

Definition of Context Free Grammars

Pumping Lemma for Regular Sets

Pumping Lemma for Context Free Languages

Definition of Pushdown Automata

Theorem Let $A \subseteq \Sigma^*$ be any language. Then the following are equivalent:

1. $A = \mathcal{L}(G)$, for some CFG G .
2. $A = \mathcal{L}(P)$, for some PDA P .
3. A is a context-free language.

$$M = (Q, \Sigma, \delta, s)$$

Q : finite set of states; $s \in Q$

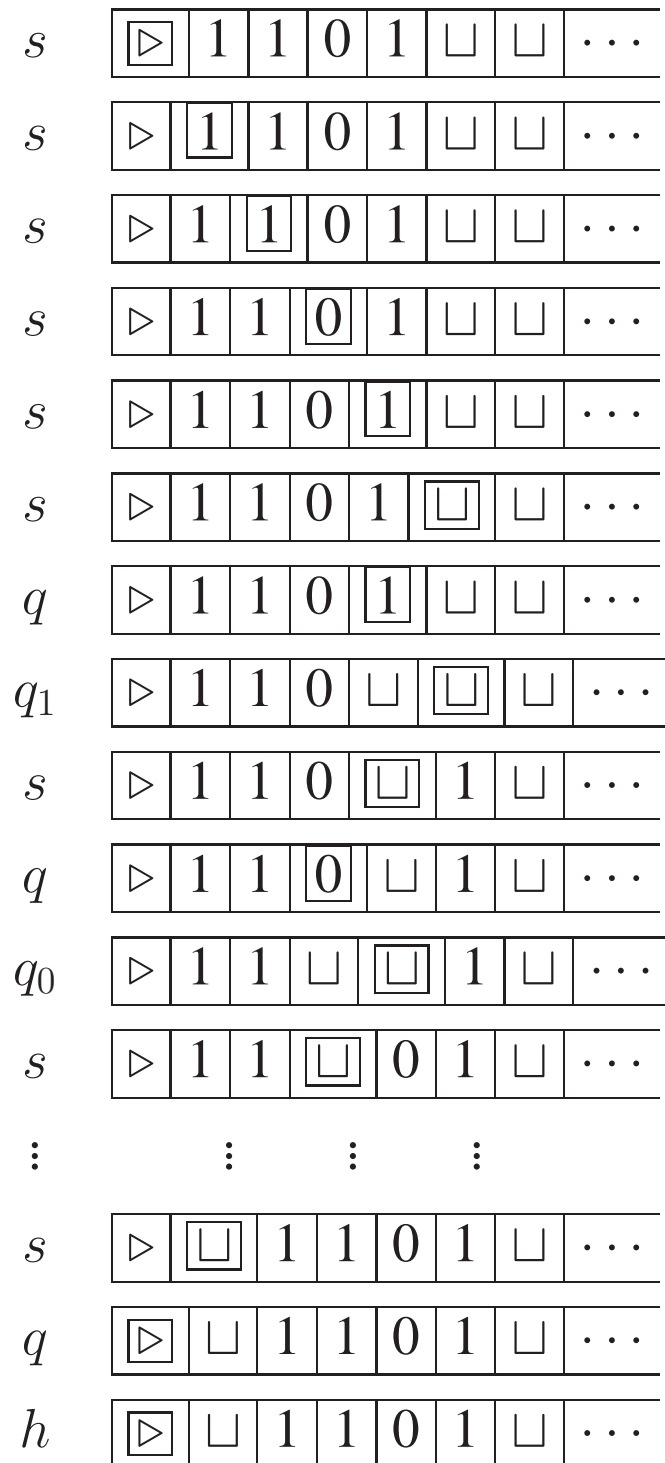
Σ : finite tape alphabet; $\triangleright, \sqcup \in \Sigma$

$\delta: Q \times \Sigma \rightarrow (Q \cup \{h\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$

s	\triangleright	1	1	0	1	\sqcup	\sqcup	\dots
-----	------------------	---	---	---	---	----------	----------	---------

A Turing machine is a DFA plus unlimited memory

mvRt.tm	s	q	q_0	q_1
0	$s, 0, \rightarrow$	q_0, \sqcup, \rightarrow		
1	$s, 1, \rightarrow$	q_1, \sqcup, \rightarrow		
\sqcup	q, \sqcup, \leftarrow		$s, 0, \leftarrow$	$s, 1, \leftarrow$
\triangleright	$s, \triangleright, \rightarrow$	$h, \triangleright, -$		
comment	find \sqcup	memorize & erase	change \sqcup to 0	change \sqcup to 1



Hilbert's Program [1901]: Give a complete axiomization of all of mathematics!

Such a complete axiomization would have provided a mechanical procedure to churn out exactly all true statements in mathematics.

This led to active interest in 1930's in the question: **“What is a mechanical procedure?”**

Church: Lambda calculus	Gödel: Recursive function	Kleene: Formal system
Markov: Markov algorithm	Post: Post machine	Turing: Turing machine

Fact: The above models are all exactly equivalent.

(And they are also equivalent to what is computable by any appropriate formal model of a real computer that has added to it a potentially unbounded amount of secondary storage.)

Church's Thesis: The intuitive idea of “effectively computable” is captured by the precise definition of computable by any of the above models.

Why is the Turing machine as powerful as any other computational model?

