Mathematics for Informatics

Carlos Areces and Patrick Blackburn

areces@loria.fr blackbur@loria.fr

http://www.loria.fr/~areces http://www.loria.fr/~blacbur

INRIA Lorraine
Nancy, France

2007/2008

Normal form theorem

Theorem

Let $f : \mathbb{N}^n \to \mathbb{N}$ *be a partially computable function. Then there is a p.r. predicate* $R : \mathbb{N}^{n+1} \to \mathbb{N}$ *such that*

$$f(x_1, \ldots, x_n) = l\left(\min_z R(x_1, \ldots, x_n, z)\right)$$

Normal form theorem

Theorem

Let $f : \mathbb{N}^n \to \mathbb{N}$ be a partially computable function. Then there is a p.r. predicate $R : \mathbb{N}^{n+1} \to \mathbb{N}$ such that

$$f(x_1, \ldots, x_n) = l\left(\min_z R(x_1, \ldots, x_n, z)\right)$$

Proof.

Let e be the number for some program for $f(x_1, \ldots, x_n)$.

Normal form theorem

Theorem

Let $f : \mathbb{N}^n \to \mathbb{N}$ be a partially computable function. Then there is a p.r. predicate $R : \mathbb{N}^{n+1} \to \mathbb{N}$ such that

$$f(x_1, \ldots, x_n) = l\left(\min_z R(x_1, \ldots, x_n, z)\right)$$

Proof.

Let e be the number for some program for $f(x_1, \ldots, x_n)$.
Remember that the instant configuration is represented as

$$\langle \text{instruction number}, \text{list representing the state} \rangle$$

Normal form theorem

Theorem

Let $f : \mathbb{N}^n \to \mathbb{N}$ be a partially computable function. Then there is a p.r. predicate $R : \mathbb{N}^{n+1} \to \mathbb{N}$ such that

$$f(x_1, \ldots, x_n) = l\left(\min_z R(x_1, \ldots, x_n, z)\right)$$

Proof.

Let e be the number for some program for $f(x_1, \ldots, x_n)$.
Remember that the instant configuration is represented as

$$\langle \text{instruction number}, \text{list representing the state} \rangle$$

The following predicate $R(x_1, \ldots, x_n, z)$ is what we need:

$$\mathrm{STP}^{(n)}(x_1, \ldots, x_n, e, r(z)) \quad \wedge$$

Normal form theorem

Theorem

Let $f : \mathbb{N}^n \to \mathbb{N}$ be a partially computable function. Then there is a p.r. predicate $R : \mathbb{N}^{n+1} \to \mathbb{N}$ such that

$$f(x_1, \dots, x_n) = l\left(\min_z R(x_1, \dots, x_n, z)\right)$$

Proof.

Let e be the number for some program for $f(x_1, \dots, x_n)$.
Remember that the instant configuration is represented as

$$\langle \text{instruction number}, \text{list representing the state} \rangle$$

The following predicate $R(x_1, \dots, x_n, z)$ is what we need:

$$\begin{gathered} \mathrm{STP}^{(n)}(x_1, \dots, x_n, e, r(z)) \quad \wedge \\ l(z) = \underbrace{\underbrace{r\left(\mathrm{SNAP}^{(n)}(x_1, \dots, x_n, e, r(z))\right)}_{\text{final state of } e \text{ with input } x_1, \dots, x_n}[1]}_{\text{value of the variable } Y \text{ in the final state}} \end{gathered}$$

Another characterization of computable functions

Theorem

A function is partially computable if it can be obtained from initial functions by means of a finite number of applications of

- *composition,*
- *primitive recursion and*
- *minimization*

Another characterization of computable functions

Theorem

A function is partially computable if it can be obtained from initial functions by means of a finite number of applications of

- *composition,*
- *primitive recursion and*
- *minimization*

Theorem

A function is computable if it can be obtain from initial functions by means of a finite number of applications of

- *composition,*
- *primitive recursion and*
- *proper minimization*
  (of the form $\min_t q(x_1, \ldots, x_n, t)$ *where there is always at least one* t *such that* $q(x_1, \ldots, x_n, t)$ *is true)*

Eliminating input variables

Let's consider the program P that uses the input variables X_1 and X_2:

$$\begin{array}{ll} \text{INSTRUCTION 1} & \#(I_1) \\ \vdots & \\ \text{INSTRUCTION k} & \#(I_k) \end{array}$$

