Exercises

Homework 1

1. Design deterministic finite automata for each of the following sets:
   (a) the set of strings in $\{4, 8, 1\}^*$ containing the substring 481;
   (b) the set of strings in $\{a\}^*$ whose length is divisible by either 2 or 7;
   (c) the set of strings $x \in \{0, 1\}^*$ such that $\#0(x)$ is even and $\#1(x)$ is a multiple of three;
   (d) the set of strings over the alphabet $\{a, b\}$ containing at least three occurrences of three consecutive b's, overlapping permitted (e.g., the string *bbbbb* should be accepted);
   (e) the set of strings in $\{0, 1, 2\}^*$ that are ternary (base 3) representations, leading zeros permitted, of numbers that are not multiples of four. (Consider the null string a representation of zero.)

2. Consider the following two deterministic finite automata.

		a	b
$\rightarrow$	1	1	2
	$2F$	2	1

		a	b
$\rightarrow$	1	2	3
	2	3	1
	$3F$	1	2

   Use the product construction to produce deterministic automata accepting (a) the intersection and (b) the union of the two sets accepted by these automata.

3. Let $M = (Q, \Sigma, \delta, s, F)$ be an arbitrary DFA. Prove by induction on $|y|$ that for all strings $x, y \in \Sigma^*$ and $q \in Q$,

$$\widehat{\delta}(q, xy) = \widehat{\delta}(\widehat{\delta}(q, x), y),$$

   where $\widehat{\delta}$ is the extended version of δ defined on all strings described in Lecture 3.

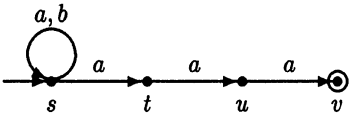
4. For $k \geq 1$ and $p \geq 2$, let

$$A_{k,p} \stackrel{\text{def}}{=} \{x \in \{0, 1, \ldots, p-1\}^* \mid x \text{ is a } p\text{-ary representation of a multiple of } k\}.$$

   In Lecture 4 we gave a DFA for the set $A_{3,2}$, the multiples of three in binary, and proved it correct. Generalize the construction and proof to arbitrary k and p.

Homework 2

1. The following nondeterministic automaton accepts the set of strings in $\{a,b\}^*$ ending in *aaa*. Convert this automaton to an equivalent deterministic one using the subset construction. Show clearly which subset of $\{s,t,u,v\}$ corresponds to each state of the deterministic automaton. Omit inaccessible states.



2. The *reverse* of a string x, denoted **rev** x, is x written backwards. Formally,

$$\textbf{rev}\ \epsilon \stackrel{\text{def}}{=} \epsilon, \qquad \textbf{rev}\ xa \stackrel{\text{def}}{=} a\,\textbf{rev}\ x.$$

For example, **rev** $abbaaab = baaabba$. For $A \subseteq \Sigma^*$, define

$$\textbf{rev}\ A \stackrel{\text{def}}{=} \{\textbf{rev}\ x \mid x \in A\}.$$

For example, **rev** $\{a, ab, aab, aaab\} = \{a, ba, baa, baaa\}$. Show that for any $A \subseteq \Sigma^*$, if A is regular, then so is **rev** A.

3. The *Hamming distance* between two bit strings x and y (notation: $H(x,y)$) is the number of places at which they differ. For example, $H(011,110) = 2$. (If $|x| \neq |y|$, then their Hamming distance is infinite.) If x is a string and A is a set of strings, the Hamming distance between x and A is the distance from x to the closest string in A:

$$H(x,A) \stackrel{\text{def}}{=} \min_{y \in A} H(x,y).$$

For any set $A \subseteq \{0,1\}^*$ and $k \geq 0$, define

$$N_k(A) \stackrel{\text{def}}{=} \{x \mid H(x,A) \leq k\},$$

the set of strings of Hamming distance at most k from A. For example, $N_0(\{000\}) = \{000\}$, $N_1(\{000\}) = \{000, 001, 010, 100\}$, and $N_2(\{000\}) = \{0,1\}^3 - \{111\}$.

Prove that if $A \subseteq \{0,1\}^*$ is regular, then so is $N_2(A)$. (*Hint*: If A is accepted by a machine with states Q, build a machine for $N_2(A)$ with states $Q \times \{0,1,2\}$. The second component tells how many errors you have seen so far. Use nondeterminism to guess the string $y \in A$ that the input string x is similar to and where the errors are.)

Homework 3

1. Give regular expressions for each of the following subsets of $\{a, b\}^*$.

   (a) $\{x \mid x$ contains an even number of a's$\}$

   (b) $\{x \mid x$ contains an odd number of b's$\}$

   (c) $\{x \mid x$ contains an even number of a's or an odd number of b's$\}$

   (d) $\{x \mid x$ contains an even number of a's and an odd number of b's$\}$

   Try to simplify the expressions as much as possible using the algebraic laws of Lecture 9. Recall that regular expressions over $\{a, b\}$ may use ϵ, $\emptyset$, a, b, and operators $+$, *, and $\cdot$ only; the other pattern operators are not allowed.

2. Give deterministic finite automata accepting the sets of strings matching the following regular expressions.

   (a) $(000^* + 111^*)^*$

   (b) $(01 + 10)(01 + 10)(01 + 10)$

   (c) $(0 + 1(01^*0)^*1)^*$

   Try to simplify as much as possible.

3. For any set of strings A, define the set

   $$\text{MiddleThirds } A = \{y \mid \exists x, z \; |x| = |y| = |z| \text{ and } xyz \in A\}.$$

   For example, $\text{MiddleThirds}\{\epsilon, a, ab, bab, bbab, aabbab\} = \{\epsilon, a, bb\}$. Show that if A is regular, then so is MiddleThirds A.

Homework 4

1. Show that the following sets are not regular.

   (a) $\{a^n b^m \mid n = 2m\}$

   (b) $\{x \in \{a,b,c\}^* \mid x \text{ is a palindrome; i.e., } x = \mathbf{rev}(x)\}$

   (c) $\{x \in \{a,b,c\}^* \mid \text{the length of } x \text{ is a square}\}$

   (d) The set PAREN of balanced strings of parentheses ( ). For example, the string ((( ) ( )) ( )) is in PAREN, but the string) (( ) is not.

2. The operation of *shuffle* is important in the theory of concurrent systems. If $x, y \in \Sigma^*$, we write $x \parallel y$ for the set of all strings that can be obtained by shuffling strings x and y together like a deck of cards; for example,

   $$ab \parallel cd = \{abcd, acbd, acdb, cabd, cadb, cdab\}.$$

   The set $x \parallel y$ can be defined formally by induction:

   $$\begin{aligned} \epsilon \parallel y &\stackrel{\text{def}}{=} \{y\}, \\ x \parallel \epsilon &\stackrel{\text{def}}{=} \{x\}, \\ xa \parallel yb &\stackrel{\text{def}}{=} (x \parallel yb) \cdot \{a\} \cup (xa \parallel y) \cdot \{b\}. \end{aligned}$$

   The shuffle of two languages A and B, denoted $A \parallel B$, is the set of all strings obtained by shuffling a string from A with a string from B:

   $$A \parallel B \stackrel{\text{def}}{=} \bigcup_{\substack{x \in A \\ y \in B}} x \parallel y.$$

   For example,

   $$\{ab\} \parallel \{cd, e\} = \{abe, aeb, eab, abcd, acbd, acdb, cabd, cadb, cdab\}.$$

   (a) What is $(01)^* \parallel (10)^*$?

   (b) Show that if A and B are regular sets, then so is $A \parallel B$. (*Hint*: Put a pebble on a machine for A and one on a machine for B. Guess nondeterministically which pebble to move. Accept if both pebbles occupy accept states.)

3. For each of the two automata

		a	b
$\rightarrow$	1	1	4
	2	3	1
	$3F$	4	2
	$4F$	3	5
	5	4	6
	6	6	3
	7	2	4
	8	3	1

		a	b
$\rightarrow$	$1F$	3	5
	$2F$	8	7
	3	7	2
	4	6	2
	5	1	8
	6	2	3
	7	1	4
	8	5	1

(a) say which states are accessible and which are not;

(b) list the equivalence classes of the collapsing relation $\approx$ defined in Lecture 13:

$$p \approx q \stackrel{\text{def}}{\iff} \forall x \in \Sigma^* \ (\widehat{\delta}(p,x) \in F \iff \widehat{\delta}(q,x) \in F);$$

(c) give the automaton obtained by collapsing equivalent states and removing inaccessible states.

Homework 5

1. The following table defines four special types of CFGs obtained by restricting productions to the form shown, where A, B represent nonterminals, a a single terminal symbol, and x a string of terminals:

Grammar type	*Form of productions*
right-linear	$A \to xB$ or $A \to x$
strongly right-linear	$A \to aB$ or $A \to \epsilon$
left-linear	$A \to Bx$ or $A \to x$
strongly left-linear	$A \to Ba$ or $A \to \epsilon$

   Prove that each of these four types of grammars generates exactly the regular sets. Conclude that every regular set is a CFL.

2. Prove that the CFG

$$S \to aSb \mid bSa \mid SS \mid \epsilon$$

   generates the set of all strings over $\{a, b\}$ with equally many a's and b's. (*Hint*: Characterize elements of the set in terms of the graph of the function $\#b(y) - \#a(y)$ as y ranges over prefixes of x, as we did in Lecture 20 with balanced parentheses.)

3. Give a CFG for the set PAREN_2 of balanced strings of parentheses of two types `( )` and `[ ]`. For example, `( [ ( ) [ ] ] ( [ ] ) )` is in PAREN_2, but `[ ( ] )` is not. Prove that your grammar is correct. Use the following inductive definition: PAREN_2 is the smallest set of strings such that

   (i) $\epsilon \in \text{PAREN}_2$;

   (ii) if $x \in \text{PAREN}_2$, then so are (x) and $[x]$; and

   (iii) if x and y are both in PAREN_2, then so is xy.

   (*Hint*: Your grammar should closely model the inductive definition of the set. For one direction of the proof of correctness, use induction on the length of the derivation. For the other direction, use induction on stages of the inductive definition of PAREN_2. The basis is (i), and there will be two cases of the induction step corresponding to (ii) and (iii).)

4. Give a PDA for the set PAREN_2 of Exercise 3 that accepts by empty stack. Specify all transitions.

Homework 6

1. Prove that the following CFG G in Greibach normal form generates exactly the set of nonnull strings over $\{a, b\}$ with equally many a's and b's:

$$S \to aB \mid bA,$$
$$A \to aS \mid bAA \mid a,$$
$$B \to bS \mid aBB \mid b.$$

(*Hint*: Strengthen your induction hypothesis to describe the sets of strings generated by the nonterminals A and B: for $x \neq \epsilon$,

$$S \xrightarrow[G]{*} x \iff \#a(x) = \#b(x),$$
$$A \xrightarrow[G]{*} x \iff ???,$$
$$B \xrightarrow[G]{*} x \iff ???.)$$

2. Construct a pushdown automaton that accepts the set of strings in $\{a, b\}^*$ with equally many a's and b's. Specify all transitions.

3. Let $b(n)$ denote the binary representation of $n \geq 1$, leading zeros omitted. For example, $b(5) = 101$ and $b(12) = 1100$. Let \$ be another symbol not in $\{0, 1\}$.

   (a) Show that the set

   $$\{b(n)\$b(n+1) \mid n \geq 1\}$$

   is not a CFL.

   (b) Suppose we reverse the first numeral; that is, consider the set

   $$\{\mathbf{rev}\, b(n)\$b(n+1) \mid n \geq 1\}.$$

   Show that this set is a CFL.

4. Recall from Exercise 3 of Homework 5 the set PAREN_2 of balanced strings of parentheses of two types, ( ) and [ ]. Give CFGs in Chomsky and Greibach normal form generating the set $\text{PAREN}_2 - \{\epsilon\}$, and prove that they are correct.

Homework 7

1. Describe a parser to parse regular expressions according to the precedence relation

   $$^* > \cdot > +$$

   Here $>$ means "has higher precedence than." The main differences you will have to account for between regular expressions and the arithmetic expressions discussed in Lecture 26 are: (i) the unary operator * comes *after* its operand, not before it as with unary minus; and (ii) the concatenation operator $\cdot$ can be omitted. Illustrate the action of your parser on the expression

   $$(a + b)^* + ab^*a.$$

2. Prove that if A is a CFL and R is a regular set, then $A \cap R$ is a CFL. (*Hint*: Use a product construction.)

3. Recall the shuffle operator $\|$ from Homework 4. Show that the shuffle of two CFLs is not necessarily a CFL. (*Hint*: Use the previous exercise to simplify the argument.)

4. Let A be any regular set. Show that the set

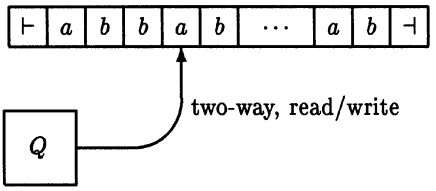
   $$\{x \mid \exists y \ |y| = 2^{|x|} \text{ and } xy \in A\}$$

   is regular.

Homework 8

1. Describe a TM that accepts the set $\{a^n \mid n \text{ is a power of } 2\}$. Your description should be at the level of the descriptions in Lecture 29 of the TM that accepts $\{ww \mid w \in \Sigma^*\}$ and the TM that implements the sieve of Eratosthenes. In particular, do not give a list of transitions.

2. A *linear bounded automaton* (LBA) is exactly like a one-tape Turing machine, except that the input string $x \in \Sigma^*$ is enclosed in left and right endmarkers $\vdash$ and $\dashv$ which may not be overwritten, and the machine is constrained never to move left of the $\vdash$ nor right of the $\dashv$. It may read and write all it wants between the endmarkers.



   (a) Give a rigorous formal definition of deterministic linearly bounded automata, including a definition of configurations and acceptance. Your definition should begin as follows: "A *deterministic linearly bounded automaton (LBA)* is a 9-tuple

   $$M = (Q, \Sigma, \Gamma, \vdash, \dashv, \delta, s, t, r),$$

   where Q is a finite set of *states*, ..."

   (b) Let M be a linear bounded automaton with state set Q of size k and tape alphabet Γ of size m. How many possible configurations are there on input x, $|x| = n$?

   (c) Argue that the halting problem for deterministic linear bounded automata is decidable. (*Hint*: You need to be able to detect after a finite time if the machine is in an infinite loop. Presumably the result of part (b) would be useful here.)

   (d) Prove by diagonalization that there exists a recursive set that is not accepted by any LBA.

3. Let A be any regular set. Show that the set

   $$\{x \mid \exists y \; |y| = |x|^2 \text{ and } xy \in A\}$$

   is regular.

Homework 9

1. Prove that the following question is undecidable. Given a Turing machine M and state q of M, does M ever enter state q on some input? (This problem is analogous to the problem of identifying *dead code*: given a PASCAL program containing a designated block of code, will that block of code ever be executed?)

2. Prove that it is undecidable whether two given Turing machines accept the same set. (This problem is analogous to determining whether two given PASCAL programs are equivalent.)

