Lessons touched by this meeting according to schedule:

* 12. 25/11/2024
  + Exercises
* 13. 26/11/2024
  + Recursive sets. Reduction. [§7.1, see also §6.1 and §9.1]

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

In other words, a set is recursive if there exists an algorithm (computable function) that can decide membership in the set - given any x ∈ ℕ, it can determine in a finite number of steps whether x belongs to A or not.

The notion of recursiveness has several important implications:

1. Decidability: The membership problem "x ∈ A?" for a recursive set A is decidable. An algorithm exists that always terminates and correctly answers yes or no.
2. Closure properties: The class of recursive sets is closed under complement, union and intersection. If A and B are recursive, then so are A̅, A ∪ B and A ∩ B.
3. Immagine che contiene testo, Carattere, schermata, algebra

   Descrizione generata automaticamenteSimple sets: All finite sets and some easily describable infinite sets like ℕ itself are recursive. The set of prime numbers is also recursive.

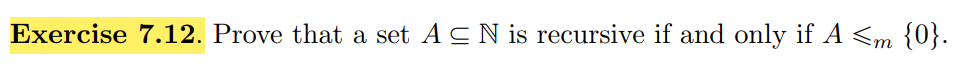
Another important implication:

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamenteReductions: If A ≤\_m B (A many-one reduces to B) and B is recursive, then A is also recursive. Conversely, if A is not recursive and A ≤\_m B, then B is not recursive either. This allows proving non-recursiveness of sets.

Immagine che contiene testo, schermata, Carattere, documento

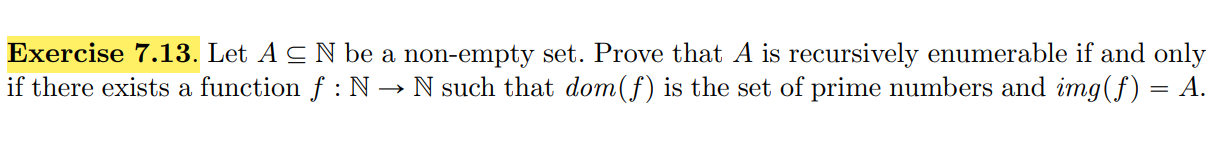
Descrizione generata automaticamenteConsider an example from the lesson:

Let’s jump immediately to related exercises:

To prove that a set A ⊆ ℕ is recursive if and only if A ≤\_m {0}, we will show both implications.

(⇒) Assume A is recursive. Then its characteristic function χ\_A is computable. Define the reduction function f : ℕ → ℕ as f(x) = 1 - χ\_A(x). Clearly, f is computable (composition of computable functions). Now, x ∈ A ⟺ χ\_A(x) = 1 ⟺ f(x) = 0 ⟺ f(x) ∈ {0}. Thus, A ≤\_m {0} via f.

(⇐) Assume A ≤\_m {0} via a computable function f. Then x ∈ A ⟺ f(x) ∈ {0} ⟺ f(x) = 0. So we can write χ\_A(x) = sg(f(x)), which is computable. Hence, A is recursive.

Therefore, A ⊆ ℕ is recursive if and only if A ≤\_m {0}.

Let ∅ ≠ A ⊆ ℕ and suppose A is recursively enumerable. We want to prove that there exists a function f : ℕ → ℕ such that dom(f) is the set of prime numbers and img(f) = A.

Since A is r.e., there exists e ∈ ℕ such that A = W\_e = dom(φ\_e). Let p\_i denote the i-th prime number. Define f as:

f(x) = φ\_e(μi. x = p\_i)

In other words, f(x) = φ\_e(i) where i is the index such that x is the i-th prime number.

We claim that f satisfies the required conditions:

1. dom(f) = {p\_i | i ∈ ℕ}, the set of all prime numbers, by definition of f.
2. img(f) = A.
   * If y ∈ A, then y = φ\_e(i) for some i. Let x = p\_i. Then f(x) = f(p\_i) = φ\_e(i) = y. So y ∈ img(f).
   * If y ∈ img(f), then y = f(x) for some prime x = p\_i. Thus, y = f(p\_i) = φ\_e(i) ∈ A.

Conversely, if A is not r.e., such an f cannot exist, because img(f) is always r.e. (direct image of r.e. set under computable function is r.e.). Thus, the claim holds.

From one last year exam:

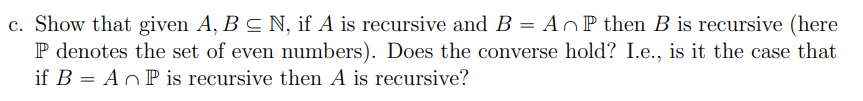


Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, calligrafia, bianco

Descrizione generata automaticamenteLet’s jump to other *exercises* of various kinds:

Proof:

Let f : ℕ → ℕ be a computable function. We define the functions g and h as follows:

g(x) = f(x) if x ∉ K

0 if x ∈ K

h(x) = x (identity function)

Here, K is the halting set, i.e., K = {x ∈ ℕ | φ\_x(x) ↓}.

