A direct computational interpretation of second-order arithmetic via update recursion

Valentin Blot

INRIA - LMF
Univ. Paris-Saclay

Second-order arithmetic

- from the point of view of provability

  first order arithmetic $\lneq$ **second order arithmetic**
  $\lneq$ **ZF set theory**

- however

  most of "usual" mathematics
  can be formalized in second order arithmetic

- lies at the heart of the reverse mathematics program

  restricted forms of comprehension

A bit of history: computational interpretations

- Interpretations of 1st order arithmetic
  - 1941: Gödel's *Dialectica* interpretation (published in 1958)
  - 1945: Kleene's *number realizability*
  - 1959: Kreisel's *modified realizability*
- Interpretations of 2nd order arithmetic
  - 1962: Spector's *bar recursion*
  - 1972: Girard's *system F*
  - 1974: Reynolds' *polymorphic λ-calculus*
  - 1994: Krivine's *classical realizability*
  - 1998: Berardi-Bezem-Coquand's *demand-driven bar recursion*
  - 2004: Berger's *update recursion*

A bit of history: computational interpretations

- Interpretations of 1st order arithmetic
  - 1941: Gödel's *Dialectica* interpretation (published in 1958)
  - 1945: Kleene's *number realizability*
  - 1959: Kreisel's *modified realizability*
- Interpretations of 2nd order arithmetic
  - 1962: Spector's *bar recursion*
  - 1972: Girard's *system F*
  - 1974: Reynolds' *polymorphic λ-calculus*
  - 1994: Krivine's *classical realizability*
  - 1998: Berardi-Bezem-Coquand's *demand-driven bar recursion*
  - 2004: Berger's *update recursion*

Two families of computational interpretations

Interpretations of second order arithmetic

- polymorphic
  - system F / polymorphic λ-calculus
  - Krivine's classical realizability
- bar recursive
  - Spector's bar recursion
  - Berardi-Bezem-Coquand's demand-driven bar recursion
  - Berger's update recursion

These two techniques interpret the same theory, but very differently.

What is the connection between bar recursion and polymorphism?

A glimpse of polymorphism and bar recursion

Polymorphism

- extension of simple types

$$T, U ::= \alpha \mid T \to U \mid \forall \alpha\, T$$

e.g. `map` works for any types α, β

$$\texttt{map} : \forall \alpha \forall \beta\, ((\alpha \to \beta) \to \texttt{list}\, \alpha \to \texttt{list}\, \beta)$$

- impredicativity
  if $x : \forall \alpha\, (\alpha \to \alpha)$, then

$$x : \forall \beta\, (\beta \to \beta) \to \forall \beta\, (\beta \to \beta)$$

with $\alpha := \forall \beta\, (\beta \to \beta)$

$\Rightarrow$ self application

$$x\, x : \forall \beta\, (\beta \to \beta)$$

Polymorphism

Logically: second order arithmetic via quantification on propositions

Extend the rules of first order arithmetic with

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall X\, A}\, X \notin FV(\Gamma) \qquad \frac{\Gamma \vdash \forall X\, A}{\Gamma \vdash A\left[B\left(x\right)/X\left(x\right)\right]}$$

second order variables instantiated with **arbitrary** propositions

- close to polymorphic λ-calculus:

  instantiating a set variable with an arbitrary proposition

  $\sim$

  instantiating a polymorphic program with an arbitrary type

Bar recursion

- only simple types
$$T, U ::= \alpha \mid T \to U$$
- a new form of recursion over well-founded trees
  - these trees are not inductively defined
  - well-foundedness of these trees is a meta-property, consequence of the continuity of programs
- termination proved via Zorn's lemma, dependent choice, bar induction...