3. Prove that the emptiness problem for deterministic linearly bounded automata (i.e., whether $L(M) = \varnothing$) is undecidable. (*Hint*: Think VALCOMPS.)

4. Prove that an r.e. set is recursive iff there exists an enumeration machine that enumerates it in increasing order.

5. For $A, B \subseteq \Sigma^*$, define

$$A/B \stackrel{\text{def}}{=} \{x \mid \exists y \in B \; xy \in A\},$$
$$A \leftarrow B \stackrel{\text{def}}{=} \{x \mid \forall y \in B \; xy \in A\}.$$

   (a) Show that if A is regular and B is any set whatsoever, then A/B and $A \leftarrow B$ are regular.

   (b) Show that even if we are given a finite automaton for A and a Turing machine for B, we cannot necessarily construct an automaton for A/B or $A \leftarrow B$ effectively.

Homework 10

1. Show that neither the set

   $$\text{TOTAL} \stackrel{\text{def}}{=} \{M \mid M \text{ halts on all inputs}\}$$

   nor its complement is r.e.

2. Consider one-tape Turing machines that are constrained not to overwrite the input string. They may write all they want on the blank portion of the tape to the right of the input string.

   (a) Show that these machines accept only regular sets. (If you're thinking, "Hey, why not just copy the input string out to the blank portion of the tape," think again ...)

   (b) Show that, despite (a), it is impossible to construct an equivalent finite automaton effectively from such a machine.

3. Show that it is undecidable whether the intersection of two CFLs is nonempty. (*Hint*: Use a variant of VALCOMPS in which every other configuration is reversed:

   $$\alpha_0\#\mathbf{rev}\,\alpha_1\#\alpha_2\#\mathbf{rev}\,\alpha_3\#\cdots\#\alpha_n.$$

   Express this set as the intersection of two CFLs.)

Homework 11

1. Recursive enumerability is intimately linked with the idea of *unbounded existential search.* Often an algorithm for accepting an r.e. set can be characterized in terms of searching for a *witness* or *proof* that a given input x is in the set.

   A binary relation R on strings over $\{0,1\}$ is called *recursive* if the set

   $$\{x\#y \mid R(x,y)\}$$

   is a recursive set. Here # is just another input symbol different from 0 or 1.

   Show that a set $A \subseteq \{0,1\}^*$ is r.e. if and only if there exists a recursive binary relation R such that

   $$A = \{x \in \{0,1\}^* \mid \exists y\ R(x,y)\}.$$

2. Show that it is undecidable whether the intersection of two given CFLs is again a CFL. (*Hint*: Use Homework 10, Exercise 3.)

Homework 12

1. A *context-sensitive grammar* (CSG) or *type 1 grammar* is a type 0 grammar that obeys the following additional restriction: all productions $\alpha \to \beta$ satisfy $|\alpha| \leq |\beta|$. Give CSGs for the following sets.

   (a) $\{x \in \{a,b,c\}^+ \mid \#a(x) = \#b(x) = \#c(x)\}$

   (b) $\{a^{n^2} \mid n \geq 1\}$

2. Show that context-sensitive grammars and nondeterministic linearly bounded automata are equivalent in the following sense:

   (a) for every context-sensitive grammar G, there is a nondeterministic LBA M such that $L(M) = L(G)$; and

   (b) for every nondeterministic LBA M, there is a context-sensitive grammar G such that $L(G) = L(M) - \{\epsilon\}$.

3. Give constructions showing that the following number-theoretic functions are primitive recursive.

   (a) $\mathbf{quotient}(x,y)$ = quotient when dividing x by y using integer division; for example, $\mathbf{quotient}(7,2) = 3$.

   (b) $\mathbf{remainder}(x,y)$ = remainder when dividing x by y using integer division; for example, $\mathbf{remainder}(7,2) = 1$.

   (c) $\mathbf{prime}(x) = 1$ if x is prime, 0 otherwise.

Miscellaneous Exercises

Finite Automata and Regular Sets

1. Let B be a set of strings over a fixed finite alphabet. We say B is *transitive* if $BB \subseteq B$ and *reflexive* if $\epsilon \in B$. Prove that for any set of strings A, A^* is the smallest reflexive and transitive set containing A. That is, show that A^* is a reflexive and transitive set containing A, and if B is any other reflexive and transitive set containing A, then $A^* \subseteq B$.

2. Consider the following pairs of deterministic finite automata. Use the product construction to produce deterministic automata accepting (i) the intersection and (ii) the union of the sets accepted by these automata.

   (a)

		a	b
$\rightarrow$	1	2	2
	$2F$	1	1

		a	b
$\rightarrow$	1	1	2
	$2F$	2	1

   (b)

		a	b
$\rightarrow$	1	2	3
	$2F$	3	1
	$3F$	1	2

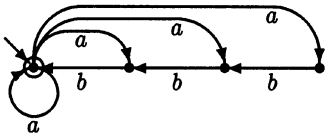
		a	b
$\rightarrow$	$1F$	3	2
	2	1	3
	$3F$	2	1

(c)

		a	b
$\rightarrow$	1	2	2
	$2F$	1	1

		a	b
$\rightarrow$	1	2	1
	$2F$	1	2

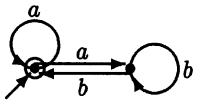
3. Consider the following nondeterministic finite automaton.



   (a) Give a string beginning with a that is *not* accepted.

   (b) Construct an equivalent deterministic automaton using the subset construction. Assuming the states are named s, t, u, v from left to right, show clearly which subset of $\{s, t, u, v\}$ corresponds to each state of the deterministic automaton. Omit inaccessible states.

4. Consider the following NFA.

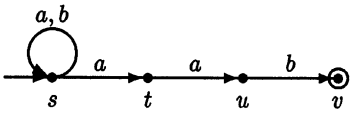


   (a) Construct an equivalent DFA using the subset construction. Omit inaccessible states.

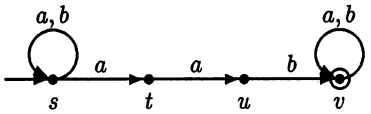
   (b) Give an equivalent regular expression.

5. Convert the following nondeterministic finite automata to equivalent deterministic ones using the subset construction. Show clearly which subset of $\{s, t, u, v\}$ corresponds to each state of the deterministic automaton. Omit inaccessible states.

   (a)



(b)



H6. Prove that NFAs are exponentially more succinct than DFAs: for any m, there exists an NFA with m states such that any equivalent DFA has at least 2^{m-1} states.

7. A convenient way of specifying automata is in terms of *transition matrices*. If the automaton has n states, the transition function δ can be specified by an $n \times n$ matrix G, indexed by states, whose u,vth entry gives the set of input symbols taking state u to state v; in symbols,

$$G_{uv} = \{a \in \Sigma \mid \delta(u,a) = v\}.$$

For example, the transition function of the automaton of Example 3.1 of Lecture 3 could be represented by the 4×4 matrix

$$\begin{bmatrix} \{b\} & \{a\} & \varnothing & \varnothing \\ \varnothing & \{b\} & \{a\} & \varnothing \\ \varnothing & \varnothing & \{b\} & \{a\} \\ \varnothing & \varnothing & \varnothing & \{a,b\} \end{bmatrix}.$$

Consider the collection of square matrices indexed by Q whose entries are subsets of Σ^*. We can define addition and multiplication on such matrices in a natural way as follows:

$$(A+B)_{uv} \stackrel{\text{def}}{=} A_{uv} \cup B_{uv},$$
$$(AB)_{uv} \stackrel{\text{def}}{=} \bigcup_{w \in Q} A_{uw} B_{wv}.$$

Let us also define the identity matrix I:

$$I_{uv} \stackrel{\text{def}}{=} \begin{cases} \{\epsilon\} & \text{if } u = v, \\ \varnothing & \text{otherwise.} \end{cases}$$

The powers of a matrix are defined inductively:

$$A^0 \stackrel{\text{def}}{=} I,$$
$$A^{n+1} \stackrel{\text{def}}{=} A^n A.$$

*S(a) Prove that

$$(A^n)_{uv} = \{x \in \Sigma^* \mid |x| = n \text{ and } \widehat{\delta}(u,x) = v\}.$$

(b) Define the asterate A^* of the matrix A to be the componentwise union of all the powers of A:

$$(A^*)_{uv} \stackrel{\text{def}}{=} \bigcup_{n \geq 0} (A^n)_{uv}.$$

Let s be the start state of the automaton and F the set of accept states. Prove that

$$L(M) = \bigcup_{t \in F} (A^*)_{st}.$$

8. Generalize Homework 2, Exercise 3 to arbitrary distance k. That is, prove that if $A \subseteq \{0,1\}^*$ is regular, then so is $N_k(A)$, the set of strings of Hamming distance at most k from some string in A.

9. (a) Show that if an NFA with k states accepts any string at all, then it accepts a string of length $k-1$ or less.

H(b) Give an NFA over a single letter alphabet that rejects some string, but the length of the shortest rejected string is strictly more than the number of states.

**H(c) Give a construction for arbitrarily large NFAs showing that the length of the shortest rejected string can be exponential in the number of states.

10. Recall from Lecture 10 that an *NFA with ϵ-transitions* is a structure

$$M = (Q, \Sigma, \epsilon, \Delta, S, F)$$

such that ϵ is a special symbol not in Σ and

$$M_\epsilon = (Q, \Sigma \cup \{\epsilon\}, \Delta, S, F)$$

is an ordinary NFA over the alphabet $\Sigma \cup \{\epsilon\}$.

Define the ϵ-closure $C_\epsilon(A)$ of a set $A \subseteq Q$ to be the set of all states reachable from some state in A under a sequence of zero or more ϵ-transitions:

$$C_\epsilon(A) \stackrel{\text{def}}{=} \bigcup_{x \in \{\epsilon\}^*} \hat{\Delta}(A, x).$$

(a) Using ϵ-closure, define formally acceptance for NFAs with ϵ-transitions in a way that captures the intuitive description given in Lecture 6.

(b) Prove that under your definition, NFAs with ϵ-transitions accept only regular sets.

(c) Prove that the two definitions of acceptance—the one given in part (a) involving ϵ-closure and the one given in Lecture 10 involving homomorphisms—are equivalent.

11. Give regular expressions for each of the following subsets of $\{a, b\}^*$. Recall that regular expressions over $\{a, b\}$ may use ϵ, $\emptyset$, a, b, and operators $+$, *, and $\cdot$ only.

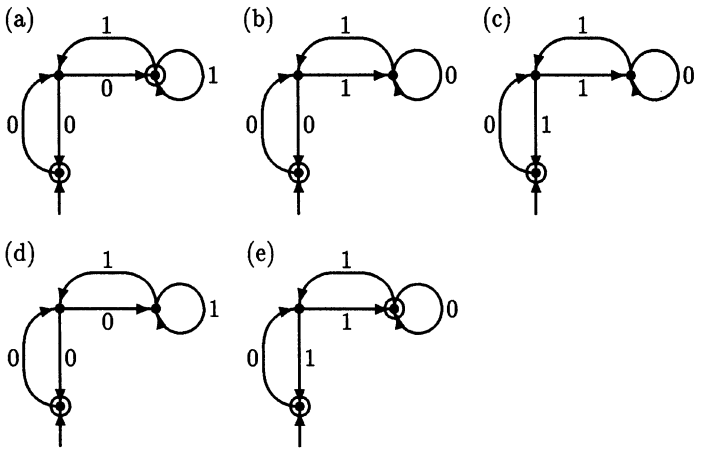
(a) $\{x \mid x$ does not contain the substring $a\}$

(b) $\{x \mid x$ does not contain the substring $ab\}$

**(c) $\{x \mid x$ does not contain the substring $aba\}$

Try to simplify the expressions as much as possible using the algebraic laws of Lecture 9.

12. Match each NFA with an equivalent regular expression.



(i) $\epsilon + 0(01^*1 + 00)^*01^*$

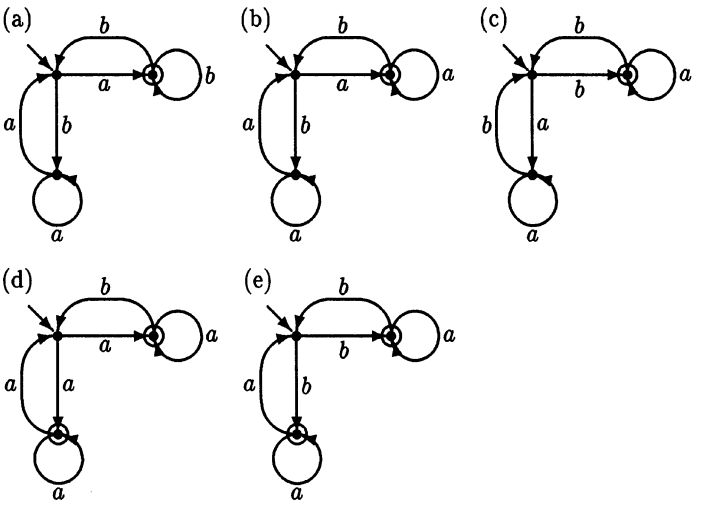
(ii) $\epsilon + 0(10^*1 + 10)^*10^*$

(iii) $\epsilon + 0(10^*1 + 00)^*0$

(iv) $\epsilon + 0(01^*1 + 00)^*0$

(v) $\epsilon + 0(10^*1 + 10)^*1$

13. Match each NFA with an equivalent regular expression.



(i) $(aa^*b + ba^*b)^*ba^*$

(ii) $(aa^*a + aa^*b)^*aa^*$

(iii) $(ba^*a + ab^*b)^*ab^*$

(iv) $(ba^*a + aa^*b)^*aa^*$

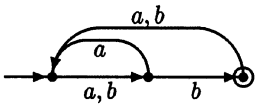
(v) $(ba^*a + ba^*b)^*ba^*$

14. Give an NFA with four states equivalent to the regular expression

$$(01 + 011 + 0111)^*.$$

Convert this automaton to an equivalent deterministic one using the subset construction. Name the states of your NFA, and show clearly which set of states corresponds to each state of the DFA. Omit inaccessible states.

15. Give a regular expression equivalent to the following automaton.



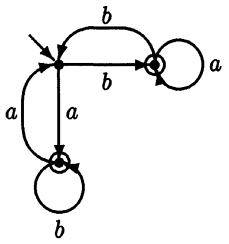
16. Give deterministic finite automata equivalent to the following regular expressions.

(a) $(00+11)^*\,(01+10)\,(00+11)^*$

(b) $(000)^*1+(00)^*1$

(c) $(0(01)^*(1+00)+1(10)^*(0+11))^*$

17. Give a regular expression equivalent to the following DFA.



18. Consider the regular sets denoted by the following pairs of regular expressions. For each pair, say whether the two sets are equal. If so, give a proof using the algebraic laws of Lecture 9; if not, give an example of a string in one that is not in the other.

