The value of Kohei's games

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

\[ 4+1 = 3+2 = 6-1 = \ldots 5 \ldots \]

a unique mathematical object under its many representations…

what about (sequential) programs?

P1: \( x=1; \ x=x+4\times x \)

P2: \( x=2; \ x=x\times 2+1 \)

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


- ...
Call-by-value games

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 evaluation as that of the type structures of the universe following the standard categorical framework developed in the context of domain theory. Mutual relationship between categories 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 programming languages, e.g. [1, 30], in which one evaluates arguments to a concerned procedure. The semantics of higher-order call-by-value evaluation has been widely studied by many recent domain theory, cf. [35, 23, 32, 12, 40, 11], through which it has been shown that the semantic framework for the call-by-value computation has a basis for introducing semi-infinite computations (see [15, 42]) for introduction to the call-by-value computation. In the framework of 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.

The present paper offers a semantic analysis of call-by-value computation based on elements of game semantics. In game semantics is modelled as specific classes of interacting processes (called strategies), with a suitable notion of composition, form a categorical universe of games. One may compare this approach to Böhm trees or to the theory of games, in both of which computation is modelled by set-theoretic functions of a kind but by objects with internal structures which reflect computational behaviour of the concerned class of computation. Game semantics has and has been used for the semantic analysis of programming languages, and has been shown to model the structure of call-by-value.

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

- 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

Set the standard for CBV.

Gave a unifying CBV/CBN framework.
Strategies are processes

Previously, hinted at: \textit{\sim CSP parallel comp.+hiding.}

Ahead of its time, a lot followed...
Strategies really are processes
Strategies really are processes

Abstract. We present a type discipline for the \( \pi \)-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 \( \pi \)-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

- \( M \cong N \Leftrightarrow [\parallel M \parallel] \cong [\parallel N \parallel] \)
Legacy

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

and finally...