Recursion Theory

Joost J. Joosten

Institute for Logic Language and Computation

University of Amsterdam

Plantage Muidergracht 24

1018 TV Amsterdam

Room P 3.26, +31 20 5256095

jjoosten@phil.uu.nl

`www.phil.uu.nl/~jjoosten`

Scaling up TM's

- In 1936 Turing showed how on TM could be input to another TM

Scaling up TM's

- In 1936 Turing showed how on TM could be input to another TM
- Even self reference became an option

Scaling up TM's

- In 1936 Turing showed how on TM could be input to another TM
- Even self reference became an option
- Self reference was central to paradoxes (Russell)

Scaling up TM's

- In 1936 Turing showed how on TM could be input to another TM
- Even self reference became an option
- Self reference was central to paradoxes (Russell)
- But also to subtle and beautiful theorems (Gödel)

Scaling up TM's

- In 1936 Turing showed how on TM could be input to another TM
- Even self reference became an option
- Self reference was central to paradoxes (Russell)
- But also to subtle and beautiful theorems (Gödel)
- How can theories talk about “themselves”?

Scaling up TM's

- In 1936 Turing showed how on TM could be input to another TM
- Even self reference became an option
- Self reference was central to paradoxes (Russell)
- But also to subtle and beautiful theorems (Gödel)
- How can theories talk about "themselves"?
- How can TM's talk about "themselves"?

Scaling up TM's

- In 1936 Turing showed how on TM could be input to another TM
- Even self reference became an option
- Self reference was central to paradoxes (Russell)
- But also to subtle and beautiful theorems (Gödel)
- How can theories talk about "themselves"?
- How can TM's talk about "themselves"?
- Gödel numbers!

Gödel numbers

- We will represent TM's by numbers

Gödel numbers

- We will represent TM's by numbers
- What are TM's?

Gödel numbers

- We will represent TM's by numbers
- What are TM's? : list of instructions over some language

Gödel numbers

- An action consists of $\langle q_i, S, A, q_j \rangle$

Gödel numbers

- An action consists of $\langle q_i, S, A, q_j \rangle$
- We code: $\mathsf{gn}(L) = p_0$

Gödel numbers

- An action consists of $\langle q_i, S, A, q_j \rangle$
- We code: $\mathsf{gn}(L) = p_0$
- We code: $\mathsf{gn}(R) = p_1$

Gödel numbers

- An action consists of $\langle q_i, S, A, q_j \rangle$
- We code: $\mathsf{gn}(L) = p_0$
- We code: $\mathsf{gn}(R) = p_1$
- We code: $\mathsf{gn}(q_i) = p_{2i+2}$
- And likewise for the S_i (see book)

Gödel numbers

- An action consists of $\langle q_i, S, A, q_j \rangle$
- We code: $\mathsf{gn}(L) = p_0$
- We code: $\mathsf{gn}(R) = p_1$
- We code: $\mathsf{gn}(q_i) = p_{2i+2}$
- And likewise for the S_i (see book)
- We code: $\mathsf{gn}(Q) = p_0^{gn(q_i)} \cdot p_1^{gn(S)} \cdot p_2^{gn(A)} \cdot p_3^{gn(q_j)}$

Gödel numbers

- An action consists of $\langle q_i, S, A, q_j \rangle$
- We code: $\mathsf{gn}(L) = p_0$
- We code: $\mathsf{gn}(R) = p_1$
- We code: $\mathsf{gn}(q_i) = p_{2i+2}$
- And likewise for the S_i (see book)
- We code: $\mathsf{gn}(Q) = p_0^{gn(q_i)} \cdot p_1^{gn(S)} \cdot p_2^{gn(A)} \cdot p_3^{gn(q_j)}$
- We code: $\mathsf{gn}(\langle Q_0, Q_1, \ldots, Q_n \rangle) = p_0^{\mathsf{gn}(Q_0)} \cdot \ldots \cdot p_n^{\mathsf{gn}(Q_n)}$

Enumerating TM's

- The e^{th} TM is empty program if e does not code a TM and is P if e is a code $\mathtt{gn}$ of some program P

