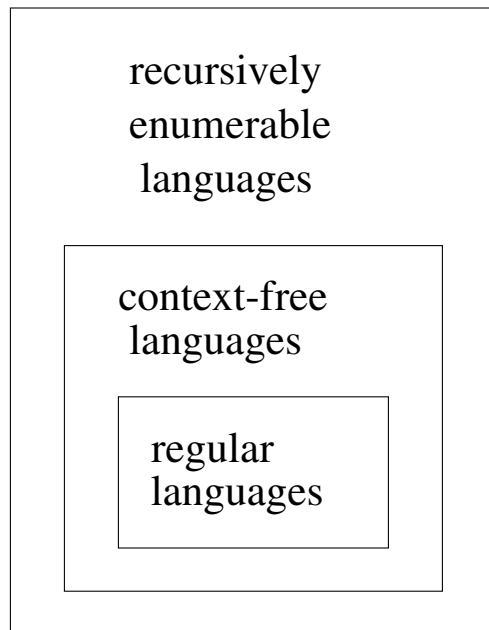


Section: Recursively Enumerable Languages

Definition: A language L is *recursively enumerable* if there exists a TM M such that $L=L(M)$.



Definition: A language L is *recursive* if there exists a TM M such that $L=L(M)$ and M halts on every $w \in \Sigma^+$.

To enumerate all $w \in \Sigma^+$ in a recursive language L :

- Let M be a TM that recognizes L ,
 $L = L(M)$.

- Construct 2-tape TM M'

Tape 1 will enumerate the strings
in Σ^+

Tape 2 will enumerate the strings
in L .

- On tape 1 generate the next
string v in Σ^+
- simulate M on v
if M accepts v , then write v on
tape 2.

To enumerate all $w \in \Sigma^+$ in a recursively enumerable language L :

Repeat forever

- Generate next string (Suppose k strings have been generated:

w_1, w_2, \dots, w_k)

- Run M for one step on w_k

Run M for two steps on w_{k-1} .

...

Run M for k steps on w_1 .

If any of the strings are accepted then write them to tape 2.

Theorem Let S be an infinite countable set. Its powerset 2^S is not countable.

Proof - Diagonalization

- **S** is countable, so it's elements can be enumerated.

$$\mathbf{S} = \{s_1, s_2, s_3, s_4, s_5, s_6 \dots\}$$

Example, $\{s_2, s_3, s_5\}$ represented by

Example, set containing every other element from **S, starting with s_1 is $\{s_1, s_3, s_5, s_7, \dots\}$ represented by**

Suppose 2^S countable. Then we can enumerate all its elements: t_1, t_2, \dots

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	\dots
t_1	0	1	0	1	0	0	1	\dots
t_2	1	1	0	0	1	1	0	\dots
t_3	0	0	0	0	1	0	0	\dots
t_4	1	0	1	0	1	1	0	\dots
t_5	1	1	1	1	1	1	1	\dots
t_6	1	0	0	1	0	0	1	\dots
t_7	0	1	0	1	0	0	0	\dots
\dots								

Theorem For any nonempty Σ , there exist languages that are not recursively enumerable.

Proof:

- A language is a subset of Σ^* .

The set of all languages over Σ is

Theorem There exists a recursively enumerable language L such that \bar{L} is not recursively enumerable.

Proof:

- Let $\Sigma = \{a\}$

Enumerate all TM's over Σ :

	a	aa	aaa	aaaa	aaaaa	...
$L(M_1)$	0	1	1	0	1	...
$L(M_2)$	1	0	1	0	1	...
$L(M_3)$	0	0	1	1	0	...
$L(M_4)$	1	1	0	1	1	...
$L(M_5)$	0	0	0	1	0	...
...						

Theorem If languages L and \bar{L} are both RE, then L is recursive.

Proof:

- $\exists M_1$ s.t. M_1 can enumerate all elements in L .

$\exists M_2$ s.t. M_2 can enumerate all elements in \bar{L} .