(i)	$(0+1)^*$	0^*+1^*
(ii)	$0(120)^*12$	$01(201)^*2$
(iii)	$\emptyset^*$	ϵ^*
(iv)	$(0^*1^*)^*$	$(0^*1)^*$
(v)	$(01+0)^*0$	$0(10+0)^*$

19. Let $\alpha = (a+b)^*ab(a+b)^*)$. Give a regular expression equivalent to the pattern $\sim\alpha$ when

(a) $\Sigma = \{a,b\}$,

(b) $\Sigma = \{a,b,c\}$.

Simplify the expressions as much as possible.

20. Prove the following theorems of Kleene algebra. Reason equationally using axioms (A.1) through (A.15) only.

[S](a) $a^*a^* = a^*$

(b) $a^*a = aa^*$

(c) $a^{**} = a^*$

(d) $(a^*b)^*a^* = (a+b)^*$

(e) $a(ba)^* = (ab)^*a$

(f) $a^* = (aa)^* + a(aa)^*$

21. Prove Lemma A.1.

*22. Prove that in the presence of Kleene algebra axioms (A.1) through (A.11), axioms (A.12) and (A.14) are equivalent (and by symmetry, so are (A.13) and (A.15)).

*23. Here is a purely algebraic version of Miscellaneous Exercise 1. An element c of a Kleene algebra $\mathcal{K}$ is said to be *reflexive* if $1 \leq c$ and *transitive* if $cc \leq c$. We say that c *contains* a if $a \leq c$. Prove that for any $a \in \mathcal{K}$,

(a) a^* is reflexive and transitive and contains a; and

(b) a^* is the least element of $\mathcal{K}$ satisfying these properties. That is, if c is any element of $\mathcal{K}$ that is reflexive and transitive and contains a, then $a^* \leq c$.

This justifies the terminology *reflexive transitive closure.*

**24. Prove Lemma A.2: the family $\mathcal{M}(n, \mathcal{K})$ of $n \times n$ matrices over a Kleene algebra $\mathcal{K}$ with the matrix operations defined as in Supplementary Lecture A again forms a Kleene algebra.

S25. Prove Theorem A.3.

HS26. For any set of strings A, define the set

$$\text{FirstHalves } A = \{x \mid \exists y\ |y| = |x| \text{ and } xy \in A\}.$$

For example, FirstHalves $\{a, ab, bab, bbab\} = \{a, bb\}$. Show that if A is regular, then so is FirstHalves A.

27. For any set of strings A, define the set

$$\text{FirstThirds } A = \{x \mid \exists y\ |y| = 2|x| \text{ and } xy \in A\}.$$

For example, FirstThirds $\{\epsilon, a, ab, bab, bbab\} = \{\epsilon, b\}$. Show that if A is regular, then so is FirstThirds A.

28. Given a set $A \subseteq \{0,1\}^*$, let

$$A' = \{xy \mid x1y \in A\}.$$

That is, A' consists of all strings obtained from a string in A by deleting exactly one 1. Show that if A is regular, then so is A'.

29. For A a set of natural numbers, define

$$\textbf{binary } A = \{\text{binary representations of numbers in } A\} \subseteq (0+1)^*,$$
$$\textbf{unary } A = \{0^n \mid n \in A\} \subseteq 0^*.$$

For example, if $A = \{2, 3, 5\}$, then

$$\textbf{binary } A = \{10, 11, 101\},$$
$$\textbf{unary } A = \{00, 000, 00000\}.$$

Consider the following two propositions:

(i) For all A, if **binary** A is regular, then so is **unary** A.

(ii) For all A, if **unary** A is regular, then so is **binary** A.

One of (i) and (ii) is true and the other is false. Which is which? Give a proof and a counterexample.

*30. Let A be a regular set. Consider the two sets

$$\{x \mid \exists n \geq 0\ \exists y \in A\ y = x^n\},$$
$$\{x \mid \exists n \geq 0\ \exists y \in A\ x = y^n\}.$$

One is necessarily regular and one is not. Which is which? Give a proof and a counterexample.

*H31. One of the following subsets of $\{a, b, \$\}^*$ is regular and the other is not. Which is which? Give proofs.

$$\{xy \mid x, y \in \{a, b\}^*,\ \#a(x) = \#b(y)\}$$
$$\{x\$y \mid x, y \in \{a, b\}^*,\ \#a(x) = \#b(y)\}$$

***H32. Two of the following three sets are always regular for any regular set A. Which are they? Give two proofs and a counterexample.

(a) $\{x \mid x^{|x|} \in A\}$

(b) $\{x \mid \exists y\ |y| = 2^{2^{|x|}} \text{ and } xy \in A\}$

(c) $\{x \mid \exists y\ |y| = \log |x| \text{ and } xy \in A\}$

**HS33. Let p be any polynomial of degree d with nonnegative integer coefficients. Show that if A is a regular set, then so is the set

$$A' = \{x \mid \exists y\ |y| = p(|x|) \text{ and } y \in A\}.$$

***34. (Seiferas and McNaughton [113]) Let $f : \mathrm{N} \to \mathrm{N}$ be a function. Call f *regularity preserving* if the set

$$\{x \mid \exists y\ |y| = f(|x|) \text{ and } xy \in A\}$$

is regular whenever A is. Call f *weakly regularity preserving* if the set

$$\{x \mid \exists y\ |y| = f(|x|) \text{ and } y \in A\}$$

is regular whenever A is. Prove that the following statements are equivalent:

(a) f is regularity preserving;

(b) f is weakly regularity preserving;

(c) for any ultimately periodic set $U \subseteq \mathrm{N}$, the set

$$f^{-1}(U) = \{m \mid f(m) \in U\}$$

is also ultimately periodic; and

(d) for any $n \in \mathrm{N}$, the set $\{m \mid f(m) = n\}$ is ultimately periodic, and f is ultimately periodic mod p for any $p \geq 1$ in the sense that

$$\forall p \geq 1\ \exists q \geq 1\ \overset{\infty}{\forall}\, n\ f(n) \equiv f(n+q) \bmod p.$$

Here $\overset{\infty}{\forall}$ means "for almost all" or "for all but finitely many." Formally,

$$\overset{\infty}{\forall}\, n\ \varphi(n) \overset{\text{def}}{\iff} \exists m \geq 0\ \forall n \geq m\ \varphi(n).$$

S35. Show that the set $\{ww \mid w \in \{0,1\}^*\}$ is not regular.

S36. Show that the set

$$\text{PRIMES} \overset{\text{def}}{=} \{a^p \mid p \text{ is prime}\}$$

is not regular.

37. Which of the following sets are regular and which are not? Give justification.

(a) $\{a^n b^{2m} \mid n \geq 0 \text{ and } m \geq 0\}$

(b) $\{a^n b^m \mid n = 2m\}$

(c) $\{a^n b^m \mid n \neq m\}$

(d) $\{a^{p-1} \mid p \text{ is prime}\}$

(e) $\{xcx \mid x \in \{a,b\}^*\}$

(f) $\{xcy \mid x,y \in \{a,b\}^*\}$

(g) $\{a^n b^{n+481} \mid n \geq 0\}$

(h) $\{a^n b^m \mid n - m \leq 481\}$

(i) $\{a^n b^m \mid n \geq m \text{ and } m \leq 481\}$

(j) $\{a^n b^m \mid n \geq m \text{ and } m \geq 481\}$

(k) $L((a^*b)^*a^*)$

(l) $\{a^n b^n c^n \mid n \geq 0\}$

(m) {syntactically correct PASCAL programs}

38. For each of the following subsets of $\{0,1\}^*$, tell whether or not it is regular. Give proof.

(a) $\{x \mid \#1(x) = 2 \cdot \#0(x)\}$

(b) $\{x \mid \#1(x) - \#0(x) < 10\}$

(c) $\{x \mid \#1(x) \cdot \#0(x) \text{ is even}\}$

39. Prove that the set $\{a^n b^m c^k \mid n,m,k \geq 0,\ n+m=k\}$ is not regular.

40. Prove that the set $\{a^i b^j \mid i \text{ is even or } j < i\}$ is not regular.

41. Prove that no infinite subset of $\{a^n b^n \mid n \geq 0\}$ is regular.

**42. Give a set $A \subseteq \{a,b\}^*$ such that neither A nor $\{a,b\}^* - A$ contains an infinite regular subset. Prove that this is true of your set.

**H43. Give a nonregular set that satisfies condition (P) of the pumping lemma for regular sets (Lecture 11); that is, such that the demon has a winning strategy. Thus (P) is a necessary but not a sufficient condition for a set to be regular.

44. Prove the following stronger versions of the pumping lemma for regular sets that give necessary and sufficient conditions for a set to be regular.

*S(a) (Jaffe [62]) A set $A \subseteq \Sigma^*$ is regular if and only if there exists $k \geq 0$ such that for all $y \in \Sigma^*$ with $|y| = k$, there exist $u,v,w \in \Sigma^*$ such that $y = uvw$, $v \neq \epsilon$, and for all $z \in \Sigma^*$ and $i \geq 0$,

$$yz \in A \iff uv^i wz \in A.$$

*(b) (Stanat and Weiss [117]) A set $A \subseteq \Sigma^*$ is regular if and only if there exists $k \geq 0$ such that for all $y \in \Sigma^*$ with $|y| \geq k$, there exist $u, v, w \in \Sigma^*$ such that $y = uvw$, $v \neq \epsilon$, and for all $x, z \in \Sigma^*$ and $i \geq 0$,

$$xuz \in A \iff xuv^iz \in A.$$

45. Let A be any subset of $\{a\}^*$ whatsoever.

*H(a) Show that A^* is regular.

**S(b) Show that

$$A^* = \{a^{np} \mid n \geq 0\} - G,$$

where G is some finite set and p is the greatest common divisor of all elements of the set $\{m \mid a^m \in A\}$. This is a generalization of the so-called postage stamp problem: any amount of postage over 7 cents can be made with some combination of 3 and 5 cent stamps.

46. Prove that the DFA with 15 states shown in Lecture 5 for the set (5.1) is minimal.

47. Minimize the following DFAs. Indicate clearly which equivalence class corresponds to each state of the new automaton.

(a)

		a	b
→	1	6	3
	2	5	6
	$3F$	4	5
	$4F$	3	2
	5	2	1
	6	1	4

(b)

		a	b
→	1	2	3
	2	5	6
	$3F$	1	4
	$4F$	6	3
	5	2	1
	6	5	4

(c)

		a	b
→	$0F$	3	2
	$1F$	3	5
	2	2	6
	3	2	1
	4	5	4
	5	5	3
	6	5	0

(d)

		a	b
→	0	3	5
	1	2	4
	2	6	3
	3	6	6
	$4F$	0	2
	$5F$	1	6
	6	2	6

(e)

		a	b
→	0	3	5
	1	6	3
	2	6	4
	3	6	6
	$4F$	0	5
	$5F$	2	4
	6	1	6

(f)

		a	b
→	0	2	5
	1	6	2
	2	6	6
	3	6	4
	$4F$	5	0
	$5F$	4	3
	6	1	6

(g)

		a	b
→	$1F$	6	4
	$2F$	7	5
	3	2	8
	4	1	8
	5	2	6
	6	3	1
	7	5	2
	8	4	2

(h)

		a	b
→	1	6	2
	2	3	6
	$3F$	2	4
	$4F$	5	3
	5	4	1
	6	1	5
	7	1	8
	8	8	7

48. Consider the DFA with states $\mathbb{Z}_5 = \{0,1,2,3,4\}$, input alphabet $\{0,1\}$, start state 0, final state 0, and transition function

$$\delta(q,i) = (q^2 - i) \bmod 5, \quad q \in \mathbb{Z}_5,\ i \in \{0,1\}.$$

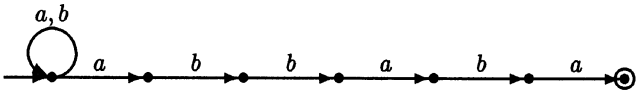
Prove that this DFA accepts exactly the set of binary strings containing an even number of 1's.

HS49. Prove the correctness of the collapsing algorithm of Lecture 14 (Theorem 14.3).

50. Let $\Sigma = \{a,b\}$. For any $x \in \Sigma^*$, define

$$\mathbf{suf}\, x = \{ux \mid u \in \Sigma^*\},$$

the set of strings ending with x. The set $\mathbf{suf}\, x$ is accepted by a nondeterministic finite automaton with $|x| + 1$ states. For example, here is a nondeterministic finite automaton for $\mathbf{suf}\, abbaba$:



(a) Draw the minimal *deterministic* finite automaton for $\mathbf{suf}\, abbaba$.

[H](b) Argue that for any x, the minimal deterministic finite automaton for **suf x has exactly $|x| + 1$ states.

**[H]51. (Greibach) Let M be an NFA, $A = L(M)$. Starting with M, do the following:

(a) reverse the transitions and interchange start and final states to get an NFA for **rev** A;

(b) determinize the resulting NFA by the subset construction, omitting inaccessible states;

(c) do the above two steps again.

Prove that the resulting automaton is the minimal DFA for A.

52. For each of the following finite automata:

(i) Give an equivalent minimal deterministic finite automaton. Don't forget to remove inaccessible states.

(ii) Give an equivalent regular expression.

(a)

		a	b
→	$1F$	2	5
	$2F$	1	4
	3	7	2
	4	5	7
	5	4	3
	6	3	6
	7	3	1

(b)

		a	b
→	$1F$	2	6
	$2F$	1	7
	3	5	2
	4	2	3
	5	3	1
	6	7	3
	7	6	5

(c)

		a	b
→	1	1	3
	$2F$	6	3
	3	5	7
	$4F$	6	1
	5	1	7
	$6F$	2	7
	7	5	3

(d)

		a	b
→	$1F$	2	5
	$2F$	1	6
	3	4	3
	4	7	1
	5	6	7
	6	5	4
	7	4	2

53. Consider the DFA with states $\mathbb{Z}_5 = \{0,1,2,3,4\}$, input alphabet $\{0,1\}$, start state 0, final state 0, and transition function

$$\delta(q,i) = (q^2 + i) \bmod 5, \quad q \in \mathbb{Z}_5,\ i \in \{0,1\}.$$

Give an equivalent minimal DFA.

54. Let $\equiv$ be any right congruence of finite index on Σ^*. Prove that any equivalence class of $\equiv$ is a regular subset of Σ^*.

55. Consider the regular set R represented by the regular expression

$$a^*b^* + b^*a^*.$$

(a) Draw the minimal DFA for R.

[H](b) Give a regular expression describing each of the equivalence classes of the Myhill–Nerode relation $\equiv_R$ defined in Lecture 16.

56. Let PAREN be the set of balanced strings of parentheses [ ]. Describe the equivalence classes of the relation $\equiv_{\text{PAREN}}$ defined in Lecture 16.

[**S]57. For strings x and y over a finite alphabet Σ, define $x \sqsubseteq y$ if x is a (not necessarily contiguous) substring of y; that is, if x can be obtained from y by deleting zero or more letters. For example,

$$abc \sqsubseteq ababac \sqsubseteq cabacbaac.$$

A subset $A \subseteq \Sigma^*$ is said to be *closed downward under* $\sqsubseteq$ if $x \in A$ whenever $x \sqsubseteq y$ and $y \in A$. Show that any subset of Σ^* closed downward under $\sqsubseteq$ is regular.