Enumerating TM's

- The e^{th} TM is empty program if e does not code a TM and is P if e is a code gn of some program P
- Instead of $\varphi_T^{(k)}$ we write $\varphi_e^{(k)}$

Enumerating TM's

- The e^{th} TM is empty program if e does not code a TM and is P if e is a code $\mathtt{gn}$ of some program P
- Instead of $\varphi_T^{(k)}$ we write $\varphi_e^{(k)}$
- We omit k whenever $k = 1$

Enumerating TM's

- The e^{th} TM is empty program if e does not code a TM and is P if e is a code $\mathtt{gn}$ of some program P
- Instead of $\varphi_T^{(k)}$ we write $\varphi_e^{(k)}$
- We omit k whenever $k = 1$
- Enumeration Theorem:

Enumerating TM's

- The e^{th} TM is empty program if e does not code a TM and is P if e is a code gn of some program P
- Instead of $\varphi_T^{(k)}$ we write $\varphi_e^{(k)}$
- We omit k whenever $k = 1$
- Enumeration Theorem:
- There is a p.c. $\varphi_z(x)$ that maps $\langle z, x \rangle$ to the output that the z^{th} TM would have on input x

Enumerating TM's

- The e^{th} TM is empty program if e does not code a TM and is P if e is a code $\mathtt{gn}$ of some program P
- Instead of $\varphi_T^{(k)}$ we write $\varphi_e^{(k)}$
- We omit k whenever $k = 1$
- Enumeration Theorem:
- There is a p.c. $\varphi_z(x)$ that maps $\langle z, x\rangle$ to the output that the z^{th} TM would have on input x
- Proof:

Enumerating TM's

- The e^{th} TM is empty program if e does not code a TM and is P if e is a code $\mathtt{gn}$ of some program P
- Instead of $\varphi_T^{(k)}$ we write $\varphi_e^{(k)}$
- We omit k whenever $k = 1$
- Enumeration Theorem:
- There is a p.c. $\varphi_z(x)$ that maps $\langle z, x \rangle$ to the output that the z^{th} TM would have on input x
- Proof:
- Long live the Church Turing Thesis!

Tm's and codes

- Enumeration Theorem uses the concept of a *Universal Turing Machine*!!!

Tm's and codes

- Enumeration Theorem uses the concept of a *Universal Turing Machine*!!!
- Note that we cannot get a version of the Enumeration Theorem for recursive functions

Tm's and codes

- Enumeration Theorem uses the concept of a *Universal Turing Machine*!!!
- Note that we cannot get a version of the Enumeration Theorem for recursive functions
- Diagonal argument again

Tm's and codes

- Enumeration Theorem uses the concept of a *Universal Turing Machine*!!!
- Note that we cannot get a version of the Enumeration Theorem for recursive functions
- Diagonal argument again
- Another fact about code of programs: Every p.c. function has infinitely many different codes (Padding Lemma)

Tm's and codes

- Looks strange, but useful: The S_n^m-theorem

Tm's and codes

- Looks strange, but useful: The S^m_n-theorem
- If $f(x, y)$ is a p.c. function, for some computable g, $f(x, y) = \varphi_{g(x)}(y)$.

Tm's and codes

- Looks strange, but useful: The S_n^m-theorem
- If $f(x, y)$ is a p.c. function, for some computable g, $f(x, y) = \varphi_{g(x)}(y)$.
- More general: for each $m, n \in \mathbb{N}$, there is a function S_n^m such that

$$\varphi_e^{m+n}(x_1, \ldots, x_m, y_1, \ldots, y_n) = \varphi_{S_n^m(e,x_1,\ldots,x_m)}(y_1, \ldots, y_n)$$

The fixed point theorem

- Kleene's Fixed point theorem (1938)

The fixed point theorem

- Kleene's Fixed point theorem (1938)
- For all computable f, there exists a k such that $\varphi_{f(k)} = \varphi_k$

The fixed point theorem

- Kleene's Fixed point theorem (1938)
- For all computable f, there exists a k such that $\varphi_{f(k)} = \varphi_k$
- Proof: consider the function h such that $\varphi_{h(x)} = \varphi_{\varphi_x(x)}$

The fixed point theorem

- Kleene's Fixed point theorem (1938)
- For all computable f, there exists a k such that $\varphi_{f(k)} = \varphi_k$
- Proof: consider the function h such that $\varphi_{h(x)} = \varphi_{\varphi_x(x)}$ (again this diagonal!)

