Enumerability

Lecture 32
Section 11.1

Robb T. Koether

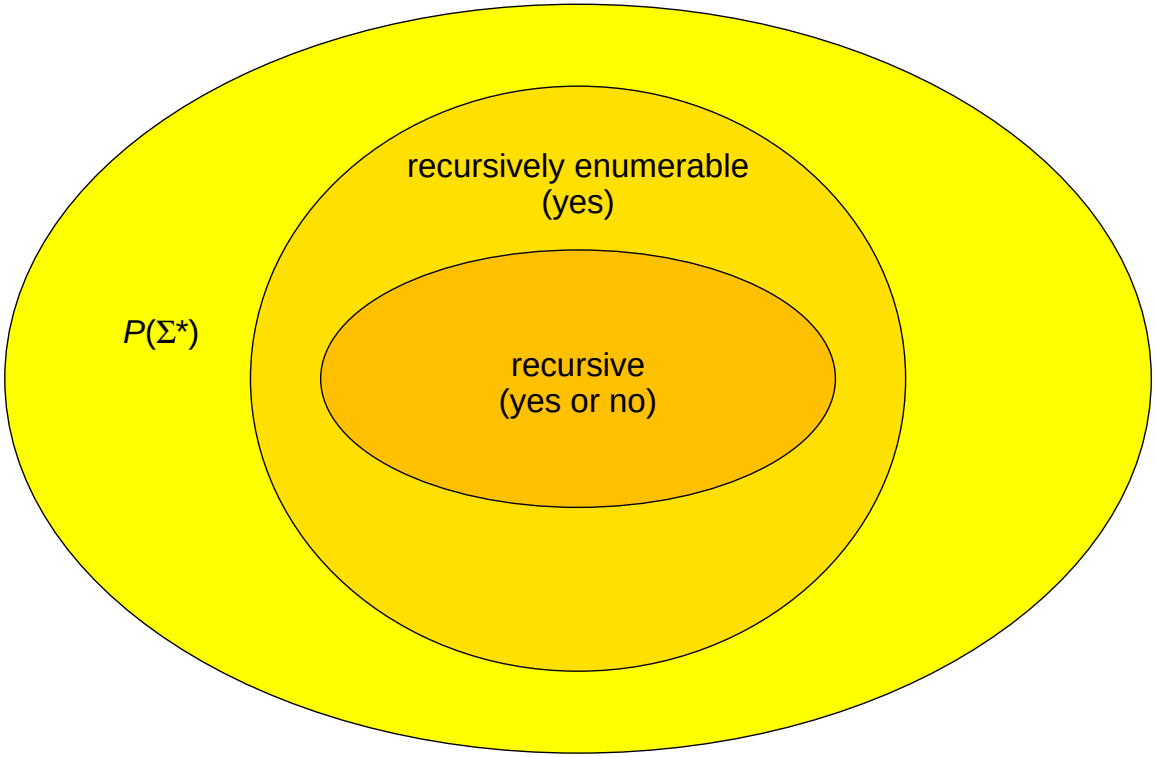
Hampden-Sydney College

Fri, Nov 11, 2016

Outline

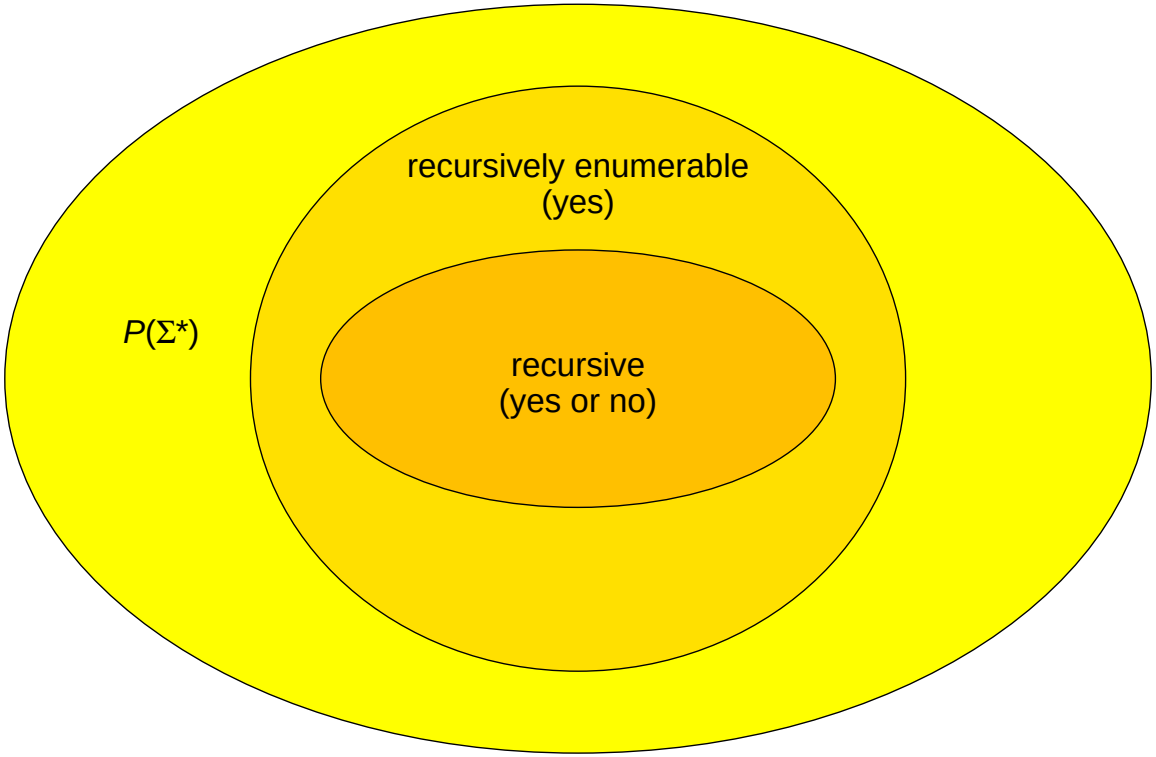
The Language Hierarchy

- Recall the language hierarchy.



The Language Hierarchy

- Recall the language hierarchy.



- Where do regular and context-free languages fit in this drawing?

A Non-Recursively Enumerable Language

- We will construct a recursively enumerable language L whose complement $\overline{L}$ is not recursively enumerable.

A Non-Recursively Enumerable Language

- We will construct a recursively enumerable language L whose complement $\overline{L}$ is not recursively enumerable.
- Let $\Sigma = \{\mathbf{a}\}$.

A Non-Recursively Enumerable Language

- We will construct a recursively enumerable language L whose complement $\overline{L}$ is not recursively enumerable.
- Let $\Sigma = \{\mathbf{a}\}$.
- Then $\Sigma^* = \{\lambda, \mathbf{a}, \mathbf{a}^2, \mathbf{a}^3, \ldots\}$.

A Non-Recursively Enumerable Language

- We will construct a recursively enumerable language L whose complement $\overline{L}$ is not recursively enumerable.
- Let $\Sigma = \{\mathbf{a}\}$.
- Then $\Sigma^* = \{\lambda, \mathbf{a}, \mathbf{a}^2, \mathbf{a}^3, \ldots\}$.
- Let $M_1, M_2, M_3, \ldots$ be an enumeration of all Turing machines over Σ.

A Non-Recursively Enumerable Language

