Focusing in Asynchronous Games

Samuel Mimram*

CEA LIST / École Polytechnique
CEA LIST, Laboratory for the Modelling and Analysis of Interacting Systems,
Point Courrier 94, 91191 Gif-sur-Yvette, France
samuel.mimram@cea.fr

Abstract. Game semantics provides an interactive point of view on proofs, which enables one to describe precisely their dynamical behavior during cut elimination, by considering formulas as games on which proofs induce strategies. We are specifically interested here in relating two such semantics of linear logic, of very different flavor, which both take into account concurrent features of the proofs: asynchronous games and concurrent games. Interestingly, we show that associating a concurrent strategy to an asynchronous strategy can be seen as a semantical counterpart of the focusing property of linear logic.

A cut-free proof in sequent calculus, when read from bottom up, progressively introduces the connectives of the formula that it proves, in the order specified by the syntactic tree constituting the formula, following the conventions induced by the logical rules. In this sense, a formula can be considered as a playground that the proof will explore. The formula describes the rules that this exploration should obey, it can thus be abstractly considered as a game, whose moves are its connectives, and a proof as a strategy to play on this game. If we follow the principle given by the Curry-Howard correspondence, and see a proof as some sort of program, this way of considering proof theory is particularly interesting because the strategies induced by proofs describe very precisely the interactive behavior of the corresponding program in front of its environment.

This point of view is at the heart of game semantics and has proved to be very successful in order to provide denotational semantics which is able to describe precisely the dynamics of proofs and programs. In this interactive perspective, two players are involved: the Proponent, which represents the proof, and the Opponent, which represents its environment. A formula induces a game which is to be played by the two players, consisting of a set of moves together with the rules of the game, which are formalized by the polarity of the moves (the player that should play a move) and the order in which the moves should be played. The interaction between the two players is formalized by a play, which is a sequence of moves corresponding to the part of the formula being explored during the cut-elimination of the proof with another proof. A proof is thus described in this

* This work has been supported by the CHOCO (ANR-07-BLAN-0324) ANR project.

F. Ferreira et al. (Eds.): CiE 2010, LNCS 6158, pp. 331–341, 2010.
© Springer-Verlag Berlin Heidelberg 2010
setting by a strategy which corresponds to the set of interactions that the proof
is willing to have with its environment.

This approach has been fruitful for modeling a wide variety of logics and pro-
gramming languages. By refining Joyal’s category of Conway games and Blass’
games, Abramsky and Jagadeesan were able to give the first fully complete
game model of the multiplicative fragment of linear logic extended with the MIX rule [3], which was later refined into a fully abstract model of PCF [4], a prototypical programming language. Here, “fully complete” and “fully abstract”
alternatively mean that the model is very precise, in the sense that every strategy
is definable (i.e. is the interpretation of a proof or a program). At exactly the
same time, Hyland and Ong gave another fully abstract model of PCF based
on a variant of game semantics called pointer games [10]. In this model, defin-
able strategies are characterized by two conditions imposed on strategies (well-
bracketing and innocence). This setting was shown to be extremely expressive:
relaxing in various ways these conditions gave rise to fully abstract models of a
wide variety of programming languages with diverse features such as references,
control, etc. [8].

Game semantics is thus helpful to understand how logic and typing regulate
computational processes. It also provides ways to analyze them (for example by
doing model checking [2]) or to properly extend them with new features [8], and
this methodology should be helpful to understand better concurrent programs.
Namely, concurrency theory being relatively recent, there is no consensus about
what a good process calculus should be (there are dozens of variants of the
π-calculus and only one λ-calculus) and what a good typing system for process
calculus should be: we believe that the study of denotational semantics of those
languages is necessary in order to reveal their fundamental structures, with a
view to possibly extending the Curry-Howard correspondence to programming
languages with concurrent features. A few game models of concurrent program-
ing languages have been constructed and studied. In particular, Ghica and
Murawski have built a fully abstract model of Idealized Algol (an imperative
programming language with references) extended with parallel composition and
mutexes [9] and Laird a game semantics of a typed asynchronous π-calculus [11].

In this paper, we take a more logical point of view and are specifically interes-
ted in concurrent denotational models of linear logic. The idea that multiplicative
connectives express concurrent behaviors is present since the beginnings of linear
logic: it is namely very natural to see a proof of $A \otimes B$ or $A \otimes B$ as a proof of $A$
in “parallel” with a proof of $B$, the corresponding introduction rules being

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \otimes B}$$

with the additional restriction that the two proofs should be “independent” in
the case of the tensor, since the corresponding derivations in premise of the
rule are two disjoint subproofs. Linear logic is inherently even more parallel: it
has the focusing property [6] which implies that every proof can be reorganized
into one in which all the connectives of the same polarity at the root of a for-

ula are introduced at once (this is sometimes also formulated using synthetic