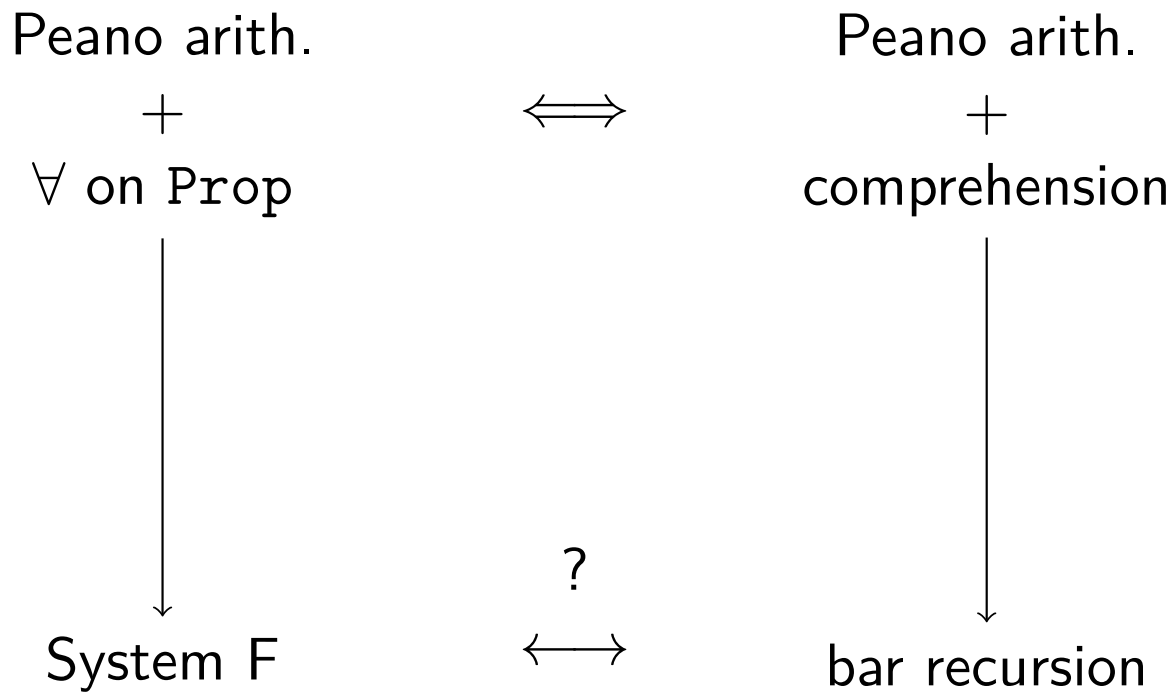
Bar recursion

Logically: second order arithmetic via the comprehension scheme

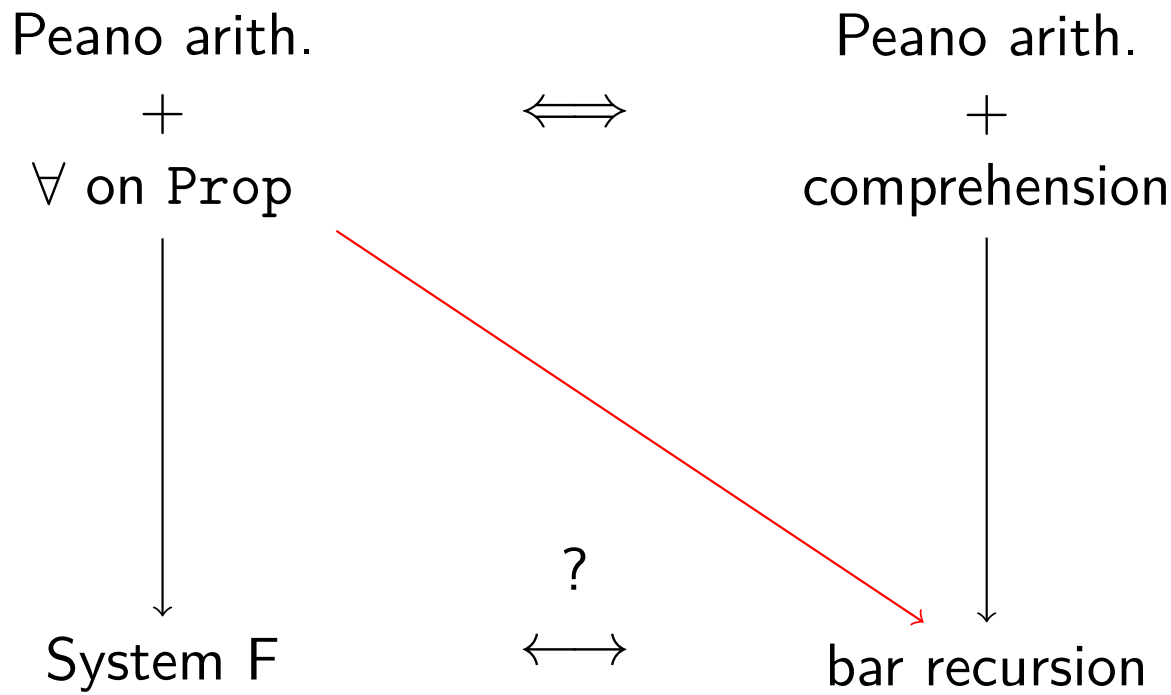
$$\exists X \forall x \left(X\left(x\right) \Leftrightarrow B\left(x\right) \right)$$

