The value of Kohei's games

$\overline{A}$	OPO...O				PO..
B		PO...P		OP...O	
$\overline{B}$		OP...O		PO...P	
C			PO...O		

Kohei Honda's legacy in semantics of programming

a tribute by Pasquale Malacaria and Nikos Tzevelekos

Theory Group, EECS, Queen Mary University of London

A chronology in games

The Full abstraction problem for PCF

4+1 = 3+2 = 6-1 = ...5 …

a unique mathematical object under its many representations…

what about (sequential) programs?

P1: x=1; x=x+4*x

P2: x=2; x=x*2+1

What is the underlying "unique" mathematical object ?

A chronology in games

- Denotational semantics provides an answer to this question.
- Programs = continuous functions on a topological space (Scott Domains)
- but there is more than just sequential programs in these domains... e.g. parallel or, a non sequential computation
- Full abstract model ~ a model with only sequential computations
- various elegant attempts to refine Scott mathematical universe failed to provide this full abstract model e.g. stability eliminated parallel or but not the or tester...

A chronology in games

- so the mathematical universe of pure sequential computations eluded researchers for many years…
- In 1993 full abstraction was achieved using Game Semantics:
  - two person games with questions/answers moves,
  - questions ‘(‘ and answers ‘)’ are well bracketed in all plays.
  - Games can be quotiented to give a topological space a la Scott
- The 1993 models (HO,AJM) solved the question for call-by-name computations. Full abstraction for call-by-value was unsolved until 1997 with the work of Kohei and Nobuko.

Kohei's chronology in games

- Kohei Honda, Nobuko Yoshida: Game Theoretic Analysis of Call-by-Value Computation. ICALP 1997: 225-236.
- Marcelo P. Fiore, Kohei Honda: Recursive Types in Games: Axiomatics and Process Representation. LICS 1998: 345-356.
- Samson Abramsky, Kohei Honda, Guy McCusker: A Fully Abstract Game Semantics for General References. LICS 1998: 334-344.
- Kohei Honda, Nobuko Yoshida: Game-Theoretic Analysis of Call-by-Value Computation. Theoretical Computer Science 221(1-2): 393-456 (1999).
- Martin Berger, Kohei Honda, Nobuko Yoshida: Sequentiality and the pi-Calculus. TLCA 2001: 29-45.
- Kohei Honda: Processes and Games. WRLA 2002: 40-69.
- …

Call-by-value games

Game Theoretic Analysis
of Call-by-Value Computati

KOHEI HONDA NOBUKO YOSHIDA

ABSTRACT. We present a general semantic universe of call-by-
based on elements of game semantics, and validate its appropriat
universe by the full abstraction result for call-by-value PCF, a
gramming language with call-by-value evaluation. The key idea
distinction between call-by-name and call-by-value as that of t
formation flow, which determines the basic form of games. In thi
computation and call-by-value computation arise as two indepe
sequential functional computation with distinct algebraic struct
the type structures of the universe following the standard catego
veloped in the context of domain theory. Mutual relationship bet
category of games and the corresponding call-by-name universe i

1. INTRODUCTION

The *call-by-value* is a mode of calling procedures widely used in i
programming languages, e.g. [1, 30], in which one evaluates ar
them to a concerned procedure. The semantics of higher-orde
call-by-value evaluation has been widely studied by many resea
domain theory, cf. [35, 23, 32, 12, 40, 11], through which it ha
semantic framework for the call-by-value computation has a basi
for call-by-name computation (see [15, 42] for introduction to t
between the semantics of call-by-value and that of call-by-na
roughly be captured as the difference in the classes of involv
name, we take any continuous functions between pointed cpos
one takes *strict* continuous functions. The latter is also equ
partial continuous functions between (possibly bottomless) cpo
to a basic algebraic difference of the induced categorical univer
The present paper offers a semantic analysis of call-by-val
different angle, based on elements of game semantics. In game
is modelled as specific classes of interacting processes (called *str*
with a suitable notion of composition, form a categorical univers
structures. One may compare this approach to Böhm trees or to
22], in both of which computation is modelled not by set-theore
kind but by objects with internal structures which reflect com
the concerned class of computation. Game semantics has its
and has been used for the semantic analysis of programming l