The fixed point theorem

- Kleene's Fixed point theorem (1938)
- For all computable f, there exists a k such that $\varphi_{f(k)} = \varphi_k$
- Proof: consider the function h such that $\varphi_{h(x)} = \varphi_{\varphi_x(x)}$ (again this diagonal!)
- Note: we do not say $h(x) = \varphi_x(x)$

The fixed point theorem

- Kleene's Fixed point theorem (1938)
- For all computable f, there exists a k such that $\varphi_{f(k)} = \varphi_k$
- Proof: consider the function h such that $\varphi_{h(x)} = \varphi_{\varphi_x(x)}$ (again this diagonal!)
- Note: we do not say $h(x) = \varphi_x(x)$
- $f \circ h$ has code e, that is, $(f \circ h)(x) = \varphi_e(x)$

The fixed point theorem

- Kleene's Fixed point theorem (1938)
- For all computable f, there exists a k such that $\varphi_{f(k)} = \varphi_k$
- Proof: consider the function h such that $\varphi_{h(x)} = \varphi_{\varphi_x(x)}$ (again this diagonal!)
- Note: we do not say $h(x) = \varphi_x(x)$
- $f \circ h$ has code e, that is, $(f \circ h)(x) = \varphi_e(x)$
- So, we can take k to be $h(e)$

The fixed point theorem

- Kleene's Fixed point theorem (1938)
- For all computable f, there exists a k such that $\varphi_{f(k)} = \varphi_k$
- Proof: consider the function h such that $\varphi_{h(x)} = \varphi_{\varphi_x(x)}$ (again this diagonal!)
- Note: we do not say $h(x) = \varphi_x(x)$
- $f \circ h$ has code e, that is, $(f \circ h)(x) = \varphi_e(x)$
- So, we can take k to be $h(e)$ (here we use that h should be total!)

Fixed points

- Note: the diagonal construction $\varphi_x(x)$ allows us to view the very same number both as the input of a program and as a program

Fixed points

- Note: the diagonal construction $\varphi_x(x)$ allows us to view the very same number both as the input of a program and as a program
- This allows some sort of self reference!

Fixed points

- Note: the diagonal construction $\varphi_x(x)$ allows us to view the very same number both as the input of a program and as a program
- This allows some sort of self reference!
- Example: there is a program that only halts on its own input

Fixed points

- Note: the diagonal construction $\varphi_x(x)$ allows us to view the very same number both as the input of a program and as a program
- This allows some sort of self reference!
- Example: there is a program that only halts on its own input
- Proof: consider the function f that maps x to the code of a TM that halts if the input equals x and loops otherwise.

Fixed points

- Note: the diagonal construction $\varphi_x(x)$ allows us to view the very same number both as the input of a program and as a program
- This allows some sort of self reference!
- Example: there is a program that only halts on its own input
- Proof: consider the function f that maps x to the code of a TM that halts if the input equals x and loops otherwise.
- Next, apply the FP THM to f.

Fixed points

- Note: the diagonal construction $\varphi_x(x)$ allows us to view the very same number both as the input of a program and as a program
- This allows some sort of self reference!
- Example: there is a program that only halts on its own input
- Proof: consider the function f that maps x to the code of a TM that halts if the input equals x and loops otherwise.
- Next, apply the FP THM to f.

Recap

- Coding: representing programs by numbers

Recap

- Coding: representing programs by numbers
- Enumeration Theorem/Universal Turing Machine

Recap

- Coding: representing programs by numbers
- Enumeration Theorem/Universal Turing Machine
- Padding Lemma
- S_n^m-theorem

Recap

- Coding: representing programs by numbers
- Enumeration Theorem/Universal Turing Machine
- Padding Lemma
- S_n^m-theorem
- Fixed point theorem

Recap

- Coding: representing programs by numbers
- Enumeration Theorem/Universal Turing Machine
- Padding Lemma
- S_n^m-theorem
- Fixed point theorem