Interpreting second-order arithmetic via update recursion

Valentin Blot

LMF, Inria, Université Paris-Saclay

Second-order arithmetic has two kinds of computational interpretations: via Spector's bar recursion [4] or via Girard-Reynolds polymorphic lambda-calculus [2, 3]. Bar recursion interprets the negative translation of the axiom of choice which, combined with an interpretation of the negative translation of the excluded middle, gives a computational interpretation of the negative translation of the axiom scheme of comprehension. It is then possible to instantiate universally quantified sets with arbitrary formulas (second-order elimination). On the other hand, polymorphic lambda-calculus interprets directly second-order elimination by means of polymorphic types. The present work aims at bridging the gap between these two interpretations by interpreting directly second-order elimination through update recursion, which is a variant of bar recursion due to Berger [1].

First, we show that a slight variant of Berger's update recursion [1] interprets the following principle:

$$\neg\neg\forall x\,(A\,(x) \vee \neg A\,(x))$$

which in turn implies the double negation of the axiom scheme of comprehension:

$$\neg\neg\exists X \forall x\,(X\,(x) \Leftrightarrow A\,(x))$$

The variant of Berger's update recursion that we use is:

$$\mathtt{ur} : ((\mathtt{nat} \to T + (T \to o)) \to o) \to (\mathtt{nat} \to T + \mathtt{unit}) \to o$$

and satisfies the following recursive equation:

$$\mathtt{ur}\,t\,u = t \begin{pmatrix} \lambda n.\mathtt{match}\,u\,n\,\mathtt{with} \\ \qquad \mathtt{inl}\,x \mapsto \mathtt{inl}\,x \\ \qquad \mathtt{inr}\,\_ \mapsto \mathtt{inr}\left(\lambda x.\mathtt{ur}\,t \begin{pmatrix} \lambda m.\mathtt{if}\,m = n \\ \quad \mathtt{then}\,\mathtt{inl}\,x \\ \quad \mathtt{else}\,u\,m \end{pmatrix}\right) \\ \mathtt{end} \end{pmatrix}$$

If we see the type $T + \mathtt{unit}$ as an option type on T, $\mathtt{ur}$ provides t with an extension of u that performs a recursive call whenever $u\,n$ is not defined. We show that $\mathtt{ur}$ interprets the formula:

$$\neg\forall x\,(A\,(x) \vee \neg A\,(x)) \Rightarrow \neg\forall x\,(A\,(x) \vee \top)$$

and hence, feeding it with $\lambda\_.\mathtt{inr}\,\mathtt{tt}$ that interprets $\forall x\,(A\,(x) \vee \top)$, we obtain an interpretation for:

$$\neg\neg\forall x\,(A\,(x) \vee \neg A\,(x))$$

Second-order arithmetic can be obtained from first-order arithmetic by adding quantification over predicates. The logical power of second-order arithmetic resides in its second-order elimination rule:

$$\forall X\,B \Rightarrow B\,[A\,(x)\,/X\,(x)]$$

that instantiates a second-order variable X with an arbitrary formula $A(x)$. Using the `ur` operator above, we are able to define inductively on the structure of B a computational interpretation of the second-order elimination rule, and therefore of second-order arithmetic.

We define a bar recursive interpretation of second-order arithmetic presented as arithmetic with quantification on predicates rather than the equivalent axiom scheme of comprehension. This presentation of second-order arithmetic is the one that most closely reflects the typing rules of polymorphic λ-calculus, and as such we make a step towards a comparison of the two families of interpretations of second-order arithmetic: bar recursion and system F.

As a future work we would like to deepen the understanding of the connection between these two principles by comparing the computational behavior of programs extracted from a single proof via the two techniques.

Another aspect that we would like to study is wether it is possible to use control operators in the interpretation of the $\forall 2e$ rule. Indeed, there is a strong connection between the negative translation of proofs and the continuation-passing style (cps) translation of programs, the latter being the Curry-Howard equivalent of the former. Calculi with control features have been designed to interpret classical proofs directly. Most of these calculi contain a notion of duality that corresponds on the logical side to the duality between a formula and its negation, and on the computational side to a call-by-name or a call-by-value evaluation strategy. During its computation, update recursion has an asymmetric behavior that consists in building a realizer of $\neg B$ by reading a realizer of B and making a recursive call with a new knowledge extended with this new realizer. This behavior corresponds to the call-by-name interpretation of the excluded middle under a cps translation. We would therefore like to have a version of update recursion that uses control operators and can be translated either to the current version through a call-by-name cps translation, or to a dual version through a call-by-value cps translation. Moreover, control operators can capture context and restore it at a later point. We would like to explore the possibility of using this property to define more intuitive versions of our interpretation of second-order elimination that could act on all instances of $X(t)$ in a formula through context capture.

While termination of `ur` can be shown using Zorn's lemma, termination of the polymorphic lambda-calculus relies on impredicative reducibility candidates. A comparison would therefore provide a new approach to understanding impredicativity.

References

[1] Ulrich Berger. A computational interpretation of open induction. In *19th IEEE Symposium on Logic in Computer Science*, page 326. IEEE Computer Society, 2004.

[2] Jean-Yves Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *2nd Scandinavian Logic Symposium*, pages 63–69. North-Holland, 1971.

[3] John Reynolds. Towards a theory of type structure. In *Programming Symposium, Paris, April 9-11, 1974*, Lecture Notes in Computer Science, pages 408–423. Springer, 1974.

[4] Clifford Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In *Recursive Function Theory: Proceedings of Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, 1962.