Lecture
Computability
(DIT312, DAT415)

Nils Anders Danielsson

2019-11-25

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:

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

X-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

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

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

X-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 $\mathsf{True}()$ if it terminates, and by $\mathsf{False}()$ otherwise.

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:

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

- The intensional halting problem is not χ-computable:

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

- 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

$$\begin{array}{l}\mathbf{rec}\ eval = \lambda\, e.\, \mathbf{case}\ e\ \mathbf{of} \\ \quad \{ \ldots \\ \quad \}\end{array}$$

Self-interpreter

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

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

Self-interpreter

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

$\mathsf{Apply}(e_1, e_2) \rightarrow \mathbf{case}\ eval\ e_1\ \mathbf{of}$
$\quad \{\, \mathsf{Lambda}(x, e) \rightarrow eval\ (subst\ x\ (eval\ e_2)\ e)$
$\quad \}$

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

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

$$\begin{aligned}
&\mathsf{Case}(e, bs) \to \mathbf{case}\ eval\ e\ \mathbf{of} \\
&\quad \{\mathsf{Const}(c, vs) \to \mathbf{case}\ lookup\ c\ bs\ \mathbf{of} \\
&\qquad\quad \{\mathsf{Pair}(xs, e') \to eval\ (substs\ xs\ vs\ e') \\
&\qquad\quad \} \\
&\quad \}
\end{aligned}$$

Exercise: Define $lookup$ and $substs$.

Self-interpreter

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

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

Quiz

Is the following partial function χ-computable?

$halts \in CExp \rightharpoonup Bool$

$halts\ p =$

if p terminates **then** true **else** undefined

X-decidable

A function $f \in A \to Bool$ 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 $e\ulcorner a \urcorner \Downarrow \ulcorner$ true $\urcorner$.
- If $f\,a =$ false then $e\ulcorner a \urcorner$ does not terminate.

The halting problem is semi-decidable

The halting problem:

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

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?

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

- The halting problem reduces to this one:

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

Pointwise equality

- Pointwise equality:

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

- The previous problem reduces to this one:

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

Termination in n steps

- Termination in n steps:

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

  $|p|$: The number of rules 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

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

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

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

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

Quiz

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

$$\textbf{let } Fun = \{\,(f, e) \mid f \in A \to Bool, e \in CExp, \\ e \text{ implements } f\} \textbf{ in}$$

$$pointwise\text{-}equal' \in Fun \times Fun \to Bool$$

$$pointwise\text{-}equal'\ ((f, \_), (g, \_)) =$$

$$\quad \textbf{if } \forall\ a \in A.\, f\ a = g\ a \textbf{ then } \text{true } \textbf{else } \text{false}$$

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

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

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

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

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

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

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

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

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

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

Coding

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

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

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

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

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

- Examples:

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

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

The reduction used above:

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

Expanded:

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

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. $\mathsf{Zero}()$
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 $[\![eval\ulcorner e\urcorner]\!] = \ulcorner [\![e]\!]\urcorner$ (for $e \in CExp$).

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:

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

Types

The reduction used above:

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

If

$$pointwise\text{-}equal' : Rep\ (\mathbb{N} \to Bool) \times Rep\ (\mathbb{N} \to Bool) \to Bool$$

then

$$halts : Rep\ A \to Bool.$$

More undecidable problems

Quiz

Is the following function χ-computable?

$optimise \in CExp \to CExp$
$optimise\ e =$
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 $Const$).

Full employment theorem for compiler writers

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

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

Computable real numbers

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

$$Interval = \{(f, e) \mid f \in \mathbb{N} \to \{-1, 0, 1\}, e \in CExp, e \text{ implements } f\}$$

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

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

Is a computable real number equal to zero?

- Is a computable real number equal to zero?

$$\begin{aligned}&\mathit{is\text{-}zero} \in \mathit{Interval} \to \mathit{Bool}\\&\mathit{is\text{-}zero}\ x = \mathbf{if}\ [\![x]\!] = 0\ \mathbf{then}\ \mathsf{true}\ \mathbf{else}\ \mathsf{false}\end{aligned}$$

- The halting problem reduces to this one:

$$\begin{aligned}&\mathit{halts} = \lambda\, p.\ \mathit{not}\ (\mathit{is\text{-}zero}\ulcorner\ \lambda\, n.\\&\quad \mathbf{case}\ \mathit{terminates\text{-}in}\ \mathsf{Pair}(\llcorner\ \mathit{code}\ p\ \lrcorner, n)\ \mathbf{of}\\&\qquad \{\mathsf{True}() \to \mathsf{One}()\\&\qquad ; \mathsf{False}() \to \mathsf{Zero}()\\&\qquad \}\urcorner)\end{aligned}$$

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.