Lecture
Computability
(DIT313, DAT415)

Nils Anders Danielsson

2022-11-21

Applications

- Inductively defined sets/structural induction: General tools for defining and proving things.
- Primitive recursion: Terminating.
- Semantics: What do programs mean?
- Computability: What can/cannot be implemented?

Today

- X-computability.
- A self-interpreter for χ.
- Reductions.
- More problems that are or are not computable.
- More about coding.

X-computability

$[\![\_]\!]$

- Define $CExp = \{\, p \in Exp \mid p \text{ is closed} \,\}$.
- The semantics as a partial function:

$$
\begin{aligned}
&[\![\_]\!] \in CExp \rightharpoonup CExp \\
&[\![p]\!] = v \ \text{ if } p \Downarrow v
\end{aligned}
$$

χ-computable functions

Assume that we have methods for representing members of the sets A and B as closed χ expressions.

A partial function $f \in A \rightharpoonup B$ is *χ-computable* (with respect to these methods) if

$$\exists\, e \in CExp.\ \forall\, a \in A.\ [\![e \ulcorner a \urcorner]\!] = \ulcorner f\ a \urcorner.$$

Note: If one side is undefined, then the other side must also be undefined.

χ-computable functions

A special case:

A (total) function $f \in A \to B$ is χ-computable if there is a closed expression e such that:

- $\forall a \in A.\ e \ulcorner a \urcorner \Downarrow \ulcorner f\ a \urcorner$.

Quiz

What would go “wrong” if we decided to represent closed χ expressions in the following way?

A closed χ expression is represented by True() if it terminates, and by False() otherwise.

Respond at `https://pingo.coactum.de/921051`.

Representation

- The choice of representation is important.
- In this course (unless otherwise noted or inapplicable): The “standard” representation.
- It might make sense to require that the representation function $\ulcorner \_ \urcorner$ is “computable”.
  - However, how should this be defined?

Examples

- Addition of natural numbers is χ-computable:

$$
\begin{aligned}
&add \in \mathbb{N} \times \mathbb{N} \to \mathbb{N} \\
&add\,(m, n) = m + n
\end{aligned}
$$

- The intensional halting problem is not χ-computable:

$$
\begin{aligned}
&halts \in \mathit{CExp} \to \mathit{Bool} \\
&halts\ p = \mathbf{if}\ p\ \textsf{terminates}\ \mathbf{then}\ \textsf{true}\ \mathbf{else}\ \textsf{false}
\end{aligned}
$$

- The semantics $[\![\_]\!]$ is computable.

Self-interpreter

Self-interpreter

Goal: Define $eval \in CExp$ satisfying

$$\forall\, e \in CExp.\, [\![eval \ulcorner e \urcorner]\!] = \ulcorner [\![e]\!] \urcorner.$$

Self-interpreter

$$\textbf{rec}\ eval = \lambda\, e.\, \textbf{case}\ e\ \textbf{of}$$
$$\{\ldots$$
$$\}$$

Self-interpreter

$$\frac{}{\textsf{lambda}\ x\ e \Downarrow \textsf{lambda}\ x\ e}$$

$$\textsf{Lambda}(x, e) \to \textsf{Lambda}(x, e)$$

Self-interpreter

$$\frac{e_1 \Downarrow \textsf{lambda}\ x\ e \qquad e_2 \Downarrow v_2 \qquad e\,[x \leftarrow v_2] \Downarrow v}{\textsf{apply}\ e_1\ e_2 \Downarrow v}$$

$$\begin{array}{l}\textsf{Apply}(e_1, e_2) \rightarrow \mathbf{case}\ \mathit{eval}\ e_1\ \mathbf{of} \\ \quad \{\,\textsf{Lambda}(x, e) \rightarrow \mathit{eval}\,(\mathit{subst}\ x\,(\mathit{eval}\ e_2)\ e) \\ \quad \}\end{array}$$

Exercise: Define subst.