Intuitive answer: Imagine any computational device. It has:

- Finitely many states
- Ability to scan limited amount per step: one page at a time
- Ability to print limited amount per step: one page at a time
- Next state determined by current state and page currently being read

Note: Without the potentially infinite supply of tape cells, paper, extra disks, extra tapes, etc. we have just a (potentially huge) **finite state machine**.

The PC on your desk, with 20 GB of hard disk is a finite state machine with over $2^{160,000,000,000}$ states!

This is better modeled as a TM with a bounded number of states, and a potentially infinite tape.

$$M(w) \equiv \begin{cases} y & \text{if } M \text{ on input } \text{“}\triangleright w \sqcup\text{”} \text{ eventually} \\ & \text{halts with output } \text{“}\triangleright y \sqcup\text{”} \\ \nearrow & \text{otherwise} \end{cases}$$

$$\Sigma_0 \equiv \Sigma - \{\triangleright, \sqcup\}; \quad \text{Usually, } \Sigma_0 = \{0, 1\}; \quad w, y \in \Sigma_0^*$$

Definition 4.1 Let $f : \Sigma_0^* \rightarrow \Sigma_0^*$ be a total or partial function. We say that f is a **partial, recursive function** iff \exists TM $M(f = M(\cdot))$, i.e., $\forall w \in \Sigma_0^*(f(w) = M(w))$. \square

Remark 4.2 *There is an easy to compute 1:1 and onto map between $\{0, 1\}^*$ and \mathbf{N} [Hw2]. Thus we can think of the contents of a TM tape as a natural number and talk about $f : \mathbf{N} \rightarrow \mathbf{N}$ being a **recursive function**.*

If the partial, recursive function f is total, i.e., $f : \mathbf{N} \rightarrow \mathbf{N}$ then we say that f is a **total, recursive function**. A partial function that is not total is called **strictly partial**.

Proposition 4.3 *The following functions are recursive. They are all total except for p_{even} .*

$$\text{copy}(w) = ww$$

$$\sigma(n) = n + 1$$

$$\text{plus}(n, m) = n + m$$

$$\text{mult}(n, m) = n \times m$$

$$\text{exp}(n, m) = n^m \quad (\text{we let } \text{exp}(0, 0) = 1)$$

$$\chi_{\text{even}}(n) = \begin{cases} 1 & \text{if } n \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$

$$p_{\text{even}}(n) = \begin{cases} 1 & \text{if } n \text{ is even} \\ \nearrow & \text{otherwise} \end{cases}$$

Proof: Exercise: please convince yourself that you can build TMs to compute all of these functions! □

Definition 4.4 Let $S \subseteq \Sigma_0^*$ or $S \subseteq \mathbf{N}$.

S is a **recursive set** iff the function χ_S is a (total) recursive function,

$$\chi_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases}$$

Examples: The following sets are recursive:

- $\{2n \mid n \in \mathbf{N}\}, \{2n + 1 \mid n \in \mathbf{N}\}$
- $\{p \in \mathbf{N} \mid p \text{ is prime}\}$
- \mathbf{N}, \emptyset
- 0^* , in fact, every regular set is recursive.
- $\{0^n 1^n \mid n \in \mathbf{N}\}$, in fact, every CFL is recursive.
- $\{0^{n^2} \mid n \in \mathbf{N}\}$

S is a **recursive set** iff the function χ_S is a (total) recursive function,

$$\chi_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases} \text{ the characteristic function of } S$$

Definition 4.5 S is a **recursively enumerable set** (S is **r.e.**) iff the function p_S is a (partial) recursive function,

$$p_S(x) = \begin{cases} 1 & \text{if } x \in S \\ \nearrow & \text{otherwise} \end{cases} \text{ the partial characteristic function of } S$$

Proposition 4.6 *If S is recursive then S is r.e.*

Proof: Suppose S is recursive and let M be the TM computing χ_S . Build M' computing the following function:

$$M'(x) = \begin{cases} 1 & \text{if } M(x) = 1 \\ \nearrow & \text{otherwise} \end{cases} = p_S(x)$$

□

\mathcal{C} a class of sets, define $\text{co-}\mathcal{C}$ the class of sets whose complements are in \mathcal{C} ,

$$\text{co-}\mathcal{C} = \{S \mid \bar{S} \in \mathcal{C}\}$$

Theorem 4.7 *S is recursive iff S and \bar{S} are both r.e.*

*Thus, **Recursive** = **r.e.** \cap **co-r.e.***

Proof: If $S \in \mathbf{Recursive}$ then χ_S is a recursive function.

Thus so is $\chi_{\bar{S}}(x) = 1 - \chi_S(x)$

Thus, S and \bar{S} are both recursive and thus both r.e.

Suppose $S \in \mathbf{r.e.} \cap \mathbf{co-r.e.}$

$$p_S = M(\cdot); \quad p_{\overline{S}} = M'(\cdot).$$

Run M and M' in parallel – sometimes called **dovetailing**:

Define $T = M \parallel M'$ on input x :

1. **for** $n := 1$ to ∞ {
2. run $M(x)$ for n steps.
3. **if** $M(x) = 1$ in n steps **then return**(1)
4. run $M'(x)$ for n steps.
5. **if** $M'(x) = 1$ in n steps **then return**(0)}

Thus, $T(\cdot) = \chi_S$ and thus $S \in \mathbf{Recursive}$. □