GAME THEORETIC ANALYSIS OF CALL-BY-VALUE COMPUTATION

KOHEI HONDA† NOBUKO YOSHIDA‡

ABSTRACT. We present a general semantic universe of call-by-value computation based on elements of game semantics, and validate its appropriateness as a semantic universe by the full abstraction result for call-by-value PCF, a generic typed programming language with call-by-value evaluation. The key idea is to consider the distinction between call-by-name and call-by-value as that of the structure of information flow, which determines the basic form of games. In this way the call-by-name computation and call-by-value computation arise as two independent instances of sequential functional computation with distinct algebraic structures. We elucidate the type structures of the universe following the standard categorical framework developed in the context of domain theory. Mutual relationship between the presented category of games and the corresponding call-by-name universe is also clarified.

1. INTRODUCTION

The *call-by-value* is a mode of calling procedures widely used in imperative and functional programming languages, e.g. [1, 40, 47], in which one evaluates arguments before applying them to a concerned procedure. The semantics of higher-order computation based on call-by-value evaluation has been widely studied by many researchers in the context of domain theory, cf. [46, 47, 31, 42, 19, 53, 16, 17], through which it has become clear that the semantic framework needed to capture the call-by-value computation has a basic difference from the one for call-by-name computation (see [23, 55] for basic introduction to the topic). The difference between the semantics of call-by-value and that of call-by-name in this context may roughly be captured as the difference in the classes of involved functions: in call-by-name, we take any continuous functions between pointed cpos, while, in call-by-value, one takes *strict* continuous functions. The latter is also equivalently presentable as partial continuous functions between (possibly bottomless) cpos. This distinction leads to a basic algebraic difference of the induced categorical universe compared to the call-by-name universe, as has been studied in [16, 19].

The present paper offers a semantic analysis of call-by-value computation from a different angle, based on elements of game semantics. In game semantics, computation is modelled as specific classes of interacting processes (called *strategies*), which, together with a suitable notion of composition, form a categorical universe with appropriate type structures. One may compare this approach to Böhm trees [8] or to sequential algorithms [9] (cf. [29]), in both of which computation is modelled not by set-theoretic functions of a certain kind but by objects with internal structures which reflect computational behaviour of the concerned class of computation. Game semantics has

Two fundamental ideas

- Information-flow in call by value is data driven.
- Strategies as processes.

