof net for `twist` $\circ$ `twist` after one step of cut elimination is:



Generally, in this fragment, we can apply this 'decomposition rule' repeatedly for tensors cut against par until all cuts are between axiom links. We can say that the whole purpose of these transformations is to match up the corresponding axiom links correctly; the 'real' information flow is then accomplished by the axiom reductions:



or more generally,



The idea, as with cut-free proofs, is to model these transformations dynamically, by the flow of information tokens, rather than by graph rewriting.

An interpretation of the multiplicative fragment can be given using just permutations on finite sets, as described in [Gir88]. The task of *characterizing* the space of proofs is then to pick out exactly those permutations which can arise as the denotation of (cut-free) proof nets, and moreover to do so in a compositional, syntax-free fashion. This is the goal of *full completeness theorems*, to be discussed in the next section. The main point to note for now is that this can indeed be done using suitable uniformity and preservation properties in a number of semantic settings; this is the content of the various full completeness theorems for Multiplicative Linear Logic which have appeared over the past few years, starting with [AJ92b] (see e.g. [HO92, BS96, Loa94a, Hag00]).

The question is, how can this view of the multiplicatives, which has no scope for expressing causal dependencies (this is exactly the sense in which it is asynchronous) be reconciled with the additives, which as we have seen are essentially concerned with choice and causality?[1]

We can in fact characterize the behavioural features of the various levels of connectives in Linear Logic using the concepts of Net Theory [Pet77]:

- the multiplicatives express *concurrency* (i.e. pure causal independence)
- the additives express *conflict* and *causality* (i.e. choice and sequencing).

(This is of course just the 'true concurrency' perspective on our previous alignment of the additives with the dynamic operations of process algebra, and the multiplicatives with the static operations.) What we are looking for is a setting in which all these features can be represented.

Of course, Net Theory itself offers such a setting. The problem with using Net Theory as a formal basis for our semantic representation is that, as we have seen, we need *two levels of description*: the formulas *and* the proofs (or: the the games *and* the strategies). While it was a relatively simple matter to describe strategies as subtrees of synchronization trees, it does not seem to be so easy to find a good, tractable description of strategies over games represented as Petri Nets. For this reason, we shall pursue a different approach.

In keeping with the discussion so far, we are looking for a semantic representation such that both:

- the account of the additives given in Blass games, and
- the account of the multiplicatives given in Geometry of Interaction

can be faithfully represented. By being a little more abstract, an elegant account meeting all these desiderata can be found.

[1]The extension of Geometry of Interaction to the additives in [Gir95], while containing much of interest, is generally agreed not to offer a fully satisfactory treatment.

7.2 A domain-theoretic formalization of games and strategies

Let us begin by revisiting Blass games, with the idea that game trees can be viewed as partial orders, in which $x \leq y$ means that the position y can be reached from the position x by playing some additional moves. This is a natural "information ordering" as in Domain theory [AJ94]; it is the prefix ordering on the plays or paths through the game tree. If we add "limit points" corresponding to the infinite branches in the game tree, we obtain a complete partial order D.

Viewed in these terms, the construction of sums and products of games as in Blass games can be described as *lifted sums* as far as the underlying domains of positions are concerned:

$$D(\coprod_{i \in I} A_i) = D(\prod_{i \in I} A_i) = (\biguplus_{i \in I} D(A_i))_{\perp}.$$

Here $\biguplus_{i \in I} D_i$ is the disjoint union of the partial orders D_i; while $D_\perp$ is the lift of D, i.e. the result of adding a new bottom element to D. It is important to understand that this partial order representation is entirely equivalent to that given using deterministic synchronization trees in Section 2. The root of the game tree for G is represented by the bottom element of $D(G)$, corresponding to the position in which no moves have yet been played. The elements of $D(G)$ immediately above $\perp_D$ will be of the form $\mathsf{in}_i(\perp_i)$, where $\perp_i$ is the bottom element of G_i. Such an element corresponds to the position where the opening move i has been made, and we are now at the root of the sub-tree G_i. As we go up in the ordering, this will correspond to the further development of the play.

Importantly, the partial order prepresentation of game trees admits an elegant representation of *strategies*, as functions on these domains of positions: $f : D \to D$, where $f(x)$ is the position obtained from x by extending it with whatever moves the strategy makes in that position. It is then immediate that $f(x) \sqsupseteq x$. Moreover, those positions where f has no moves to make (e.g. because 'it is not its turn') are exactly the *fixpoints* of f. In the usual way, we require computationally reasonable strategies to be monotonic and continuous. Finally, as a useful normalizing condition, we require strategies to be *idempotent*: $f^2 = f$. To understand this, consider f applied to $f(x)$. The only moves made in $f(x)$ which were not already made in x are those made by f itself: $f(x)$ contains no more information supplied by the Opponent (i.e. the environment) than x did. Hence anything f decides to do at $f(x)$ it should have already been able to decide to do at x, and we require that $f(f(x)) = f(x)$. Of course, this allows several moves to be made in a block by a player. This possibility already exists in Blass games, e.g. Opponent must move twice initially in

$$(A \times B) \times (C \times D).$$

An important point is that strategies may not be well-defined at all positions. In general there are some positions that can never be reached by following that strategy. To mesh with the requirement that strategies are increasing functions, we adjoin a top element to the domain of positions D, writing this as $D^\top$. We represent f being undefined at x by $f(x) = \top$.

In summary, strategies (for either player) are represented as continuous closure operators on $D^\top$, which under modest assumptions on D (bounded completeness) is a complete lattice. We can completely specify a game as a structure (D, S^P, S^O), where D is the domain of positions, S^P is the set of legal strategies for P, and S^O is the set of legal counter-strategies, i.e. strategies for O.