- We will construct a recursively enumerable language L whose complement $\overline{L}$ is not recursively enumerable.
- Let $\Sigma = \{\mathbf{a}\}$.
- Then $\Sigma^* = \{\lambda, \mathbf{a}, \mathbf{a}^2, \mathbf{a}^3, \ldots\}$.
- Let $M_1, M_2, M_3, \ldots$ be an enumeration of all Turing machines over Σ.
- Consider the strings $\mathbf{a}^i$, for $i = 1, 2, 3, \ldots$.

A Non-Recursively Enumerable Language

- We will construct a recursively enumerable language L whose complement $\overline{L}$ is not recursively enumerable.
- Let $\Sigma = \{\mathbf{a}\}$.
- Then $\Sigma^* = \{\lambda, \mathbf{a}, \mathbf{a}^2, \mathbf{a}^3, \ldots\}$.
- Let $M_1, M_2, M_3, \ldots$ be an enumeration of all Turing machines over Σ.
- Consider the strings $\mathbf{a}^i$, for $i = 1, 2, 3, \ldots$.
- Define a language L as

$$L = \{\mathbf{a}^i \mid \mathbf{a}^i \in L(M_i)\}.$$

A Non-Recursively Enumerable Language

- Then

$$\overline{L} = \{\mathbf{a}^i \mid \mathbf{a}^i \notin L(M_i)\}.$$

A Non-Recursively Enumerable Language

- Then
$$\overline{L} = \{\mathbf{a}^i \mid \mathbf{a}^i \notin L(M_i)\}.$$
- Suppose that $\overline{L}$ is recursively enumerable.

A Non-Recursively Enumerable Language

- Then
$$\overline{L} = \{\mathbf{a}^i \mid \mathbf{a}^i \notin L(M_i)\}.$$
- Suppose that $\overline{L}$ is recursively enumerable.
- Then $\overline{L}$ is accepted by some Turing machine M_k.

