A puzzle about Gödel's numbering

James Avery[1] Jean-Yves Moyen[1] Jakob Grue Simonsen[1]
Jean-Yves.Moyen@lipn.univ-paris13.fr

[1]Datalogisk Institut
University of Copenhagen

Supported by the VILLUM FONDEN network for Experimental Mathematics in Number Theory, Operator Algebras, and Topology; the Marie Curie action "Walgo" program H2020-MSCA-IF-2014 number 655222; and the Danish Council for Independent Research *Sapere Aude* grant "Complexity via Logic and Algebra" (COLA).

October 6-7 2016

Part 1: the puzzle

Gödel's numberings

Gödel's numbering

Historically introduced by K. Gödel to encode (arithmetical) formulas into numbers.

Allows to manipulate formulas as arithmetical objects and thus write formulas about formulas.

Proof of Gödel's Incompleteness Theorem boils down to "This sentence has no proof".

Gödel's numbering

Historical numbering: assign a number to each symbol and use power of the n-th prime to say that the symbol is in n-th position.

Example:
"`0`" is 6, "`=`" is 5. "`0=0`" is $2^6 \times 3^5 \times 5^6 = 243,000,000$.

Example: "the first symbol of φ is `0`" = "the power of 2 in the encoding of φ is 6".

Used to encode Turing Machines into numbers and thus create Universal Turing Machine.

Gödel's numberings

There are many ways to encode stuff into numbers.

Example: letters can be encoded using the ASCII code ('H' is $1001000_{(2)} = 72$).
Example: strings can be encoded using ASCII + leading '1' ("Hello" is $1, 1001000, 1100101, 1101100, 1101100, 1101111_{(2)} = 53, 900, 686, 959$).

Example: images into `.bmp` files, read as one big binary number.

Example: anything stored into your computer...

In general: any injective function into the natural numbers can be considered as a Gödel's numbering.

Gödel's numbering of program

A Gödel's numbering of programs allow to manipulate programs with other programs.

Example: compilation is a manipulation between Gödel's numbering of the source and object files.

A Gödel's numbering does not need to be computable!

Example: encode uniformly terminating programs into even numbers and other into odd numbers.

Operators on programs

Binary operators on programs

If `Pgms` is a set of programs (`C`, Turing machines, ...) we can define (binary) operators on it. $\mathbb{F} : \texttt{Pgms} \times \texttt{Pgms} \to \texttt{Pgms}$.

Example: sequential composition, parallel composition (with or without communication), ..., other things?

Operators can be complicated: sequential composition of `C` programs requires some α-conversions + cleaning headers (conflicting `#define`) + ...

Operators can be non-computable: "if `p` and `q` compute inverse functions, then let $\mathbb{F}(\texttt{p}, \texttt{q})$ be $\lambda x.x$, otherwise ..."

Operators and numberings

A binary operator on programs and a Gödel's numbering of programs automatically define a binary operator on numbers.

Example: if $\mathtt{p}$ is encoded by 132, $\mathtt{q}$ by 93 and $\mathbb{F}(\mathtt{p}, \mathtt{q}) = \mathtt{r}$, encoded by 32789; then $\mathrm{F}(132, 93) = 32789$.

The puzzle

The puzzle

Can you choose:

- a ~~Turing-complete programming language~~ ICC system;
- a Gödel's numbering for it;
- a binary operator on it;

such that the induced operator on numbers is "as simple as possible"?

Example: start by looking into sequential or parallel compositions (still, many choices of such operators).

Example: can the operator on numbers be increasing? convex? polynomial? addition? concatenation? "continuous"? injective? other properties?