We can define the strategies for either player inductively on (well-founded) Blass games, just as we did using the synchronization tree representation. For products, we define

$$S^P(\prod_{i \in I} A_i) = \{\langle \sigma_i | i \in I \rangle \mid \forall i \in I.\, \sigma_i \in S^P(A_i)\}$$

where

$$\begin{array}{rcl} \langle \sigma_i | i \in I \rangle(\bot) & = & \bot \\ \langle \sigma_i | i \in I \rangle(\mathsf{in}_i(x)) & = & \mathsf{in}_i(\sigma_i(x)). \end{array}$$

This corresponds exactly to the idea that Player must first wait for Opponent to choose an $i \in I$, and then plays according to some strategy for A_i.

$$S^P(\coprod_{i \in I} A_i) = \{\mathsf{in}_i(\sigma) \mid i \in I \wedge \sigma \in S^P(A_i)\}$$

where

$$\begin{array}{rcl} \mathsf{in}_i(\sigma)(\bot) & = & \mathsf{in}_i(\sigma(\bot)) \\ \mathsf{in}_i(\sigma)(\mathsf{in}_i(x)) & = & \mathsf{in}_i(\sigma(x)) \\ \mathsf{in}_i(\sigma)(\mathsf{in}_j(x)) & = & \top, \quad (i \neq j). \end{array}$$

Again, this corresponds to the idea that a strategy for Opponent will firstly choose some i, then play according to a strategy for Opponent in A_i. Note that the last case above covers 'unreachable states' for $\mathsf{in}_i(\sigma)$.

We can also define the duality directly in terms of its effect on the set of strategies for either player: the reversal of rôles means that strategies for P turn into strategies for O, and vice versa:

$$(D, S^P, S^O)^\perp = (D, S^O, S^P).$$

Clearly with this definition $G^{\perp\perp} = G$. Also, we define $S^O(G) = S^P(G^\perp)$.

For the case of well-founded Blass games, we recover exactly the same class of strategies as in our earlier definition in terms of synchronization trees. For such games, D is the domain of finite and infinite sequences under the prefix ordering corresponding to the paths through the game tree, and the conventions about who is to play are captured by the fact that, for all $x \in D$, either $\sigma(x) = x$ for all $\sigma \in S^P$ (it is Opponent's turn to move), or $\tau(x) = x$ for all $\tau \in S^O$ (it is Player's turn). However, this is a very special case of our general setting; and we will overcome the problems with Blass games precisely by allowing situations in which *both* players can move.

To do this, we shall interpret the game boards for the multiplicatives differently to Blass: by a true concurrency rather than an interleaving representation. In our setting, this is simply a matter of defining

$$D(A \otimes B) = D(A \invamp B) = D(A) \times D(B),$$

the cartesian product of domains. It is important to understand how this relates to our earlier discussion of true concurrency. A typical situation in which there are two causally indepdent events a and b might be represented by a game $D \times E$, where say a corresponds to the increase in information $\bot \sqsubseteq d$ in D, while b is represented by the increase in information $\bot \sqsubseteq e$ in E. Then the performance of these two actions *in either order*

$$(\bot, \bot) \sqsubseteq (d, \bot) \sqsubseteq (d, e)$$

corresponding to the trace ab, or

$$(\bot, \bot) \sqsubseteq (\bot, e) \sqsubseteq (d, e)$$

corresponding to ba, will indeed be identified, with the single outcome (d, e). Indeed, cartesian product of event structures is standardly used to model parallel composition in 'partial order' or 'true concurrency' semantics [WN95].

For the strategies for tensor, we define

$$S^P(A \otimes B) = \{\sigma \times \tau \mid \sigma \in S^P(A) \wedge \tau \in S^P(B)\}$$

where $\sigma \times \tau(x, y) = (\sigma(x), \tau(y))$. (This is really smash product with respect to $\top$; if either $\sigma(x) = \top$ or $\tau(y) = \top$, then the result is $\top$.) This exactly captures the idea of informational independence between Player's actions in A and in B (cf. [AJ92b]). How Player moves in A depends only on the information available in A, and similarly for B.

In order to define the counter-strategies for the Tensor (and hence the strategies for Par and Linear implication, and eventually the morphisms in the category of concurrent games), we introduce the most important feature of our formalization: the elegant treatment it affords of composition of strategies. Suppose firstly that $\sigma \in S^P$ and $\tau \in S^O$ in a game (D, S^P, S^O). How do we play σ off against τ? We define

$$\begin{array}{rcl} \langle \sigma | \tau \rangle & = & \mathbf{Y}(\sigma \circ \tau) = \bigvee_{k \in \omega} (\sigma \circ \tau)^k(\bot) \\ & = & \bigvee_{k \in \omega} (\tau \circ \sigma)^k(\bot) = \mathbf{Y}(\tau \circ \sigma). \end{array}$$

The fact that these two least fixpoints coincide follows easily from the fact that σ and τ are continuous closures; in fact, this is a special case of the construction of the *join* of two closure operators. Thus $\langle \sigma | \tau \rangle \in D$ is the position we reach as a result of playing σ against τ. The equality of the two formulas above also shows that this is independent of all questions about 'who starts'.

Now given closure operators σ on $(D \times E)^\top$ and τ on $(E \times F)^\top$, we want to 'compose' them to get a closure $\sigma; \tau$ on $(D \times F)^\top$. We define this as follows:

$$\sigma; \tau(x, z) = (\pi_1 \circ \sigma(x, y), \pi_2 \circ \tau(y, z))$$

where

$$y = \langle \pi_2 \circ \sigma(x, -) | \pi_1 \circ \tau(-, z) \rangle.$$

That is, given input in D and F, we play σ and τ off against each other in E relative to this input, and obtain their external response taking into account their interaction with each other.

In particular, if σ is a closure on $(D \times E)^\top$, it induces an 'action' taking closures on $D^\top$ to closures on $E^\top$, $\alpha \mapsto \alpha;\sigma$, and a 'coaction' taking closures on $E^\top$ back to closures on $D^\top$, $\beta \mapsto \sigma;\beta$. E.g. $\sigma;\beta(x) = \pi_1 \circ \sigma(x, y)$, where $y = \langle \pi_2 \circ \sigma(x, -) | \beta \rangle$.