You may use *Higman's lemma*: any subset of Σ^* has a finite $\sqsubseteq$-base. A $\sqsubseteq$-*base* of a set X is a subset $X_0 \subseteq X$ such that for all $y \in X$ there exists an $x \in X_0$ such that $x \sqsubseteq y$. Higman's lemma is equivalent to saying that the set of $\sqsubseteq$-minimal elements of any $X \subseteq \Sigma^*$ is finite. You need not prove Higman's lemma.

58. (Kapur) Allow a concise representation of strings by using exponents to denote repeated substrings. For example,

$$0101010101000 = (01)^5 0^3 = 0(10)^5 0^2.$$

Denote by $/x/$ the length of the most concise representation of string x (exponents are given in binary). Let $|x|$ denote the ordinary length of x. Let R be a regular set.

*(a) Show that there exist constants c and d depending only on R such that for all $x \in R$, there exists another string $y \in R$ such that $|y| - |x| \leq d$ and $/y/ \leq c \log |y|$.

**(b) Answer the same question with the condition $|y| - |x| \leq d$ replaced by the condition $|y| = |x|$.

***S59. Here is a generalization of nondeterminism. An *alternating finite automaton* (AFA) is a 5-tuple

$$M = (Q, \Sigma, \delta, F, \alpha)$$

where Q is a finite set of *states*, Σ is a finite *input alphabet*, $F : Q \to \{0,1\}$ is the characteristic function of a set of *final states*, that is,

$$F(q) = \begin{cases} 1 & \text{if } q \text{ is a final state,} \\ 0 & \text{otherwise,} \end{cases}$$

δ is the *transition function*

$$\delta : (Q \times \Sigma) \to ((Q \to \{0,1\}) \to \{0,1\}),$$

and α is the *acceptance condition*

$$\alpha : (Q \to \{0,1\}) \to \{0,1\}.$$

Intuitively, a computation of M generates a computation tree whose depth is the length of the input string. The function F gives a labeling of 0 or 1 to the leaves of this computation tree. For all $q \in Q$ and $a \in \Sigma$, the Boolean function

$$\delta(q,a) : (Q \to \{0,1\}) \to \{0,1\}$$

takes a Boolean labeling on states at level i and computes a new labeling at level $i-1$; this is used to pass Boolean labels 0 or 1 back up the computation tree. The machine accepts if the labeling at level 0 satisfies α. An NFA is just an AFA in which the Boolean functions α and $\delta(q,a)$ compute the Boolean "or" of some subset of the inputs.

Formally, the transition function δ uniquely determines a map

$$\hat{\delta} : (Q \times \Sigma^*) \to ((Q \to \{0,1\}) \to \{0,1\}),$$

defined inductively as follows: for $q \in Q$, $a \in \Sigma$, and $x \in \Sigma^*$,

$$\hat{\delta}(q,\epsilon)(u) = u(q),$$
$$\hat{\delta}(q,ax)(u) = \delta(q,a)(\lambda p.(\hat{\delta}(p,x)(u))).$$

(Here "$\lambda p \ldots$" means "the function which on input p computes ...;" see Lecture 37.) The machine is said to *accept* $x \in \Sigma^*$ if

$$\alpha(\lambda p.(\hat{\delta}(p,x)(F))) = 1.$$

Prove that a set $A \subseteq \Sigma^*$ is accepted by a k-state alternating finite automaton if and only if its reverse $\mathbf{rev}\, A$ is accepted by a 2^k-state deterministic finite automaton.

[H]60. Show that minimal-state NFAs are not necessarily unique.

61. (Vardi [122]) In this exercise we show that two-way nondeterministic finite automata accept only regular sets. Let M be a 2NFA with states Q, start states S, accept state t, and transition function

$$\Delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to 2^{Q \times (\{L,R\})}.$$

Assume without loss of generality that whenever M accepts, it moves its head all the way to the right endmarker $\dashv$ and enters its accept state t.

(a) Let $x = a_1 a_2 \cdots a_n \in \Sigma^*$, $a_i \in \Sigma$, $1 \leq i \leq n$. Let $a_0 = \vdash$ and $a_{n+1} = \dashv$. Argue that x is *not* accepted by M iff there exist sets $W_i \subseteq Q$, $0 \leq i \leq n+1$, such that

- $S \subseteq W_0$;
- if $u \in W_i$, $0 \leq i \leq n$, and $(v, R) \in \Delta(u, a_i)$, then $v \in W_{i+1}$;
- if $u \in W_i$, $1 \leq i \leq n+1$, and $(v, L) \in \Delta(u, a_i)$, then $v \in W_{i-1}$; and
- $t \notin W_{n+1}$.

[H](b) Using (a), show that $\sim L(M)$, hence $L(M)$, is regular.

62. Prove Lemma B.8.

63. Prove that if a bisimulation between two NFAs is a one-to-one correspondence on the states, then it is an isomorphism.

64. Prove that if NFAs M and N are bisimilar, then the relation (B.1) of Supplementary Lecture B gives a bisimulation between the deterministic automata obtained from M and N by the subset construction.

65. Prove that two DFAs are bisimilar if and only if they accept the same set.

66. Prove Lemma C.11.

67. Prove Lemma D.2.

68. Prove that the relation $\equiv_A$ defined in the statement of the Myhill–Nerode theorem for term automata (Theorem D.3) is a congruence on the term algebra $T_\Sigma(A)$.

Miscellaneous Exercises

Pushdown Automata and Context-Free Languages

69. Consider the following context-free grammar G:

$$S \rightarrow ABS \mid AB,$$
$$A \rightarrow aA \mid a,$$
$$B \rightarrow bA.$$

Which of the following strings are in $L(G)$ and which are not? Provide derivations for those that are in $L(G)$ and reasons for those that are not.

(a) $aabaab$

(b) $aaaaba$

(c) $aabbaa$

(d) $abaaba$

*H70. Consider the context-free grammar G with start symbol S and productions

$$S \rightarrow aAB \mid aBA \mid bAA \mid \epsilon,$$
$$A \rightarrow aS \mid bAAA,$$
$$B \rightarrow aABB \mid aBAB \mid aBBA \mid bS.$$

Prove that $L(G)$ is the language consisting of all words that have exactly twice as many a's as b's.

71. Give a grammar with no ϵ- or unit productions generating the set $L(G) - \{\epsilon\}$, where G is the grammar

$$S \to aSbb \mid T,$$
$$T \to bTaa \mid S \mid \epsilon.$$

72. Give grammars in Chomsky and Greibach normal form for the following context-free languages.

(a) $\{a^n b^{2n} c^k \mid k, n \geq 1\}$

(b) $\{a^n b^k a^n \mid k, n \geq 1\}$

(c) $\{a^k b^m c^n \mid k, m, n \geq 1,\ 2k \geq n\}$

(d) $\{a, b\}^* - \{\text{palindromes}\}$

73. Let $\Sigma = \{0, 1\}$. Let $\overline{x}$ denote the Boolean complement of x; that is, the string obtained from x by changing all 0's to 1's and 1's to 0's. Let **rev** x denote the reverse of x; that is, the string x written backwards. Consider the set

$$A = \{x \mid \mathbf{rev}\ x = \overline{x}\}.$$

For instance, the strings 011001 and 010101 are in A but 101101 is not.

(a) Give a CFG for this set.

(b) Give grammars in Chomsky and Griebach normal form for $A - \{\epsilon\}$.

*74. Consider the set of all strings over $\{a, b\}$ with no more than twice as many a's as b's:

$$\{x \in \{a, b\}^* \mid \#a(x) \leq 2\#b(x)\}.$$

(a) Give a CFG for this set, and prove that it is correct.

(b) Give a pushdown automaton for this set. Specify completely all data (states, transitions, etc.) and whether your machine accepts by final state or empty stack. Show sample runs on the input strings *aabbaa*, *aaabbb*, and *aaabaa*.

75. Our definition of patterns and regular expressions in Lecture 7 was a little imprecise since it did not mention parentheses. Make this definition precise by using a CFG to specify the set of regular expressions over an alphabet Σ. The grammar you come up with should have terminal symbols $\Sigma \cup \{\epsilon, \emptyset, +, \cdot, (,), ^*\}$.

76. Consider the set

$$a^*b^*c^* - \{a^n b^n c^n \mid n \geq 0\},$$

the set of all strings of a's followed by b's followed by c's such that the number of a's, b's, and c's are not all equal.

(a) Give a CFG for the set, and prove that your grammar is correct.

(b) Give an equivalent PDA.

*S77. What set is generated by the following grammar?

$$S \rightarrow bS \mid Sa \mid aSb \mid \epsilon$$

Give proof.

S78. For $A, B \subseteq \Sigma^*$, define

$$A/B \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \exists y \in B \; xy \in A\}.$$

Prove that if L is a CFL and R is a regular set, then L/R is CFL.

H79. Show that the context-free languages are closed under homomorphic images and preimages.

H80. (Ginsburg and Rice [45]) Show that any context-free subset of $\{a\}^*$ is regular.

81. The specification of **while** programs at the beginning of Supplementary Lecture I is rather imprecise. Give a rigorous context-free specification.

82. Prove that the set

$$\text{PRIMES} \stackrel{\text{def}}{=} \{a^p \mid p \text{ is prime}\}$$

is not context-free.

**H83. Show that $\{a, b\}^* - \{a^n b^{n^2} \mid n \geq 0\}$ is not context-free.

84. Which of the following sets are context-free and which are not? Give grammars for those that are context-free and proof for those that are not.

(a) $\{a^n b^m c^k \mid n, m, k \geq 1 \text{ and } (2n = 3k \text{ or } 5k = 7m)\}$

(b) $\{a^n b^m c^k \mid n, m, k \geq 1 \text{ and } (2n = 3k \text{ and } 5k = 7m)\}$

(c) $\{a^n b^m c^k \mid n, m, k \geq 1 \text{ and } (n \neq 3m \text{ or } n \neq 5k)\}$

(d) $\{a^n b^m c^k \mid n, m, k \geq 1 \text{ and } (n \neq 3m \text{ and } n \neq 5k)\}$

(e) $\{a^n b^m c^k \mid n, m, k \geq 1 \text{ and } n + k = m\}$

(f) $\{a^i b^j c^k d^\ell \mid i, j, k, \ell \geq 1,\ i = j,\ k = \ell\}$

(g) $\{a^i b^j c^k d^\ell \mid i, j, k, \ell \geq 1,\ i = k,\ j = \ell\}$

(h) $\{a^i b^j c^k d^\ell \mid i, j, k, \ell \geq 1,\ i = \ell,\ j = k\}$

85. Say whether the following sets are (i) regular, (ii) context-free but not regular, or (iii) not context-free. Give justification.

(a) $\{x \in \{a, b, c\}^* \mid \#a(x) = \#b(x) = \#c(x)\}$

(b) $\{a^j \mid j \text{ is a power of } 2\}$

(c) $\{x \in \{0, 1\}^* \mid x \text{ represents a power of 2 in binary}\}$

(d) $L(a^* b^* c^*)$

(e) the set of all balanced strings of parentheses of three types, `( ) [ ] { }`

(f) $\{a^n b^m \mid n \neq m\}$

(g) $\{a^n b^m c^k d^\ell \mid 2n = 3k \text{ or } 5m = 7\ell\}$

(h) $\{a^n b^m c^k d^\ell \mid 2n = 3k \text{ and } 5m = 7\ell\}$

(i) $\{a^n b^m c^k d^\ell \mid 2n = 3m \text{ and } 5k = 7\ell\}$

(j) $\{a^n b^m c^k d^\ell \mid 2n = 3\ell \text{ and } 5k = 7m\}$

(k) $\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i > j \text{ and } j > k\}$

(l) $\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } (i > j \text{ or } j > k)\}$

(m) $\{x \in \{a, b\}^* \mid \#a(x) > \#b(x)\}$

(n) $\{a^m b^n \mid m, n \geq 0,\ 5m + 3n = 24\}$

(o) $\{a^m b^n \mid m, n \geq 0,\ 5m - 3n = 24\}$

**[H]86. Give a non-context-free set that satisfies the condition of the pumping lemma for CFLs given in Lecture 22; that is, such that the demon has a winning strategy.

87. Let Σ be a finite alphabet. For a set $A \subseteq \Sigma^*$, define

$$\textbf{cycle}\ A = \{yx \mid xy \in A\},$$
$$\textbf{permute}\ A = \{y \mid \exists x \in A\ \forall a \in \Sigma\ \#a(x) = \#a(y)\}.$$

For example, if $\Sigma = \{a, b, c\}$ and $A = \{aaabc\}$, then

$$\textbf{cycle}\ A = \{aaabc, aabca, abcaa, bcaaa, caaab\},$$
$$\textbf{permute}\ A = \{aaabc, aabca, abcaa, bcaaa, caaab,$$
$$aabac, abaca, bacaa, acaab, caaba,$$
$$abaac, baaca, aacab, acaba, cabaa,$$
$$baaac, aaacb, aacba, acbaa, cbaaa\}.$$

Which of the following propositions are true and which are false? Give proof.

(a) For all $A \subseteq \Sigma^*$, if A is regular, then so is **cycle** A.

(b) For all $A \subseteq \Sigma^*$, if A is regular, then so is **permute** A.

[H](c) For all $A \subseteq \Sigma^*$, if A is context-free, then so is **cycle A.

(d) For all $A \subseteq \Sigma^*$, if A is context-free, then so is **permute** A.

88. Recall the shuffle operator $\|$ from Homework 4.

(a) Show that if L is context-free and R is regular, then $L \| R$ is context-free.

*(b) If L is a DCFL, is $L \| R$ necessarily a DCFL? Give proof.

*89. For $A, B \subseteq \Sigma^*$, define

$$A/B \stackrel{\text{def}}{=} \{x \mid \exists y \in B\ xy \in A\}$$
$$A \leftarrow B \stackrel{\text{def}}{=} \{x \mid \forall y \in B\ xy \in A\}.$$

Exactly one of the following two statements is true.

(a) If L is context-free, then so is L/Σ^*.

(b) If L is context-free, then so is $L \leftarrow \Sigma^*$.

Which is true? Give a proof and a counterexample.

90. Let $\Sigma = \{a, b, c\}$. Exactly one of the following four statements is true.

(a) For any $A \subseteq \Sigma^*$, if A is regular, then so is $\{xx \mid x \in A\}$.

(b) For any $A \subseteq \Sigma^*$, if A is regular, then so is $\{x \mid xx \in A\}$.

(c) For any $A \subseteq \Sigma^*$, if A is context-free, then so is $\{xx \mid x \in A\}$.

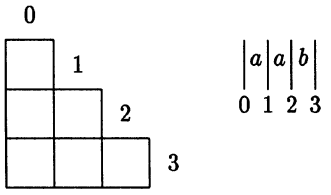
*H(d) For any $A \subseteq \Sigma^*$, if A is context-free, then so is $\{x \mid xx \in A\}$.

Which is true? Give a proof and three counterexamples.

