

1 Recursiveness and computability

1.1 Recursive functions

Definition 0.1 (Primitive recursive function).

Proposition 0.1 (Primitive recursive functions).

The following are primitive recursive:

- Addition: $x + 0 = x, x + S(y) = S(x + y)$
- Predecessor: $pd(0) = 0, pd(x + 1) = x$
- Integer difference: $x - 0 = x, x - (y + 1) = pd(x - y)$

Proof. • Define $\begin{cases} f(x, 0) = I_1^1(x) \\ f(x, y + 1) = h(x, y, f(x, y)) \\ h(x, y, z) = S(I_3^3(x, y, z)) \end{cases}$

□

Definition 0.2 (Recursive μ -operator).

content...

Definition 0.3 (Recursive function).

A function f is defined from a relation R by μ -recursion if:

1. R is a regular predicate, i.e. $\forall \vec{x}. \exists y. R(\vec{x}, y)$
2. $f(\vec{x}) = \mu y. R(\vec{x}, y)$

Theorem 1 (Kleene's Normal Form Theorem).

There exist:

- U primitive recursive function
- for each n , primitive recursive predicates T_n

such that for any recursive function $f^{(n)}$ there is a number e for which:

- $\forall \vec{x}. \exists y. T_n(e, \vec{x}, y)$

- $f(\vec{x}) = U(\mu y.T_n(e, \vec{x}, y))$

Proof. Let $f^{(n)}$ be a recursive function. This function can be obtained algorithmically in different forms according to the inductive procedure that we use to generate recursive functions. Once we fix this procedure, we have fixed an algorithmic description of every recursive function. This description is in essence a sequence of elementary steps or computations. The theorem says that for any input \vec{x} , there is a computation giving the value of f at \vec{x} and that the final result can be obtained through primitive recursive operations using at most one minimization. The function U plays the role of an interpreter, extracting the result value from the computation code. Note also that programs are treated as data, i.e. recursive functions are also reduced to numbers via indexes. This is done so that we can define numeric functions U and T_n .

Recursive function coding:

$$O \mapsto \langle 0 \rangle$$

$$S \mapsto \langle 1 \rangle$$

$$I_i^n \mapsto \langle 2, n, i \rangle$$

$$g(h_1(\cdot), \dots, h_m(\cdot)) \mapsto \langle 3, b_1, \dots, b_m, a \rangle$$

$$\text{prim. recursion}(g, h) \mapsto \langle 4, a, b \rangle$$

$$\text{bounded recursion}(g) \mapsto \langle 5, a \rangle$$

□

Corollary 1.1.

The set of recursive functions is countable.

1.2 Recursive sets

Definition 1.1 (Recursive set).

A set is recursive iff its characteristic function is recursive.

2 Basic recursion theory

2.1 Partial recursive functions

Definition 1.2 (Partial function).

A partial function is a function that may be undefined for some arguments. The set of arguments for which it is defined is called its domain.

Definition 1.3 (Partial recursive μ -operator).

A function f is defined from a relation R by μ -recursion if $f(\vec{x}) \simeq \mu y.R(\vec{x}, y)$.

Definition 1.4 (Partial recursive functions).

The class of partial recursive functions is the smallest class of functions

1. Containing the initial functions O, S, I_i^n
2. Closed under composition: $\varphi(\vec{x}) \simeq \psi(\gamma_1(\vec{x}), \dots, \gamma_m(\vec{x}))$
3. Closed under primitive recursion:

$$\varphi(\vec{x}, 0) \simeq \psi(\vec{x}), \varphi(\vec{x}, y+1) = \gamma(\vec{x}, y, \varphi(\vec{x}, y))$$

4. Closed under unrestricted μ -recursion:

$$\varphi(\vec{x}) \simeq \mu y[\forall z \leq y.\psi(\vec{x}, z) \downarrow \wedge \psi(\vec{x}, y) \simeq 0]$$

Theorem 2 (Normal form theorem for partial recursive functions).

There exist:

- U primitive recursive function
- for each n , primitive recursive predicates T_n

such that for any recursive function $\varphi^{(n)}$ there is a number e for which:

- $\varphi(x_1, \dots, x_n) \downarrow \iff \exists y.T_n(e, \vec{x}, y)$
- $\varphi(\vec{x}) \simeq U(\mu y.T_n(e, \vec{x}, y))$

Corollary 2.1 (Recursive as partial recursive).

The recursive functions are exactly the total and partial recursive functions.

Recall the normal form expression $U(\mu y.T_n(e, \vec{x}, y))$. In the total setting, it is shown that for e an index of a total recursive function, this gives a value. In

particular, the minimized predicate must be regular. But what if e does not correspond to any recursive function? The predicate may not be minimizable, i.e. the corresponding search is not terminating. This problem is removed with the particular μ -operator. In that setting, $\varphi_e^n(\vec{x}) \simeq U(\mu y.T_n(e, \vec{x}, y))$ always defines a partial function.

