Recursive functions:
a reminder

Primitive recursion, informally

To define a function $h(x, z)$ by primitive recursion, we need to describe what happens

- in the case where $z = 0$, and
- in the case where z is of the form $y + 1$ for some y, using the value of $h(x, y)$.

Example: multiplication

Below we have a primitive recursive definition of multiplication, in terms of $+$.

$$x \cdot 0 = 0$$

$$x \cdot (y+1) = x \cdot y + x$$

Note that this looks almost like a realistic computer program. An even more realistic version might look like

$$mult(x,y) =$$
$$if\ (y = 0)\ then\ 0\ else\ mult(x, y-1) + x.$$

Primitive recursion, formally

Definition. If $f : N^k \longrightarrow N$ and $g : N^{k+2} \longrightarrow N$, then the function $h : N^{k+1} \longrightarrow N$ is said to be defined by **primitive recursion** from f and g if

$$h(x, 0) = f(x)$$
$$h(x, s(y)) = g(x, y, h(x, y))$$

where x stands for $x_1, \ldots, x_k$.

Definition of primitive recursive functions

Definition. The class of **primitive recursive functions** is defined as follows:

- The **zero** function z, the **successor** function s, and all **projection** functions p_i^k are primitive recursive.
- Functions which arise by **composition** Cn or **primitive recursion** Pr from primitive recursive functions are also primitive recursive.

Towards general recursion

- Some functions (e.g. the Ackermann function) are not primitive recursive.
- Informally, this is because primitive recursive functions do not allow while-loops, i.e. constructs of the form

  “WHILE some condition holds, DO X”.
- Formally, instead of WHILE loops, we add a construct called **minimization**.

Definition of minimization

The **minimization** of a function $f : N^{k+1} \rightarrow N$ is defined as follows (where x stands for $x_1, \ldots, x_k$):

$$\mathrm{Mn}[f](x) = \begin{cases} y & \text{if } f(x,y) = 0 \text{ and for all } i < y, \\ & f(x,i) \text{ is defined and } \neq 0 \\ \\ \perp & \text{otherwise} \end{cases}$$

Algorithm for $\mathrm{Mn}[f](x)$

The algorithm for $\mathrm{Mn}[f](x)$ looks as follows:

```
y = 0;
while(not(f(x,y) = 0)) {
    y = y+1;
}
return y;
```

This can fail to halt for two reasons: either because $f(x,i)$ fails to halt for some i, or because $f(x,i) \neq 0$ for all i.

Definition of recursive functions

Definition. The class of **recursive functions** is defined as follows:

- The functions s and z are recursive, and so are all projections p_i^k.
- Functions built from recursive ones by using composition Cn or primitive recursion Pr are also recursive.
- Functions built from recursive ones by minimization Mn are also recursive.

Recursive relations (Part 1/2)

Basic idea

- The main idea behind this lecture is the notion of **decidable set** (or relation).
- Informally, a set A is called **effectively decidable** if there is an effective procedure that returns
  - “Yes” if $x \in A$, and
  - “No” if $x \notin A$.

Basic idea

- We shall deal mainly with the more technical notion of **recursively decidable set**, which establishes a link between sets and total recursive functions.
- In particular, we shall study the important special case of **primitive recursive sets**.
- These sets are very useful for “using recursive functions as a programming language”.

Relations

- Sets whose elements are pairs (x, y) are often called **relations**.
- E.g., the lesser-than relation $<$ on N is represented as the set of pairs

$$\{(x, y) \in N \times N : x < y\}.$$

- If R is a relation, we write $R(x, y)$ interchangeably with $(x, y) \in R$.

k-place relations

- More generally, we consider k-place relations, which are sets whose elements are k-tuples $(x_1, x_2, \ldots, x_k)$.
- E.g. the line below is a definition of a 3-place relation on the natural numbers:

$$R(x, y, z) \quad \text{iff} \quad x = y + z.$$

The characteristic function of a subset

The following notion establishes an important link between functions and sets resp. relations:

Definition. The **characteristic function** of a subset A of a set B is the function

$$\xi_A : B \rightarrow \{0,1\}$$

given by

$$\xi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A. \end{cases}$$

Example

The characteristic function $\xi_< : N \times N \longrightarrow \{0, 1\}$ of the lesser-than relation $<$ on N is given as follows:

$$\xi_<(x, y) = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise.} \end{cases}$$

Effectively decidable relations

The main idea behind this lecture is captured by the following definition.

Definition. A k-place relation R is called **effectively decidable** if there is an effective procedure that returns

- "Yes" (or 1) if $R(x_1, \ldots, x_k)$ holds, and
- "No" (or 0) if $R(x_1, \ldots, x_k)$ does not hold;

or, more precisely, if ξ_R is effectively computable.

Recursive relations

To enable technical progress, we shall use a more specific version of decidability:

Definition. A k-place relation R on the natural numbers is called **recursively decidable**, or simply **recursive**, if ξ_R is recursive.

“Recursive” vs. “effectively decidable”

Which of the two claims is true?

