Lessons touched by this meeting according to schedule:

* 20. 16/12/2024
  + Recursive functionals (recursive operators, in the book)
  + Myhill-Shepherdson Theorem
  + First recursion theorem [§10.1, §10.2, §10.3, without proofs]
* 21. 17/12/2024
  + Immagine che contiene testo, schermata, Carattere, algebra

    Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere

    Descrizione generata automaticamenteSecond recursion theorem [§11.1, §11.2]

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteMyhill-Shepherdson Theorem:

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamente

The First Recursion Theorem is an important result in computability theory that is used in several key ways in the exercises provided:

1. Proving the existence of fixed points for recursive functionals. The theorem states that every recursive functional has a least fixed point that is computable. This is applied in examples like showing that the Ackermann functional has the Ackermann function as its least fixed point.
2. Classifying sets as recursively enumerable or not. The theorem can be used to show that certain sets, like the set A={x | φ\_x(y)=x^2 for infinitely many y}, are likely recursively enumerable but not recursive. The proof sketch leverages the First Recursion Theorem.
3. Proving undecidability results. As a corollary, the theorem allows proving Rice's Theorem, which states that any non-trivial property of computable functions is undecidable. This in turn is used to prove the undecidability of the Halting Problem.
4. Showing the non-extensionality of certain sets. The theorem is applied to prove that the Halting Set K={x | φ\_x(x)↓} is not saturated (i.e. extensional). The diagonalization argument in the proof relies on the recursion theorem.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteThe Second Recursion Theorem, also known as Kleene's Fixed Point Theorem, states that for any total computable function f : N → N, there exists a fixed point e ∈ N such that φ\_e = φ\_f(e). In other words, there is a program e that, when executed, behaves exactly like the program obtained by applying f to e.

This theorem is used in the course to:

1. Show the non-extensionality of certain sets. For example, the Halting Set K = {x | φ\_x(x)↓} can be proven to be non-extensional using the Second Recursion Theorem. The proof involves constructing a program that behaves differently when given its own index as input.
2. Demonstrate the existence of self-referential programs. The Second Recursion Theorem allows us to create programs that can access their own source code or index during execution. This is a powerful technique for constructing counterexamples and proving the limitations of computable functions.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteLet’s use these proofs as exercises:

Immagine che contiene testo, ricevuta, Carattere, bianco

Descrizione generata automaticamenteConsider some real examples (exercises):

Immagine che contiene testo, Carattere, calligrafia, bianco

Descrizione generata automaticamente

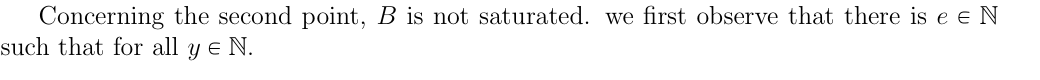


Immagine che contiene testo, schermata, Carattere, algebra

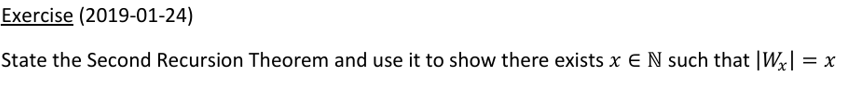
Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteLet’s jump to some more exercises:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, calligrafia, Carattere, schermata

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Define:

g(x,y) = {

y if y < x

↑ otherwise

}

By the s-m-n theorem, there exists a total computable function s : N → N such that:

φs(x)(y) = g(x,y) for all x,y ∈ N

By the Second Recursion Theorem, there exists e ∈ N such that: φe = φs(e)

Therefore:

φe(y) = φs(e)(y) = g(e,y) = {

y if y < e

↑ otherwise

}

This means:

We = dom(φe) = {y ∈ N | y < e}

Therefore:

|We| = |{y ∈ N | y < e}| = e

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteLet’s jump to some recursiveness exercises:

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamente

Immagine che contiene testo, ricevuta, Carattere, schermata

Descrizione generata automaticamente

For 8.1:

A is not r.e.: Consider the identity function id ∉ A since dom(id) = N is infinite.

However, we can find a finite subfunction θ ⊆ id defined as:

θ(x) = {

0 if x = 0

1 if x = 1

↑ otherwise

}

Clearly θ ∈ A since |dom(θ)| = 2. By Rice-Shapiro theorem, A is not r.e.

For 8.2:

For A = {x ∈ N : x ∈ Wx ∩ Ex}, we first prove A is not recursive by showing K ≤m A.

Let's define:

g(x,y) = {

y if x ∈ K

↑ otherwise

}

g is computable since g(x,y) = y · sc\_K(x).

By the s-m-n theorem, there exists s: N → N total computable such that φs(x)(y) = g(x,y).

We claim s is a reduction function K ≤m A. Indeed:

* If x ∈ K then φs(x)(y) = y for all y, so s(x) ∈ Ws(x) and s(x) ∈ Es(x), thus s(x) ∈ A
* If x ∉ K then φs(x)(y) ↑ for all y, so s(x) ∉ Ws(x), thus s(x) ∉ A

Additionally, A is r.e. since:

scA(x) = 1(μw.H(x,x,(w)1) ∧ S(x,(w)1,x,(w)2))

is computable.

Therefore A is r.e. but not recursive, which implies Ā is not r.e.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente