COMS 331: Theory of Computing, Fall 2021
Semester Notes

1 Alphabets and Strings

Def: an alphabet is a finite set of objects that we regard as symbols
ex: 0,1, a,b

Def: a string is over an alphabet
ex: 1101 is a string over 0,1

Notation: Σ^* is the set of strings over Σ
ex: $\{0,1\}^*$ is the set of binary strings
the length of a string $x \in \{0,1\}^*$ is the total number of occurrences of symbols in x
ex: $|1101| = 4$

Question: What is $\emptyset^*$?
Answer: $\emptyset^* = \{\lambda\}$
Intelligent first answer: $\emptyset^* = \emptyset$ (wrong)
$\Sigma^* = \{a_1, a_2, \cdots, a_n | n \geqslant 0 \text{ and each } a_i \in \Sigma\}$
Further cautions: $\{a, b\} = \{b, a\}$, but $ab \neq ba$ (unless $a = b$)
$\{a, a, b\} = \{a, b\}$, but $aab \neq ab$

Def: The concatenation of two strings $x, y \in \Sigma^*$ is the string xy
ex: the concatenation of 1101 and 001 is 1101001
- not an invertable operation, can't ask which strings were concatenated

Properties of Concatenation:
- Associative: $x(yz) = (xy)z$
- Identity: $\lambda x = x\lambda$
- Additivity of length: $|xy| = |x| + |y|$
- Not commutative

Notation: for $x \in \Sigma^*$ and $a \in \Sigma$
$\#(a, x) = \#_a(x) =$ the number of a's in x
Def: If x and y are strings, then x is a prefix of y ($x \sqsubseteq y$),
if there exists $z \in \Sigma^*$ such that $xz = y$

Remarks: For all $x \in \Sigma^*$, $\lambda \sqsubseteq x$ (because you can take $z = x$), and
$x \sqsubseteq x$ (because can take $z = \lambda$)

Def: x is a proper prefix of $y \Rightarrow x \not\sqsubseteq y$, if $x \sqsubseteq y$ and $x \neq y$

Notation: the standard enumeration of $\{0,1\}^*$ is
$\lambda, 0, 1, 00, 01, 11, 000, 001, \cdots$
$s_0, s_1, s_2, \cdots$
The natural number represented by a string $x \in \{0,1\}^*$ is the number $bnum(x)$ defined by the following recursion:
$bnum(\lambda) = 0$
$bnum(x0) = 2bnum(x)$
$bnum(x1) = 2bnum(x) + 1$

2 Sets

Set Operations: If A and B are sets, then
$A \cup B = \{x | x \in A \text{ or } x \in B\}$
$A \cap B = \{x | x \in A \text{ and } x \in B\}$
$A - B = \{x | x \in A \text{ and } x \notin B\}$
$A \subseteq B =$ every element of A is an element of B
$A \not\subseteq B = A$ is a proper subset of $B \Rightarrow A \subseteq B$ and $A \neq B$

Def: A language over an alphabet Σ is a set of $A \subseteq \Sigma^*$
ex: The set
PRIMES $= \{x \in \{0,1\}^* | x$ is the binary representation of a prime number$\}$
is a language over $\{0,1\}$

Def: The concatenation of two languages $A, B \subseteq \Sigma^*$ is the language
$AB = \{xy | x \in A \text{ and } y \in B\}$
Note: If A and B are finite, then $|AB| \leqslant |A| * |B|$,
has to be $\leqslant$ because there could be repeats.

Def: A prefix set is a language $A \subseteq \{0,1\}^*$ with the property that no element of A is a prefix of another element of A.
Kraft's inequality proves that this is true.

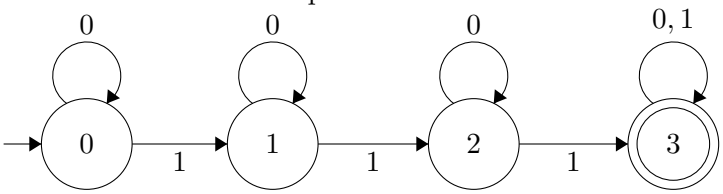
3 Finite Automata and Regular Sets

Def: A deterministic finite automata (or DFA) is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$ where
· Q is a finite set whose elements are states
· Σ is an alphabet, called the input alphabet
· $s \in Q$ is the start state
· $F \subseteq Q$ is the set of accepting states
· $\delta : Q \times E \to Q$ is the transition function
The set of strings $L(M)$ accepted by M is a language over Σ

Example: Let $M = (Q, \Sigma, \delta, s, F)$ where $Q = \{0, 1, 2, 3\}$, $\Sigma = \{0, 1\}$, $F = \{3\}$ and δ is given by the following table:

$\delta(q, a)$	0	1
0	0	1
1	1	2
2	2	3
3	3	3

The DFA can also be represented as:



M accepts the language $L(M) = \{x \in \{0, 1\}^* | \#(1, x) \geqslant 3\}$

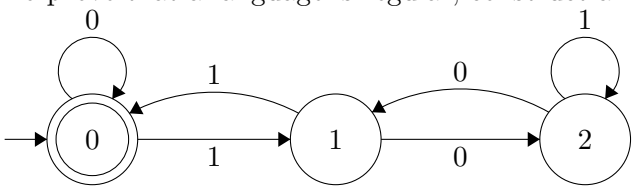
Def: The extended transition function $\hat{\delta} : Q \times \Sigma^* \to Q$ intuitively defines a path for an arbitrary string x from $q \in Q$ to $\hat{\delta}(q, x)$.
Formally, $\hat{\delta}$ is defined recursively as follows:
· $\hat{\delta}(q, \lambda) = q$
· For $x \in \Sigma^*$ and $a \in \Sigma$, $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

Def: Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA, and let $x \in \Sigma^*$,
1. M accepts x if $\hat{\delta}(s, x) \in F$
2. M rejects x if $\hat{\delta}(s, x) \notin F$
3. $L(M) = \{x \in \Sigma^* | M \text{ accepts } x\}$

4 Regular Languages

Def: A language $A \subseteq \Sigma^*$ is regular if there is a DFA M such that $L(M) = A$.