Eliminating input variables

Let's consider the program P that uses the input variables X_1 and X_2:

INSTRUCTION 1 $\quad \#(I_1)$

$\vdots$

INSTRUCTION k $\quad \#(I_k)$

It computes the function $f : \mathbb{N}^2 \to \mathbb{N}$

$$f(x, y) = \Psi_P^{(2)}(x, y)$$

$$\#(P) = [\#(I_1), \ldots, \#(I_k)] - 1$$

Eliminating input variables

Let's consider the program P that uses the input variables X_1 and X_2:

INSTRUCTION 1 $\#(I_1)$

$\vdots$

INSTRUCTION k $\#(I_k)$

It computes the function $f : \mathbb{N}^2 \to \mathbb{N}$

$$f(x, y) = \Psi_P^{(2)}(x, y)$$

$$\#(P) = [\#(I_1), \ldots, \#(I_k)] - 1$$

Look for the number of the program P_0 for $f_0 : \mathbb{N} \to \mathbb{N}, f_0(x) = f(x, 0)$

Eliminating input variables

Let's consider the program P that uses the input variables X_1 and X_2:

INSTRUCTION 1 $\quad \#(I_1)$

$\vdots$

INSTRUCTION k $\quad \#(I_k)$

It computes the function $f : \mathbb{N}^2 \to \mathbb{N}$

$$f(x, y) = \Psi_P^{(2)}(x, y)$$

$$\#(P) = [\#(I_1), \ldots, \#(I_k)] - 1$$

Look for the number of the program P_0 for $f_0 : \mathbb{N} \to \mathbb{N}$, $f_0(x) = f(x, 0)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

Eliminating input variables

Let's consider the program P that uses the input variables X_1 and X_2:

INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

It computes the function $f : \mathbb{N}^2 \to \mathbb{N}$

$$f(x,y) = \Psi_P^{(2)}(x,y)$$

$$\#(P) = [\#(I_1), \ldots, \#(I_k)] - 1$$

Look for the number of the program P_0 for $f_0 : \mathbb{N} \to \mathbb{N}$, $f_0(x) = f(x, 0)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

It computes the function $f_0 : \mathbb{N} \to \mathbb{N}$

$$f_0(x) = \Psi_{P_0}^{(1)}(x)$$

$$\#(P_0) = [109, 110, \#(I_1), \ldots, \#(I_k)] - 1$$

Eliminating input variables

Look for the number of the program P_1 for $f_1 : \mathbb{N} \to \mathbb{N}, f_1(x) = f(x, 1)$

Eliminating input variables

Look for the number of the program P_1 for $f_1 : \mathbb{N} \to \mathbb{N}, f_1(x) = f(x, 1)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
$X_2 \leftarrow X_2 + 1$	26
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

Eliminating input variables

Look for the number of the program P_1 for $f_1 : \mathbb{N} \to \mathbb{N}, f_1(x) = f(x, 1)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
$X_2 \leftarrow X_2 + 1$	26
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

It computes the function $f_1 : \mathbb{N} \to \mathbb{N}$

$$f_1(x) = \Psi_{P_1}^{(1)}(x)$$

$$\#(P_1) = [109, 110, 26, \#(I_1), \ldots, \#(I_k)] - 1$$

Eliminating input variables

Look for the number of the program P_1 for $f_1 : \mathbb{N} \to \mathbb{N}, f_1(x) = f(x, 1)$

$[A]\ X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
$X_2 \leftarrow X_2 + 1$	26
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

It computes the function $f_1 : \mathbb{N} \to \mathbb{N}$

$$f_1(x) = \Psi^{(1)}_{P_1}(x)$$

$$\#(P_1) = [109, 110, 26, \#(I_1), \ldots, \#(I_k)] - 1$$

Look for the program P_2 for $f_2 : \mathbb{N} \to \mathbb{N}, f_2(x) = f(x, 2)$

Eliminating input variables

Look for the number of the program P_1 for $f_1 : \mathbb{N} \to \mathbb{N}, f_1(x) = f(x, 1)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
$X_2 \leftarrow X_2 + 1$	26
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