- builds sets from arbitrary propositions: $\left\{ x \in \mathbb{N} \,|\, B\left(x\right) \right\}$
- bar recursion builds such a set by performing recursion on a subset of $\mathcal{P}\left(\mathbb{N}\right)$ represented as a tree

The big picture

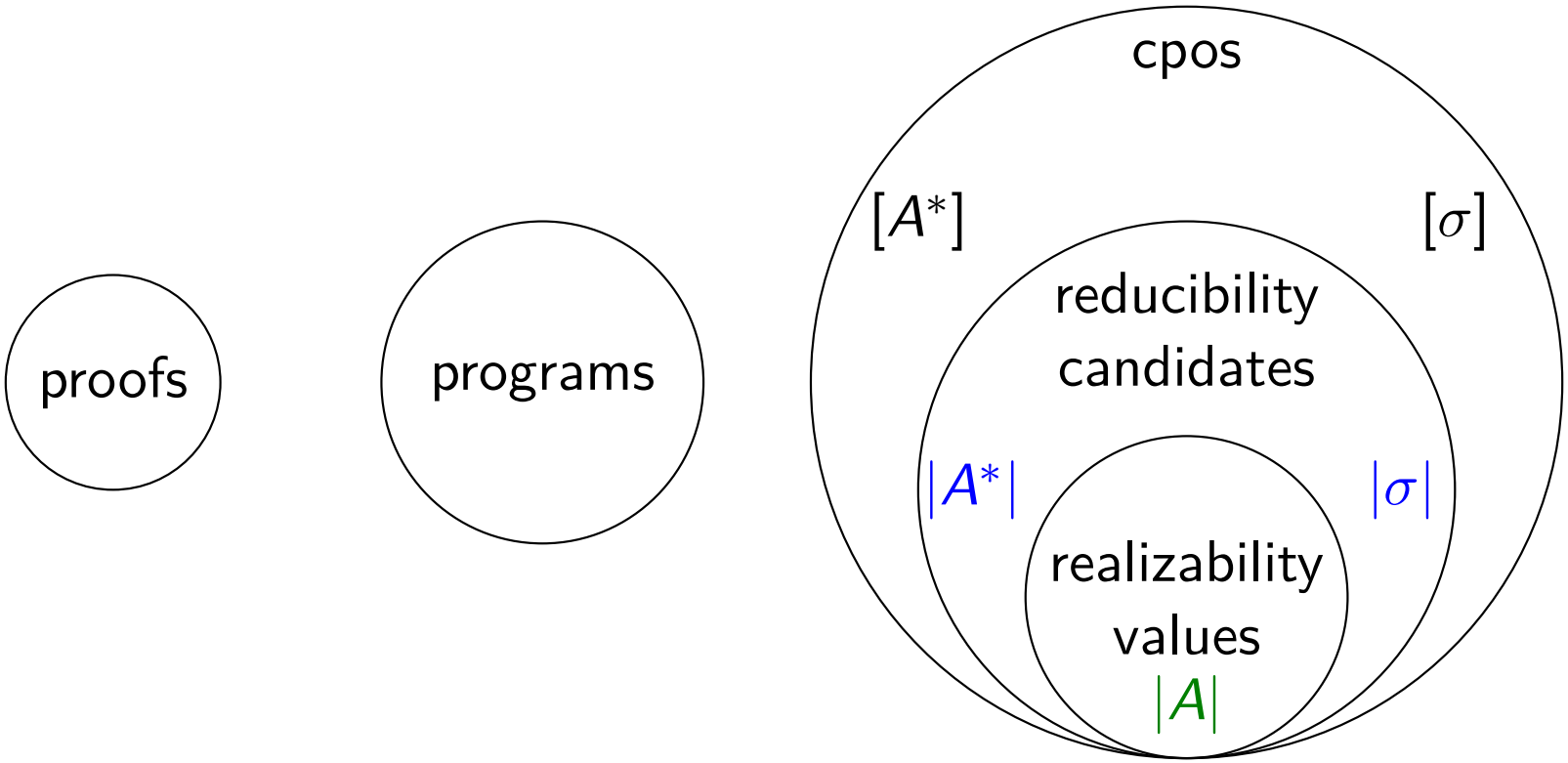


The big picture



A realizability model for second order arithmetic

From programs to termination and correctness



$M : \sigma$ $\qquad$ $[M] \in |\sigma| \subseteq [\sigma]$

$$\frac{\pi}{\vdash A} \qquad \pi^* : A^* \qquad [\pi^*] \in |A| \subseteq |A^*| \subseteq [A^*]$$