Example: Prove that the language $A = \{x \in \{0,1\}^* |\ 3|bnum(x)\}$
(i.e. num divisible by 3) is regular.
To prove that a language is regular, construct a DFA for it, as seen below:



Product Construction: Let $M = (Q_A \times Q_B, \Sigma, \delta, s, F)$ where
$\delta : (Q_A \times Q_B) \times \Sigma \to Q_A \times Q_B$ is defined by
$\delta((q,r),a) = (\delta_A(q,a), \delta(r,a))$ for all $q \in Q_A, r \in Q_B$, and $a \in \Sigma$.
$s = (s_A, s_B)$ and $F = \{(q,r) \in Q_A \times Q_B | q \in F_A \text{ or } r \in F_B\}$
for intersection, can use and instead of or for F.

When proving a language is regular, if you can split it up and prove each part is regular, then you can declare the combined language regular per the product construction.

4.1 Singleton Languages

- Given regular languages A and B, $A \cup B$, $A \cap B$ and A complements B are all regular.
- It is true that every singleton language is regular by problem 16 on homework 3.
- It is also true that any finite language is regular by problem 17 on homework 3.

Example 1. Prove that for all $x \in \Sigma^*$, the singleton language $\{x\}$ is regular.

Σ^* is an infinitely long set of *finite* strings. Given any $x \in \Sigma^*$ it is true that x is of finite length. Furthermore, $\{x\}$ is a finite language consisting of a finite string. You can always construct a DFA for a finitely long string, and for every language that you can make a DFA for, it is regular. Therefore, $\{x\}$ is regular.

Example 2. Prove that every finite language $A \subseteq \Sigma^*$ is regular.

A finite language means that it is a finite set of a finite number of inputs (i.e. each input is of finite length). Consider the finite set $Z = \{s_1, s_2, \cdots, s_n\} \subseteq \Sigma^*$. Split Z into singleton languages $Z_1 = \{s_1\}$, $Z_2 = \{s_2\}$, $\cdots$, $Z_n = \{s_n\}$. By the example above, each singleton language $Z_1, Z_2, \cdots, Z_n$ is regular. It is known that the union of two regular languages is a regular language. Therefore, by performing a union on all $Z_1, Z_2, \cdots, Z_n$, this shows that Z is regular. Thus, every finite language is regular.

5 Equivalence Relations

5.1 Review of Equivalence Relations

- An equivalence relation on a set X is a binary relation $\equiv$ on X that is reflexive, symmetric, and transitive.
- If $\equiv$ is an equivalence relation on X and $x \in X$, then the $\equiv$ equivalence class of x is the set $[x]_{\equiv} = [x] = \{y \in X | x \equiv y\}$
- If X is a set and $\equiv$ is an equivalence relation on X, then the quotient of X by $\equiv$ is the set $X/_{\equiv} = \{[x]_{\equiv} | x \in X\}$
- a partition of a set X is a collection P of subsets B of X with properties:
  - each element of P is nonempty
  - $\bigcup_{B \in P} B = X$
  - For all $B_1, B_2 \in P, B_1 \neq B_2 \Rightarrow B_1 \cap B_2 = \emptyset$

5.2 Fundamental Theorem of Equivalence Relations

Let X be a set.

1. For every equivalence relation $\equiv$ on X, $X/_{\equiv}$ is a partition of X.
2. For every partition P of X, the relation $\equiv$ on X defined by $x \equiv y$ if and only if there exists a $B \in P$ such that $x, y \in B$.

Def: If $\equiv$ is an equivalence relation on a set S, then a set $A \subseteq S$ respects $\equiv$ if for all $x, y \in S$, $x \equiv y \Rightarrow [x \in A \textit{ iff } y \in A]$

Obs: If $\equiv$ is an equivalence relation on a set S, then $\forall A \subseteq S$, the following conditions are equivalent:
(1) A respects $\equiv$
(2) There is a set $I \subseteq S/_{\equiv}$ such that $A = \bigcup_{C \in I} C$

Def: For a DFA $M = (Q, \Sigma, \delta, s, F)$, define the relation $\equiv_M$ on Σ^* by $x \equiv_M y$ *iff* $\hat{\delta}(s, x) = \hat{\delta}(s, y)$

Observation 1: For each $q \in Q$, let $C_{m,q} = \{x \in \Sigma^* | \hat{\delta}(s, x) = q\}$
Define a state $q \in Q$ to be reachable in M if $C_{m,q} \neq \emptyset$.
Otherwise, q is unreachable in M.

Observation 2: An equivalence class $\equiv$ on a set S has finite index if
$|S/_{\equiv} < \infty$, i.e. if it has only finitely many equivalence classes.
Otherwise, $\equiv$ has infinite index.