91. Using the grammar

$$S \to AB, \qquad A \to a, \qquad B \to AB \mid b,$$

run the CKY algorithm on the string aab. Draw a table like the following one and fill it in completely.



92. (a) Modify the CKY algorithm to count the number of parse trees of a given string and to construct one if the number is nonzero.

(b) Test your algorithm of part (a) on the grammar

$$S \to ST \mid a,$$
$$T \to BS,$$
$$B \to +$$

and string

$$a + a + a + a.$$

(Sanity check: the string has five parse trees.)

***H93. Let $D \subseteq \Sigma^*$ be a DCFL. One of the following sets is always a DCFL, the other is not necessarily. Which is which? Give proof for both.

(a) $\{x \mid \exists a \in \Sigma\ xa \in D\}$

(b) $\{x \mid \exists a \in \Sigma\ ax \in D\}$

Conclude that the family of DCFLs is not closed under reversal.

94. Let Σ be a fixed finite signature as described in Supplementary Lecture C. Give an unambiguous context-free grammar for the set of ground terms over Σ. *Unambiguous* means that there is exactly one parse tree for each ground term. Prove that your grammar is correct and that it is unambiguous.

**95. The context-free language $\{a^n b^n \mid n \geq 0\}$ is the unique $\subseteq$-minimal solution of the equation

$$\mathbf{X} = a\mathbf{X}b + \epsilon.$$

In general, let Σ be a finite alphabet. Consider finite systems of equations of the form

$$\begin{aligned} \mathbf{X}_1 &= \mathcal{E}_1(\mathbf{X}_1, \ldots, \mathbf{X}_n), \\ &\vdots \\ \mathbf{X}_n &= \mathcal{E}_n(\mathbf{X}_1, \ldots, \mathbf{X}_n), \end{aligned}$$

where the $\mathbf{X}_i$ are variables ranging over subsets of Σ^* and the $\mathcal{E}_i$ are regular expressions over $\mathbf{X}_1, \ldots, \mathbf{X}_n$ and Σ.

(a) Argue that any such system has a unique minimal solution

$$X_1, \ldots, X_n \in (2^{\Sigma^*})^n.$$

You may want to use the *Knaster–Tarski theorem*: any monotone map on a complete partial order has a unique least fixpoint. A map f on a partially ordered set is *monotone* if $x \leq y \rightarrow f(x) \leq f(y)$. A partially ordered set is *complete* if every subset of that set has a least upper bound.

(b) For the $X_1, \ldots, X_n$ of part (a), show that X_1 is a context-free language.

(c) Show that all context-free languages arise in this way.

Miscellaneous Exercises

Turing Machines and Effective Computability

96. Give a Turing machine with input alphabet $\{a\}$ that on input a^m halts with a^{m^2} written on its tape. Describe the operation of the machine both informally and formally. Be sure to specify all data.

97. The Euclidean algorithm computes the greatest common divisor (GCD) of two nonnegative integers:

```
procedure gcd(m,n):
if n = 0 then return(m)
         else return(gcd(n,m mod n))
```

Give a Turing machine with input alphabet $\{a, \#\}$ that on input $a^m \# a^n$ halts with $a^{\gcd(m,n)}$ written on its tape. Describe the operation of the machine both informally and formally. Be sure to specify all data.

98. Prove that the class of r.e. sets is closed under union and intersection.

99. A *queue machine* is like a Turing machine, except that it has a queue instead of a tape. It has a finite queue alphabet Γ and a finite input alphabet $\Sigma \subseteq \Gamma$. If $x \in \Sigma^*$ is the input, the machine starts in its start state s with $x\$$ in the queue, where $\$$ is a special symbol in $\Gamma - \Sigma$. In each step, it removes a symbol from the front of the queue. Based on

that symbol and the current state, it pushes a string $z \in \Gamma^*$ onto the back of the queue and enters a new state according to the transition function δ. It accepts by emptying its queue.

[S](a) Give a rigorous formal definition of these machines, including a definition of configurations and acceptance. Your definition should begin as follows: "A *queue machine* is a sextuple

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

where ..."

[***HS](b) Prove that queue machines and Turing machines are equivalent in power.

100. A *one-counter automaton* is an automaton with a finite set of states Q, a two-way read-only input head, and a separate counter that can hold any nonnegative integer. The input $x \in \Sigma^*$ is enclosed in endmarkers $\vdash, \dashv \notin \Sigma$, and the input head may not go outside the endmarkers. The machine starts in its start state s with its counter empty and with its input head pointing to the left endmarker $\vdash$. In each step, it can test its counter for zero. Based on this information, its current state, and the symbol its input head is currently reading, it can either add one to its counter or subtract one, move its input head either left or right, and enter a new state. It accepts by entering a distinguished final state t.

(a) Give a rigorous formal definition of these machines, including a definition of acceptance. Your definition should begin as follows: "A *one-counter automaton* is a septuple

$$M = (Q, \Sigma, \vdash, \dashv, s, t, \delta),$$

where ..."

[*](b) Prove that the membership problem for deterministic one-counter automata is decidable: given M and x, does M accept x?

[H](c) Prove that the emptiness problem is undecidable: given a one-counter automaton M, is $L(M) = \varnothing$?

101. A *ray automaton* consists of an infinite number of deterministic finite automata $A_0, A_1, A_2, \ldots$ arranged in a line. The automata all have the same set of states Q, the same start state s, and the same transition function δ except A_0, which has a different transition function δ_0 since it has no left neighbor. They all start simultaneously in their initial state s and execute synchronously. In each step, each A_i moves to a new state, depending on its own current state and the current states of its

immediate left and right neighbors, according to its transition function. The ray automaton is said to *halt* if A_0 ever enters a distinguished final state t. There is no input alphabet.

(a) Give a rigorous formal definition of ray automata, including a definition of execution and halting. Your definition should begin as follows: "A *ray automaton* is a quintuple

$$\mathcal{A} = (Q, s, t, \delta_0, \delta),$$

where Q is a finite set of *states*, ..."

(b) Prove that the halting problem for ray automata is undecidable.

(c) Is the halting problem for ray automata semidecidable? Why or why not?

102. A *deterministic two-dimensional Turing machine* is like a Turing machine except that instead of a one-dimensional tape it has a two-dimensional tape that is like a chessboard, infinite in all directions. It has a finite input alphabet Σ and a finite tape alphabet Γ containing Σ as a subset. If $x \in \Sigma^*$ is the input, $|x| = n$, the machine starts in its start state s with x written in tape cells $(0,1), (0,2), \ldots, (0,n)$, the origin $(0,0)$ containing a special symbol $O \in \Gamma - \Sigma$, and all other cells (i,j) containing a special blank symbol $\sqcup \in \Gamma - \Sigma$. It has a read/write head initially pointing to the origin. In each step, it reads the symbol of Γ currently occupying the cell it is scanning. Depending on that symbol and the current state of the finite control, it writes a symbol of Γ on that cell, moves one cell either north, south, east, or west, and enters a new state, according to its transition function δ. It accepts its input by erasing the entire board; that is, filling all cells with $\sqcup$.

H(a) Give a rigorous formal definition of these machines, including a definition of configurations, the next configuration relation, and acceptance. Try to be as precise as possible. Your definition should begin as follows: "A *two-dimensional Turing machine* is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \sqcup, O, s, \delta),$$

where Q is a finite set of *states*, ..."

(b) Argue that two-dimensional Turing machines and ordinary Turing machines are equivalent in the sense that each can simulate the other. Describe the simulations *informally* (i.e., no transitions) but in sufficient detail that transitions implementing your description could readily be written down.

103. A nondeterministic Turing machine is one with a multiple-valued transition relation. Give a formal definition of these machines. Argue that every nondeterministic TM can be simulated by a deterministic TM.

104. Show that the type 0 grammars (see Lecture 36) generate exactly the r.e. sets.

105. For $A, B \subseteq \Sigma^*$, define

$$A/B \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid \exists y \in B \; xy \in A\}.$$

(a) Show that if A and B are r.e., then so is A/B.

*H(b) Show that every r.e. set can be represented as A/B with A and B CFLs.

106. Is it decidable, given $M\#y$, whether the Turing machine M ever writes a nonblank symbol on its tape on input y? Why or why not?

107. Is it decidable for TMs M whether $L(M) = \mathbf{rev}\, L(M)$? Give proof.

108. Tell whether the following problems are decidable or undecidable. Give proof.

(a) Given a TM M and a string y, does M ever write the symbol # on its tape on input y?

(b) Given a CFG G, does G generate all strings except ϵ?

(c) Given an LBA M, does M accept a string of even length?

(d) Given a TM M, are there infinitely many TMs equivalent to M?

109. Tell whether or not the following sets are r.e. Give proof.

(a) $\{(M, N) \mid M$ takes fewer steps than N on input $\epsilon\}$

(b) $\{M \mid M$ takes fewer than 481^{481} steps on some input$\}$

(c) $\{M \mid M$ takes fewer than 481^{481} steps on at least 481^{481} different inputs$\}$

(d) $\{M \mid M$ takes fewer than 481^{481} steps on all inputs$\}$

110. Show that the set $\{M \mid M$ accepts at least 481 strings$\}$ is r.e. but not co-r.e.

111. One of the following sets is r.e. and the other is not. Which is which? Give proof for both.

    (a) $\{M \mid L(M)$ contains at least 481 elements$\}$

    (b) $\{M \mid L(M)$ contains at most 481 elements$\}$

112. Show that the set

$$\{M \mid M \text{ halts on all inputs of length less than } 481\}$$

    is r.e., but its complement is not.

[S]113. Let M range over Turing machine descriptions. Show that neither the set

$$\text{REG} \stackrel{\text{def}}{=} \{M \mid L(M) \text{ is a regular set}\}$$

    nor its complement is recursively enumerable.

114. Let $|M|$ denote the length of the description of the Turing machine M. Are the following problems decidable? Give proof.

    (a) Does a given Turing machine M take at least $|M|$ steps on some input?

    (b) ...on all inputs?

115. Tell whether the following problems are decidable or undecidable, and give proof:

    (a) whether a given TM runs for at least 481^{481} steps on input a^{481};

    (b) whether a given TM ever reenters its start state on any input;

    *(c) whether a given Turing machine will ever move its head left more than ten times on input a^{481};

    *(d) whether a given Turing machine will ever print more than 481 nonblank symbols on input a^{481}.

116. Think for two minutes about why the following problems are undecidable, but don't write anything down:

    (a) whether two given C++ programs compute the same function;

    (b) whether a given C++ program will ever get into an infinite loop on some input;

(c) whether a given μ-recursive function is total;

(d) whether a given λ-term reduces to normal form.

117. Show that the following problems of pairs of Turing machines are undecidable:

(a) whether $L(M) = L(N)$;

(b) whether $L(M) \subseteq L(N)$;

(c) whether $L(M) \cap L(N) = \varnothing$;

(d) whether $L(M) \cap L(N)$ is a recursive set;

(e) whether $L(M) \cap L(N)$ is finite.

**118. Formalize and prove the following extension of Rice's theorem that has the results of Exercise 117 as special cases: every nontrivial property of *pairs* of r.e. sets is undecidable.

119. Let G and G' denote context-free grammars over $\{a, b\}$. Prove that the following problems are undecidable:

H(a) whether $L(G) = L(G')$;

(b) whether $L(G) \subseteq L(G')$;

*(c) whether $L(G) = L(G)L(G)$.

120. One of the following problems is decidable and the other is not. Which is which? Give proof for both.

(a) Given a CFL L and a regular set R, is $L \subseteq R$?

(b) Given a CFL L and a regular set R, is $R \subseteq L$?

H121. Prove that the following problems are undecidable:

(a) whether a given CFL is a DCFL;

(b) whether the intersection of two given CFLs is a CFL;

(c) whether the complement of a given CFL is a CFL;

**(d) whether the union of two given DCFLs is a DCFL.

122. Prove that it is undecidable whether a given LBA halts on all inputs.

[H]123. Show that the finiteness problem for Turing machines reduces to the finiteness problem for LBAs.

124. Prove that it is undecidable whether a given LBA accepts a regular set.

125. Consider the following context-sensitive productions.

$$S \to bSb,$$
$$S \to AcA,$$
$$Ab \to A,$$
$$Ab \to b,$$
$$bA \to b,$$
$$bA \to A.$$

Let G be the grammar given by all the rules except for the last, and let G' the grammar given by all the rules including the last. One of $L(G)$ and $L(G')$ is regular, and the other is context-free but not regular. Which is which, and why?

126. Give a set over a *single letter alphabet* in each of the following classes, or explain why such a set does not exist:

(a) regular;

(b) DCFL but not regular;

(c) CFL but not DCFL;

(d) recursive but not CFL;

(e) r.e. but not recursive;

(f) not r.e.

127. Prove that every infinite regular set contains a non-r.e. subset.

*[H]128. Prove that every infinite r.e. set contains an infinite recursive subset.

*129. In this exercise we will prove a kind of fixpoint theorem for Turing machines known as the *recursion theorem.*

If M is a TM, let $M(x)$ denote the contents of M's tape at the point that M halts on input x, provided M does indeed halt on input x. If M does not halt on input x, then $M(x)$ is undefined.

A partial function $\sigma : \Sigma^* \to \Sigma^*$ is said to be a *computable function* if $\sigma(x) = M(x)$ for some Turing machine M. In addition, σ is a *total computable function* if M is total.

Let M_x be the TM whose encoding over Σ^* is x.

Theorem **(Recursion theorem)** *Let $\sigma : \Sigma^* \to \Sigma^*$ be any total computable function. Then there exists a string u such that*

$$L(M_u) = L(M_{\sigma(u)}).$$

(a) Let $\sigma : \Sigma^* \to \Sigma^*$ be a given total computable function, say computable by a total TM K. Let N be a TM that on input x computes a *description* of a machine that does the following on input y:

- constructs M_x;
- runs M_x on input x;
- if it halts, runs K on $M_x(x)$;
- interprets the result of that computation, $K(M_x(x))$, as the description of a TM, and simulates that TM on the original input y, accepting or rejecting as that machine accepts or rejects, respectively.

Argue that N is total and that

$$L(M_{N(x)}) = L(M_{\sigma(M_x(x))}).$$

(b) Let v be a description of the machine N; that is, $N = M_v$. Argue that $N(v)$ is the desired fixpoint of σ.

[H]130. Give a short proof of Rice's theorem using the recursion theorem (see Miscellaneous Exercise 129).

[**H]131. A TM is *minimal* if it has the fewest states among all TMs that accept the same set. Prove that there does not exist an infinite r.e. set of minimal TMs.

132. [H](a) Show that there does not exist an r.e. list of Turing machines such that every machine on the list is total (i.e., halts on all inputs) and every recursive set is represented by some machine on the list.

[**H](b) Show that there exists an r.e. list of Turing machines such that every machine on the list accepts a recursive set and every recursive set is represented by some machine on the list.

133. In addition to the usual constructs of **while programs (simple assignment, conditional, while loop, sequential composition), add a print statement

print x **and halt**

that prints the current value of a variable and halts. Call two programs *equivalent* if for all initial values of the variables, one program halts iff the other does, and whenever they both halt, they print the same value.