Self-interpreter

$$\frac{e\,[x \leftarrow \mathsf{rec}\ x\ e] \Downarrow v}{\mathsf{rec}\ x\ e \Downarrow v}$$

$$\mathsf{Rec}(x, e) \rightarrow eval\ (subst\ x\ \mathsf{Rec}(x, e)\ \ e)$$

Self-interpreter

$$\frac{es \Downarrow^{\star} vs}{\mathsf{const}\ c\ es \Downarrow \mathsf{const}\ c\ vs}$$

$\mathsf{Const}(c, es) \rightarrow \mathsf{Const}(c, map\ eval\ es)$

Exercise: Define map.

Self-interpreter

$$\frac{e \Downarrow \textsf{const}\ c\ vs \qquad Lookup\ c\ bs\ xs\ e' \qquad e'\ [xs \leftarrow vs] \mapsto e'' \qquad e'' \Downarrow v}{\textsf{case}\ e\ bs \Downarrow v}$$

$$
\begin{array}{l}
\textsf{Case}(e, bs) \rightarrow \textbf{case}\ eval\ e\ \textbf{of} \\
\quad \{\, \textsf{Const}(c, vs) \rightarrow \textbf{case}\ lookup\ c\ bs\ \textbf{of} \\
\qquad \{\, \textsf{Pair}(xs, e') \rightarrow eval\ (substs\ xs\ vs\ e') \\
\qquad \} \\
\quad \}
\end{array}
$$

Exercise: Define $lookup$ and $substs$.

Self-interpreter

$$
\begin{array}{l}
\textbf{rec}\ eval = \lambda\, e.\ \textbf{case}\ e\ \textbf{of} \\
\quad \{\ \mathsf{Lambda}(x, e) \rightarrow \mathsf{Lambda}(x, e) \\
\quad ;\ \mathsf{Apply}(e_1, e_2) \rightarrow \textbf{case}\ eval\ e_1\ \textbf{of} \\
\qquad \{\ \mathsf{Lambda}(x, e) \rightarrow eval\ (subst\ x\ (eval\ e_2)\ e)\ \} \\
\quad ;\ \mathsf{Rec}(x, e) \qquad \rightarrow eval\ (subst\ x\ \mathsf{Rec}(x, e)\ e) \\
\quad ;\ \mathsf{Const}(c, es) \rightarrow \mathsf{Const}(c, map\ eval\ es) \\
\quad ;\ \mathsf{Case}(e, bs) \quad \rightarrow \textbf{case}\ eval\ e\ \textbf{of} \\
\qquad \{\ \mathsf{Const}(c, vs) \rightarrow \textbf{case}\ lookup\ c\ bs\ \textbf{of} \\
\qquad\quad \{\ \mathsf{Pair}(xs, e') \rightarrow eval\ (substs\ xs\ vs\ e')\ \} \\
\qquad \} \\
\quad \}
\end{array}
$$

Note: $subst$, map, $lookup$ and $substs$ are meta-variables that stand for (closed) expressions.

Quiz

Is the following partial function χ-computable?

$$
\begin{aligned}
&halts \in CExp \rightharpoonup Bool \\
&halts\ p = \\
&\quad \textbf{if}\ p\ \text{terminates}\ \textbf{then}\ \text{true}\ \textbf{else}\ \text{undefined}
\end{aligned}
$$

Respond at `https://pingo.coactum.de/921051`.

Quiz

Is the following partial function χ-computable?

$$halts \in CExp \rightharpoonup Bool$$
$$halts\ p =$$
$$\quad \mathbf{if}\ p \text{ terminates } \mathbf{then} \text{ true } \mathbf{else} \text{ undefined}$$

One implementation: $\lambda\, p.\, (\lambda\, \_.\, \mathsf{True}())\ (eval\ p)$.

X-decidable

A function $f \in A \rightarrow Bool$ is χ-*decidable* if it is χ-computable. If not, then it is χ-*undecidable*.