Observation 3: For every DFA M, the equivalence relation $\equiv_M$ has finite index.
An equivalence relation $\equiv$ is right-invariant if, for all $x, y, z \in \Sigma^*$
$x \equiv y \Rightarrow xz \equiv yz$.

Observation 4: For every DFA M, $\equiv_M$ is right-invariant.

Observation 5: For every DFA M, $L(M) = \bigcup_{q \in F} C_{m,q}$.

Note: Observations 2 and 5 tell us that $L(M)$ respects $\equiv_M$. Taken together, observations 1-5 tell us that every regular language $A \subseteq \Sigma^*$ respects a right-invariant equivalence relation of finite index on Σ^*, namely the relation $\equiv_M$, where M is any DFA deciding A.

Def: The canonical equivalence relation of a language $A \subseteq \Sigma^*$ is the relation $\equiv_A$ on Σ^* defined by:
$x \equiv_A y$ *iff* for all $z \in \Sigma^*$, $xy \in A \Leftrightarrow yz \in A$, i.e., if you can attach a z to x, y that can distinguish x, y, you cannot decide the language.

Observation 6: For every language $A \subseteq \Sigma^*$, $\equiv_A$ is a right-invariant equivalence relation on Σ^*

Def: If $\equiv$ and $\approx$ are equivalence relations on a set S, then $\equiv$ refines $\approx$ if, $\forall x, y \in S, x \equiv y \Rightarrow x \approx y$.

Remarks:
- every equivalence relation defines itself
- every equivalence relation refines $\forall x, y, x \approx y$
- equality refines every equivalence relation

Observation 7: Let $\equiv$ and $\approx$ be equivalence relations on S. If $\equiv$ refines $\approx$, $\equiv$ has finite index $\Rightarrow \approx$ has finite index.

Lemma 8: Let $A \subseteq \Sigma^*$, and let $\equiv$ be a right-invariant equivalence relation on Σ^*, The following conditions are equivalent:
(1) A respects $\equiv$
(2) $\equiv$ refines $\equiv_A$

6 Proving Languages Not Regular

6.1 Myhill-Nerode

Theorem 9: For every language $A \subseteq \Sigma^*$, the following conditions are equivalent:
(1) A is regular
(2) The relation $\equiv_A$ has finite index.
(3) A respects some right-invariant equivalence relation of finite index on Σ^*

The Myhill-Nerode relation for A satisfies:
(1) $x \equiv y \Rightarrow xa \equiv ya$ for $x, y \in \Sigma^*$ and $a \in \Sigma$
(2) $x \equiv y \Rightarrow (x \in A \Leftrightarrow y \in A)$
(3) $\equiv$ is of finite index.
(4) Let M be a DFA with no inaccessible states. $x \equiv_M y \Leftrightarrow \hat{\delta}(s, x) = \hat{\delta}(s, y)$

Example: The language $A = \{0^n 1^n | n \in \mathbb{N}\}$ is not regular. Proof:
Let A be as given. It suffices by Myhill-Nerode to prove that $\equiv_A$ has infinite index.
Let $m, n \in \mathbb{N}$. Since $0^m 1^m \in A$ and $0^n 1^n \notin A$, and A respects $\equiv_A$,
it must be the case that $O^m 1^m \not\equiv_A 0^n 1^m$. Since $\equiv_A$ is right-invariant,
it follows that $0^m \not\equiv 0^n$. Thus, $\equiv_A$ does not have finite index.

6.2 Ordinal Extensions

- For each set $A \in \Sigma^*$ and each $j \in \mathbb{Z}^+$ not exceeding $|A|$, the jth element of A is thus unambiguously defined.
- For each $A \subseteq \Sigma^*$ and $x \in \Sigma^*$, let $A_x = \{y \in \Sigma^* | xy \in A\}$ be the set of all A-extensions of x. A_x is the set of all strings you can add to x and still be in A.
- Define the set $A_x^{(j)}$ as follows:
  - If $j \leqslant |A|$, then $A_x^{(j)} = \{y\}$, where y is the jth A-extension of x.
  - If $j > |A|$, then $A_x^{(j)} = \emptyset$
  - Note that $|A_x^{(j)}| \leqslant 1$ in any case.
- For each $A, B \subseteq \Sigma^*$ and $j \in \mathbb{Z}^+$, let $A_B^{(j)} = \bigcup_{x \in B} A_x^{(j)}$ be the set of all jth extensions of elements of B, and let $A^{(j)} = A_{\Sigma^*}^{(j)}$ be the set of all jth A-extensions.

Example. Prove that the language $\{0^k 1^m 0^n \mid n = k + m\}$ is not regular.

$LetA = \{0^k 1^m 0^n \mid n = k + m\}$, and $k = m = p$, $p \in \mathbb{N}$. Let $n = 2p$. Consider an x s.t. $x = 0^p 1^p$, and the first extension of x is $xA_x^1 \in A$.
Then consider y s.t. $y = 0^{2p}$, and concur that $xy \in A$ is accepted. It is true that for every p, there is one y, and y is dependent on p. Because p is infinite, y is also infinite. Then, $|y| = |A_x^1| = \infty$, showing that A is not regular.

6.2.1 Terminology

Def: A language $A \subseteq \Sigma^*$ has bounded ordinal extensions if there exists $m \in \mathbb{Z}^+$ such that, for all $j \in \mathbb{Z}^+$, $|A^{(j)}| \leqslant m$. Otherwise, A has unbounded ordinal extensions.