One of the following problems is decidable and the other is undecidable. Which is which? Justify your answers.

(a) Given a program, does there exist an equivalent program with at most one **while** loop?

(b) Given a program, does there exist an equivalent program with no **while** loops?

134. This question is for those who know something about propositional logic. A *propositional Horn clause* is a disjunction of literals with at most one positive literal. The clause

$$\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n \vee Q$$

is often written as

$$P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow Q,$$

and the clause

$$\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n$$

can be written as

$$P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow \bot,$$

where $\bot$ denotes falsity. Any single positive literal Q is also a Horn clause.

(a) Show that the emptiness problem for context-free languages (i.e., given a context-free grammar G, deciding whether $L(G) = \varnothing$) reduces to the satisfiability problem for finite conjunctions of Horn clauses, and vice versa.

(b) Since the satisfiability of propositional formulas is decidable, what can we conclude about the decidability of the emptiness problem for CFLs?

[H]135. Show that the finiteness problem for regular sets and context-free languages (i.e., whether a given machine/grammar accepts/generates a finite set) is decidable.

136. Show that FIN $\leq_T$ REG. In other words, suppose you are given an oracle that will always answer questions of the form "Is $L(M)$ a regular set?" truthfully. Show how to use such an oracle to decide questions of the form "Is $L(M)$ finite?"

**137. Prove Theorem J.1.

*[H]138. Let $HP_1 \stackrel{\text{def}}{=} HP$, and let HP_{n+1} be the halting problem for oracle Turing machines with oracle HP_n, $n \geq 1$; that is,

$$HP_{n+1} \stackrel{\text{def}}{=} \{M\#x \mid M \text{ is an oracle TM with oracle } HP_n, \\ M \text{ halts on input } x\}.$$

The oracle need not be represented in the description of the oracle machine M. Show that $HP_n \in \Sigma_n^0 - \Pi_n^0$.

139. Show that the integer square root function is primitive recursive. On input n, the function should return the greatest integer less than or equal to the square root of n.

[H]140. A language B is said to be *computable in linear time* if there exists a deterministic Turing machine M and a constant $c > 0$ such that $L(M) = B$ and M always halts within cn steps on inputs of length n. Show that there exists a recursive set that is not computable in linear time.

141. Show that the Turing reducibility relation $\leq_T$ is reflexive and transitive and that $\leq_m$ refines $\leq_T$.

142. Prove that the following sets are $\leq_m$-complete for the given classes:

(a) EMPTY is $\leq_m$-complete for Π_1^0;

*(b) TOTAL is $\leq_m$-complete for Π_2^0;

**(c) COF is $\leq_m$-complete for Σ_3^0;

**(d) the set

$$REG \stackrel{\text{def}}{=} \{M \mid L(M) \text{ is a regular set}\}$$

is $\leq_m$-complete for Σ_3^0.

*[H]143. Prove that there exists a total computable function $f : \mathbb{N} \to \mathbb{N}$ that is not provably total in Peano arithmetic.

Hints for Selected Miscellaneous Exercises

6. Look for a clue in Lecture 5.

9. (b) Build an NFA with seven states arranged in loops of length two and five. Assign start and final states so that the shortest rejected string is of length $9 = 2 \cdot 5 - 1$.

   (c) The product of a set of distinct primes is exponential in their sum.

26. Given a DFA M for A, build an NFA M' for FirstHalves A that implements the following idea: put a white pebble on the start state of M and green and blue pebbles on an arbitrary guessed state of M. Never move the green pebble. Move the white pebble forward in response to input symbols and move the blue pebble forward according to some nondeterministically guessed input symbol. Accept if the white and green pebbles occupy the same state and the blue pebble occupies an accept state when the input string is exhausted. Describe M' formally and prove that it accepts the set FirstHalves A. Presumably the set of states of M' will be the set $Q \times Q \times Q$ encoding the positions of the three pebbles.

    In general, you will see several problems of the form

    show that if A is a regular set, then so is A',

where A' is some set formed by massaging A in some way. Exercise 2 of Homework 2 about **rev** A and Exercise 3 of Homework 3 about MiddleThirds A are of this type. Most of these problems can be solved by applying the following five-step protocol:

Step 1 Assume we are given a deterministic finite automaton

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

accepting A. We want to build a nondeterministic automaton

$$M' = (Q', \Sigma, \Delta', S', F')$$

accepting A'. Come up with an intuitive design of M' in terms of moving pebbles around on the states of M. Think about the initial configuration of pebbles, how the pebbles should move in response to each input symbol, and what the accepting configurations should be.

Step 2 Write down a formal description of Q', Σ, Δ', S', and F' that formally captures your intuition about moving the pebbles developed in step 1. The first thing to think about is what the states Q' should be. You need to figure out how to encode formally the information that the new machine needs to remember at each step. Make sure the types are right; for example, whatever you decide the set of states Q' should be, the start state should be an element of Q' and the set of accept states F' should be a subset of Q'. If you are designing a deterministic machine M', then δ' should be a function $Q' \times \Sigma \to Q'$. If M' is to be nondeterministic, then you should have $\Delta' : Q' \times \Sigma \to 2^{Q'}$.

Step 3 In step 2, you defined a transition function Δ' of M'. Most likely, Δ' was defined in terms of the transition function δ of M. State a lemma extending this relationship to a relationship between $\widehat{\delta}$ and $\widehat{\Delta}'$.

Step 4 Prove the lemma stated in step 3 by induction on $|x|$. The proof will most likely use the standard inductive definitions of $\widehat{\delta}$ and $\widehat{\Delta}'$, as well as the definition you gave in step 2 of Δ' in terms of δ.

Step 5 Prove that $L(M') = A'$. The proof will generally use the lemma proved in step 4 and the definitions of S' and F'.

Step 1 is usually not much of a problem, since it is usually easy to see how to move the pebbles. Steps 2 and 3 typically give the most trouble. If the lemma in step 3 is formulated correctly, the proofs in 4 and 5 should be fairly routine.

An example of an application of this protocol is given in the solution to this exercise on p. 358.

31. One of the sets is $\{a, b\}^*$.

32. Use the matrix representation of Miscellaneous Exercise 7.

33. Use the fact that

$$p(n+1) = \sum_{i=0}^{d} \frac{p^{(i)}(n)}{i!},$$

where $p^{(i)}$ denotes the ith derivative of p.

43. Try

$$\bigcup_{n \geq 0} (a^+c)^n (b^+c)^n + (a+b+c)^* cc(a+b+c)^*$$

with $k = 3$.

45. (a) By Theorem 12.3, it suffices to show that A^* is ultimately periodic. You can take as period the length of the smallest nonnull element of A. This is not necessarily the smallest period, but it will do.

49. For the direction ($\Rightarrow$), use induction on the stages of the algorithm. For the direction ($\Leftarrow$), use induction on $|x|$.

50. (b) Consider the relation

$$y \equiv z \overset{\text{def}}{\iff} \text{overlap}(y, x) = \text{overlap}(z, x),$$

where $\text{overlap}(y, x)$ is the longest string that is both a suffix of y and a prefix of x. Use the Myhill–Nerode theorem.

51. Suppose that you start with a DFA for B with no inaccessible states, reverse the transitions and exchange the start and final states to get an NFA for **rev** B, then construct an equivalent DFA using the subset construction, omitting inaccessible states. Prove that the resulting automaton is the minimal DFA for **rev** B.

55. (b) Two of them are ϵ and aa^*.

60. $aa^* \equiv a^*a$.

61. (b) Build an NFA whose states are subsets of Q. Let the machine guess the sets W_i.

70. Prove inductively that

$$S \xrightarrow[G]{*} x \iff \#a(x) = 2\#b(x),$$
$$A \xrightarrow[G]{*} x \iff \#a(x) = 2\#b(x) + 1,$$
$$B \xrightarrow[G]{*} x \iff \#a(x) = 2\#b(x) - 2.$$

Think about the graph of the function $\#a(y) - 2\#b(y)$ for prefixes y of x.

79. Everything you need can be found in Lecture 10.

80. Use Parikh's theorem (Theorem H.1) and the theorem on ultimate periodicity (Theorem 12.3).

83. Use Parikh's theorem (Theorem H.1) and the fact that the complement of a semilinear subset of $\mathbb{N}^k$ is semilinear.

86. Show that the condition (P) of the pumping lemma for regular sets given in Lecture 11 implies the condition of the pumping lemma for context-free languages given in Lecture 22. Give a non-context-free set satisfying (P). A slightly modified version of the hint for Miscellaneous Exercise 43 should do.

87. (c) Build a PDA that pushes and pops antimatter.

90. (d) Consider the set

$$A = \{a^n b^n c^m a^m b^k c^k \mid n, m, k \geq 1\}.$$

93. (a) Let

$$M = (Q, \Sigma, \Gamma, \delta, \perp, \dashv, s, \varnothing)$$

be a DPDA for D that accepts by empty stack. Prove that for $p, q \in Q$ and $a \in \Sigma$,

$$\{\gamma \in \Gamma^* \mid (p, a, \gamma) \xrightarrow[M]{*} (q, \epsilon, \epsilon)\}$$

is a regular set.

(b) See the end of Lecture 27.

99. (b) Simulating a queue machine with a Turing machine is easy. The other direction is tricky. Make the queue of the queue machine contain a representation of the configuration of the Turing ma-

chine. The hard part is simulating a left move of the Turing machine. You need to go all the way around the queue.

You might try breaking your solution into two steps:

(i) First invent a new kind of Turing machine, a *one-way* Turing machine. These machines can only move right on the tape. When they see the right endmarker, they magically jump back to the left endmarker. Show that one-way machines can simulate ordinary TMs. Simulate a left move of the ordinary TM by pushing a marker all the way around to the right.

(ii) Simulate one-way machines with queue machines.

100. (c) Think VALCOMPS.

102. (a) The infinite checkerboard should be $\mathbb{Z} \times \mathbb{Z}$, where $\mathbb{Z}$ is the set of integers $\{\ldots, -2, -1, 0, 1, 2, 3, \ldots\}$. Tape contents should be modeled by functions $f : \mathbb{Z} \times \mathbb{Z} \to \Gamma$, which assign a tape symbol in Γ to each cell $(i, j) \in \mathbb{Z} \times \mathbb{Z}$.

105. (b) Think VALCOMPS.

119. (a) Take G to be $S \to aS \mid bS \mid \epsilon$.

121. Think VALCOMPS.

123. Think VALCOMPS.

128. Use Exercise 4 of Homework 9.

130. Let P be a nontrivial property of the r.e. sets. Then there exist TMs $M_\top$ and $M_\bot$ such that $P(L(M_\top)) = \top$ and $P(L(M_\bot)) = \bot$. Show that if it were decidable for TMs M whether $P(L(M)) = \top$, then we could construct a total computable map σ with no fixpoint, contradicting the recursion theorem (see Miscellaneous Exercise 129).

131. Use the recursion theorem (see Miscellaneous Exercise 129).

132. (a) Diagonalize.

(b) Let $\leq$ be an arbitrary computable linear order on the set of input strings. Given M, let M' be a machine that on input x simulates M on all $y \leq x$.

135. Use the pumping lemma.

138. Diagonalize.

140. Construct a list of total Turing machines that run in linear time such that every set computable in linear time is accepted by some machine on the list. Build a machine that diagonalizes over this list.

143. Diagonalize.

Solutions to Selected Miscellaneous Exercises

7. (a) By induction on n.

Basis

$$
\begin{aligned}
(A^0)_{uv} &= I_{uv} \\
&= \begin{cases} \{\epsilon\} & \text{if } u = v, \\ \varnothing & \text{otherwise} \end{cases} \\
&= \{x \in \Sigma^* \mid |x| = 0 \text{ and } \widehat{\delta}(u, x) = v\}.
\end{aligned}
$$

Induction step

$$
\begin{aligned}
&(A^{n+1})_{uv} \\
&= (A^n A)_{uv} \\
&= \bigcup_{w \in Q} (A^n)_{uw} A_{wv} \\
&= \bigcup_{w \in Q} \{x \in \Sigma^* \mid |x| = n \text{ and } \widehat{\delta}(u, x) = w\} \cdot \{a \in \Sigma \mid \delta(w, a) = v\} \\
&= \{xa \in \Sigma^* \mid |x| = n \text{ and } \exists w\ \widehat{\delta}(u, x) = w \text{ and } \delta(w, a) = v\} \\
&= \{xa \in \Sigma^* \mid |x| = n \text{ and } \widehat{\delta}(u, xa) = v\} \\
&= \{y \in \Sigma^* \mid |y| = n + 1 \text{ and } \widehat{\delta}(u, y) = v\}.
\end{aligned}
$$

20. (a) We'll show the inequality in both directions using the axioms (A.1) through (A.15).

Since $a^* = 1 + aa^*$ by (A.10), we have $aa^* \leq a^*$ by the definition of $\leq$. Then $a^*a^* \leq a^*$ follows from (A.14).

Conversely,

$$\begin{aligned} a^*a^* &= a^*(1 + aa^*) && \text{by (A.10)} \\ &= a^* + a^*aa^* && \text{by (A.8);} \end{aligned}$$

therefore, $a^* \leq a^*a^*$ by the definition of $\leq$.

25. We show first that A^*b is a solution of $x = Ax + b$. By Lemma A.2, we have that $A^* = AA^* + I$; this is just axiom (A.10) of Kleene algebra. Multiplying both sides by b and distributing, we get $A^*b = AA^*b + b$, which is just $x = Ax + b$ with A^*b substituted for x.

Now we wish to show that A^*b is the least solution. Let c be any other solution; then $c = Ac + b$. The array c is a vector of length n over the Kleene algebra $\mathcal{K}$. Form a square matrix C by juxtaposing n copies of c. Form the matrix B from b similarly. Then $C = AC + B$. By Lemma A.2 and axiom (A.12) of Kleene algebra, $A^*B \leq C$, therefore $A^*b \leq c$.

26. We show that if A is regular, then so is FirstHalves$A$ using the five-step protocol given in the hint for this exercise on p. 351.

Step 1 Let

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

be a DFA for A. Here is an informal description of an NFA M' for FirstHalves A in terms of pebbles. There will be a white pebble, a green pebble, and a blue pebble on the automaton at any point in time. We start with the white pebble on the start state of M and the blue and green pebbles together on a nondeterministically chosen state of M. The initial position of the blue and green pebbles is a guess as to where M will be after scanning x. In each step, we move the white pebble forward according to the input symbol and move the blue pebble forward according to some nondeterministically chosen symbol. The green pebble never moves. When the end of the input x is reached, we accept iff the white pebble and green pebble occupy the same state and the blue pebble occupies an accept state. The white pebble will occupy the state $\widehat{\delta}(s, x)$, since we moved it according to the input x. The blue pebble will occupy some state q reachable from the position of the green pebble under some string y such that $|y| = |x|$. If the white and green pebbles occupy the same state and the blue

pebble occupies an accept state, then we can concatenate x and y to get a string twice as long as x accepted by M.