Theorem 3 (Enumeration theorem).

For each e , φ_e^n is a partial recursive function of n variables.

If ψ is a partial recursive function of n variables, then there is e such that $\psi \simeq \varphi_e^n$

There is a partial recursive function $\varphi^{(n+1)}(e, \vec{x}) \simeq \varphi_e^n(\vec{x})$

In summary, φ is a partial recursive function that enumerates all partial recursive functions with its parameter e .

Theorem 4 (Padding lemma).

Let e be an index for a partial recursive function φ .

We can effectively generate infinitely many indices of the same function.

Proof. It would suffice to apply the identity function to the coding of φ or to add and subtract the same quantity. Note that the computation trees appearing in the proof of the normal form theorem have equations as nodes. So the proof is phrased in terms of "adding redundant equations" \square

Theorem 5 (S_n^m /parametrization theorem).

Let $m, n \in \omega$. There is a primitive recursive, injective function S_n^m such that:

$$\varphi_{S_n^m(e, x_1, \dots, x_n)}(y_1, \dots, y_m) \simeq \varphi_e(x_1, \dots, x_n, y_1, \dots, y_m)$$

Proof. We take a partial recursive function coded by a program e and we assume that some of its parameters x_1, \dots, x_n are fixed. The result is some function γ . The theorem says that we can compute a program $S_n^m(e, x_1, \dots, x_n)$ using only primitive recursive means.

The computation of γ can be described by the function coded by e followed by the equation $\gamma(y_1, \dots, y_m) \simeq \psi(C_{x_1}, \dots, C_{x_n}, y_1, \dots, y_m)$ (recall the computation trees). This function is clearly partial recursive and its index depends on the

code e and the fixed constant, call it $S_n^m(e, x_1, \dots, x_n)$. By construction the index function is injective and its arithmetization is clearly primitive recursive. \square

Note that the enumeration and parametrization theorems play an inverse role translating arguments to indices and viceversa. This is the essence of partial evaluation and supercompilation.

2.2 Recursively enumerable sets

We find the set analog of partial recursive function. Since characteristic functions are total we need to find an alternative definition.

Definition 5.1 (Recursively enumerable set).

A set is recursively enumerable iff its the domain of a partial recursive function.

Notation 5.1.

Note W_e^n the domain of φ_e^n and $W_{e,s}^n$ the domain of $\varphi_{e,s}^n$.

Theorem 6 (Normal form theorem for r.e. relations).

An n-ary relation $P^{(n)}$ is r.e. iff one of the following holds:

- For some recursive relation R :

$$P^{(n)} = \{\vec{x}. \exists y. R^{(n+1)}(\vec{x}, y)\}$$

- For some index e :

$$\exists e. P^{(n)} = W_e^n = \{\vec{x}. \exists y. T_n(e, \vec{x}, y)\}$$

Theorem 7 (Numerical characterization of r.e. sets).

The r.e. sets are exactly the diophantine sets, i.e. sets of the form:

$$\{a. \exists \vec{x}. p(a, \vec{x}) = 0\}$$

where p is a polynomial with integer coefficients and $a \in \mathbb{Z}$, $\vec{x} \in \mathbb{Z}^n$. As a consequence there is no *algorithm* solving Hilbert's tenth problem.

TODO: there should be a match between this set and the expression of theorem 6. But they appear to list different kinds of elements.

Theorem 8 (Graph theorem).

A function φ is partial recursive iff its graph is r.e.

A function f recursive iff its graph is recursive.

The theorem confirms that the choice of the *domain* is good for the canonical translation from sets to functions given by characteristic functions and from functions to sets given by graph.

2.3 Indices and enumerations

Definition 8.1 (System of indices).

A system of indices ψ is a family of maps $\psi^n : \omega \rightarrow \mathcal{P}^n$.

Note $\psi_e^n := \psi^n(e)$.

ψ satisfies enumeration if $\forall n. \exists a. \psi_a^{n+1}(e, x_1, \dots, x_n) \simeq \psi_e^n(x_1, \dots, x_n)$.

ψ satisfies parametrization if $\forall m, n. \exists s$ total recursive. $\psi_{s(e, x_1, \dots, x_n)}^m(y_1, \dots, y_m) \simeq \psi_e^{m+n}(x_1, \dots, x_n, y_1, \dots, y_m)$

Note φ the standard system of indices.

ψ is acceptable if $\exists f, g$ total recursive. $\psi_e^n \simeq \varphi_{f(e)}^n, \varphi_e^n \simeq \psi_{g(e)}^n$

Proposition 8.1 (Acceptable system characterization).

A system of indices is acceptable \iff it satisfies enumeration and parametrization.

3 Measures of complexity

References