Def: A language A has infinite ordinal extensions if there exists $j \in \mathbb{Z}^+$ such that $|A^{(j)}| = \infty$. Clearly, a language with infinite ordinal extensions must have unbounded ordinal extensions.

Example: $A = \{0^n 1^n | n \in \mathbb{N}\}$ is not regular. Proof:
For each $n \in \mathbb{N}$, $1^n \in A^{(1)}$, because 1^n is the first A-extension of 0^n.

Theorem: Every regular language has bounded ordinal extensions.
$\therefore$ To prove that A is not regular, it suffices to show that A has unbounded ordinal extensions.
$\therefore$ For this, it suffices to show that A has infinite ordinal extensions.

Theorem: If A has unbounded ordinal extensions, then A is not regular.

Lemma: Let $\approx$ be a right-invariant equivalence relation on Σ^*, and let $A \subseteq \Sigma^*$.
If A respects $\approx$, then $\forall j \in \mathbb{Z}$, $|A^{(j)}| \leqslant |\Sigma^*/_{\approx}$.

7 Computability

7.1 Turing Machines

Def: Per the Church-Turing thesis, a function is computable if and only if there is a Turing machine that computes it.

Def: A Turing machine TM is a 9-tuple $M = (Q, \Sigma, \Gamma, \vdash, \square, \delta, s, t, r)$
Q a finite set of states
$\vdash$ is the left end marker
$\square$ is the empty cell
$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
$\Sigma \cup \{\vdash, \square\} \subseteq \Gamma$, and no transition changes $\vdash$
For all $a \in \Gamma$, $\delta(t, a) = (t, a, R)$ and $\delta(r, a) = (r, a, R)$, i.e.
whether it accepts/rejects, goes endlessly anyways.

Example. Give the formal description for a Turing machine that accepts the language

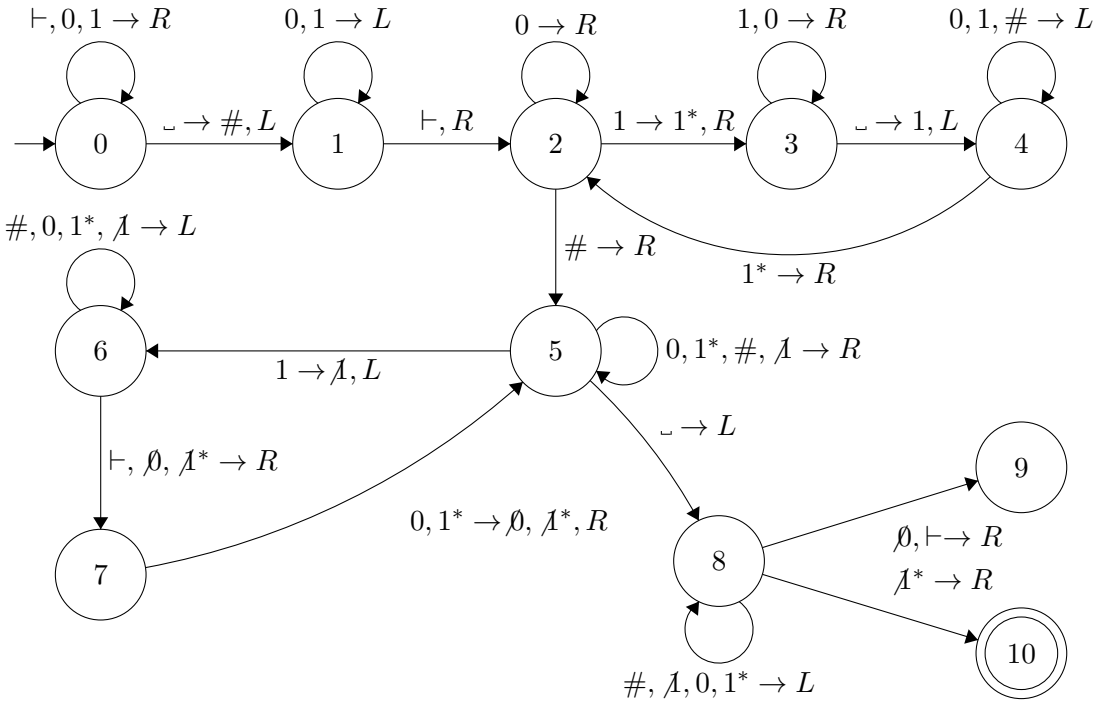
$$\{x \mid \text{the } \#(1, x)\text{-th symbol of } x \text{ is } 1\}$$

with $\Sigma = \{0, 1\}$.

Let $\Gamma = \Sigma \cup \{\vdash, \textvisiblespace, \#, 1^*, \not{1}, \not{1}^*, \not{0}\}$

Intuition (informal description):
Read until first blank, replace with #
Go back to beginning ($\vdash$)
(1) Go right until reach a 1
    When reach a 1, change the 1 to 1^*
    Go to first $\sqcup$ after #, change to a 1
    Go back to last 1^*
    Repeat (1)
Go right until # (have covered all the original ones by now)
Go right to first 1 after #
    Change to $\not{1}$
    Go back to start symbol or last canceled symbol
After canceling all 1's, go to last canceled and accept if one, reject if not.

Formal description:



Five methods for proving a language not regular.
(1) Brute force: Find inequivalent x,y that DFA must confuse
(2) Myhill-Nerode
(3) Pumping Lemma
(4) Kolmogorov Complexity
(5) Ordinal Extensions

In principle, (1) and (2) always work with extra ad hoc cleverness.
(3), (4), (5) are systematic, and usually work.

