Typed Realizability for First-Order Classical Analysis

Valentin Blot

Dept. of Computer Science, University of Bath, United Kingdom
v.blot@bath.ac.uk

In realizability we associate to each formula a set of programs which behave like a proof of the formula and that we call its realizers. This allows to give computational content to the axioms of a theory, and to choose quite freely the programming language, independently from the logical system. It is even possible to consider untyped programming languages, as was the case in the first realizability model from Kleene [7], in which a realizer may be any recursive function. It is however still possible to consider a typed language, as did Kreisel in his modified realizability model [8]. Both models from Kleene and Kreisel gave computational interpretation to Heyting arithmetic, the intuitionistic variant of Peano arithmetic.

Gödel’s negative translation [4] allowed for the so-called indirect interpretations of classical logic through intuitionistic interpretations of negatively translated classical proofs. This allowed to interpret full Peano arithmetic using the realizability models of Kleene and Kreisel. Much later, Griffin’s discovery [6] that the \texttt{call/cc} control operator could be typed with the law of Peirce opened the possibility for a direct interpretation of classical logic, using programming languages with control features. Following this path, Parigot defined the $\lambda\mu$-calculus [12], a language for Gentzen’s classical sequent calculus for which Selinger axiomatized the universal categorical model [13]. On another side, Krivine considered untyped $\lambda$-calculus extended with the \texttt{call/cc} operator to give a realizability interpretation to classical second-order Zermelo-Fraenkel set theory [9], later extended to handle the axiom of dependent choice [10, 11].

In this talk I will present a direct realizability interpretation for first-order classical logic which, contrary to Krivine’s and similarly to Kreisel’s (but for classical logic), uses typed programs as realizers. The classical proofs are interpreted in a categorical model of the language $\mu$PCF, which is a combination of the functional Turing-complete language PCF with the control features of call-by-name $\lambda\mu$-calculus. A careful analysis of relativization motivates the distinction between negative and positive formulas, the former having both a truth value and a falsity value which are orthogonal to each other as in Krivine’s work, but the latter having only a truth value, which is not closed by double-orthogonality. A suitable and non-restrictive requirement on the classical proofs (right structural rules are forbidden for positive formulas) allows to prove the soundness of the interpretation. The model is validated by proving that the usual terms of Gödel’s system T realize the axioms of Peano arithmetic. Concerning extraction of programs from proofs, Friedman’s trick is implemented through the use of an external $\mu$-variable rather than through the replacement of the $\bot$ formula by an existential statement, which allows for a simpler and more effective interpretation in some models (typically the models based upon game semantics). The choice of having a free $\mu$-variable $\kappa$ in the realizers may be seen as a way to avoid Friedman’s $A$-translation. Indeed, in Friedman’s original work, the translation is obtained by replacing each basic predicate $P$ with the disjunction $P \lor A$. In classical sequent calculus, the right-hand context is meant to be interpreted as a disjunction, so adding a fixed $\mu$-variable to this context corresponds to applying Friedman’s translation on programs instead of proofs. This variable is also used to define the orthogonality relation between our truth and falsity values.
Interpreting the axiom of dependent choice in a classical setting is much more complicated than interpreting arithmetic. To interpret it, Spector defined the bar recursor [14] and used it in Gödel’s Dialectica interpretation [5] (a computational interpretation similar to realizability). A more uniform version was then used to give an indirect realizability interpretation of countable choice [1], and a version with an implicit termination condition was later defined and used to interpret, still in an indirect realizability setting, the double-negation shift principle and therefore the axiom of dependent choice [2].

In the direct interpretation described in the talk and under some assumptions on the model of μPCF, the bar recursion operator [2] realizes the axiom of dependent choice, similarly to what was done in a previous work [3]. The particular implementation of Friedman’s trick then allows to obtain an extraction result on Π^0_2 formulas provable in classical analysis (Peano arithmetic + the axiom of dependent choice).

This work has been submitted recently as a journal paper.

References