X-semi-decidable

A function $f \in A \to Bool$ is χ*-semi-decidable* if there is a closed expression e such that, for all $a \in A$:

- If $f\ a =$ true then $e \ulcorner a \urcorner \Downarrow \ulcorner \text{true} \urcorner$.
- If $f\ a =$ false then $e \ulcorner a \urcorner$ does not terminate.

The halting problem is semi-decidable

The halting problem:

$$halts \in CExp \rightarrow Bool$$
$$halts\ p = \textbf{if}\ p\ \text{terminates}\ \textbf{then}\ \text{true}\ \textbf{else}\ \text{false}$$

A program witnessing the semi-decidability:

$$\lambda\, p.\, (\lambda\, \_.\, \mathsf{True}())\ (eval\ p)$$

Reductions

Reductions (one variant)

A χ-reduction of $f \in A \rightharpoonup B$ to $g \in C \rightharpoonup D$ consists of a proof showing that, if g is χ-computable, then f is χ-computable.

Reductions (one variant)

A χ-reduction of $f \in A \rightharpoonup B$ to $g \in C \rightharpoonup D$ consists of a proof showing that, if g is χ-computable, then f is χ-computable.

- If f is reducible to g, and f is not computable, then g is not computable.
- Last week we proved that the halting problem is undecidable by reducing another problem to it.

More (un)decidable problems

Semantic equality

- Are two closed χ expressions semantically equal?

$$\begin{aligned}
&\mathit{equal} \in \mathit{CExp} \times \mathit{CExp} \rightarrow \mathit{Bool} \\
&\mathit{equal}\,(e_1, e_2) = \\
&\quad \textbf{if}\ [\![e_1]\!] = [\![e_2]\!]\ \textbf{then}\ \mathsf{true}\ \textbf{else}\ \mathsf{false}
\end{aligned}$$

- The halting problem reduces to this one:

$$\mathit{halts} = \lambda\, p.\, \mathit{not}\,(\mathit{equal}\,\mathsf{Pair}(p, \ulcorner \mathbf{rec}\ x = x \urcorner))$$

Pointwise equality

- Pointwise equality:

$$\begin{aligned}&\mathit{pointwise\text{-}equal} \in \mathit{CExp} \times \mathit{CExp} \to \mathit{Bool}\\&\mathit{pointwise\text{-}equal}\,(e_1, e_2) =\\&\quad \mathbf{if}\ \forall\ e \in \mathit{CExp}.\, [\![e_1\ e]\!] = [\![e_2\ e]\!]\\&\quad \mathbf{then}\ \mathsf{true}\ \mathbf{else}\ \mathsf{false}\end{aligned}$$

- The previous problem reduces to this one:

$$\begin{aligned}&\mathit{equal} = \lambda\, p.\, \mathbf{case}\ p\ \mathbf{of}\\&\quad \{\, \mathsf{Pair}(e_1, e_2) \to\\&\qquad\quad \mathit{pointwise\text{-}equal}\\&\qquad\qquad \mathsf{Pair}(\mathsf{Lambda}(\mathsf{Zero}(), e_1),\\&\qquad\qquad\qquad \mathsf{Lambda}(\mathsf{Zero}(), e_2))\\&\quad \}\end{aligned}$$

Termination in n steps

- Termination in n steps:

$$
\begin{aligned}
&\mathit{terminates\text{-}in} \in \mathit{CExp} \times \mathbb{N} \to \mathit{Bool} \\
&\mathit{terminates\text{-}in}\,(e, n) = \\
&\quad \mathbf{if}\,\exists\, v.\ \exists\, p \in e \Downarrow v. \mid p \mid \leq n \\
&\quad \mathbf{then}\ \text{true}\ \mathbf{else}\ \text{false}
\end{aligned}
$$

  $\mid p \mid$: The number of rules ($\Downarrow$, not $\Downarrow^\star$) in the derivation tree.

- Decidable: We can define a variant of the self-interpreter that tries to evaluate e but stops if more than n rules are needed.