(5) short easy proofs.
(4) intuitive (talks about information)

7.2 Two string-pairing functions

$\langle x, y\rangle = 0^{|x|}1xy$
$\langle x, y\rangle = bd(x)01y$, where $bd(1101) = 11110011$ (bit double)

7.3 Cantor space

$P(\{0,1\}^*) = \{$languages over alphabet $\{0,1\}\}$
$\{0,1\}^* = \{s_0, s_1, s_2, \cdots\}$
$A \subseteq \{0,1\}^* = \{s_0, s_1, s_2, \cdots\}$
$x_A \in \{0,1\}^\infty = 0\ 1\ 1\ 0\ 1$ (bitmap of decisions made in A (no, yes, yes, no, yes, $\cdots$)

7.4 Reductions

Def: Let $A, B \subseteq \{0,1\}^*$. A many-one reduction (a.k.a. m-reduction, or $\leqslant_m$-reduction) of A to B is a computable function $f : \{0,1\}^* \to \{0,1\}^*$ s.t. for all $x \in \{0,1\}^*$, $x \in A \Leftrightarrow f(x) \in B$.

- We say that A is $\leqslant_m$-reducible to B, and we write $A \leqslant_m B$ if there exists $\leqslant_m$-reduction f of A to B

Reducibilities:
$\leqslant_T$ Turing reducibilities
$\leqslant_m$
$\leqslant_T^p$ polynomial time Turing reducibility
$\leqslant_m^p$ polnomial $\leqslant_m$

7.5 Decidability and Computability

Def: A language $A \subseteq \Sigma^*$ is decidable if there is a TM that (halts on every input and) decides A.

Def: A function $f : \{0,1\}^* \to \{0,1\}^*$ is computable if there is a TM that computes it.

Def: A partial function $f :\subseteq \{0,1\}^* \to \{0,1\}^*$ is computable if there is a TM that define it. It might not be defined on all inputs.

Def: For each $i \in \mathbb{N}$, the i-th TM is the TM M_i s.t. $\#(M_i) = u_i$, then $M_0, M_1, M_2, \cdots$ is the standard enumeration of all TMs
Note that a minor variant of U, say $\hat{U}$, has the property that

$\forall(i \in \mathbb{N} \wedge x \in \{0,1\}^*), \hat{U}(\langle i, x\rangle) \cong M_i(x)$

Notation: $\phi_i \subseteq \{0,1\}^* \to \{0,1\}^*$ is the partial-function computed by M_i, then $\phi_0, \phi_1, \phi_2, \cdots$ is the standard enumeration of all computable partial functions.

Notation: $M(x)\downarrow$, M halts on input x, $M(x)\uparrow$, M doesn't halt on input.

Def: The rapidly growing function (Note: max $\emptyset$=0) $G : \mathbb{N} \to \mathbb{N}$, is defined by $G(N) = 1 + max\{\phi_k(l)$ s.t. $0 \leqslant k \leqslant n$, $0 \leqslant l \leqslant n$, and $\phi_k(l)\downarrow\}$

Observation 1: G is total, i.e., $G(n)$ is defined for all n.

Observation 2: G is non-decreasing, i.e., $G(n) \leqslant G(n+1)$ for all n.

Def: The halting problem is the set $H = \{(k,l) \in \mathbb{N} \times \mathbb{N} | M_k(l)\downarrow\}$

Observation 3: If H is undecidable, then G is computable.

Lemma 4: G grows faster than any computable function. That is, for every computable function $f : \mathbb{N} \to \mathbb{N}$, we have $G(n) > f(n)$ for all but infinitely many n.

Corollary 5: G is not computable.

Theorem 6: H is undecidable.

Lemma 4 Proof: Let $f : \mathbb{N} \to \mathbb{N}$ be computable. Then, there is an index $i \in \mathbb{N}$ s.t. $\phi_i = f$. Then, for all $n \geqslant i$, $G(n)$ is defined as previously, $n \geqslant i \Rightarrow \phi_i(n)$ is in the $G(n)$ table.

$$\begin{aligned} G(n) &\geqslant 1 + \phi_i(n) \\ &> \phi_i(n) \\ &= f(n) \ \square \end{aligned}$$

Note: Once you know that the halting problem H is undecidable, you usually prove that other languages A are undecidable by proving that $H \leqslant_m A$.

Def: A language $A \subseteq \{0,1\}$ is computable enumerable (CE) if there is a TM M s.t. $L(M) = A$ where $L(M)$ is the set of all strings accepted by M (M is not required to halt on strings not in A)

Def: A language $A \subseteq \{0,1\}^*$ is co-computably enumerable (co-CE) if $\{0,1\}^* - A$ is CE (i.e. the complement of A is CE)

The classes DEC (decidable), CE, and co-CE are defined in the now-obvious ways:
DEC $\subseteq$ CE
$H \in$ CE - DEC (H is in CE, not decidable)

co-DEC = DEC
DEC $\subseteq$ co-CE
$\mathcal{C} \subseteq \mathcal{D}$ =co-$\mathcal{C} \subseteq$ co-$\mathcal{D}$, where $\mathcal{C}, \mathcal{D}$ are classes
CE $\cap$ co-CE = DEC

Fact: For every language $A \subseteq \{0,1\}^*$, the following conditions are equivalent:
(1) A is CE
(2) There is a DEC language $B \subseteq \{0,1\}^*$ s.t. for all $x \in \{0,1\}^*$,
$x \in A \Leftrightarrow (\exists w \in \{0,1\}^*)\langle x, w\rangle \in B$
(3) There is a CE language $B \subseteq \{0,1\}^*$ s.t. for all $x \in \{0,1\}^*$,
$x \in A \Leftrightarrow (\exists w \in \{0,1\}^*)\langle x, w\rangle \in B$

Fact: CE is close (downward) under $\leqslant_m$, meaning that $A \leqslant_m B \subseteq$ CE $\Rightarrow A \in$ CE.
DEC, and co-CE are also closed in this manner.