It computes the function $f_1 : \mathbb{N} \to \mathbb{N}$

$$f_1(x) = \Psi^{(1)}_{P_1}(x)$$

$$\#(P_1) = [109, 110, 26, \#(I_1), \ldots, \#(I_k)] - 1$$

Look for the program P_2 for $f_2 : \mathbb{N} \to \mathbb{N}, f_2(x) = f(x, 2)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
$X_2 \leftarrow X_2 + 1$	26
$X_2 \leftarrow X_2 + 1$	26
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

Eliminating input variables

Look for the number of the program P_1 for $f_1 : \mathbb{N} \to \mathbb{N}, f_1(x) = f(x, 1)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
$X_2 \leftarrow X_2 + 1$	26
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

It computes the function $f_1 : \mathbb{N} \to \mathbb{N}$

$$f_1(x) = \Psi_{P_1}^{(1)}(x)$$

$$\#(P_1) = [109, 110, 26, \#(I_1), \ldots, \#(I_k)] - 1$$

Look for the program P_2 for $f_2 : \mathbb{N} \to \mathbb{N}, f_2(x) = f(x, 2)$

[A] $X_2 \leftarrow X_2 - 1$	109
IF $X_2 \neq 0$ GOTO A	110
$X_2 \leftarrow X_2 + 1$	26
$X_2 \leftarrow X_2 + 1$	26
INSTRUCTION 1	$\#(I_1)$
$\vdots$	
INSTRUCTION k	$\#(I_k)$

It computes the function $f_2 : \mathbb{N} \to \mathbb{N}$

$$f_2(x) = \Psi_{P_2}^{(1)}(x)$$

$$\#(P_2) = [109, 110, 26, 26, \#(I_1), \ldots, \#(I_k)] - 1$$

Parameters theorem

There is a program P_i for the function $f_i(x) = f(x, i)$

The transformation $\#(P) \mapsto \#(P_i)$ is p.r., i.e., there is a function $S : \mathbb{N}^2 \to \mathbb{N}$ p.r. such that given $\#(P), x_2$ it computes $\#(P_{x_2})$:

$$S(x_2, y) = \left(2^{109} \cdot 3^{110} \cdot \prod_{j=1}^{x_2} p_{j+2}^{26} \cdot \prod_{j=1}^{|y+1|} p_{j+x_2+2}{}^{(y+1)[j]} \right) - 1$$

Parameters theorem

There is a program P_i for the function $f_i(x) = f(x, i)$

The transformation $\#(P) \mapsto \#(P_i)$ is p.r., i.e., there is a function $S : \mathbb{N}^2 \to \mathbb{N}$ p.r. such that given $\#(P), x_2$ it computes $\#(P_{x_2})$:

$$S(x_2, y) = \left(2^{109} \cdot 3^{110} \cdot \prod_{j=1}^{x_2} p_{j+2}^{26} \cdot \prod_{j=1}^{|y+1|} p_{j+x_2+2}{}^{(y+1)[j]} \right) - 1$$

Theorem

There is a p.r. function $S : \mathbb{N}^2 \to \mathbb{N}$ such that

$$\Phi_y^{(2)}(x_1, x_2) = \Phi_{S(x_2,y)}^{(1)}(x_1)$$

Parameters theorem

There is a program P_i for the function $f_i(x) = f(x, i)$

The transformation $\#(P) \mapsto \#(P_i)$ is p.r., i.e., there is a function $S : \mathbb{N}^2 \to \mathbb{N}$ p.r. such that given $\#(P), x_2$ it computes $\#(P_{x_2})$:

$$S(x_2, y) = \left(2^{109} \cdot 3^{110} \cdot \prod_{j=1}^{x_2} p_{j+2}^{26} \cdot \prod_{j=1}^{|y+1|} p_{j+x_2+2}{}^{(y+1)[j]} \right) - 1$$

Theorem

There is a p.r. function $S : \mathbb{N}^2 \to \mathbb{N}$ such that

$$\Phi_y^{(2)}(x_1, x_2) = \Phi_{S(x_2,y)}^{(1)}(x_1)$$

Theorem

For each $n, m > 0$ there is an inyective p.r. function $S_m^n : \mathbb{N}^{n+1} \to \mathbb{N}$ such that

$$\Phi_y^{(n+m)}(x_1, \dots, x_m, u_1, \dots, u_n) = \Phi_{S_m^n(u_1,\dots,u_n,y)}^{(m)}(x_1, \dots, x_m)$$