A Non-Recursively Enumerable Language

- Then
$$\overline{L} = \{\mathbf{a}^i \mid \mathbf{a}^i \notin L(M_i)\}.$$
- Suppose that $\overline{L}$ is recursively enumerable.
- Then $\overline{L}$ is accepted by some Turing machine M_k.
- Consider the string $\mathbf{a}^k$.

A Non-Recursively Enumerable Language

- Then
$$\overline{L} = \{\mathbf{a}^i \mid \mathbf{a}^i \notin L(M_i)\}.$$
- Suppose that $\overline{L}$ is recursively enumerable.
- Then $\overline{L}$ is accepted by some Turing machine M_k.
- Consider the string $\mathbf{a}^k$.
- Either $\mathbf{a}^k \in \overline{L}$ or $\mathbf{a}^k \notin \overline{L}$.

A Non-Recursively Enumerable Language

- Suppose that $\mathbf{a}^k \in \overline{L}$.

A Non-Recursively Enumerable Language

- Suppose that $\mathbf{a}^k \in \overline{L}$.
- Then $\mathbf{a}^k$ is accepted by M_k. That is, $\mathbf{a}^k \in L(M_k)$.

A Non-Recursively Enumerable Language

- Suppose that $\mathbf{a}^k \in \overline{L}$.
- Then $\mathbf{a}^k$ is accepted by M_k. That is, $\mathbf{a}^k \in L(M_k)$.
- But that implies that $\mathbf{a}^k \in L$, which implies that $\mathbf{a}^k \notin \overline{L}$, a contradiction.

A Non-Recursively Enumerable Language

- Suppose that $\mathbf{a}^k \in \overline{L}$.
- Then $\mathbf{a}^k$ is accepted by M_k. That is, $\mathbf{a}^k \in L(M_k)$.
- But that implies that $\mathbf{a}^k \in L$, which implies that $\mathbf{a}^k \notin \overline{L}$, a contradiction.
- So it must be the case that $\mathbf{a}^k \notin \overline{L}$.

A Non-Recursively Enumerable Language

- Suppose that $\mathbf{a}^k \in \overline{L}$.
- Then $\mathbf{a}^k$ is accepted by M_k. That is, $\mathbf{a}^k \in L(M_k)$.
- But that implies that $\mathbf{a}^k \in L$, which implies that $\mathbf{a}^k \notin \overline{L}$, a contradiction.
- So it must be the case that $\mathbf{a}^k \notin \overline{L}$.
- Then $\mathbf{a}^k \in L$.

A Non-Recursively Enumerable Language

- Suppose that $\mathbf{a}^k \in \overline{L}$.
- Then $\mathbf{a}^k$ is accepted by M_k. That is, $\mathbf{a}^k \in L(M_k)$.
- But that implies that $\mathbf{a}^k \in L$, which implies that $\mathbf{a}^k \notin \overline{L}$, a contradiction.
- So it must be the case that $\mathbf{a}^k \notin \overline{L}$.
- Then $\mathbf{a}^k \in L$.
- This implies that $\mathbf{a}^k \in L(M_k)$, which implies that $\mathbf{a}^k \in \overline{L}$, again a contradiction.

A Non-Recursively Enumerable Language

- Suppose that $\mathbf{a}^k \in \overline{L}$.
- Then $\mathbf{a}^k$ is accepted by M_k. That is, $\mathbf{a}^k \in L(M_k)$.
- But that implies that $\mathbf{a}^k \in L$, which implies that $\mathbf{a}^k \notin \overline{L}$, a contradiction.
- So it must be the case that $\mathbf{a}^k \notin \overline{L}$.
- Then $\mathbf{a}^k \in L$.
- This implies that $\mathbf{a}^k \in L(M_k)$, which implies that $\mathbf{a}^k \in \overline{L}$, again a contradiction.
- The conclusion must be that $\overline{L}$ is not accepted by any Turing machine, i.e., $\overline{L}$ is not recursively enumerable.

A Non-Recursively Enumerable Language

- Is L recursively enumerable?

A Non-Recursively Enumerable Language

- Is L recursively enumerable?
- Yes, but the argument requires a bit of hand-waving.

A Non-Recursively Enumerable Language

- Is L recursively enumerable?
- Yes, but the argument requires a bit of hand-waving.
- We build a Turing machine M that does the following.

A Non-Recursively Enumerable Language

- Is L recursively enumerable?
- Yes, but the argument requires a bit of hand-waving.
- We build a Turing machine M that does the following.
  - It reads $\mathbf{a}^i$.