Now we can define

$$\begin{array}{rl} S^O(A \otimes B) = \{\sigma : (D(A) \times D(B))^\top \mid & \forall \alpha \in S^P(A).\, \alpha;\sigma \in S^O(B) \\ \wedge & \forall \beta \in S^P(B).\, \sigma;\beta \in S^O(A)\}. \end{array}$$

Again, by De Morgan duality,

$$A \parr B = (A^\perp \otimes B^\perp)^\perp, \quad A \multimap B = A^\perp \parr B.$$

In particular,

$$\begin{array}{rl} S^P(A \multimap B) = \{\sigma : (D(A) \times D(B))^\top \mid & \forall \alpha \in S^P(A).\, \alpha;\sigma \in S^P(B) \\ \wedge & \forall \beta \in S^O(B).\, \sigma;\beta \in S^O(B)\}. \end{array}$$

This is a symmetric, 'classical' version of the familiar logical relations condition: σ's action carries P strategies on A to P strategies on B, and its coaction carries counter-strategies on B to counter-strategies on A.

7.3 Connection with Geometry of Interaction

As to the issue of faithfully reflecting geometry of interaction ideas in modelling the multiplicatives, we recall that the basic idea in geometry of interaction is that multiplicative proofs are modelled by permutations. Note that in any symmetric monoidal category, the symmetric group $S(n)$ acts in a canonical way on the tensor power

$$\bigotimes^n A \;=\; \underbrace{A \otimes \ldots \otimes A}_{n}.$$

In the original form of Geometry of Interaction [Gir89] (dubbed 'particle style GoI' in [Abr96]) the monoidal structure used for the representation is coproduct (disjoint union). In our setting, we can use products to exactly the same effect. Thus in our example of the sequent

$$\vdash \alpha^\perp \parr \alpha^\perp, \alpha \otimes \alpha$$

the two proofs can be represented as maps on a product

$$D \times D \times D \times D$$

by

$$(a, b, c, d) \mapsto (c, d, a, b)$$

for the identity, and

$$(a, b, c, d) \mapsto (d, c, b, a)$$

for the twist.

This idea, although basically correct, requires a slight elaboration in order to mesh with our approach of using closure operators to model strategies. We turn permutations σ as above into closure operators γ by defining

$$\gamma(x) = x \vee \sigma(x).$$

Thus for example the permutation corresponding to the identity as above will give rise to the closure operator

$$(a, b, c, d) \mapsto (a \vee c, b \vee d, c \vee a, d \vee b).$$

The axiom $\vdash \alpha^{\perp}, \alpha$ is interpreted by the twist map on $D \times D$

$$(x, y) \mapsto (y, x)$$

and then by the closure

$$(x \vee y, x \vee y)$$

which forces both components to the diagonal.

In this fashion, we can claim to have achieved our objective of faithfully embedding both the Blass games treatment of the additives, and the GoI treatment of the multiplicatives, within our model.

7.4 Connection with 'New Foundations'

It should be noted that mathematically, the concurrent games model is merely a rephrasing of the 'New Foundations for the Geometry of Interaction' introduced by the present author and Radha Jagadeesan in [AJ92a]. As demonstrated in [Abr96, AHS00], this and the original form of geometry of interaction in [Gir89] are indeed both instances of a single general scheme, as far as the interpretation of the multiplicatives and exponentials are concerned. However, as already emphasized in [AJ92a], the New Foundations setting has the representational capacity to capture the causality inherent in the additives, which the 'particle style' GoI of [Gir89] does not. The advantage of the presentation of concurrent games in terms of closure operators is that the connection with traditional forms of game semantics, and Blass games in particular, becomes much clearer. However, the New Foundations presentation remains important, particularly since we can use it to pick out those strategies satisfying significant domain-theoretic properties such as stability and sequentiality, which turn out to be important for full completeness, i.e. for characterizing which strategies arise from proofs. These properties apply to the underlying NFGoI functions f, rather than to the closure operators σ they induce via

$$\sigma(x) = x \vee f(x).$$

7.5 Composition revisited

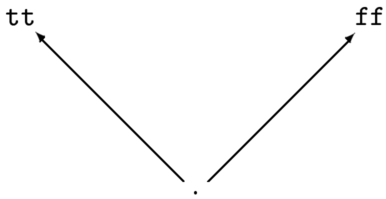
We now illustrate how the concurrent games model overcomes the problem with composition which arises with Blass games, with reference to the example discussed in section 4.

For illustration, we fix the interpretation of α to be the game $\mathbb{B} = I \oplus I$, where

$$I = (\mathbf{1}, \{\mathsf{id}_1\}, \{\mathsf{id}_1\})$$

the one-point domain in which no moves are possible (i.e. the sole point is the root of the game tree), and in which the identity function is a valid strategy for both P and O. Note that $\mathbb{B}$ can be seen as a type of 'booleans':

$$D(\mathbb{B}) = (\mathbf{1} \uplus \mathbf{1})_\perp =$$



the flat domain of booleans, with

$$S^P = \{\mathsf{in}_{\mathtt{tt}}, \mathsf{in}_{\mathtt{ff}}\} \qquad S^O(\mathbb{B}) = \{\mathsf{id}_{\mathbb{B}}\}$$

where for $b \in \{\mathtt{tt}, \mathtt{ff}\}$

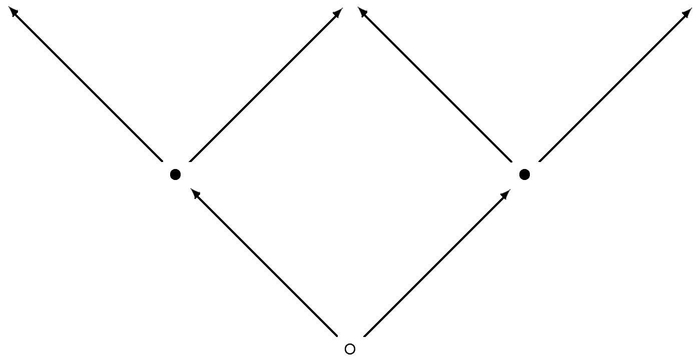
$$\mathsf{in}_b(\perp) = \mathsf{in}_b(b) = b, \quad \mathsf{in}_b(b') = \top \quad (b \neq b').$$