Def: Languages $A, B \in \{0,1\}^*$ are $\leqslant_m$-equivalent, and we write $A \equiv_m B$, if
$A \leqslant_m B$ and $B \leqslant_m A$.
Equivalence classes of $\equiv_m$ are called $\equiv_m$-degrees.

Def: Let $A \subseteq \{0,1\}^*$ be a language, and let $\mathcal{C}$ be a class of languages.
(1) A is $\leqslant_m$-hard for $\mathcal{C}$ if, for ever $B \in \mathcal{C}$, $\mathcal{C}$.
(at least as hard as anything in $\mathcal{C}$ w/ respect to this reducibility)
(2) A is $\leqslant_m$-complete for $\mathcal{C}$ if $A \in \mathcal{C}$ and A is $\leqslant_m$-hard for $\mathcal{C}$.

Corollary: Recall that the halting problem $H = \{(k,l) \in N \times N | M_k(l) \downarrow\}$ is CE.
H is not co-CE.

Theorem: H is $\leqslant_m$-complete for $\mathcal{C}$.

Def: An input/output property of TMs is a set $I \in \mathbb{N}$ s.t., for all
$i, j \in \mathbb{N}$, $\phi_i = \phi_j \Rightarrow [i \in I \Leftrightarrow j \in I]$

An I/O property I of TMs is trivial if $I = \emptyset$ or $I = \mathbb{N}$. Otherwise, I is non-trivial.

Theorem: (Rice's Theorem) states that every non-trivial I/O property of TMs is $\leqslant_m$-hard for CE or $\leqslant_m$-hard for co-CE hence undecidable in any case.

Example 1. Let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be computable. Prove: If $f(n) < f(n+1)$ for all $n \in \mathbb{N}$, then $\{f(n) \mid n \in \mathbb{N}\}$ is decidable.

Need to show that $f(n)$ is decidable when $f(n) < f(n+1)$.
TM: on input x

(1) Enumerate n (implies that $n = 0, 1, 2, \cdots$)
Compute $f(n)$
if $f(n) < x$ **then**
repeat (1)
else if $f(n) == x$ **then**
accept
else if $f(n) > x$ **then**
reject
end if

Thus constructing a decider for it and accepting when $x = f(n)$.

Recall that a real number $x \in \mathbb{R}$ is *computable* if there is a computable function $f : \mathbb{N} \longrightarrow \mathbb{Q}$ such that, for all $r \in \mathbb{N}$,

$$|f(r) - x| \leq 2^{-r}.$$

Example 2. Let $x, y \in \mathbb{R}$. Prove: If x and y are computable, then x+y is computable.

With x being computable, there is a computable function $|f(r) - x| \leqslant 2^{-r}$. The same applies for y, $|g(r) - y| \leqslant 2^{-r}$. To show that $x + y$ is also computable, define $h(r) = f(r) + g(r)$ and show that $|h(r) - (x + y)| \leqslant 2^{-r}$ holds. For this problem, consider $r + 1$, $h(r) = f(r + 1) + g(r + 1)$.

$$\begin{aligned} |f(r+1) + g(r+1) - (x+y)| &\leqslant 2 \cdot 2^{-(r+1)} \\ &\leqslant 2^1 \cdot 2^{-r-1} \\ &\leqslant 2^{-r} \end{aligned}$$

Considering that x, y and their respective functions are computable, we needed to show that for $x + y$ and respectively $h(r)$ is computable. Because $h(r)$ holds as shown above, $x + y$ is computable.

8 Algorithmic Information Theory

8.1 Kolmogorov Complexity

Def: Let M be a TM and $x \in \{0,1\}^*$. The (plain) Kolmogorov Complexity of x with respect to M is $C_m(x) = \min\{|\pi| \text{ s.t. } \pi \in \{0,1\}^* \text{ and } M(\pi) = x\}$ where $\min\emptyset = \infty$.

Intuition: π is a description of a program for x in the "language of M". $C_m(x)$ is the information content of x with respect to the algorithm M.

Def: A TM U is optimal if, for every TM M, there is a constant $c_m \in \mathbb{N}$ s.t. $\forall x \in \{0,1\}^*$, $C_U(x) \leqslant C_m(x) + c_m$.

c_m is (in practice) never more than a few thousand bits (a trivial amount).

Theorem 1: (optimality theorem) Every universal TM is optimal.

Theorem 2: For every TM M, there is an optimality constant $c_m \in \mathbb{N}$ s.t. for all $x \in \{0,1\}^*$, $C(x) \leqslant C_m(x) + c_m$

Theorem 3: There is a constant $a \in \mathbb{N}$ s.t. $\forall x \in \{0,1\}^*$, $C(x) \leqslant |x| + a$.