Autoreferencing programs

- in the proof of the Halting Problem we built a program P such that, when it is executed with its own program number (i.e., $\#(\mathrm{P})$), it turns into a contradiction.
- in general, we can assume that programs known their own program number

Autoreferencing programs

- in the proof of the Halting Problem we built a program P such that, when it is executed with its own program number (i.e., $\#(\mathrm{P})$), it turns into a contradiction.
- in general, we can assume that programs known their own program number
- but if a program P knows it's own program number, it could, for example, return it's own number, i.e. $\#(P)$

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ *is partially computable, then there is* e *such that*

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is partially computable, then there is e such that

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Proof.

Let S_n^1 be the function in the Parameters theorem:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n).$$

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is partially computable, then there is e such that

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Proof.

Let S_n^1 be the function in the Parameters theorem:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n).$$

The function $g(S_n^1(v, v), x_1, \ldots, x_n)$ is partially computable,

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is partially computable, then there is e such that

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Proof.

Let S_n^1 be the function in the Parameters theorem:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n).$$

The function $g(S_n^1(v, v), x_1, \ldots, x_n)$ is partially computable, hence there is d such that

$$g(S_n^1(v, v), x_1, \ldots, x_n) \quad = \quad \Phi_d^{(n+1)}(x_1, \ldots, x_n, v)$$

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ *is partially computable, then there is e such that*

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Proof.

Let S_n^1 be the function in the Parameters theorem:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n).$$

The function $g(S_n^1(v, v), x_1, \ldots, x_n)$ is partially computable, hence there is d such that

$$\begin{aligned} g(S_n^1(v, v), x_1, \ldots, x_n) &= \Phi_d^{(n+1)}(x_1, \ldots, x_n, v) \\ &= \Phi_{S_n^1(d,v)}^{(n)}(x_1, \ldots, x_n) \end{aligned}$$

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is partially computable, then there is e such that

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Proof.

Let S_n^1 be the function in the Parameters theorem:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n).$$

The function $g(S_n^1(v, v), x_1, \ldots, x_n)$ is partially computable, hence there is d such that

$$\begin{aligned} g(S_n^1(v, v), x_1, \ldots, x_n) &= \Phi_d^{(n+1)}(x_1, \ldots, x_n, v) \\ &= \Phi_{S_n^1(d,v)}^{(n)}(x_1, \ldots, x_n) \end{aligned}$$

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is partially computable, then there is e such that

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Proof.

Let S_n^1 be the function in the Parameters theorem:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n).$$

The function $g(S_n^1(v, v), x_1, \ldots, x_n)$ is partially computable, hence there is d such that

$$\begin{aligned} g(S_n^1(v, v), x_1, \ldots, x_n) &= \Phi_d^{(n+1)}(x_1, \ldots, x_n, v) \\ &= \Phi_{S_n^1(d,v)}^{(n)}(x_1, \ldots, x_n) \end{aligned}$$

d is fixed; v is variable.

Recursion theorem

Theorem

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ *is partially computable, then there is e such that*

$$\Phi_e^{(n)}(x_1, \dots, x_n) = g(e, x_1, \dots, x_n)$$

Proof.

Let S_n^1 be the function in the Parameters theorem:

$$\Phi_y^{(n+1)}(x_1, \dots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \dots, x_n).$$

The function $g(S_n^1(v, v), x_1, \dots, x_n)$ is partially computable, hence there is d such that

$$\begin{aligned} g(S_n^1(v, v), x_1, \dots, x_n) &= \Phi_d^{(n+1)}(x_1, \dots, x_n, v) \\ &= \Phi_{S_n^1(d,v)}^{(n)}(x_1, \dots, x_n) \end{aligned}$$

d is fixed; v is variable. Choose $v = d$ and $e = S_n^1(d, d)$.

Aplications of the Recursion theorem

Corollary

If $g : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$ *is partially computable, then there are* *infinite* e *such that*

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Aplications of the Recursion theorem

Corollary

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ *is partially computable, then there are infinite* e *such that*

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Proof.

In the proof of the previous theorem, there are infinite d such that

$$\Phi_d^{(n+1)} = g(S_n^1(v, v), x_1, \ldots, x_n).$$

$S_n^1(v, v)$ is inyective and hence there are infinite

$$e = S_n^1(d, d).$$

□

Quines

A quine is a program that when executed, it returns as output the same program.