We can now compute the underlying domains of the sequents:

$$\begin{array}{rcl} D(\alpha^\perp \oplus \alpha^\perp, \alpha \oplus \alpha) & = & D((\alpha^\perp \oplus \alpha^\perp) \parr (\alpha \oplus \alpha)) \\ & = & (\mathbb{B} \uplus \mathbb{B})_\perp \times (\mathbb{B} \uplus \mathbb{B})_\perp \\ & = & D(\alpha^\perp \& \alpha^\perp, \alpha \& \alpha). \end{array}$$

Although these domains are identical, if we take account of the polarities on the additive connectives, which are reflected in the definitions of the sets of strategies for the corresponding games, we can distinguish between them. For example,

$$D(\alpha^\perp \oplus \alpha^\perp) =$$



Here the nodes marked $\bullet$ are where O is to move, and hence must be fixpoints for all P-strategies; and conversely for the $\circ$-node.

We now consider the interpretations of the three proofs discussed in section 4. Firstly, $c = [\![\Pi_1]\!]$ has the definition

$$c(\bot,\bot) = (\mathsf{in}_{\mathtt{tt}}(\bot), \mathsf{in}_{\mathtt{tt}}(\bot))$$
$$c(\mathsf{in}_{\mathtt{tt}}(x), \mathsf{in}_{\mathtt{tt}}(y)) = (\mathsf{in}_{\mathtt{tt}}(x \vee y), \mathsf{in}_{\mathtt{tt}}(x \vee y))$$
$$c(\mathsf{in}_{\mathtt{ff}}(x), y) = c(x, \mathsf{in}_{\mathtt{ff}}(y)) = \top.$$

Note that Π_3, the proof which introduces the two occurrences of $\oplus$ in the opposite order, has *the same denotation* $[\![\Pi_3]\!] = e = c$; in both cases, the two introductions (or: choices at the $\oplus$-nodes) are performed concurrently.

Finally $d = [\![\Pi_2]\!]$ has the definition

$$d(x,\bot) = (x,\bot) \qquad d(\bot,y) = (\bot,y)$$

$$d(\mathsf{in}_i(x), \mathsf{in}_j(y)) = (\mathsf{in}_i(x \vee y), \mathsf{in}_j(x \vee y)).$$

Again, this has the same denotation as the proof which introduces the two &'s in the opposite order. (However, it should be noted that in general, strategies must be taken modulo partial equivalence relations as in [AM99] in order to obtain the unicity properties for product and coproduct (i.e. the commutative conversions in sequent calculus terms, or η-conversions in λ-calculus terminology). Interestingly, partial equivalence relations are also needed in Ludics [Gir01]).

Now one can check, unfolding the definitions, that

$$c;(d;e) = c = e = (c;d);e.$$

For example, the fixpoint in $c;d$ converges after two iterations, with the resulting behaviour

$$c;d(\bot,\bot) = (\mathsf{in}_{\mathtt{tt}}(\bot), \bot)$$

$$c;d(\mathsf{in}_{\mathtt{tt}}(x), \mathsf{in}_j(y)) = (\mathsf{in}_{\mathtt{tt}}(x \vee y), \mathsf{in}_j(x \vee y)).$$

This strategy, of type

$$\vdash \alpha^{\bot} \oplus \alpha^{\bot}, \alpha \& \alpha$$

exemplifies how *global polarization no longer holds* in the concurrent games model. *Both* P *and* O can move initially, P to make the choice at the $\oplus$, O to make the choice at the &. After these choices have been made (determining an axiom link in the corresponding proof net) a copy-cat is then played between the chosen occurrences of the atom. See [AJ92a] for many examples of such fixpoint computations, and a detailed analysis of the correspondence with cut elimination.

Thus when we compose in either order, $(c;d);e$ or $c;(d;e)$, the result is the same; P is no longer blocked by polarization constraints from making the choices at *both* occurrences of $\oplus$—truly concurrently.

7.6 A more concrete presentation

The foregoing mathematical development may have seemed abstract, and hard to relate to one's intuitions. Let us start again, and give an essentially equivalent, but more concrete presentation.

Our notion of concurrent games can be thought of as played between two 'teams' of players, distributed around the 'game board', in an aynchronous fashion.

The basic action in a game is *making a move*, which is chosen from among several possible alternatives. The information content of the move is the *choice* or decision as to which move to play.

We fix a set C of 'cells' as labels for the various choices or decisions that have to be made during the course of a play of the game, and a set V of 'values', enumerating the possible choices. So the 'event' (c, v) means that alternative v was chosen at cell c. A state of the game board—a position of the game—is then represented by a partial function from cells to values

$$\{(c_1, v_1), \ldots, (c_k, v_k)\}$$

(so there is *no* information about the order in which decisions were taken).

We can take $C = V = \mathbb{N}$ for a 'universal' choice of game board, into which any other (subject to countability) can be embedded. The positions are then partial functions on natural numbers, which under inclusion form one of the basic examples of a *domain* [AJ94]. Moreover, as play progresses, and more moves are made, the position *increases* in the natural information ordering:

$$\{(c_1, v_1), (c_2, v_2)\} \stackrel{(c_3,v_3)}{\longrightarrow} \{(c_1, v_1), (c_2, v_2), (c_3, v_3)\}.$$

Call this universal game board $\mathbb{G}$.

A *strategy* is a function

$$f : \mathbb{G} \longrightarrow \mathbb{G}.$$

For the reasons already explained in the previous section, this function should be monotonic and continuous, increasing ($x \subseteq f(x)$), and idempotent ($f(f(x)) = f(x)$).