CBV is data driven

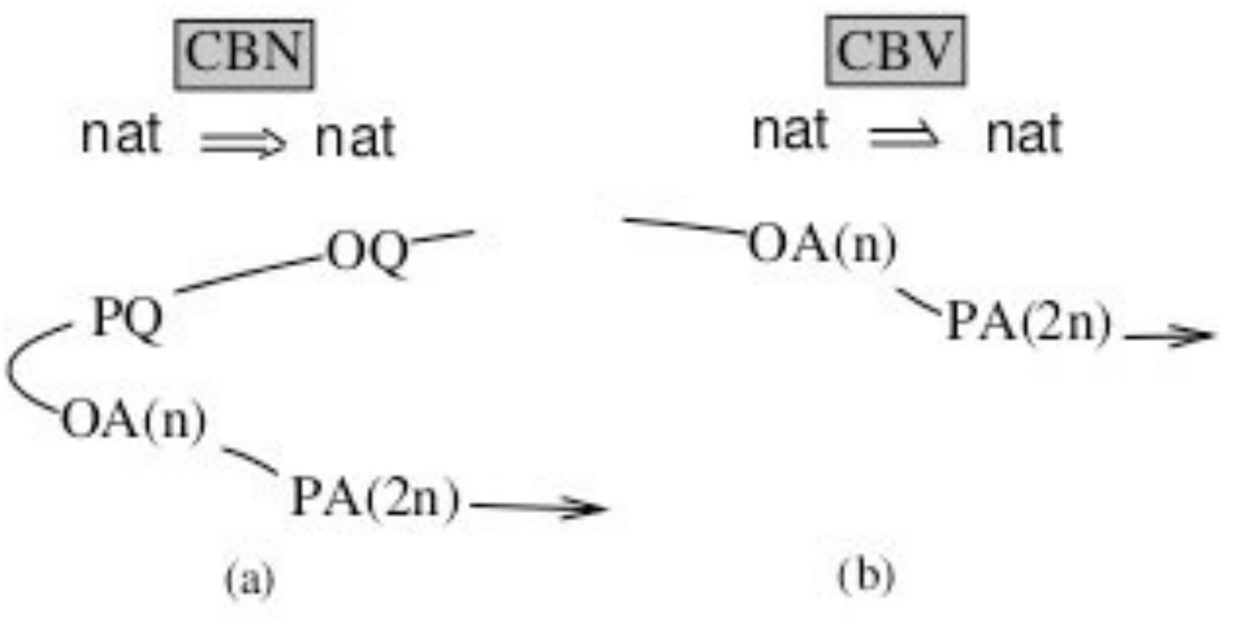

Figure 1

- A natural intuition reflecting
- CBN (fun x: M)N -> M[x:=N] whereas
- CBV (fun x.M)N -> M[x:=v] where v is the value of N
- technically challenging to make it work, e.g. bracketing?

CBV is data driven

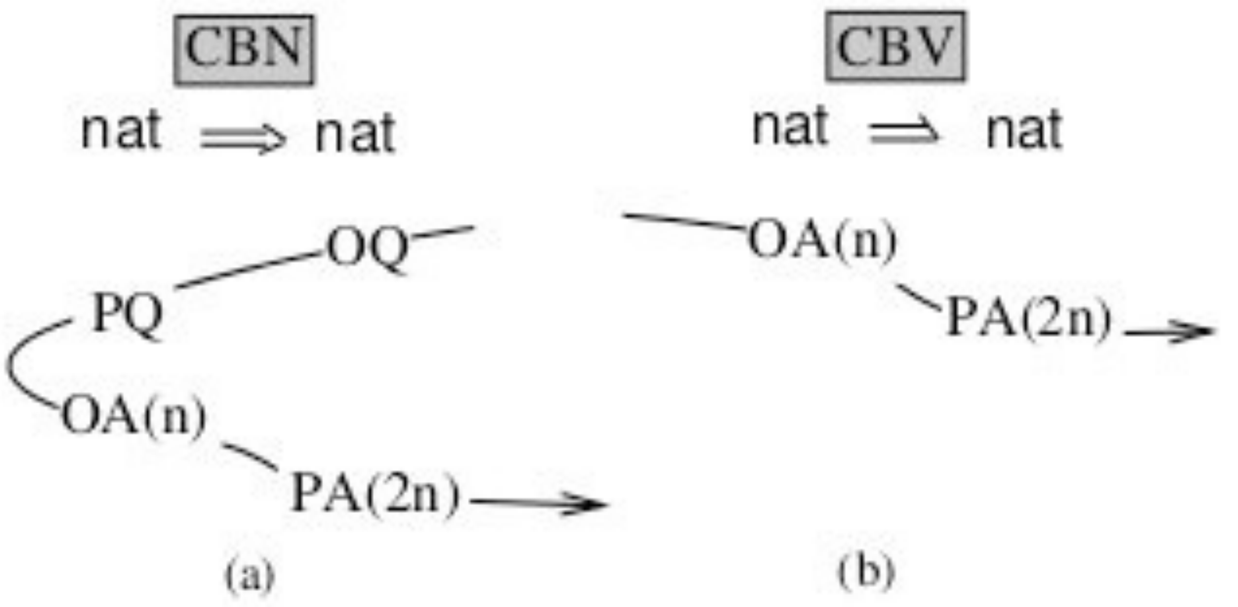

Figure 1

- Set the standard for CBV.
- Gave a unifying CBV/CBN framework.

Strategies are processes

APPENDIX B. PROCESS REPRESENTATION OF STRATEGIES

B.6. **Composition.** We write $\mathsf{bn}(l)$ for the set of names in l which is not its subject.

(i) (hiding) Given a process P and a set of names α, we define $(\nu\alpha)P$ by the following recursion:

$$(\nu\alpha)0 \stackrel{\text{def}}{=} 0$$

$$(\nu\alpha_1)(\nu\alpha_2)P \stackrel{\text{def}}{=} (\nu\alpha_1 \cup \alpha_2)P$$

$$(\nu\alpha)l.P \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } \mathsf{fn}(l) \subset \alpha \\ l.(\nu\alpha \backslash \mathsf{bn}(l))P & \text{if else} \end{cases}.$$

$$(\nu\alpha)\sum l.P \stackrel{\text{def}}{=} \sum(\nu\alpha)l.P$$

(ii) (parallel composition) Given two processes $P \stackrel{\text{def}}{=} \sum_i l_i.P_i$ and $Q \stackrel{\text{def}}{=} \sum_j l'_j.Q_j$, we define:

$$P \mid Q = \sum_{l_i = \overline{l'_j}} (\nu\, \mathsf{bn}(l_i))(P_i \mid Q_j) + \sum_i l_i.(P_i \mid Q) + \sum_j l'_j.(P \mid Q_j)$$