Step 2 Now let's do this formally. Define the NFA

$$M' = (Q', \Sigma, \Delta', S', F')$$

as follows. We take the states of M' to be $Q' \stackrel{\text{def}}{=} Q^3$, the set of ordered triples of elements of Q. For $(p, q, r) \in Q'$, the first component models the position of the white pebble, the second models the position of the green pebble, and the third models the position of the blue pebble.

The transition function Δ' must be a function

$$\Delta' : Q' \times \Sigma \to 2^{Q'}.$$

For any $p, q, r \in Q$ and $a \in \Sigma$, we define

$$\Delta'((p, q, r), a) \stackrel{\text{def}}{=} \{(\delta(p, a), q, \delta(r, b)) \mid b \in \Sigma\}.$$

These are the possible next pebble positions after (p, q, r) on input $a \in \Sigma$. The first component $\delta(p, a)$ says that we move the white pebble according to the input symbol a; the second component q says that we leave the green pebble where it is; and the third component $\delta(r, b)$ says that we move the blue pebble according to $b \in \Sigma$. All possible b are included, which reflects the idea that M' is guessing the next symbol of the string y.

We define the start states of M' to be

$$S' \stackrel{\text{def}}{=} \{(s, t, t) \mid t \in Q\},$$

modeling all possible initial configurations of pebbles. The white pebble initially occupies the start state of M and the green and blue pebbles occupy an arbitrary nondeterministically chosen state of M.

Finally, we take the accept states of M' to be

$$F' \stackrel{\text{def}}{=} \{(u, u, v) \mid u \in Q,\ v \in F\},$$

indicating that we accept provided the white and green pebbles occupy the same state and the blue pebble occupies an accept state.

Step 3 Our formal definition specifies a relationship between δ and Δ'. Now let's try to extend it to a relationship between $\widehat{\delta}$ and $\widehat{\Delta}'$. Intuitively, after scanning a string x of length n starting in some start state (s, q, q), the machine M' can be in any state of the form $(\widehat{\delta}(s, x), q, \widehat{\delta}(q, y))$ for some $y \in \Sigma^n$.

Lemma *For any* $x \in \Sigma^*$,

$$\widehat{\Delta}'(S', x) = \{(\widehat{\delta}(s, x), q, \widehat{\delta}(q, y)) \mid q \in Q,\ y \in \Sigma^{|x|}\}.$$

Step 4 We prove the lemma of step 3 by induction on $|x|$. For the basis $x = \epsilon$, we use the base clauses (3.1) and (6.1) in the inductive definitions of $\widehat{\delta}$ and $\widehat{\Delta}'$ and the definition of S':

$$\begin{aligned}\widehat{\Delta}'(S', \epsilon) &= S' \\ &= \{(s, q, q) \mid q \in Q\} \\ &= \{(\widehat{\delta}(s, \epsilon), q, \widehat{\delta}(q, \epsilon)) \mid q \in Q\} \\ &= \{(\widehat{\delta}(s, \epsilon), q, \widehat{\delta}(q, y)) \mid q \in Q,\ y \in \Sigma^0\}.\end{aligned}$$

For the induction step, assume that the lemma is true for x; that is,

$$\widehat{\Delta}'(S', x) = \{(\widehat{\delta}(s, x), q, \widehat{\delta}(q, y)) \mid q \in Q,\ y \in \Sigma^{|x|}\}.$$

We want to show that it is true for xa; that is,

$$\widehat{\Delta}'(S', xa) = \{(\widehat{\delta}(s, xa), q, \widehat{\delta}(q, yb)) \mid q \in Q,\ y \in \Sigma^{|x|},\ b \in \Sigma\},$$

where $a \in \Sigma$. The argument uses the inductive definitions of $\widehat{\Delta}'$ and $\widehat{\delta}$, the induction hypothesis, and the definition of Δ' in terms of δ given in step 2:

$$\begin{aligned}&\widehat{\Delta}'(S', xa) \\ &= \bigcup_{(p,q,r) \in \widehat{\Delta}'(S', x)} \Delta'((p, q, r), a) && \text{by (6.2)} \\ &= \bigcup_{q \in Q,\ y \in \Sigma^{|x|}} \Delta'((\widehat{\delta}(s, x), q, \widehat{\delta}(q, y)), a) && \text{induction hypothesis} \\ &= \bigcup_{q \in Q,\ y \in \Sigma^{|x|}} \{(\delta(\widehat{\delta}(s, x), a), q, \delta(\widehat{\delta}(q, y), b)) \mid b \in \Sigma\} \\ & && \text{definition of } \Delta' \\ &= \bigcup_{q \in Q,\ y \in \Sigma^{|x|}} \{(\widehat{\delta}(s, xa), q, \widehat{\delta}(q, yb)) \mid b \in \Sigma\} && \text{by (3.2)} \\ &= \{(\widehat{\delta}(s, xa), q, \widehat{\delta}(q, yb)) \mid q \in Q,\ y \in \Sigma^{|x|},\ b \in \Sigma\}.\end{aligned}$$

□

Step 5 Finally, we prove $L(M') = \text{FirstHalves}\, L(M)$. For any $x \in \Sigma^*$,

$$\begin{aligned}&x \in L(M') \\ &\iff \widehat{\Delta}'(S', x) \cap F' \neq \varnothing \\ &\iff \{(\widehat{\delta}(s, x), q, \widehat{\delta}(q, y)) \mid q \in Q,\ y \in \Sigma^{|x|}\} && \text{the lemma of step 3} \\ &\quad \cap \{(u, u, v) \mid u \in Q,\ v \in F\} \neq \varnothing && \text{definition of } F' \\ &\iff \exists y \in \Sigma^{|x|}\ \exists q \in Q\ \widehat{\delta}(s, x) = q \text{ and } \widehat{\delta}(q, y) \in F \\ &\iff \exists y \in \Sigma^{|x|}\ \widehat{\delta}(\widehat{\delta}(s, x), y) \in F \\ &\iff \exists y \in \Sigma^{|x|}\ \widehat{\delta}(s, xy) \in F && \text{Homework 1, Exercise 3}\end{aligned}$$

$$\iff \exists y \in \Sigma^{|x|}\ xy \in L(M)$$
$$\iff x \in \text{FirstHalves}\, L(M). \qquad \square$$

33. Using the fact that

$$p(n+1) = \sum_{i=0}^{d} \frac{p^{(i)}(n)}{i!},$$

where $p^{(i)}$ denotes the ith derivative of p, we have

$$p^{(j)}(n+1) = \sum_{i=0}^{d-j} \frac{p^{(i+j)}(n)}{i!} = \sum_{k=j}^{d} \frac{p^{(k)}(n)}{(k-j)!},$$

therefore

$$\frac{p^{(j)}(n+1)}{j!} = \sum_{k=j}^{d} \binom{k}{j} \frac{p^{(k)}(n)}{k!}. \tag{1}$$

Also,

$$\frac{p^{(j)}(0)}{j!} = a_j, \tag{2}$$

where a_j is the coefficient of n^j in $p(n)$. Let $M = (Q, \Sigma, \Delta, s, F)$ be an NFA for A. We will build a DFA $M' = (Q', \Sigma, \delta', s', F')$ for A'. Let B be the square Boolean matrix indexed by states of M such that

$$B_{uv} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \exists a \in \Sigma\ v \in \Delta(u,a), \\ 0 & \text{otherwise.} \end{cases}$$

Let B^n be the nth Boolean power of B, B^0 the identity matrix. One can show by induction on n that

$$(B^n)_{uv} = \begin{cases} 1 & \text{if } \exists y \in \Sigma^n\ v \in \widehat{\Delta}(\{u\}, y), \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

In other words, $(B^n)_{uv} = 1$ iff there exists a path of length n from u to v in the graph representation of the automaton.

Now consider the set of all square Boolean matrices indexed by the states of M. The states Q' of M' will be the set of all sequences $(C_0, C_1, \ldots, C_d)$ of $d+1$ such matrices, which we denote by $(C_i \mid 0 \le i \le d)$. Define

$$\delta'((C_i \mid 0 \le i \le d), a) = (\prod_{k=j}^{d} C_k^{\binom{k}{j}} \mid 0 \le j \le d),$$
$$s' = (B^{a_i} \mid 0 \le i \le d),$$
$$F' = \{(C_i \mid 0 \le i \le d) \mid \exists q \in F\ (C_0)_{sq} = 1\}.$$

Lemma *Let $x \in \Sigma^*$. Then*

$$\widehat{\delta}'(s', x) = (B^{p^{(i)}(|x|)/i!} \mid 0 \le i \le d).$$

Proof. By induction on $|x|$.

Basis

$$\begin{aligned}
\widehat{\delta}'(s', \epsilon) &= s' && \text{definition of } \widehat{\delta}' \\
&= (B^{a_i} \mid 0 \le i \le d) && \text{definition of } s' \\
&= (B^{p^{(i)}(0)/i!} \mid 0 \le i \le d) && \text{by (2)} \\
&= (B^{p^{(i)}(|\epsilon|)/i!} \mid 0 \le i \le d).
\end{aligned}$$

Induction step

Assume true for x. Then

$$\begin{aligned}
\widehat{\delta}'(s', xa) &= \delta'(\widehat{\delta}'(s', x), a) && \text{definition of } \widehat{\delta}' \\
&= \delta'((B^{p^{(i)}(|x|)/i!} \mid 0 \le i \le d), a) && \text{induction hypothesis} \\
&= (\prod_{k=j}^{d} (B^{p^{(k)}(|x|)/k!})^{\binom{k}{j}} \mid 0 \le j \le d) && \text{definition of } \delta' \\
&= (B^{\sum_{k=j}^{d} \binom{k}{j} p^{(k)}(|x|)/k!} \mid 0 \le j \le d) \\
&= (B^{p^{(i)}(|x|+1)/i!} \mid 0 \le i \le d) && \text{by (1)} \\
&= (B^{p^{(i)}(|xa|)/i!} \mid 0 \le i \le d). && \square
\end{aligned}$$

Theorem $L(M') = A'$.

Proof.

$$\begin{aligned}
& x \in L(M') \\
& \iff \widehat{\delta}'(s', x) \in F' && \text{definition of acceptance} \\
& \iff (B^{p^{(i)}(|x|)/i!} \mid 0 \le i \le d) \in F' && \text{by the lemma} \\
& \iff \exists q \in F \; B^{p(|x|)}(s, q) = 1 && \text{definition of } F' \\
& \iff \exists y \in \Sigma^{p(|x|)} \; F \cap \widehat{\Delta}(\{s\}, y) \neq \varnothing && \text{by (3)} \\
& \iff \exists y \in \Sigma^{p(|x|)} \; y \in L(M) && \text{definition of acceptance} \\
& \iff x \in A' && \text{definition of } A' \qquad \square
\end{aligned}$$

35. Let

$$A = \{ww \mid w \in \{0,1\}^*\}.$$

Note that $\{ww \mid w \in \{0\}^*\}$ is regular—it is just the set of strings of 0's of even length.

To show that A is nonregular, we will show that we have a winning strategy in the demon game. The demon picks some k. Now we take $x = 0$, $y = 1^k$, and $z = 01^k$. Then $xyz = 01^k01^k$, which is in A, and $|y| = k$. The demon must now choose u, v, w such that $y = uvw$ and $v \neq \epsilon$. Say the demon picks u, v, w of lengths j, m, n, respectively. Then $k = j + m + n$ and $m > 0$. But whatever the demon picks, we can win by taking $i = 0$:

$$xuv^0wz = xuwz = 01^{k-m}01^k,$$

which is not in A because it is not of the form ww for any w.

36. Recall that a number is *prime* if it is greater than 1 and has no divisors other than 1 and itself. PRIMES is an example of a single letter alphabet set that is not regular because the elements of the set do not appear with any predictable pattern.

Suppose the demon chooses k. You choose $x = z = \epsilon$ and $y = a^p$, where p is the smallest prime greater than k (Euclid proved that there exist infinitely many primes, so p exists). Then $xyz = a^p \in$ PRIMES and $|y| = p > k$. The demon must now choose u, v, w such that $y = uvw$ and $v \neq \epsilon$. Say the lengths of u, v, w are j, m, n, respectively. Then $k = j + m + n$ and $m > 0$. You now need to find i such that $xuv^iwz \notin$ PRIMES (i.e., $|xuv^iwz|$ is not prime). But

$$|xuv^iwz| = j + im + n = p + (i-1)m,$$

so we need to find i such that $p + (i-1)m$ is not prime. Take $i = p+1$. Then

$$p + (i-1)m = p + pm = p(1+m),$$

which is not prime since it has factors p and $1 + m$. You win.

44. (a) First we show that the given condition is necessary for regularity. Let $A \subseteq \Sigma^*$ be a regular set, and let k be the number of states of a DFA for A. Then for all $y \in \Sigma^*$ with $|y| = k$, the automaton repeats a state while scanning y. Let v be a nonnull substring of y such that the automaton is in the same state just before and just after scanning v, and let $y = uvw$. Then for all $z \in \Sigma^*$ and $i \geq 0$, the automaton is in the same state after scanning $yz = uvwz$ as after scanning uv^iwz; therefore,

$$yz \in A \iff uv^iwz \in A.$$

Now we show that the given condition is sufficient for regularity. Let $A \subseteq \Sigma^*$ such that A satisfies the given condition. For any $x \in \Sigma^*$ with $|x| \geq k$, by applying the condition with $i = 0$

as many times as necessary, we can repeatedly delete nonnull substrings of x until we obtain a string x' of length $k-1$ or less such that for all $z \in \Sigma^*$,

$$xz \in A \iff x'z \in A.$$

This says that $x \equiv_A x'$, where $\equiv_A$ is the relation (16.1). Since every $\equiv_A$-class contains a string of length $k-1$ or less, the relation $\equiv_A$ is of finite index. By the Myhill–Nerode theorem, A is regular.

45. (b) Equivalently, if $A \subseteq \mathbb{N}$ and $\widehat{A}$ is the smallest subset of $\mathbb{N}$ containing A and 0 and closed under addition, then A^* consists of all but finitely many multiples of gcd A.

First we consider the case $|A| = 2$. Let $m\mathbb{N}$ denote the set of all nonnegative multiples of m, and let $m\mathbb{N} + n\mathbb{N}$ denote the set of all sums $am + bn$ for $a, b \geq 0$. Write $A \sim B$ if A and B differ by a finite set; that is, if the set $(A - B) \cup (B - A)$ is finite.

Lemma *Let m and n be positive integers, $g = \gcd(m, n)$. Then*

$$m\mathbb{N} + n\mathbb{N} \sim g\mathbb{N}.$$

Moreover, $\operatorname{lcm}(m, n) - m - n$ is the largest multiple of g not expressible in the form $am + bn$ with $a, b \geq 0$.

Proof. We first show a special case: if m and n are relatively prime, then $m\mathbb{N} + n\mathbb{N} \sim \mathbb{N}$. Moreover, the largest number not expressible in the form $am + bn$ with $a, b \geq 0$ is $mn - m - n$.