What about unreachable positions? We need an 'error element' $\top$ for situations when the strategy is applied to a position which could never have been reached if we had been following that strategy. To preserve the increasingness property, this error element must be top in the information ordering. So a strategy is a continuous closure operator on the complete lattice $\mathbb{G}^\top$ obtained by adjoining a top element to $\mathbb{G}$. The set of all such closure operators

$$\mathcal{S} = \mathsf{Cl}(\mathbb{G}^\top)$$

is our universal space of strategies.

Various sub-classes of $\mathcal{S}$ are important. In particular, there are constraints on strategies which correspond to important computational properties. One of these is sequentiality [Cur86].

A closure operator $\sigma \in \mathcal{S}$ has an *output function* $f : \mathbb{G} \rightarrow \mathbb{G}$ if

$$\forall x \in \mathtt{dom}(\sigma).\, \sigma(x) = f(x) \vee x$$

where

$$\mathtt{dom}(\sigma) = \{x \in \mathbb{G} \mid \sigma(x) \neq \top\}.$$

We recall the domain-theoretic conditions of *stability* and *sequentiality* [AC98]. These make sense because our underlying domain $\mathbb{G}$ is simple and concrete—it is in fact a 'concrete domain' in the technical sense of Kahn and Plotkin [KP78]. Indeed, because it is simply a (countable) product of flat domains (in fact, $\mathbb{G} = \mathbb{N}_{\perp}{}^{\omega}$), we can use Vuillemin's original definition of sequentiality [Vui73]: a function $f : \mathbb{G} \to \mathbb{G}$ is sequential if:

$$\forall c \in C. \forall x \in \mathbb{G}. f(x)(c)\uparrow \;\;\Rightarrow\;\; \forall y \supseteq x. f(y)(c)\uparrow$$

$$\vee \;\; (\exists c'. x(c')\uparrow \wedge \forall y \supseteq x. f(y)(c)\downarrow \;\Rightarrow\; y(c')\downarrow).$$

There says that there is some particular cell in the input which *must* get filled before we can fill the given output cell. Classically, this condition excludes the parallel-or function, Berry's 'Gustave function', etc. [Vui73, KP78, Cur86].

We take the 'constraint' of having a sequential output function as picking out those strategies which are to be regarded as computing in a deterministic fashion, and hence sequentially implementable. In support of this contention, we have the fact that, when we make our universe of sequential strategies into a model for a sequential functional language such as PCF (the most elegant way of doing this uses realizability ideas [Abr00]), then it is fully abstract, and even universal if we restrict to effective strategies. Moreover, by Loader's results [Loa96], this is about as good a characterization of sequentiality as we are likely to get.

Let us give a first indication of how we can use our simple universal (type-free) space of strategies to recover the same structure we displayed in a typed version in the previous section.

We split the 'address space' by fixing

$$C = C_1 \uplus C_2$$

(C_1, C_2 infinite). Note that

$$\mathbb{G} = V_{\perp}^{C} = V_{\perp}^{C_1 + C_2} \cong V_{\perp}^{C_1} \times V_{\perp}^{C_2} \cong \mathbb{G} \times \mathbb{G}.$$

So we can regard any $\sigma \in \mathcal{S}$ as

$$\sigma : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G} \times \mathbb{G}.$$

The most basic example of a strategy is the *asynchronous copy-cat*:

$$\mathtt{id}(x, y) = (x \vee y, x \vee y).$$

The corresponding output function is the twist map:

$$\mathtt{tw}(x, y) = (y, x)$$

since

$$(x \vee y, x \vee y) = (x, y) \vee (y, x).$$

7.7 The process model

This may still be too abstract for some. We can also describe the model in operational terms with a little process calculus.

We define a syntax of terms by the BNF

$$P \ ::= \ c?x \rightarrow P \mid c!e \mid P \| Q \mid 0 \mid \top \mid (\nu c)P \mid X \mid \texttt{rec}\, X.\, P \mid \ldots$$

We define a structural congruence on these terms in the style of the (asynchronous) π-calculus [Mil99]. The main novel feature is that $\top$ acts as a zero, and we have

$$c!v_1 \parallel c!v_2 \equiv \top \ \ (v_1 \neq v_2).$$

We can then define a reduction semantics, again in the style of the π-calculus, but with one key difference: we have the reduction

$$c!v \parallel (c?x \rightarrow P) \quad \longrightarrow \quad c!v \parallel P[v/x]$$

where in CCS or the π-calculus we would have the rule

$$c!v \parallel (c?x \rightarrow P) \quad \longrightarrow \quad P[v/x].$$

This property of *persistence* of outputs (which corresponds to the information increasingness in the domain-theoretic presentation) is more akin to concurrent constraint programming [Sar93].

We can connect this operational view very directly to our view of strategies as closure operators by defining a denotational semantics for terms of our process calculus as closure operators, and proving a correspondence between this denotational semantics and the operational semantics. We can then see the process calculus terms as providing a syntax for our semantic objects, and moreover making the computational behaviour of strategies explicit via the associated operational semantics.

We give a couple of clauses for this denotational semantics.

$$[\![c?x \rightarrow P]\!](a) = \begin{cases} a, & a(c)\uparrow \\ [\![P[v/x]]\!](a), & a(c) = v \end{cases}$$

$$[\![c!v]\!](a) = \begin{cases} a \cup \{(c, v)\} & \text{if consistent} \\ \top & \text{otherwise} \end{cases}$$

Parallel composition is interpreted as the join of closure operators; note the connection to the composition operators defined in Section 5.

$$[\![P \| Q]\!] = \bigvee_{k \in \omega} ([\![P]\!] \circ [\![Q]\!])^k = \bigvee_{k \in \omega} ([\![Q]\!] \circ [\![P]\!])^k.$$

Applied to any a, this gives the least common fixpoint of P and Q above a. Note that it doesn't matter who starts!