1. “Recursive relations are effectively decidable.”
2. “Effectively decidable relations are recursive.”

Primitive recursive relations

Definition. A k-place relation R on the natural numbers is called **primitive recursive** if its characteristic function ξ_R is primitive recursive

(So every primitive recursive relation is recursive.)

Example

The relation below is primitive recursive.

$$R(y) \quad \text{iff} \quad y > 0$$

ξ_R is the signum function sg seen earlier, which can be defined by primitive recursion as follows:

$$\begin{aligned} sg(0) &= 0 \\ sg(y+1) &= 1. \end{aligned}$$

Example

The lesser-than relation $<$ is primitive recursive, because its characteristic function $\xi_<$ can be written as

$$sg(y \dot{-} x),$$

where the function $\dot{-}$ is defined by primitive recursion as follows:

$$x \dot{-} 0 = x$$

$$x \dot{-} s(y) = pred(x \dot{-} y).$$

Example

The **identity relation**, which holds if and only if

$$x = y,$$

is primitive recursive. Its characteristic function is

$$sg(x \dot{-} y) + sg(y \dot{-} x).$$

Boolean connectives for relations

- The **conjunction** of two relations R_1 and R_2 is the relation S defined as follows:

$$S(x_1, \ldots, x_k) \text{ iff } R_1(x_1, \ldots, x_k) \text{ and } R_2(x_1, \ldots, x_k).$$

- The **disjunction** of two relations R_1 and R_2 is the relation S defined as follows:

$$S(x_1, \ldots, x_k) \text{ iff } R_1(x_1, \ldots, x_k) \text{ or } R_2(x_1, \ldots, x_k).$$

Remark: this is just new terminology and notation for the intersection ($\cap$) and union ($\cup$) of sets.

Boolean connectives for relations

- The **complement** of a k-place relation R on the natural numbers is the relation S defined as follows:

$$S(x_1, \ldots, x_k) \quad \text{iff} \quad \text{not } R(x_1, \ldots, x_k).$$

Boolean connectives for relations

Proposition. Let R and S be k-place relations on the natural numbers.

- If R and S are (primitive) recursive, then so are their conjunction and disjunction.
- If R is (primitive) recursive, then so is its complement.

Proof. See lecture.

Definition by cases

Definition. A function $f(x,y)$ is given by **definition by cases** if

$$f(x,y) = \begin{cases} g_1(x,y) & \text{if } C_1(x,y) \\ \quad\vdots & \\ g_n(x,y) & \text{if } C_n(x,y), \end{cases}$$

where $g_1,\ldots,g_n$ are functions, and $C_1,\ldots,C_n$ are relations that are

- mutually exclusive, i.e., for no x,y do more than one of them hold, and
- collectively exhaustive, i.e., for any x,y at least one of them holds.

Example

The maximum function, which returns the larger of its to arguments, has a convenient definition by cases:

$$\max(x, y) = \begin{cases} y & \text{if } x < y \\ y & \text{if } x = y \\ x & \text{if } x > y. \end{cases}$$

Definition by cases

Proposition.

1. If the functions $g_1, \ldots, g_n$ are recursive and the relations $C_1, \ldots, C_n$ are recursive, then f is recursive.
2. If the functions $g_1, \ldots, g_n$ are primitive recursive and the relations $C_1, \ldots, C_n$ are primitive recursive, then f is primitive recursive.

Proof. See lecture.

A non-recursive relation

Let $f_0, f_1, f_2, \ldots$ be an enumeration of the recursive functions. (It follows from the exercise about coding recursive functions that such an enumeration exists.) The “self-halting” relation $self$ given by

$$self(x) \quad \text{iff} \quad f_x(x) \text{ is defined.}$$

is not recursive. This follows from a diagonal argument (see lecture for proof).

Substitution

Given a relation $R(y_1, \ldots, y_m)$ and **total** functions $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$, the relation $R^*(x_1, \ldots, x_n)$ obtained by **substitution** from the f_i and R is defined by

$$R^*(x_1, \ldots, x_n)$$

iff

$$R(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)).$$

Example

The relation R defined as follows

$$R(x, y, v) \quad \text{iff} \quad x \cdot v = y$$

is obtained by substitution from the identity relation and the functions

$$f_1(x, y, v) = x \cdot v \qquad\qquad f_2(x, y, v) = y.$$

Substitution

Proposition. Let R^* be the relation obtained by substitution from total functions $f_1, \ldots, f_m$ and an m-place relation R.

- If R and the f_i are recursive, then R^* is recursive.
- If R and the f_i are primitive recursive, then R^* is primitive recursive.

Proof. See lecture.

Exercise

Let R be a (primitive) recursive two-place relation. Show that the relations below are (primitive) recursive:

1. the **converse** of R, given by $S(x, y)$ iff $R(y, x)$;
2. the **diagonal** of R, given by $D(x)$ iff $R(x, x)$;
3. for any natural number m, the **vertical and horizontal sections** of R, given by

$$R_m(y) \text{ iff } R(m, y),$$
$$R^m(x) \text{ iff } R(x, m).$$