A Non-Recursively Enumerable Language

- Is L recursively enumerable?
- Yes, but the argument requires a bit of hand-waving.
- We build a Turing machine M that does the following.
  - It reads $\mathbf{a}^i$.
  - It sends the integer i to a "converter" that uses i to build a representation of the Turing machine M_i.

A Non-Recursively Enumerable Language

- Is L recursively enumerable?
- Yes, but the argument requires a bit of hand-waving.
- We build a Turing machine M that does the following.
  - It reads $\mathbf{a}^i$.
  - It sends the integer i to a "converter" that uses i to build a representation of the Turing machine M_i.
  - Both M_i and $\mathbf{a}^i$ are fed into the universal Turing machine U.

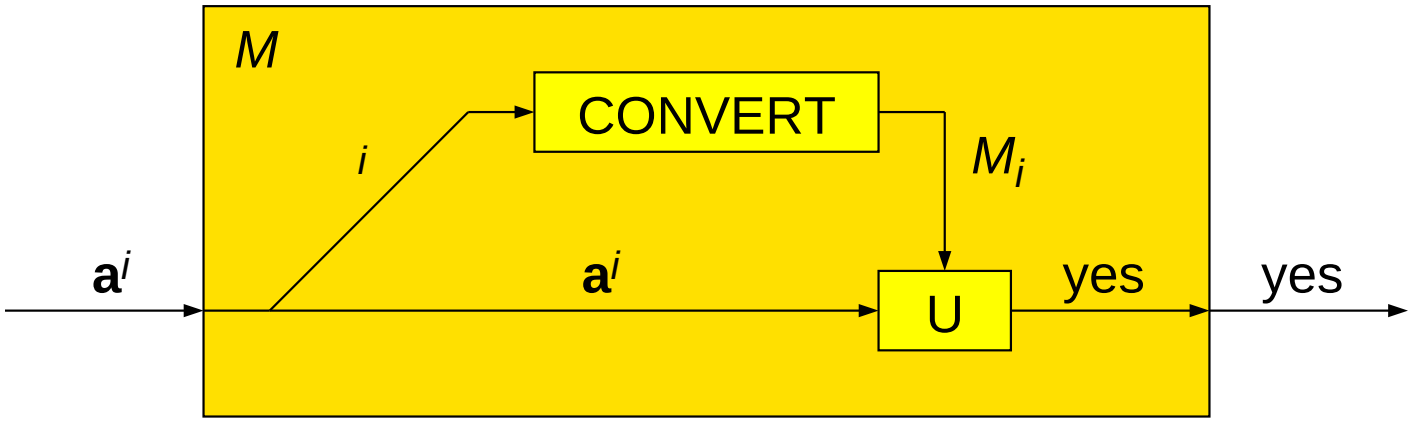
A Non-Recursively Enumerable Language

- Is L recursively enumerable?
- Yes, but the argument requires a bit of hand-waving.
- We build a Turing machine M that does the following.
  - It reads $\mathbf{a}^i$.
  - It sends the integer i to a "converter" that uses i to build a representation of the Turing machine M_i.
  - Both M_i and $\mathbf{a}^i$ are fed into the universal Turing machine U.
  - The result, if any, is reported.

A Non-Recursively Enumerable Language

- Is L recursively enumerable?
- Yes, but the argument requires a bit of hand-waving.
- We build a Turing machine M that does the following.
  - It reads $\mathbf{a}^i$.
  - It sends the integer i to a "converter" that uses i to build a representation of the Turing machine M_i.
  - Both M_i and $\mathbf{a}^i$ are fed into the universal Turing machine U.
  - The result, if any, is reported.
- If $\mathbf{a}^i \in L(M_i)$, i.e., if $\mathbf{a}^i \in L$, then U, and therefore M, will accept $(M_i, \mathbf{a}^i)$.

A Non-Recursively Enumerable Language



Outline

A Recursively Enumerable, but Non-Recursive, Language

Theorem

If a language is and its complement are both recursively enumerable, then they are both recursive.

A Recursively Enumerable, but Non-Recursive, Language

Proof.

- Let L be a recursively enumerable language whose complement $\overline{L}$ is also recursively enumerable.

A Recursively Enumerable, but Non-Recursive, Language

Proof.

- Let L be a recursively enumerable language whose complement $\overline{L}$ is also recursively enumerable.
- Then L is accepted by some Turing machine M_1 and $\overline{L}$ is accepted by some Turing machine M_2.

A Recursively Enumerable, but Non-Recursive, Language

Proof.