To determine if a string w is in L or not in L

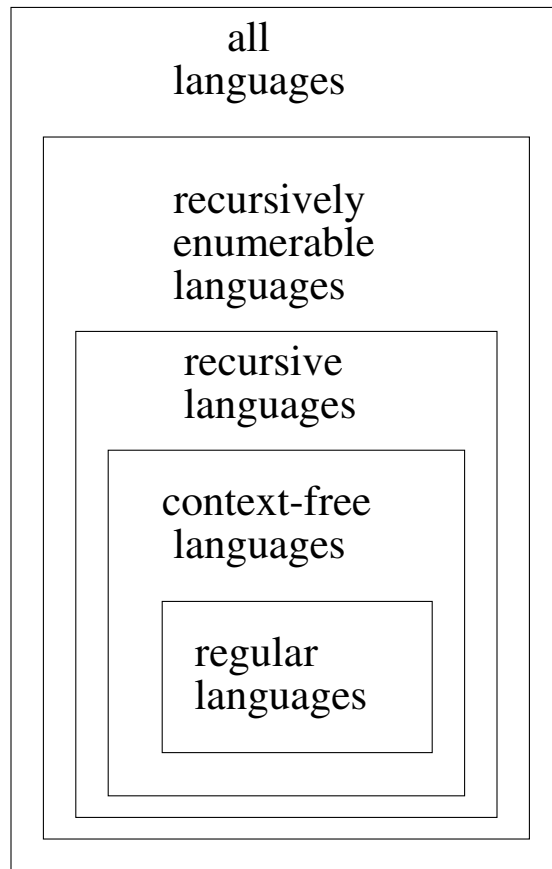
Theorem: If L is recursive, then \bar{L} is recursive.

Proof:

- L is recursive, then there exists a TM M such that M can determine if w is in L or w is not in L .

Construct TM M' that does the following. M' first simulates TM M .

Hierarchy of Languages:



Definition A grammar $G=(V,T,S,P)$ is *unrestricted* if all productions are of the form

$$u \rightarrow v$$

where $u \in (V \cup T)^+$ and $v \in (V \cup T)^*$

Example:

Let $G=(\{S,A,X\},\{a,b\},S,P)$, $P=$

$$S \rightarrow bAaaX$$

$$bAa \rightarrow abA$$

$$AX \rightarrow \lambda$$

Example Find an unrestricted grammar G s.t. $L(G) = \{a^n b^n c^n \mid n > 0\}$

$G = (V, T, S, P)$

$V = \{S, A, B, D, E, X\}$

$T = \{a, b, c\}$

$P =$

- 1) $S \rightarrow AX$**
- 2) $A \rightarrow aAbc$**
- 3) $A \rightarrow aBbc$**
- 4) $Bb \rightarrow bB$**
- 5) $Bc \rightarrow D$**
- 6) $Dc \rightarrow cD$**
- 7) $Db \rightarrow bD$**
- 8) $DX \rightarrow EXc$**

$S \Rightarrow AX \Rightarrow aAbcX \Rightarrow aaAbcbcbX \Rightarrow$
 $aaaBbcbcbcbX$

Theorem If G is an unrestricted grammar, then $L(G)$ is recursively enumerable.

Proof:

- List all strings that can be derived in one step.

List all strings that can be derived in two steps.

Theorem If L is recursively enumerable, then there exists an unrestricted grammar G such that $L=L(G)$.

Proof:

- L is recursively enumerable.

\Rightarrow there exists a TM M such that $L(M)=L$.

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$q_0w \vdash^* x_1q_fx_2$ for some $q_f \in F$,
 $x_1, x_2 \in \Gamma^*$

Construct an unrestricted grammar G s.t. $L(G)=L(M)$.

$S \xRightarrow{*} w$

Three steps

1. $S \xRightarrow{*} B \dots B \# xq_fyB \dots B$

2. $B \dots B \# xq_fyB \dots B \xRightarrow{*}$
 $B \dots B \# q_0wB \dots B$

3. $B \dots B \# q_0wB \dots B \xRightarrow{*} w$

Definition A grammar G is *context-sensitive* if all productions are of the form

$$x \rightarrow y$$

where $x, y \in (V \cup T)^+$ and $|x| \leq |y|$

Definition L is context-sensitive (CSL) if there exists a context-sensitive grammar G such that $L=L(G)$ or $L=L(G) \cup \{\lambda\}$.

Theorem For every CSL L not including λ , \exists an LBA M s.t. $L=L(M)$.

Theorem If L is accepted by an LBA M , then \exists CSG G s.t. $L(M)=L(G)$.

Theorem Every context-sensitive language L is recursive.

Theorem There exists a recursive language that is not CSL.