Using this calculus, we can construct the copy-cat strategy described previously as an abstract closure operator in more concrete terms as a process. (We use the splitting of the address space introduced in the previous section. For each cell c, we think of it occurring once in the 'left part' of the address space, C_1, as $\mathtt{l}(c)$, and once in the right part C_2, as $\mathtt{r}(c)$). Now we can define the 'copy-cat process' as:

$$\mathtt{id} \;=\; \|_{c \in C} \, (\mathtt{l}(c)?x \to \mathtt{r}(c)!x \parallel \mathtt{r}(c)?x \to \mathtt{l}(c)!x).$$

We can think of this process as a big team of players, one for each matched pair of cells. This process copies the contents of either element of the pair into the other as soon as it is filled. Of course, if the pair of cells have incompatible contents, then this will fail. However, if we play this copy-cat strategy against any other strategy following the type constraints of Linear Logic, then this kind of failure can never happen.

8 Full Completeness

The usual notion of completeness for a logic is with respect to provability; Full Completeness is with respect to *proofs*.

Let $\mathcal{M}$ be a model of the formulas *and* proofs of a logic $\mathcal{L}$. Typically this means that $\mathcal{M}$ is a category with structure of an appropriate kind, such that the formulas of $\mathcal{L}$ denote objects of $\mathcal{M}$, proofs Π in $\mathcal{L}$ of entailments $A \vdash B$ denote morphisms

$$[\![\Pi]\!] : [\![A]\!] \longrightarrow [\![B]\!],$$

and the convertibility of proofs in $\mathcal{L}$ with respect to cut-elimination is soundly modelled by the equations between morphisms holding in $\mathcal{M}$. We say that $\mathcal{M}$ is *fully complete* for $\mathcal{L}$ if for all formulas A, B of $\mathcal{L}$, every morphism $f : [\![A]\!] \to [\![B]\!]$ in $\mathcal{M}$ is the denotation of some proof Π of $A \vdash B$ in $\mathcal{L}$: $f = [\![\Pi]\!]$. Thus the full completeness of $\mathcal{M}$ means that it characterizes 'what it is to be a proof in $\mathcal{L}$' in a very strong sense. If $\mathcal{M}$ is defined in a syntax-independent way, this is a true semantic characterization of the 'space' of proofs spanned by $\mathcal{L}$.

The notion of Full Completeness was introduced in [AJ92b][2], and a Full Completeness theorem was proved for a game semantics of Multiplicative Linear Logic (with the MIX rule). This was followed by a series of papers which established full completeness results for a variety of models with respect to various versions of Multiplicative Linear Logic, e.g. [HO92, BS96, Loa94a, Loa94b, Tan97, DHPP99, Hag00].

The proofs of full completeness which have appeared to date fall into two broad classes:

- Proofs using decomposition arguments

[2]There were a number of significant precursors, as noted in [AJ92b], including representation theorems in category theory [FS91], full abstraction results in programming language semantics [Mil75, Plo77], Plotkin's characterization of definability in the λ-calculus using logical relations [Plo80], studies of parametric polymorphism [BFSS90, HRR89], and the completeness conjecture in [Gir91]. However, the contribution of [AJ92b] was to clearly identify and formulate this issue as a precise and interesting research programme, and to prove the first in what has become quite a rich sequence of results.

- 'Geometric' proofs via proof nets.

The first class of proofs is based on the 'last rule = first move' idea. One decomposes a (sequential) strategy into an opening move, or an initial P-response to an opening O-move, and some sub-strategies. The initial protocol is matched with a sequent calculus rule, or a term formation scheme in a λ-calculus setting. One then recursively applies the decomposition to the sub-strategies, uncovering the sequent proof or term which denotes the strategy step by step.

This is typical of the proofs of definability and full completeness which have been given for a range of λ-calculus based programming languages [AMcC99, Abr00a], and for various fragments of Intuitionistic Logic and Intuitionistic Linear Logic [MO99, MO00], and also applies, *modo grosso*, to the full completeness proof for Ludics with respect to $\mathrm{MALL}_{\mathsf{foc}}$ [Gir01]. This form of proof is obviously well-adapted to sequential models.

By contrast, the second class of proof, beginning with the original full completeness result in [AJ92b], first establishes a connection beteen the semantic objects (in our case, strategies) and proof structures, or 'pre-proofs'; and then shows that the 'geometric' constraints picking out those structures which are proof nets, i.e which correspond to real proofs, must be satisfied by the proof structures arising from the semantic objects.

The first step typically uses uniformity arguments of a fairly general nature, while the second step makes more delicate use of specific features of the model.

The full completeness proof for concurrent games with respect to MALL [AM98, AM99] is of this general form. The connection made is with the proof nets for MALL introduced in [Gir95a], which eliminate additive boxes in favour of boolean weights, which distribute information about causal dependencies around the proof net. An important point is that what seemed to be mere technical machinery in [Gir95a] acquires a much clearer semantic status in the full completeness argument. In particular, the boolean weights are derived from inherent functional (or, in process terms: causal) dependencies in the model; and the 'monomial condition', which plays a key technical rôle in the proof of the Sequentialization Theorem in [Gir95a], turns out to correspond exactly to the semantic constraint of *stability* on the strategies in the concurrent games model.

We thus have the interesting situation of two rather different proofs: one for the sequential case (Ludics with respect to $\mathrm{MALL}_{\mathsf{foc}}$) and one for the concurrent case (concurrent games with respect to MALL). Nevertheless, the scope of the two results is entirely analogous: both concern the Π^1 fragment of MALL resp. $\mathrm{MALL}_{\mathsf{foc}}$, without the multiplicative units. It is a challenge for further work to relate them.

9 Concluding Remarks

We have sketched two ways out of the structural dilemma which arose with Blass games. There are good arguments for pursuing both:

- The focussing version of Linear Logic offers more control over proof search (the purpose for which it was originally introduced [AP91]), and is a good setting for studying sequential models of classical systems, analogous to the $\lambda\mu$-calculus for Classical Logic.

- The concurrent approach allows the direct study of the original system. It is close to Geometry of Interaction and Proof nets. It may offer a principled way of describing and controlling true concurrency, just as classical logic may for non-deterministic computation.

Ultimately, one would hope to relate these two directions in a coherent fashion.