Representation

- How do we represent a χ-computable function?
- For instance a member of the set

$$\{ f \in \mathbb{N} \rightarrow \mathbb{N} \mid f \text{ is } \chi\text{-computable} \}.$$

- By the representation of one of the closed expressions witnessing the computability of the function. However, which one?
- One solution: Switch to

$$\{ (f, e) \mid f \in \mathbb{N} \rightarrow \mathbb{N}, e \in \mathit{CExp}, e \text{ implements } f \},$$

and define $\ulcorner (f, e) \urcorner = \ulcorner e \urcorner$.

Quiz

Is the following problem χ-decidable for $A = Bool$? What if $A = \mathbb{N}$?

$$\begin{aligned}
&\textbf{let } Fun = \{ (f, e) \mid f \in A \to Bool, e \in CExp, \\
&\qquad\qquad\qquad e \text{ implements } f \} \textbf{ in} \\
&\textit{pointwise-equal}' \in Fun \times Fun \to Bool \\
&\textit{pointwise-equal}' \; ((f, _), (g, _)) = \\
&\quad \textbf{if } \forall\, a \in A.\, f\ a = g\ a \textbf{ then } \text{true} \textbf{ else } \text{false}
\end{aligned}$$

Hint: Use $eval$ or $terminates\text{-}in$.

Respond at `https://pingo.coactum.de/921051`.

Pointwise equality of computable functions in $Bool \to Bool$

- The function $pointwise\text{-}equal'$ is decidable.
- Implementation:

$$
\begin{array}{l}
pointwise\text{-}equal' = \lambda\, p.\, \textbf{case}\ p\ \textbf{of} \\
\quad \{\, \mathsf{Pair}(f, g) \to \\
\quad\quad and\ (equal_{Bool}\ (eval\ \mathsf{Apply}(f, \ulcorner \mathsf{True}() \urcorner)) \\
\quad\quad\quad\quad\quad\quad\quad\quad (eval\ \mathsf{Apply}(g, \ulcorner \mathsf{True}() \urcorner))) \\
\quad\quad\quad\quad (equal_{Bool}\ (eval\ \mathsf{Apply}(f, \ulcorner \mathsf{False}() \urcorner)) \\
\quad\quad\quad\quad\quad\quad\quad\quad (eval\ \mathsf{Apply}(g, \ulcorner \mathsf{False}() \urcorner))) \\
\quad \}
\end{array}
$$

Pointwise equality of computable functions in $Bool \to Bool$

- The function $pointwise\text{-}equal'$ is decidable.
- Implementation:

$$
\begin{array}{l}
pointwise\text{-}equal' = \lambda\, p.\, \mathbf{case}\ p\ \mathbf{of} \\
\quad \{\mathsf{Pair}(f, g) \to \\
\quad\quad and\ (equal_{Bool}\ (eval\ \ulcorner \llcorner f \lrcorner\ \mathsf{True}()\ \urcorner) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad (eval\ \ulcorner \llcorner g \lrcorner\ \mathsf{True}()\ \urcorner)) \\
\quad\quad\quad\quad (equal_{Bool}\ (eval\ \ulcorner \llcorner f \lrcorner\ \mathsf{False}()\ \urcorner) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad (eval\ \ulcorner \llcorner g \lrcorner\ \mathsf{False}()\ \urcorner)) \\
\quad \}
\end{array}
$$

Pointwise equality of computable functions in $\mathbb{N} \to Bool$

- The function $pointwise\text{-}equal'$ is undecidable.
- The halting problem reduces to it:

$$
\begin{aligned}
&halts = \lambda\, p.\, not\, (pointwise\text{-}equal' \\
&\quad \mathsf{Pair}(\ulcorner \lambda\, n.\, terminates\text{-}in\, \mathsf{Pair}(\llcorner\, code\ p\, \lrcorner, n)\, \urcorner, \\
&\qquad \ulcorner \lambda\, \_.\, \mathsf{False}()\, \urcorner))
\end{aligned}
$$