(iii) Finally given $\sigma_1 : A \to B$ and $\sigma_2 : B \to C$, we consider the sorting for two concerned pre-arenas such that their name assignments on sorts on B coincide, while those on A and C are disjoint. Let β be the set of names used for the initial sorts of B. Let P_1 and P_2 be the resulting process representations. Then we define:

$$P_1; P_2 \stackrel{\text{def}}{=} (\nu\beta)(P_1 \quad P_2)$$

- Previously, hinted at: *~CSP parallel comp.+hiding.*
- Ahead of its time, a lot followed...

Strategies **really are** processes

Recursive Types in Games:
Axiomatics and Process Representation
(Extended Abstract)

Marcelo Fiore* Kohei Honda
<marcelo@cogs.susx.ac.uk> <kohei@dcs.ed.ac.uk>
COGS, Univer

Abstract

This paper presents two basic resul
mantics of FPC, a metalanguage with
ponentials and recursive types. First w
account of the category of games **G** *in*
fering a fundamental structural analys
well as a transparent way to prove com
As a consequence we obtain an intensi
result through a standard definability
extend the category **G** *by introducing*
G, *with optimised strategies; we sh*
tional semantics in **G**, *gives a compi*
into core Pict codes (the asynchronous
without summation). The process re
a pioneering idea of Hyland and O
we advance their representation by in
ally well-founded optimisation techniq
the setting to encompass the rich typ
The resulting code gives basic insigh
between the abstract, categorical, type
implementations.

1 Introduction

Games in semantics. Recently, th
and strategies have been used for con
ical models of programming language
18, 2, 22, 15, 3]. The basic common i
works is to construct a universe of s
which a program phrase is modelled a
ternal structure reflecting its computati

Sequentiality and the π-Calculus

Martin Berger[1], Kohei Honda[1], and Nobuko Yoshida[2]

[1] Queen Mary, University of London, U.K
[2] University of Leiceste

Abstract. We present a type discipline fo
cisely captures the notion of sequential funct
cific class of name passing interactive beha
allows direct interpretation of both call-by-
quential functions. The precision of the repr
by way of a fully abstract encoding of PC
typed π-calculus can be used as a descriptiv
of programming languages without losing th
ties. Close correspondence with games sema
reasoning techniques are together used to es

1 Introduction

This paper studies a type discipline for the π-c
the notion of sequential functional computatio
tation is demonstrated by way of a fully abstr
studies have shown that while operational encod
guage constructs into the π-calculus are possib
[28,32]: we necessarily lose information by such
a source term M will generally result in a pro

Processes and Games

Kohei Honda[1,2]

Department of Computer Science,
Queen Mary, University of London,
London, E1 4NS, UK.

Abstract

A general theory of computing is important, if we wish to have a common mathematical footing based on which diverse scientific and engineering efforts in computing are uniformly understood and integrated. A quest for such a general theory may take different paths. As a case for one of the possible paths towards a general theory, this paper establishes a precise connection between a game-based model of sequential functions by Hyland and Ong on the one hand, and a typed version of the π-calculus on the other. This connection has been instrumental in our recent

Strategies **really are** processes

> **Abstract.** We present a type discipline for the π-calculus which precisely captures the notion of sequential functional computation as a specific class of name passing interactive behaviour. The typed calculus allows direct interpretation of both call-by-name and call-by-value sequential functions. The precision of the representation is demonstrated by way of a fully abstract encoding of PCF. The result shows how a typed π-calculus can be used as a descriptive tool for a significant class of programming languages without losing the latter's semantic properties. Close correspondence with games semantics and process-theoretic reasoning techniques are together used to establish full abstraction.

- Game semantics = game-typed pi-calculus
- $M \cong N \Leftrightarrow [\![M]\!] \cong [\![N]\!]$

Legacy

- Call-by-value games = Honda-Yoshida games
- Games are Processes hence, Programs are Processes...! (~ process logic -> program logic)

and finally...