termination $\qquad$ correctness

Interpretation of second order logic

- classical logic
  - Needed for bar recursion
  - Requires $|\bot| = |\forall X\, X| \neq \emptyset$ (otherwise $|\neg A|$ is $\emptyset$ or $[A^*]$)
  - X^* must be non-trivial, we choose $X^* = \iota$
- interpretation
  - formulas

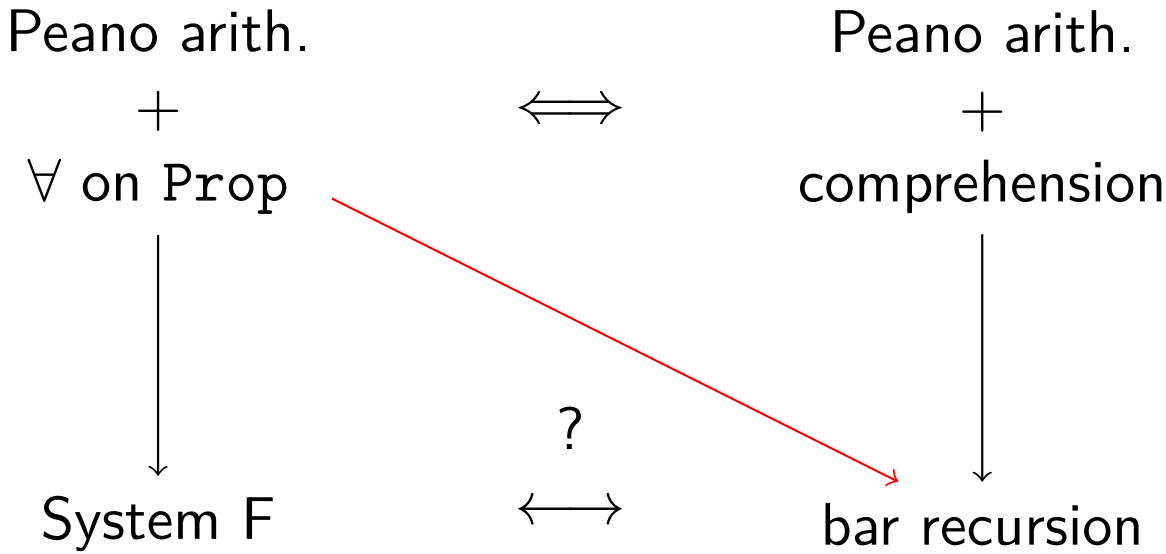
$$(X\,(t))^* = \iota \qquad (A \Rightarrow B)^* = A^* \to B^*$$
$$(\forall x\, A)^* = \iota \to A^* \qquad (\forall X\, A)^* = A^*$$

second order quantification is uniform

  - proofs
    - first order logic: standard
    - axioms: $\left(\dfrac{}{\forall x \forall y\,(\mathrm{S}\,x = \mathrm{S}\,y \Rightarrow x = y)}\right)^* = \lambda xyp.p$
    - second order

$$\left(\frac{\dfrac{\pi}{A}}{\forall X\, A}\right)^* = \pi^* \qquad \left(\frac{\dfrac{\pi}{\forall X\, A}}{A\,[B/X]}\right)^* \text{ built from } \pi^* \text{ and } \begin{matrix}\text{update}\\ \text{recursion}\end{matrix}$$

Conclusion and future works



- Classical logic / continuation-passing-style
  - current interpretation has a call-by-name flavour
  - we are in the target of a negative translation
  - more direct interpretation with control operators?
- New termination proof for System F?
- Reverse mathematics
  - computational consequences of restrictions to comprehension?
  - computational versions of RCA_0, ACA_0, $\Pi^1_1 CA_0$?