Coding

$\llcorner \_ \lrcorner$

One way to give a semantics to $\llcorner \_ \lrcorner$:

- $\llcorner \_ \lrcorner$ is a constructor of a variant of Exp:

$$\frac{e \in Exp}{\llcorner e \lrcorner \in \overline{Exp}} \qquad \frac{e_1 \in \overline{Exp} \qquad e_2 \in \overline{Exp}}{\mathsf{apply}\ e_1\ e_2 \in \overline{Exp}} \qquad \cdots$$

- This variant is the domain of $\ulcorner \_ \urcorner$:

$$\begin{aligned} \ulcorner \_ \urcorner &\in \overline{Exp} \to Exp \\ \ulcorner \llcorner e \lrcorner \urcorner &= e \\ \ulcorner \mathsf{apply}\ e_1\ e_2 \urcorner &= \mathsf{Apply}(\ulcorner e_1 \urcorner, \ulcorner e_2 \urcorner) \\ &\vdots \end{aligned}$$

$\llcorner \_ \lrcorner$

- Examples:

$$\begin{aligned} \ulcorner \llcorner f \lrcorner \,\mathsf{True}()\urcorner &= \mathsf{Apply}(f, \ulcorner \mathsf{True}() \urcorner) \\ \ulcorner eval \llcorner code\ e \lrcorner \urcorner &= \mathsf{Apply}(\ulcorner eval \urcorner, code\ e) \end{aligned}$$

- Note that you do not have to use $\llcorner \_ \lrcorner$.

⌞ — ⌟

The reduction used above:

$$
\begin{aligned}
&\mathit{halts} = \lambda\, p.\ \mathit{not}\ (\mathit{pointwise\text{-}equal}'\\
&\quad \mathsf{Pair}(\ulcorner \lambda\, n.\ \mathit{terminates\text{-}in}\ \mathsf{Pair}(\llcorner\ \mathit{code}\ p\ \lrcorner, n)\ \urcorner,\\
&\qquad\quad \ulcorner \lambda\, \_.\ \mathsf{False}()\ \urcorner))
\end{aligned}
$$

Expanded:

$$
\begin{aligned}
&\lambda\, p.\ \mathit{not}\ (\mathit{pointwise\text{-}equal}'\\
&\quad \mathsf{Pair}(\mathsf{Lambda}(\ulcorner\, n\, \urcorner,\\
&\qquad\qquad \mathsf{Apply}(\ulcorner\, \mathit{terminates\text{-}in}\, \urcorner,\\
&\qquad\qquad\qquad \mathsf{Const}(\ulcorner\, \mathsf{Pair}\, \urcorner,\\
&\qquad\qquad\qquad\quad \mathsf{Cons}(\mathit{code}\ p,\\
&\qquad\qquad\qquad\qquad \mathsf{Cons}(\mathsf{Var}(\ulcorner\, n\, \urcorner), \mathsf{Nil}())))))),\\
&\qquad\quad \ulcorner \lambda\, \_.\ \mathsf{False}()\ \urcorner))
\end{aligned}
$$

Coding

Probably not what you want:

$$\lambda\, p.\ulcorner\, eval\, p\,\urcorner = \lambda\, p.\, \mathsf{Apply}(\ulcorner\, eval\,\urcorner, \mathsf{Var}(\ulcorner\, p\,\urcorner))$$

If p corresponds to 0:

$$\lambda\, p.\, \mathsf{Apply}(\ulcorner\, eval\,\urcorner, \mathsf{Var}(\mathsf{Zero}()))$$

The argument p is ignored.

Coding

Perhaps more useful:

$$\lambda\, p.\ulcorner eval \llcorner code\ p \lrcorner \urcorner = \lambda\, p.\,\mathsf{Apply}(\ulcorner eval \urcorner, code\ p)$$

For any closed expression e:

$$(\lambda\, p.\ulcorner eval \llcorner code\ p \lrcorner \urcorner)\ \ulcorner e \urcorner \Downarrow \ulcorner eval \ulcorner e \urcorner \urcorner$$

Quiz

What is the result of evaluating $(\lambda p.\, eval \ulcorner eval \llcorner code\ p \lrcorner \urcorner)\ \ulcorner \mathsf{Zero}() \urcorner$?

