Recursive languages

- A language $L \subseteq \Sigma^*$ is *recursively enumerable* if $L = L(M)$, for some Turing machine M.

$$w \longrightarrow \boxed{\text{M}} \longrightarrow \begin{cases} \text{yes,} & \text{if } w \in L \\ \text{no,} & \text{if } w \notin L \\ M \text{ does not halt,} & \text{if } w \notin L \end{cases}$$

- A language $L \subseteq \Sigma^*$ is *recursive* if $L = L(M)$ for some Turing machine M that halts on all inputs $w \in \Sigma^*$.

$$w \longrightarrow \boxed{\text{M}} \longrightarrow \begin{cases} \text{yes,} & \text{if } w \in L \\ \text{no,} & \text{if } w \notin L \end{cases}$$

- **Lemma.** L is recursive iff both L and $\overline{L} = \Sigma^* \setminus L$ are recursively enumerable.

Enumerating languages

- An *enumerator* is a Turing machine M with extra output tape T, where symbols, once written, are never changed.
- M writes to T words from Σ^*, separated by \$.
- Let $G(M) = \{w \in \Sigma^* \mid w \text{ is written to } T\}$.

Some results

- **Lemma.** For any finite alphabet Σ, there exists a Turing machine that generates the words $w \in \Sigma^*$ in *canonical ordering* (i.e., $w \prec w' \Leftrightarrow |w| < |w|$ or $|w| = |w|$ and $w \prec_{lex} w'$).
- **Lemma.** There exists a Turing machine that generates all pairs of natural numbers (in binary encoding).

  Proof: Use the ordering $(0,0)$, $(1,0)$, $(0,1)$, $(2,0)$, $(1,1)$, $(0,2)$, ...
- **Proposition.** L is recursively enumerable iff $L = G(M)$, for some Turing machine M.

Computing functions

- Unary encoding of natural numbers: $i \in \mathbb{N} \mapsto \underbrace{||\ldots|}_{i \text{ times}} = |^i$

  (binary encoding would also be possible)
- *M computes* $f : \mathbb{N}^k \to \mathbb{N}$ with $f(i_1, \ldots, i_k) = m$:
  - Start: $|^{i_1} 0 |^{i_2} 0 \ldots |^{i_k}$
  - End: $|^m$
- *f partially recursive:*

$$i_1, \ldots, i_k \longrightarrow \boxed{\text{M}} \longrightarrow \begin{cases} \text{halts with } f(i_1, \ldots, i_k) = m, \\ \text{does not halt, i.e., } f \text{ undefined.} \end{cases}$$

- *f recursive:*

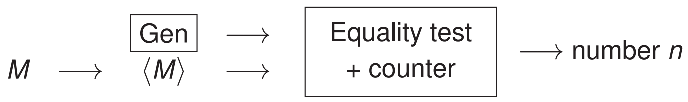
$$i_1, \ldots, i_k \longrightarrow \boxed{\text{M}} \longrightarrow \text{halts with } f(i_1, \ldots, i_k) = m.$$

Turing machines codes

- May assume
$$M = (Q, \{0,1\}, \{0,1,\#\}, \delta, q_1, \#, \{q_2\})$$
- Unary encoding
$$0 \mapsto 0, 1 \mapsto 00, \# \mapsto 000, L \mapsto 0, R \mapsto 00$$
- $\delta(q_i, X) = (q_j, Y, R)$ encoded by
$$0^i 1 \underbrace{0 \ldots 0}_{X} 1 0^j 1 \underbrace{0 \ldots 0}_{Y} 1 \underbrace{0 \ldots 0}_{R}$$
- δ encoded by
$$111\, \texttt{code}_1\, 11\, \texttt{code}_2\, 11 \ldots 11\, \texttt{code}_r\, 111$$
- Encoding of Turing machine M denoted by $\langle M \rangle$.

Numbering of Turing machines

- **Lemma.** There exists a Turing machine that generates the natural numbers in binary encoding.
- **Lemma.** There exists a Turing machine *Gen* that generates the binary encodings of all Turing machines.
- **Proposition.** The language of Turing machine codes is recursive.
- **Corollary.** There exist a bijection between the set of natural numbers, Turing machine codes and Turing machines.



Diagonalization

- Let w_i be the i-th word in $\{0,1\}^*$ and M_j the j-th Turing machine.
- Table T with $t_{ij} = \begin{cases} 1, & \text{if } w_i \in L(M_j) \\ 0, & \text{if } w_i \notin L(M_j) \end{cases}$

$j \longrightarrow$ / $i \downarrow$

	1	2	3	4	...
1	0	1	1	0	...
2	1	1	0	1	...
3	0	0	1	0	...
⋮	⋮	⋮	⋮	⋮	

- *Diagonal language* $L_d = \{w_i \in \{0,1\}^* \mid w_i \notin L(M_i)\}$.
- **Theorem.** L_d is not recursively enumerable.
- *Proof:* Suppose $L_d = L(M_k)$, for some $k \in \mathbb{N}$. Then
$$w_k \in L_d \Leftrightarrow w_k \notin L(M_k),$$
contradicting $L_d = L(M_k)$.