Corollary 4: For all $x \in \{0,1\}^*$, $C(x) < \infty$

Theorem 5: Let $n, r \in \mathbb{N}$. If we choose $x \in \{0,1\}^n$ uniformly at random, then the $\text{Prob}[C(x) \geqslant n - r] > 1 - 2^{-r}$

Corollary 6: For every $n \in \mathbb{N}$, $\exists x \in \{0,1\}^n$ s.t. $C(x) \geqslant n$.

Intuition: We call a string $x \in \{0,1\}^*$ random if $C(x) \approx |x|$. Sometimes we impose a precise, yet arbitrary threshold and call x random if $C(x) \geqslant |x|$. In this latter sense, correlation 6 tells us that there are random strings of every length. Note that this notion identifies randomness with incompressibility.

Theorem 7: (conservation of information) For every computable partial function $f :\subseteq \{0,1\}^* \to \{0,1\}^*$, there is a constant $c_f \in \mathbb{N}$ s.t. for all x in the domain of f, $C(f(x)) \leqslant C(x) + c_f$.

The intuition is that whatever you are doing when you compute, you are not creating new information. The amount of information in the output is never more than the input.

Notation: Recall the standard enumeration s_n, and recall $|s_n| = \lfloor \log_2(n+1) \rfloor$. The Kolmogorov complexity of a natural number $n \in \mathbb{N}$ is $C(n) = C(s_n)$.

Observation 8: There is a constant $a \in \mathbb{N}$ s.t. $\forall n \in \mathbb{N}$, $C(n) \leqslant \log(n+1) + a$.

Corollary 9: For every computable partial function $f :\subseteq \mathbb{N} \to \{0,1\}^*$, there is a constant $b_f \in \mathbb{N}$ s.t. for every n in the domain of f, $C(f(n)) \leqslant \log(n+1) + b_f$.

Observation 10: $\lim_{n\to\infty} C(n) = \infty$. That is, for every $m \in \mathbb{N}$, the condition $C(x) > m$ holds for all but finitely many $x \in \{0,1\}^*$.

Theorem 11: The Kolmogorov complexity function C is not computable. In fact, if $f :\subseteq \{0,1\}^* \to \mathbb{N}$ is any computable partial function that is a lower bound for C on its domain (i.e. $f(x) < C(x)$ for all x in domain f), then f is bounded (i.e. there is a constant $m \in \mathbb{N}$ s.t. $f(x) \leqslant m$ holds for all x in the domain of f).

$C(x) =$ plain Kolmogorov complexity of x
$K(x) =$ Kolmogorov complexity of x

Fact: A sequence $s \in \{0,1\}^\infty$ is <u>random</u> if there is a constant $c \in \mathbb{N}$ such that for all $n \in \mathbb{N}$, $K(s[0 \cdots n-1]) \geqslant n - c$.

Example. Prove: For every lossless data compression scheme (f, g), there is a constant $c_{(f,g)} \in \mathbb{N}$ such that, for all $x \in \{0,1\}^*$,

$$C(x) \leqslant |f(x)| + c_{(f,g)}$$

Given a TM M and that (f, g) are both computable, $M(f(x)) = g(f(x))$. Theorem 2 is defined as $C(x) \leqslant C_m(x) + c_m$. Thus, $C(x) \leqslant C_m(x) + c_{(f,g)}$, and then $C(x) \leqslant |f(x)| + c_{(f,gO}$. This proves that there is a constant $c_{(f,g)} \in \mathbb{N}$ such that the original proposition is true.

8.2 Number Theory

Notation:

· $p_0, p_1, p_2, \cdots$ is the enumeration of all prime numbers in order. Thus, $p_0 = 2$, $p_1 = 3$, $p_2 = 5$, etc..

· For $n \in \mathbb{Z}^+$, $\pi(n) = |\{i | p_i \leqslant n\}|$ is the number of prime numbers $\leqslant n$. We thus have the values...

n	1	2	3	4	5	6	$\cdots$
$\pi(n)$	0	1	2	2	3	3	$\cdots$

which just keeps track of the number of primes.

· Gauss conjectured $\lim_{n\to\infty} \dfrac{\pi(n)}{\frac{n}{ln(n)}} = 1$

Lemma 14: There is a constant $c \in \mathbb{N}$ s.t., for all $n > 1$, $C(n) \leqslant \pi(n)[3 + 2\log\log(n)] + c$.

Obs. 15: There exists infinitely many n s.t. $C(n) \geqslant \log(n) - 1$.

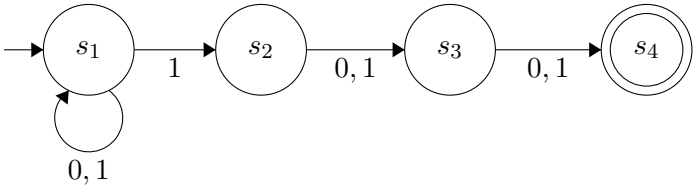
Theorem 16: There exists infinitely many n s.t. $\pi(n) > \dfrac{\log(N)}{3\log\log(n)}$.

Corollary 17: There are infinitely many prime numbers.

9 Nondeterministic Finite Automata

Example: For each $k \in \mathbb{Z}^+$, let $A_k = \{x \in \{0,1\}^* | \text{the } k\text{-th to last bit is } 1\}$.