1. Nothing
2. Zero()
3. $\ulcorner \mathsf{Zero}() \urcorner$
4. $\ulcorner\ulcorner \mathsf{Zero}() \urcorner\urcorner$
5. $\ulcorner\ulcorner\ulcorner \mathsf{Zero}() \urcorner\urcorner\urcorner$
6. $\ulcorner\ulcorner\ulcorner\ulcorner \mathsf{Zero}() \urcorner\urcorner\urcorner\urcorner$

Recall that $[\![eval \ulcorner e \urcorner]\!] = \ulcorner [\![e]\!] \urcorner$ and $[\![code \ulcorner e \urcorner]\!] = \ulcorner\ulcorner e \urcorner\urcorner$ (for $e \in CExp$).

Respond at `https://pingo.coactum.de/921051`.

Quiz

What is the result of evaluating
$(\lambda p.\, eval \ulcorner eval \llcorner code\; p \lrcorner \urcorner) \ulcorner \mathsf{Zero}() \urcorner$?

$$
\begin{aligned}
&[\![(\lambda p.\, eval \ulcorner eval \llcorner code\; p \lrcorner \urcorner) \ulcorner \mathsf{Zero}() \urcorner]\!] &=\\
&[\![(\lambda p.\, eval\, \mathsf{Apply}(\ulcorner eval \urcorner, code\; p)) \ulcorner \mathsf{Zero}() \urcorner]\!] &=\\
&[\![eval\, \mathsf{Apply}(\ulcorner eval \urcorner, code \ulcorner \mathsf{Zero}() \urcorner)]\!] &=\\
&[\![eval\, \mathsf{Apply}(\ulcorner eval \urcorner, \ulcorner \ulcorner \mathsf{Zero}() \urcorner \urcorner)]\!] &=\\
&[\![eval \ulcorner eval \ulcorner \mathsf{Zero}() \urcorner \urcorner]\!] &=\\
&\ulcorner [\![eval \ulcorner \mathsf{Zero}() \urcorner]\!] \urcorner &=\\
&\ulcorner \ulcorner [\![\mathsf{Zero}()]\!] \urcorner \urcorner &=\\
&\ulcorner \ulcorner \mathsf{Zero}() \urcorner \urcorner
\end{aligned}
$$

Recall: $[\![eval \ulcorner e \urcorner]\!] = \ulcorner [\![e]\!] \urcorner$, $[\![code \ulcorner e \urcorner]\!] = \ulcorner \ulcorner e \urcorner \urcorner$.

Types

- The language χ is untyped.
- However, it may be instructive to see certain programs as typed.

Types

- $Rep\ A$: Representations of programs of type A.
- Some examples:

$$
\begin{aligned}
&\mathsf{True}() &&: Bool\\
&\ulcorner \mathsf{True}() \urcorner &&: Rep\ Bool\\
&\ulcorner \mathsf{true} \urcorner &&: Bool\\
&\lambda f.\,\lambda x.\,f\ x &&: (A \to B) \to A \to B\\
&\lambda f.\,\lambda x.\,\mathsf{Apply}(f, x) &&: Rep\ (A \to B) \to\\
& && \quad Rep\ A \to Rep\ B\\
&eval &&: Rep\ A \to Rep\ A\\
&code &&: Rep\ A \to Rep\ (Rep\ A)\\
&terminates\text{-}in &&: Rep\ A \times \mathbb{N} \to Bool\\
&\ulcorner terminates\text{-}in \urcorner &&: Rep\ (Rep\ A \times \mathbb{N} \to Bool)
\end{aligned}
$$