- Let L be a recursively enumerable language whose complement $\overline{L}$ is also recursively enumerable.
- Then L is accepted by some Turing machine M_1 and $\overline{L}$ is accepted by some Turing machine M_2.
- Then we build a Turing machine M that does the following.

□

A Recursively Enumerable, but Non-Recursive, Language

Proof.

- Let L be a recursively enumerable language whose complement $\overline{L}$ is also recursively enumerable.
- Then L is accepted by some Turing machine M_1 and $\overline{L}$ is accepted by some Turing machine M_2.
- Then we build a Turing machine M that does the following.
  - Read a word w.

A Recursively Enumerable, but Non-Recursive, Language

Proof.

- Let L be a recursively enumerable language whose complement $\overline{L}$ is also recursively enumerable.
- Then L is accepted by some Turing machine M_1 and $\overline{L}$ is accepted by some Turing machine M_2.
- Then we build a Turing machine M that does the following.
  - Read a word w.
  - Send copies of w to M_1 and M_2.

A Recursively Enumerable, but Non-Recursive, Language

Proof.

- Let L be a recursively enumerable language whose complement $\overline{L}$ is also recursively enumerable.
- Then L is accepted by some Turing machine M_1 and $\overline{L}$ is accepted by some Turing machine M_2.
- Then we build a Turing machine M that does the following.
  - Read a word w.
  - Send copies of w to M_1 and M_2.
  - If M_1 accepts w, report acceptance.

□

A Recursively Enumerable, but Non-Recursive, Language

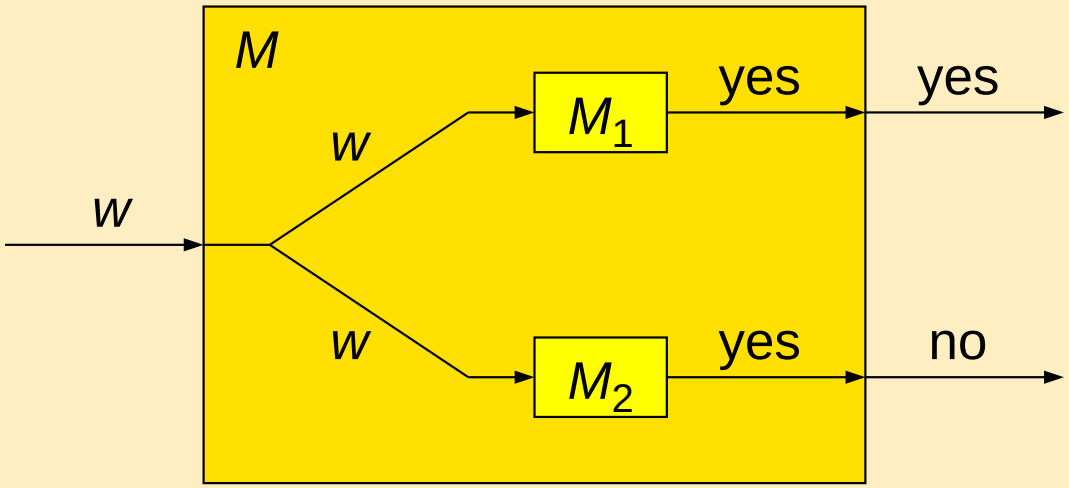
Proof.

- Let L be a recursively enumerable language whose complement $\overline{L}$ is also recursively enumerable.
- Then L is accepted by some Turing machine M_1 and $\overline{L}$ is accepted by some Turing machine M_2.
- Then we build a Turing machine M that does the following.
  - Read a word w.
  - Send copies of w to M_1 and M_2.
  - If M_1 accepts w, report acceptance.
  - If M_2 accepts w, report rejection.

□

A Recursively Enumerable, but Non-Recursive, Language

Proof.



A Recursively Enumerable, but Non-Recursive, Language

Proof.

- The Turing machine M decides L.
- By reversing the outputs "yes" and "no," we get a Turing machine that decides $\overline{L}$.
- Therefore, L and $\overline{L}$ are recursive.

□

Outline

Enumerability

Theorem

A language L is enumerable in canonical order if and only if L is recursive.

Enumerability

Theorem

A language L is enumerable (in some order) if and only if L is recursively enumerable.

Corollary

If L is recursively enumerable, but not recursive, then L is enumerable, but L cannot be enumerated in any standard order.

Outline

Collected

Collected on Monday, Nov. 11

- Section 10.3 Exercises 2a, 3.
- Section 10.4 Exercises 5, 8.
- Section 11.1 Exercise 3.

Outline

Assignment

Homework

- Section 11.1 Exercises 14, 16, 17, 19.