Game Semantics and Uniqueness of Type Inhabitance in the Simply-Typed $\lambda$-Calculus

Pierre Bourreau and Sylvain Salvati
LaBRI - INRIA Sud-Ouest
351, Cours de la Libération
33405 Talence Cedex, France
{bourreau,salvati}@labri.fr

Abstract. The problem of characterizing sequents for which there is a unique proof in intuitionistic logic was first raised by Mints [Min77], initially studied in [BS82] and later in [Aot99]. We address this problem through game semantics and give a new and concise proof of [Aot99]. We also fully characterize a family of $\lambda$-terms for Aoto’s theorem. The use of games also leads to a new characterization of principal typings for simply-typed $\lambda$-terms. These results show that game models can help proving strong structural properties in the simply-typed $\lambda$-calculus.

Keywords: games semantics, simply-typed $\lambda$-calculus, principal typing, coherence theorem, uniqueness of type inhabitance.

1 Introduction

Coherence theorems in category theory are used to ensure the equality of the composition of certain morphisms. In particular such conditions have been studied for cartesian closed categories - models of the simply-typed $\lambda$-calculus - in [BS82], a result which was later extended in [Aot99]. These results imply that a unique $\lambda$-term inhabits a given typing that verifies certain syntactic constraints. In [BS82], the exhibited typings are balanced while in [Aot99] they are negatively non-duplicated. A question that arises from these results, is that of the consequences of these constraints on types on their inhabitants. It can easily be observed that balanced typings are exactly inhabited by affine $\lambda$-terms. It was showed in [Kan07] that the family of almost linear $\lambda$-terms was included in that of the terms inhabiting negatively non-duplicated typings. One of the aims of this paper is to completely characterize the family of $\lambda$-terms that can be typed with negatively non-duplicated typings.

More precisely, we show that using game semantics leads to a concise proof of Aoto’s theorem, and in general, gives an accurate method to address structural properties of simply-typed terms. The dialogic-game representation of proofs originates in [Lor59,Lor68,Bla92] and while game semantics has been widely used to study programming language semantics [AM99,Ho00] and $\lambda$-calculus semantics [Hug00,GFH99,KNO02], to our knowledge it has never been used to address issues on proofs/terms structures, while it offers two main advantages:
first, it brings closer representations of typings and \(\lambda\)-terms which helps associating families of \(\lambda\)-terms with families of typings; second, it provides the analysis of proofs with a fine grained and natural access to the interplay of atomic types occurring in sequents. Our study leads indeed to a rather simple proof of Aoto’s theorem, and to a syntactic characterization of the inhabitants of negatively non-duplicating types as first-order copying \(\lambda\)-terms, which extends both notions of linear and almost linear terms. These \(\lambda\)-terms could be depicted in Kanazawa’s vocabulary as almost affine\(^1\). From a more general perspective, game semantics offers a simple way of investigating the relationship between typings and their inhabitants. As an example, we give a new characterization of the principal typings of \(\beta\)-normal terms.

In this paper, we will first recall basic notions on the simply-typed \(\lambda\)-calculus and introduce typing games and strategies; in the third section the correspondence between strategies and simply-typed terms will be presented; finally, we give a new characterization of principal typings, a concise proof of [Aot99] and a full characterization of the terms for [Aot99].

2 Preliminaries

2.1 Simply-Typed \(\lambda\)-Calculus

Let \(\mathcal{A}\) be a countable set of atomic types. The set \(\mathcal{T}(\mathcal{A})\) of simple types built upon \(\mathcal{A}\) is the smallest set built as the closure of \(\mathcal{A}\) under the right-associative connector \(\to\). We call a type substitution \(\sigma\) an endomorphism of \(\mathcal{T}(\mathcal{A})\) i.e. a function that verifies \(\sigma(\alpha \to \beta) = \sigma(\alpha) \to \sigma(\beta)\), for \(\alpha, \beta\) in \(\mathcal{T}(\mathcal{A})\). Note that a type substitution is completely defined by the values it takes on \(\mathcal{A}\). A type relabelling \(\sigma\) is a type substitution such that \(\sigma(\mathcal{A})\) is included in \(\mathcal{A}\). Finally, a type renaming denotes a bijective type substitution.

Let us consider a countable set of variables \(\mathcal{V}\). The set \(\Lambda\) of \(\lambda\)-terms built on \(\mathcal{V}\) is inductively defined with the following syntactic rules:

\[
\Lambda ::= \mathcal{V} \mid \lambda \mathcal{V}.\Lambda \mid (\Lambda \Lambda)
\]

We write \(\lambda\)-terms with the usual conventions, omitting sequences of \(\lambda\)’s and unnecessary parentheses. For a term \(M\), the notions of free variables (noted \(FV(M)\)), bound variables (\(BV(M)\)) and variables (\(V(M) = FV(M) \cup BV(M)\)) are defined as usual. We also take for granted the notions of \(\alpha\)-conversion, \(\beta\)-reduction and \(\eta\)-conversion. A precise definition of all these notions can be found in [Bar84]. A context is a \(\lambda\)-term with a hole, built according to the following rules:

\[
A \underline{[]} ::= [] \mid \lambda \mathcal{V}.A \underline{[]} \mid A \underline{[]} A \mid AA \underline{[]}
\]

\(^1\) This result was independently proved by Kanazawa in a yet unpublished work. Nevertheless, we believe the use of game semantics allows to express more directly the relation between syntactic properties of types and their inhabitants, giving a simpler proof.