We end with some more specific challenges and questions:

- Both for Ludics and concurrent games, the challenge of extending the full completeness results to the exponentials remains. This may require a better treatment of the multiplicative units than either currently offers.
- Once the exponentials have been analyzed, one may hope to turn the lens of fully complete models onto systems which can be interpreted into Linear Logic, notably Classical logic. One tantalizing question which remains unclear: is there true concurrency lurking somewhere in the computational interpretation of classical logic?
- It would also be interesting to look at bounded systems such as Light Linear Logic [Gir98], and the connections to complexity classes.

Appendix: a brief review of MALL

The formulas of the system are built from *literals*, i.e. propositional atoms α, β, ... and their negations $\alpha^\perp$, $\beta^\perp$, ..., by the grammar

$$A \ ::= \ \alpha \mid \alpha^\perp \mid A \otimes A \mid A \parr A \mid A \oplus A \mid A \& A.$$

Here $\otimes$ and $\parr$ are the *multiplicative* connectives, while $\oplus$, $\&$ are the *additive* connectives. Negation is definitionally extended to general formulas by the equations

$$(A \otimes B)^\perp = A^\perp \parr B^\perp \qquad (A \parr B)^\perp = A^\perp \otimes B^\perp$$

$$(A \oplus B)^\perp = A^\perp \& B^\perp \qquad (A \& B)^\perp = A^\perp \oplus B^\perp$$

$$A^{\perp\perp} = A.$$

A sequent in MALL is an expression $\vdash \Gamma$, where Γ is a finite multiset of formulas.

The rules of MALL are as follows.

Axiom/Cut

$$\frac{}{\vdash A, A^\perp}\ \mathsf{Id} \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\ \mathsf{Cut}$$

Multiplicatives

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\ \otimes \qquad \frac{\Gamma, A, B}{\vdash \Gamma, A \parr B}\ \parr$$

Additives

$$\frac{\vdash \Gamma, A}{\vdash \Gamma A \oplus B}\ \oplus\mathsf{L} \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma A \oplus B}\ \oplus\mathsf{R} \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \& B}\ \&$$

References

[AJ92a] S. Abramsky, R. Jagadeesan, New foundations for the geometry of interaction, *Information and Computation*, 111(1):53-119, 1994. Conference version appeared in LiCS '92.

[AJ92b] S. Abramsky and R. Jagadeesan. Games and Full Completeness for Multiplicative Linear Logic, *Journal of Symbolic Logic*, (1994), vol. 59, no. 2, 543–574. Conference version appeared in FSTTCS '92.

[AJ94] S. Abramsky and A. Jung. Domain Theory. *Handbook of Logic in Computer Science. Volume III: Semantic Structures*, pages 1–168. S. Abramsky, D. M. Gabbay and T. S. E. Maibaum, editors. Oxford University Press 1994.

[Abr96] S. Abramsky. Retracing Some Paths in Process Algebra. In *Proceedings of CONCUR 96*, Lecture Notes in Computer Science Vol. 1119, pp. 1-17. Springer-Verlag 1996.

[Abr97] S. Abramsky. Semantics of Interaction. In *Semantics and Logics of Computation*, A. Pitts and P. Dybjer, eds. Publications of the Newton Institute, pp. 1–31. Cambridge University Press, 1997.

[AHS00] S. Abramsky, E. Hagverdi and P. Scott. Geometry of Interaction and models of Linear Combinatory Logic. Submitted for publication. 2000.

[AMcC99] S. Abramsky and G. McCusker. Game Semantics. In *Computational Logic*. U. Berger and H. Schwichtenberg eds. Springer-Verlag 1999, 1–56.

[Abr00] S. Abramsky. Process Realizability. In *Foundations of Secure Computation*, F. L. Bauer and R. Steinbrüggen, eds. IOS Press 2000, 167–180.

[Abr00a] S. Abramsky. Axioms for Definability and Full Completeness. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, G. Plotkin, C. Stirling and M. Tofte, eds. MIT Press, 2000, pp. 55–75.

[AM98] S. Abramsky and P.-A. Melliès. Concurrent games and full completeness. Draft paper, 37 pages, 1998.

[AM99] S. Abramsky and P.-A. Melliès. Concurrent games and full completeness. *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, 431–44, 1999.

[AC98] R. Amadio, P.-L. Curien, *Domains and Lambda-calculi.* Cambridge University Press, 1998.

[AP91] J.-M. Andreoli and R. Pareschi. Linear objects: logical processes with built-in inheritance. *New Generation Computing* 9(3–4):445–473, 1991.

[BFSS90] S. Bainbridge, P. J. Freyd, A. Scedrov and P. Scott. Functorial Polymorphism. *Theoretical Computer Science* 70:35–64, 1990.

[BB96] F. Barbanera and S. Berardi. A symmetric lambda calculus for classical program extraction. *Information and Computation* 125:103–117, 1996.

[Bla72] A. Blass. Degrees of indeterminacy of games. *Fundamenta Mathematica* LXXVII:151–162, 1972.

[Bla92] A. Blass, A game semantics for linear logic, *Annals of Pure and Applied Logic*, 56:183-220, North-Holland, 1992.

[Bla] A. Blass. Personal communication.

[BS96] R. Blute and P. J. Scott. Linear Läuchli Semantics. *Annals of Pure and Applied Logic* 77:101–142, 1996.

[BW90] J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge University Press 1990.

[Cur86] P.-L. Curien. *Categorical combinators, sequential algorithms and functional programming*. Pitman 1986. Revised edition, Birkhauser 1993.

[DR93] V. Danos and L. Regnier (1993) Local and Asynchronous beta-reduction (An analysis of Girard's Execution Formula), in: *Proc. of Eighth International Symposium on Logic in Computer Science*, IEEE Press, pp. 296-306, 1993.

[DR95] V. Danos and L. Regnier, Proof-nets and the Hilbert space, in [GLR95], pp. 307–328.

[DJS96] V. Danos and J.-B. Joinet and H. Schellinx. A new deconstructive logic: Linear Logic. *Journal of Symbolic Logic* 62(3), 1996.