Types

The reduction used above:

$$\begin{aligned} &\mathit{halts} = \lambda\, p.\ \mathit{not}\ (\mathit{pointwise\text{-}equal}' \\ &\quad \mathsf{Pair}(\ulcorner \lambda\, n.\ \mathit{terminates\text{-}in}\ \mathsf{Pair}(\llcorner\ \mathit{code}\ p\ \lrcorner, n)\ \urcorner, \\ &\qquad\quad \ulcorner \lambda\, \_.\ \mathsf{False}()\ \urcorner)) \end{aligned}$$

If

$$\begin{aligned} &\mathit{pointwise\text{-}equal}' : \\ &\quad \mathit{Rep}\ (\mathbb{N} \to \mathit{Bool}) \times \mathit{Rep}\ (\mathbb{N} \to \mathit{Bool}) \to \mathit{Bool} \end{aligned}$$

then

$$\mathit{halts} : \mathit{Rep}\ A \to \mathit{Bool}.$$

More undecidable problems

Quiz

Is the following function χ-computable?

$$optimise \in CExp \rightarrow CExp$$
$$optimise\ e =$$

some optimally small expression with the same semantics as e

Size: The number of constructors in the abstract syntax (Exp, Br, $List$, not Var or $Const$).

Respond at `https://pingo.coactum.de/921051`.

Full employment theorem for compiler writers

- An optimally small non-terminating closed expression is equal to $\mathbf{rec}\ x = x$ (for some x).
- The halting problem reduces to this one:

$$
\begin{array}{l}
\mathit{halts} = \lambda\, p.\, \mathbf{case}\ \mathit{optimise}\ p\ \mathbf{of} \\
\quad \{\, \mathsf{Rec}(x, e) \rightarrow \mathbf{case}\ e\ \mathbf{of} \\
\qquad \{\, \mathsf{Var}(y) \quad \rightarrow \mathsf{False}() \\
\qquad ;\ \mathsf{Rec}(x, e) \rightarrow \mathsf{True}() \\
\qquad ;\ \ldots \\
\qquad \} \\
\quad ;\ \ldots \\
\quad \}
\end{array}
$$

Computable real numbers

- Computable reals can be defined in many ways.
- One example, using signed digits:

$$
\begin{aligned}
& \mathit{Interval} = \\
& \quad \{ (f, e) \mid f \in \mathbb{N} \to \{-1, 0, 1\}, e \in \mathit{CExp}, \\
& \qquad\qquad e \text{ implements } f \}
\end{aligned}
$$

$$
\begin{aligned}
& [\![\_]\!] \in \mathit{Interval} \to [-1, 1] \\
& [\![(f, \_)]\!] = \textstyle\sum_{i=0}^{\infty} f\ i \cdot 2^{-i-1}
\end{aligned}
$$

- Why signed digits? Try computing the first digit of $0.00000\ldots + 0.11111\ldots$ (in binary notation).

Is a computable real number equal to zero?

- Is a computable real number equal to zero?

$$\begin{array}{l} \textit{is-zero} \in \textit{Interval} \rightarrow \textit{Bool} \\ \textit{is-zero}\ x = \textbf{if}\ [\![x]\!] = 0\ \textbf{then}\ \mathsf{true}\ \textbf{else}\ \mathsf{false} \end{array}$$

- The halting problem reduces to this one:

$$\begin{array}{l} \textit{halts} = \lambda\, p.\ \textit{not}\ (\textit{is-zero}\ \ulcorner \lambda\, n. \\ \quad \textbf{case}\ \textit{terminates-in}\ \mathsf{Pair}(\llcorner\ \textit{code}\ p\ \lrcorner, n)\ \textbf{of} \\ \quad\quad \{\ \mathsf{True}() \rightarrow \mathsf{One}() \\ \quad\quad ;\ \mathsf{False}() \rightarrow \mathsf{Zero}() \\ \quad\quad \}\urcorner) \end{array}$$

Undecidable problems

- A list on Wikipedia.
- A list on MathOverflow.

Summary

- X-computability.
- A self-interpreter for χ.
- Reductions.
- More problems that are or are not computable.
- More about coding.