For example:

```
char*f="char*f=%c%s%c;main()
{printf(f,34,f,34,10);}%c";
main(){printf(f,34,f,34,10);}
```

Quines

Is there an e such that $\Phi_e(x) = e$?

Quines

Is there an e such that $\Phi_e(x) = e$?

Yes, the empty program has number 0 and computes the constant function 0, i.e. $\Phi_0(x) = 0$.

Quines

Is there an e such that $\Phi_e(x) = e$?

Yes, the empty program has number 0 and computes the constant function 0, i.e. $\Phi_0(x) = 0$.

Corollary

There are infinite e such that $\Phi_e(x) = e$*.*

Quines

Is there an e such that $\Phi_e(x) = e$?

Yes, the empty program has number 0 and computes the constant function 0, i.e. $\Phi_0(x) = 0$.

Corollary

There are infinite e such that $\Phi_e(x) = e$.

Proof.

Let's consider the function $g : \mathbb{N}^2 \to \mathbb{N}, g(z, x) = z$.

Quines

Is there an e such that $\Phi_e(x) = e$?

Yes, the empty program has number 0 and computes the constant function 0, i.e. $\Phi_0(x) = 0$.

Corollary

There are infinite e such that $\Phi_e(x) = e$.

Proof.

Let's consider the function $g : \mathbb{N}^2 \to \mathbb{N}, g(z, x) = z$.
Applying the Recursion theorem, there are infinite e such that

$$\Phi_e(x) = g(e, x) = e.$$

□

Fixed point theorem

Theorem

If $f : \mathbb{N} \to \mathbb{N}$ *is computable, then there is* e *such that* $\Phi_{f(e)}(x) = \Phi_e(x)$.

Fixed point theorem

Theorem

If $f : \mathbb{N} \to \mathbb{N}$ is computable, then there is e such that $\Phi_{f(e)}(x) = \Phi_e(x)$.

Proof.

Consider the function $g : \mathbb{N}^2 \to \mathbb{N}, g(z, x) = \Phi_{f(z)}(x)$. Applying the Recursion theorem, there is an e such that

$$\Phi_e(x) = g(e, x) = \Phi_{f(e)}(x)$$

$\square$

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable.

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total $\Rightarrow g(e, y) \uparrow$ for any y

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total $\Rightarrow g(e, y) \uparrow$ for any $y \Rightarrow \Phi_e(y) \uparrow$ for any y

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total $\Rightarrow g(e, y) \uparrow$ for any $y \Rightarrow \Phi_e(y) \uparrow$ for any $y \Rightarrow \Phi_e$ is not total

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total $\Rightarrow g(e, y) \uparrow$ for any $y \Rightarrow \Phi_e(y) \uparrow$ for any $y \Rightarrow \Phi_e$ is not total
- Φ_e is not total

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total $\Rightarrow g(e, y) \uparrow$ for any $y \Rightarrow \Phi_e(y) \uparrow$ for any $y \Rightarrow \Phi_e$ is not total
- Φ_e is not total $\Rightarrow g(e, y) = 0$ for any y

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total $\Rightarrow g(e, y) \uparrow$ for any $y \Rightarrow \Phi_e(y) \uparrow$ for any $y \Rightarrow \Phi_e$ is not total
- Φ_e is not total $\Rightarrow g(e, y) = 0$ for any $y \Rightarrow \Phi_e(y) = 0$ for any y

Exercise

Prove that $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} 1 & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Suppose f is computable. We can define

$$[A] \qquad \text{IF } f(X) = 1 \text{ GOTO } A$$

and then

$$g(x, y) = \begin{cases} \uparrow & \Phi_x \text{ is total} \\ 0 & \text{otherwise} \end{cases}$$

is partially computable. By the Recursion theorem, let e be such that $\Phi_e(y) = g(e, y)$.

- Φ_e is total $\Rightarrow g(e, y) \uparrow$ for any $y \Rightarrow \Phi_e(y) \uparrow$ for any $y \Rightarrow \Phi_e$ is not total
- Φ_e is not total $\Rightarrow g(e, y) = 0$ for any $y \Rightarrow \Phi_e(y) = 0$ for any $y \Rightarrow \Phi_e(y) = 0$ is total