[DHPP99] H. Devarajan, D. Hughes, G. Plotkin and V. Pratt, Full completeness of the multiplicative linear logic of Chu spaces. In *Proceedings of the Fourteenth International Symposium on Logic in Computer Science*, 234–243, IEEE Press, 1999.

[FS91] P. J. Freyd and A. Scedrov. *Categories, allegories*. North-Holland Mathematical Library, Vol. 39, Elsevier 1990.

[Gir87] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir88] J.-Y. Girard. Multiplicatives. *Rendiconti del Seminario Matematico (Torino)*, Universita Pol. Torino, 1988.

[Gir89] J.-Y. Girard, Geometry of Interaction I: Interpretation of System F. In *Logic Colloquium '88*, ed. R. Ferro et al. North-Holland 1989, pp. 221-260.

[Gir90] J.-Y. Girard, Geometry of Interaction II: Deadlock- free Algorithms. *COLOG-88*, P. Martin-Lof, G. Mints, eds. Springer LNCS Vol. 417, 1990, pp. 76-93.

[Gir91] J.-Y. Girard. A new constructive logic: classical logic. *Mathematical Structures in Computer Science* 1:255–296, 1991.

[Gir95] J.-Y. Girard, Geometry of Interaction III: Accommodating the Additives, in [GLR95], pp. 1-42.

[GLR95] J.-Y. Girard, Y. Lafont, L. Regnier, eds. *Advances in Linear Logic*, London Math. Soc. Lecture Note Series Vol. 222, Camb. Univ. Press, 1995.

[Gir95a] J.-Y. Girard. Proof-nets: the parallel syntax for proof theory. In *Logic and Algebra*, Marcel Dekker, 1995.

[Gir98] J.-Y. Girard. Light Linear Logic. *Information and Computation* 143:175–204, 1998.

[Gir01] J.-Y. Girard. Locum Solus. To appear in *Mathematical Structures in Computer Science*, 2001.

[GLT89] J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and Types*. Cambridge University Press 1989.

[Hag00] E. Haghverdi. *A Categorical Approach to Linear Logic, Geometry of Proofs and Full Completeness*. PhD Thesis, University of Ottawa, 2000.

[HS95] J. Hintikka and G. Sandu. What is the logic of parallel processing? *International Journal of the Foundations of Computer Science* 6:27–49, 1995.

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

[HRR89] M. Hyland, E. P. Robinson and G. Rosolini. Algebraic types in PER models. In *Proceedings of the Fifth Conference on Mathematical Foundations of Programming Semantics*, Springer LNCS Vol. 442, pp. 333-350, 1989.

[HO92] M. Hyland and C.-H. L. Ong. *Fair Games and Full Completeness for Multiplicative Linear Logic without the MIX rule*. Unpublished Manuscript, 1992.

[KP78] G. Kahn and G. Plotkin. Concrete Domains. *Theoretical Computer Science*, 121:187–277, 1993. Appeared as TR IRIA-Laboria 336 in 1978.

[Lai99] J. D. Laird. *A semantic analysis of control*. Ph.D. thesis, University of Edinburgh, 1999.

[Lai01] J. D. Laird. A deconstruction of non-deterministic classical cut-elimination. To appear in the Proceedings of TLCA 2001, 2001.

[Loa94a] R. Loader. Linear Logic, Totality and Full Completeness. In *Proceedings of the Ninth International Symposium on Logic in Computer Science*, IEEE Press, 1994.

[Loa94b] R. Loader, *Models of lambda calculi and linear logic: structural, equational and proof-theoretic characterisations*. PhD Thesis, Oxford University, 1994.

[Loa96] R. Loader. Finitary PCF is not decidable. To appear in *Theoretical Computer Science*.

[Mil75] R. Milner. Proceeses: a mathematical model of computing agents. *Logic Colloquium, Bristol 1973*, North-Holland 1975, pp. 157–174.

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

[Mil99] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.

[MO99] A. S. Murawski and C.-H. L. Ong. Exhausting strategies, joker games, and full completeness for IMLL with Unit. *Electronic Notes in Theoretical Computer Science* 29, 1999. Proceedings of the 8th Conference on Category Theory and Computer Science.

[MO00] A. S. Murawski and C.-H. L. Ong. Discreet games, Light Affine Logic, and PTIME computation. In *Proceedings of CSL2000: Annual Conference of the European Association of Computer Science Logic*, Springer LNCS Vol. 1862, 427–441, 2000.

[Ong96] C.-H. L. Ong. A semantic view of classical proofs: type-theoretical, categorical and denotational characterizations. In *Proceedings of the Eleventh International Symposium on Logic in Computer Science*, 230–241, IEEE Press, 1996.

[Par92] M. Parigot. $\lambda\mu$ calculus: an algorithmic interpretation of classical natural deduction. In *Proc. International Conference on Logic Programming and Automated Reasoning* 190–201. Springer, 1992.

[NPW81] M. Nielsen and G. Plotkin and G. Winskel. Petri nets, event structures and domains, Part 1. *Theoretical Computer Science* 13:85–108, 1981.

[Pet77] C. A. Petri. *Non-sequential processes*. Technical report ISF-77-05, GMD-ISF, 1977.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science* 5:223–255, 1977.

[Plo80] G. D. Plotkin. Lambda definability in the full type hierarchy. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 363–373, Academic Press, 1980.

[Sar93] V. Saraswat. *Concurrent Constraint Programming*. MIT Press, 1993.

[Tan97] A. Tan. *Full completeness for models of linear logic*. Ph.D. thesis, University of Cambridge, 1996.

[Urb00] C. Urban. *Classical Logic and Computation*. Ph.D. thesis, Cambridge University, 2000.

[Vui73] J. Vuillemin. *Proof Techniques for Recursive programs*. Ph.D. thesis, Stanford University, 1973.

[WN95] G. Winskel and M. Nielsen. Models for Concurrency. In *Handbook of Logic in Computer Science Vol. 4*, S. Abramsky, D. M. Gabbay and T. S. E. Maibaum, eds. Oxford University Press 1995, 1–148.