NFA:

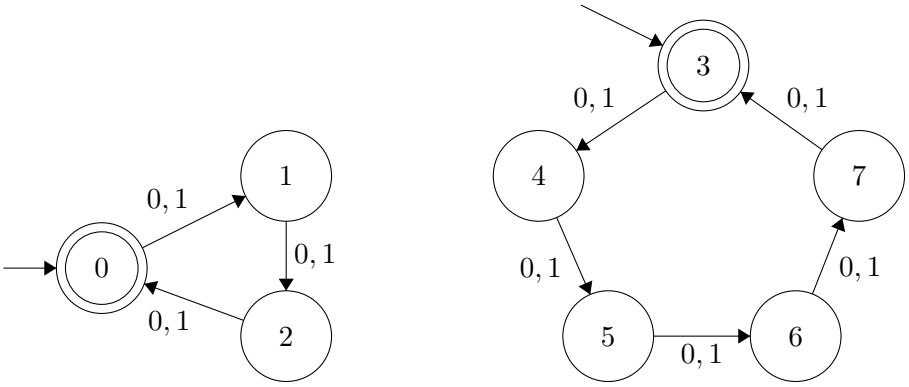


Def: A nondeterministic finite automata (NFA) is a 5-tuple $N = (Q, \Sigma, \Delta, S, F)$ where
· Q, Σ, F are the same as in DFA
· $S \subseteq Q$ is the set of start states
· $\Delta : Q \times E \to P(Q)$ is the transition function

Intuition: An NFA $N = (Q, \Sigma, \Delta, S, F)$ may start in any state in S. Like a DFA, it reads an input string $x \in \Sigma^*$ one symbol at a time. If it is in state q at the time t and reads $a \in \Sigma$, then at time $t + 1$ it may be in any state $q' \in \Delta(q, a)$.

"The magic:" If there is some way for N to accept, (proceedint as above) then it does so, and we say that N accepts x, otherwise N rejects x.

Example: Design an NFA that accepts the language
$A = \{x \in \{0, 1\}^* \text{ s.t. } |x| \text{ is divisible by 3 or 5}\}$.



Formal Semantics of NFAs:

- Let $N = (Q, \Sigma, \Delta, S, F)$ be an NFA. We define the extended transition function: $\hat{\Delta} : P(Q) \times \Sigma^* \to P(Q)$ as follows.
- We want $\hat{\Delta}(A, x)$ to be the set of all states that are reachable from A by processing x.
- The definition of $\hat{\Delta}(A, x)$ is recursive.
  $\hat{\Delta}(A, \lambda) = A$
  $\hat{\Delta}(A, xa) = \cup_{q \in \hat{\Delta}(A,x)} \Delta(q, a)$

Def: An NFA N accepts a string $x \in \Sigma^*$ if $\hat{\Delta}(s, x) \cap F = \emptyset$, otherwise N rejects x.

Fact: For every DFA M, there is an NFA N s.t. $L(M) = L(N)$

Theorem: For every NFA N, there is a DFA M s.t. $L(M) = L(N)$.

10 Grammar

Def: A sequence $S \in \{0,1\}^\infty$ is normal if and only if no finite-state gambler can make unbounded money placing fair bets on it.
$\therefore$ Normality is finite-state randomness.

Def: A context-free-grammar is a 4-tuple
$G = (N, \Sigma, P, S)$ where
· N is an alphabet of non-terminal symbols
· Σ is an alphabet of terminal symbols
· $N \cap \Sigma = \emptyset$
· $S \in \mathbb{N}$ is the start symbol
· P is a finite set of productions, each of which is of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (N \cup \Sigma)*$

Semantics: Let $G = (N, \Sigma, P, S)$ be a CFG. For $\alpha, \beta \in (N \cup \Sigma)^*$, we say that β is derivable from α in one step, and we write $\alpha \overset{①}{G} \beta$. If there exists a production $A \to \gamma$ in P and strings $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ such that $\alpha = \alpha_1 A \alpha_2$ and $\beta = \alpha_1 \gamma \alpha_2$.

For each $n \in \mathbb{N}$, define the relation $\overset{n}{G}$ on $(N \cup \Sigma)^*$
· $\alpha \overset{0}{G} \beta \Leftrightarrow \alpha = \beta$
· $\alpha \overset{n+1}{G} \beta \Leftrightarrow \exists \gamma \in (N \cup \Sigma)^*$ s.t. $\alpha \overset{n}{G} \gamma$ and $\gamma \overset{①}{G} \beta$

Def: If $\alpha, \beta \in (N \cup \Sigma)^*$, then β is derivable from α, and we write $\alpha \overset{*}{G} \beta$ if there exists $n \in \mathbb{N}$ s.t. $\alpha \overset{n}{G} \beta$.

Def: The language generated by G is $L(G) = \{x \in \Sigma^* | S \overset{*}{G} x\}$

Def: A context-free-language is a language $A \subseteq \Sigma^*$ for which there exists a CFG G s.t. $L(G) = A$.

Terminology: Let $G = (N, \Sigma, P, S)$ be a CFG.
· a sequential form of G is a string $\alpha \in (N \cup \Sigma)^*$ s.t. $S \overset{*}{G} \alpha$
· a sentence of G is a terminal string $x \in \Sigma^*$ s.t. $S \overset{*}{G} x$

Def: A right-linear grammar is a CFG all of whose productions are of the form $A \to xB$ or $A \to x$ where $A, B \in N$ and $x \in \Sigma^*$.

Def: A strongly right-linear grammar is a right-linear grammar, all of whose productions are of the form $A \to aB$ or $A \to \lambda$, where $A, B \in N$ and $a \in \Sigma$.