First, let's verify that f = g ∘ h:

(g ∘ h)(x) = g(h(x))

= g(x)

= f(x) if x ∉ K

0 if x ∈ K

= f(x) for all x, since f is total

Now, we prove that g and h are not computable.

h is clearly computable, as it is the identity function.

Assume, for the sake of contradiction, that g is computable. Then we could use g to decide the halting set K as follows:

For any x ∈ ℕ, compute g(x) and f(x).

If g(x) = f(x), then x ∉ K.

If g(x) = 0 and f(x) ≠ 0, then x ∈ K.

This procedure would give us a computable characteristic function for K:

χ\_K(x) = 1 if g(x) = 0 and f(x) ≠ 0

0 if g(x) = f(x)

However, we know that K is not recursive (as proven using the recursion theorem in a previous example). Therefore, our assumption that g is computable must be false.

Thus, g is not computable, and f is the composition of the non-computable function g and the computable function h.

Since f was arbitrary, this proves that every computable function is the composition of two non-computable functions.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

The key relationship between decidability/semi-decidability and recursive/recursively enumerable sets can be expressed through characteristic functions and predicates (important for later reasoning)

A set A ⊆ N is recursive (decidable) if and only if its characteristic function χ\_A is computable:

A set A ⊆ N is recursively enumerable (r.e.) or semi-decidable if and only if its semi-characteristic function sc\_A is computable:

sc\_A(x) = {

1 if x ∈ A

↑ if x ∉ A

}

The Structure Theorem for semi-decidable predicates states that P(x⃗) is semi-decidable if and only if there exists a decidable predicate Q(t,x⃗) such that:

P(x⃗) ≡ ∃t.Q(t,x⃗)

This is crucial because it:

1. Characterizes semi-decidable predicates in terms of decidable ones via existential quantification

2. Shows that semi-decidable predicates can be expressed as projections of decidable predicates

3. Leads to the Projection Theorem which states that if P(x,y⃗) is semi-decidable, then ∃x.P(x,y⃗) is also semi-decidable

These theorems provide powerful tools for:

- Proving predicates are semi-decidable by expressing them in terms of decidable predicates

- Showing closure properties of semi-decidable predicates under existential quantification

- Understanding the relationship between decidability and semi-decidability

- Constructing new semi-decidable predicates from existing ones

The theorems also help explain why semi-decidable predicates are not closed under complementation and universal quantification, which is key for understanding undecidability results.

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteExample of usage of such notions:

Immagine che contiene testo, Carattere, schermata, linea

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

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

Descrizione generata automaticamenteComing back to other stuff:

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, linea

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

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

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

Descrizione generata automaticamente

Let's tackle this exercise step by step, as requested.

1. Define reduction for A ≤\_m B:

We say that A ≤\_m B (A is many-one reducible to B) if there exists a computable function f : ℕ → ℕ such that for all x ∈ ℕ, x ∈ A ⟺ f(x) ∈ B.

2. Is it true that if A is recursive and B is finite, not empty then A ≤\_m B?

No, this is not true in general. Here's a counterexample:

Let A = ℕ\{0} (the natural numbers without 0) and B = {0}.

A is recursive (its characteristic function is computable) and B is finite, not empty.

But A ≰\_m B. If there were a reduction f : ℕ → ℕ with x ∈ A ⟺ f(x) ∈ B, then f(x) would have to be 0 for all x ∈ A. But 0 ∉ A, so there can't be such an f.

3. Without finiteness hypothesis for B:

If we don't assume B to be finite, then the statement "if A is recursive and B is not empty then A ≤\_m B" does hold.

Here's why:

Let b be any element of B (which exists as B is not empty). Define f : ℕ → ℕ as follows:

f(x) = b if x ∈ A

some fixed element not in B if x ∉ A

Since A is recursive, f is computable. And x ∈ A ⟺ f(x) = b ∈ B. So f is a reduction from A to B.

4. Condition which allows to restore the property when B infinite:

The property "if A is recursive then A ≤\_m B" holds whenever B is not empty and A ≠ ℕ\{0}.

Indeed, if A = ∅, the constant function f(x) = b ∈ B is a reduction.

If A ≠ ∅ and A ≠ ℕ\{0}, choose a ∈ A and a' ∉ A. Then the function

f(x) = b if x ∈ A

b' if x ∉ A

where b ∈ B and b' ∉ B is a computable reduction from A to B.

So in summary, the reduction property fails only in the specific case where A = ℕ\{0} and B is a singleton. In all other cases where A is recursive and B is not empty, we can find a reduction from A to B.