Suppose $mn-m-n$ were so expressible, say $am+bn = mn-m-n$. Then $(a+1)m + (b+1)n = mn$. Since m and n are relatively prime, m must divide $b+1$ and n must divide $a+1$. The smallest values of a and b for which this is true would be $a = n-1$ and $b = m-1$, which are already too big:

$$(n-1)m + (m-1)n = 2mn - m - n > mn - m - n.$$

Now let's show that $mn - m - n + 1$ is expressible. Let $u < n$ and $v < m$ such that $vn - um = 1$. (The numbers u and v can be produced by an extended version of the Euclidean GCD algorithm.) Take $a = n - u - 1$ and $b = v - 1$. Then

$$\begin{aligned} am + bn &= (n-u-1)m + (v-1)n \\ &= mn - um - m + vn - n \\ &= mn - m - n + 1. \end{aligned}$$

Now we proceed by induction. Suppose we have some $am + bn \geq mn - m - n + 1$. Since

$$(u-1)m + (m-v-1)n = um - m + mn - vn - n$$

$$= mn - m - n - 1,$$

we must have either $a \geq u$ or $b \geq m - v$. If the former, take $a' = a - u$ and $b' = b + v$ to get

$$\begin{aligned} a'm + b'n &= (a-u)m + (b+v)n \\ &= am - um + bn + vn \\ &= am + bn + 1. \end{aligned}$$

If the latter, take $a' = a + n - u$ and $b' = b - m + v$, and again

$$\begin{aligned} a'm + b'n &= (a+n-u)m + (b-m+v)n \\ &= am + mn - um + bn - mn + vn \\ &= am + bn + 1. \end{aligned}$$

If m and n are not relatively prime, say $g = \gcd(m,n)$, then everything is scaled by g. Any $am + bn$ is a multiple of g since g divides m and n, and the largest multiple of g not so expressible is

$$((m/g)(n/g) - m/g - n/g)g = \operatorname{lcm}(m,n) - m - n.$$

Now we use this to show that for any $A \subseteq \mathrm{N}$, $\widehat{A}$ consists of all but finitely many multiples of $g = \gcd A$. This follows from the observation that $\gcd A = \gcd X$ for some finite subset $X \subseteq A$ and from applying the lemma iteratively to obtain

$$\sum_{m \in X} \sum m\mathrm{N} \sim g\mathrm{N}. \qquad \square$$

49. Suppose first that $\{p,q\}$ is marked. We proceed by induction on the stages of the algorithm. If $\{p,q\}$ is marked in step 2, then either $p \in F$ and $q \notin F$ or vice versa, therefore $p \not\approx q$ (take $x = \epsilon$ in the definition of $\approx$). If it is marked in step 3, then for some $a \in \Sigma$, $\{\delta(p,a), \delta(q,a)\}$ was marked at some earlier stage. By the induction hypothesis, $\delta(p,a) \not\approx \delta(q,a)$, therefore $p \not\approx q$ by Lemma 13.5.

    Conversely, suppose $p \not\approx q$. By definition, there exists an $x \in \Sigma^*$ such that either $\widehat{\delta}(p,x) \in F$ and $\widehat{\delta}(q,x) \notin F$ or vice versa. We proceed by induction on the length of x. If $x = \epsilon$, then either $p \in F$ and $q \notin F$ or vice versa, so $\{p,q\}$ is marked in step 2. If $x = ay$, then either $\widehat{\delta}(\delta(p,a),y) \in F$ and $\widehat{\delta}(\delta(q,a),y) \notin F$ or vice versa. By the induction hypothesis, $\{\delta(p,a), \delta(q,a)\}$ is eventually marked by the algorithm, and $\{p,q\}$ will be marked in the following step.

57. A subset of Σ^* is closed downward under $\sqsubseteq$ iff its complement is closed upward under $\sqsubseteq$. Since the complement of any regular set is regular, it suffices to show that all upward-closed sets are regular.

Let $\min X$ denote the set of $\sqsubseteq$-minimal elements of X, and let

$$y\uparrow = \{x \in \Sigma^* \mid y \sqsubseteq x\}.$$

Then X is upward-closed iff

$$\begin{aligned} X &= \{x \in \Sigma^* \mid \exists y \in \min X \; y \sqsubseteq x\} \\ &= \bigcup_{y \in \min X} y\uparrow . \end{aligned}$$

By Higman's lemma, this is a finite union of sets of the form $y\uparrow$. Since a finite union of regular sets is regular, it suffices to show that any $y\uparrow$ is regular. But

$$a_1 a_2 \cdots a_n\uparrow = L(\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_n \Sigma^*).$$

59. To construct a DFA from an AFA, let

$$A = (Q_A, \Sigma, \delta_A, F_A, \alpha_A)$$

be the given AFA, $|Q_A| = k$. Let Q_D be the set of all functions $Q_A \to \{0,1\}$. Define the DFA

$$D = (Q_D, \Sigma, \delta_D, F_D, s_D),$$

where

$$\delta_D(u,a)(q) = \delta_A(q,a)(u), \tag{4}$$

$$F_D = \alpha_A, \tag{5}$$

$$s_D = F_A. \tag{6}$$

To construct an AFA from a DFA, let

$$D = (Q_D, \Sigma, \delta_D, F_D, s_D)$$

be the given DFA, $|Q_D| = k$. Let Q_A be any set of size $\lceil \log k \rceil$ and identify each element of Q_D with a distinct function $Q_A \to \{0,1\}$. Define the AFA

$$A = (Q_A, \Sigma, \delta_A, F_A, \alpha_A),$$

where δ_A, F_A, and α_A are defined such that (4), (5), and (6) hold. (For $u \notin Q_D$, define $\delta_A(q,a)(u)$ arbitrarily.)

In both reductions, one can show by induction on $|x|$ that for any $q \in Q_A$, $u \in Q_D$, and $x \in \Sigma^*$,

$$\hat{\delta}_D(u,x)(q) = \hat{\delta}_A(q, \mathbf{rev}\, x)(u).$$

In particular,

$$x \in L(D) \iff F_D(\hat{\delta}_D(s_D, x)) = 1$$

$$\iff \alpha_A(\widehat{\delta}_D(F_A, x)) = 1$$
$$\iff \alpha_A(\lambda q.(\widehat{\delta}_D(F_A, x)(q))) = 1$$
$$\iff \alpha_A(\lambda q.(\widehat{\delta}_A(q, \mathbf{rev}\ x)(F_A))) = 1$$
$$\iff \mathbf{rev}\ x \in L(A).$$

77. The set generated is $\{a, b\}^*$. We can prove this by induction on string length. The null string is generated in one step. For nonnull strings, either the string begins with b, in which case we use the first production; or ends with a, in which case we use the second production; or neither, in which case we use the third production. In any case the induction hypothesis implies that the rest of the string can be generated.

78. Let P be a PDA for L and M a DFA for R. Build a PDA for L/R that on input x scans x and simulates P, then when it comes to the end of the input x, guesses the string y and continues to simulate P (from the same configuration where it left off) but also runs M simultaneously on the guessed y starting from the start state. It accepts if both L and M accept. Thus it accepts its original input x if it was successfully able to guess a string y such that $xy \in L(P)$ and $y \in L(M)$; that is, if there exists y such that $xy \in L$ and $y \in R$.

    Here is an alternative proof using homomorphisms. Suppose the alphabet is $\{a, b\}$. Let $\{a', b'\}$ be another copy of the alphabet disjoint from $\{a, b\}$. Let h be the homomorphism that erases marks; that is, $h(a) = h(a') = a$ and $h(b) = h(b') = b$. Let g be the homomorphism that erases the unmarked symbols and erases the marks on the marked symbols; that is, $g(a) = g(b) = \epsilon$, $g(a') = a$, $g(b') = b$. Then

$$L/R = g(h^{-1}(L) \cap \{a', b'\}^* R). \tag{7}$$

    This is a CFL, since CFLs are closed under homomorphic preimage, intersection with regular set, and homomorphic image.

    To see (7), first consider the set $h^{-1}(L)$. This is the set of all strings that look like strings in L, except that some of the symbols are marked. Now intersect with the regular set $\{a', b'\}^* R$. This gives the set of strings of the form $x'y$ such that the symbols of x' are marked, those of y are unmarked, $xy \in L$, and $y \in R$. Now apply g to this set of strings. Applied to a string $x'y$ as described above, we would get x. Therefore, the resulting set is the set of all x such that there exists y such that $x'y \in h^{-1}(L) \cap \{a', b'\}^* R$; in other words, such that $xy \in L$ and $y \in R$. This is L/R.

99. (a) A *queue machine* is a sextuple

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

where

- Q is a finite set of *states*,
- Σ is a finite *input alphabet*,
- Γ is a finite *queue alphabet*,
- $\$ \in \Gamma - \Sigma$ is the *initial queue symbol*,
- $s \in Q$ is the *start state*, and
- $\delta : Q \times \Gamma \to Q \times \Gamma^*$ is the *transition function*.

A *configuration* is a pair $(q, \gamma) \in Q \times \Gamma^*$ giving the current state and current contents of the queue. The *start configuration* on input x is the pair $(s, x\$)$. The *next configuration relation* $\xrightarrow[M]{1}$ is defined as follows: if

$$\delta(p, A) = (q, \gamma),$$

then

$$(p, A\alpha) \xrightarrow[M]{1} (q, \alpha\gamma).$$

The relation $\xrightarrow[M]{*}$ is the reflexive transitive closure of $\xrightarrow[M]{1}$. An *accept configuration* is any configuration of the form (q, ϵ), where $q \in Q$ and ϵ is the null string. The queue machine M is said to *accept* $x \in \Sigma^*$ if

$$(s, x\$) \xrightarrow[M]{*} (q, \epsilon) \qquad \text{for some } q \in Q.$$

(b) To simulate a queue machine U on a Turing machine T, let T maintain the contents of U's queue on its tape and simulate the action of U, shuttling back and forth from the front to the back of the simulated queue. Each simulated step of U consists of T moving to the front of the simulated queue, erasing the first symbol and remembering it in the finite control, then moving to the back of the simulated queue to write symbols. The simulated queue migrates to the right on T's tape, but that's okay, because there's plenty of room. The machine T accepts if its tape ever becomes completely blank, which indicates that the simulated queue of U is empty.

The simulation in the other direction is much harder. Given a Turing machine T, we build a queue machine U that simulates

moves of T, using the queue to maintain a description of T's current configuration. We will represent configurations by strings such as

$$\vdash a\ b\ a\ a\ b\ a\ q\ b\ b\ a\ \$$$

for example; exactly one of the symbols is a state of T (q in this example), and its position in the string indicates the position of the tape head of T, which is immediately to the right of the state. If the state is just before the \$, this models T scanning a blank cell to the right of the portion of its tape represented in the configuration.

The queue machine U will also have an "internal queue" that will hold two symbols of this configuration; since this is only a finite amount of information, the internal queue can be encoded in the finite control of U. The remaining portion of the configuration will be held in the external queue. Since configurations have at least three symbols (state, left endmarker, \$), even if the tape is blank, the external queue will never be prematurely emptied.

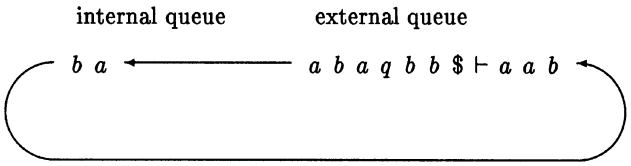
Let x be the input string. The queue machine U starts with $x\$$ in its external queue. It enters a state representing an internal queue of

$$s \vdash$$

The internal and external queues concatenated together represent the start configuration of T on input x:

$$s \vdash x\ \$$$

A **rotate** operation consists of rotating the symbols on the internal and external queues as follows:



Formally, this is done by popping the first element off the front of the external queue and pushing the front element of the internal queue onto the back of the external queue, then changing state to reflect the new contents of the internal queue. In this example, after one rotation we would have

$$a\ a \qquad\qquad b\ a\ q\ b\ b\ \$ \vdash a\ a\ b\ b$$

on the internal and external queues, respectively.

We simulate a step of T as follows. If the rightmost symbol of the internal queue is not a state of T, we **rotate** until this becomes true. In this example, after three more steps we would have

$$a\ q \qquad\qquad b\ b\ \$ \vdash a\ a\ b\ b\ a\ a\ b$$

Now we have the symbol that T is scanning (say b) at the head of the external queue and the current state of T (say q) rightmost on the internal queue. We read b at the head of the queue. If b is not \$, we simulate a move of T as follows.

- If $\delta_T(q, b) = (p, c, R)$ and the leftmost symbol of the internal queue is d, we push d onto the back of the external queue and make $(c\,p)$ the new internal queue.
- If $\delta_T(q, b) = (p, c, L)$ and the leftmost symbol of the internal queue is d, we push p onto the back of the external queue and make $(d\,c)$ the new internal queue.

In the example above, this would give

$$a\ p \qquad\qquad b\ \$ \vdash a\ a\ b\ b\ a\ a\ b\ a$$

if $\delta_T(q, b) = (p, a, R)$, and

$$a\ a \qquad\qquad b\ \$ \vdash a\ a\ b\ b\ a\ a\ b\ q$$

if $\delta_T(q, b) = (p, a, L)$. If the symbol at the head of the external queue is \$, then this indicates that the tape head of T is scanning a blank symbol to the right of the portion of the tape represented in the configuration. For example,

$$b\ q \qquad\qquad \$ \vdash a\ a\ b\ b\ b\ a\ a\ b\ a\ b$$

In this case, when we pop \$ we don't simulate a move of T immediately; first we insert an extra blank symbol $\sqcup$ between the state and the \$. We do this by pushing both symbols in the internal queue onto the back of the external queue and making the new internal queue $(\sqcup\ \$)$. In this example, the resulting queue contents would be

$$\sqcup\ \$ \qquad\qquad \vdash a\ a\ b\ b\ b\ a\ a\ b\ a\ b\ b\ q$$

We continue to simulate moves of T. If T ever enters its accept state, then U goes into a little subroutine that just empties its external queue, thereby accepting.

113. Let

$$\text{REG} = \{M \mid L(M) \text{ is regular}\}.$$

A corollary of Rice's theorem for r.e. sets (Theorem 34.2) is that any semidecidable property of the r.e. sets is monotone. That is, if P is a property of r.e. sets such that $\{M \mid P(L(M))\}$ is r.e., A and B are r.e. sets, $A \subseteq B$, and $P(A)$, then $P(B)$. Applying this corollary with $P(C)$ = "C is regular," $A = \varnothing$, and $B = \{a^n b^n \mid n \geq 0\}$ gives immediately that REG is not r.e. Similarly, applying the corollary with $P(C)$ = "C is not regular," $A = \{a^n b^n \mid n \geq 0\}$, and $B = \Sigma^*$ gives immediately that REG is not co-r.e.

References

[1] W. ACKERMANN, Zum Hilbertschen Aufbau der reellen Zahlen, *Math. Annalen*, 99 (1928), pp. 118–133.

[2] A.V. AHO AND J.D. ULLMAN, *The Theory of Parsing, Translation, and Compiling, Vol. I: Parsing*, Prentice Hall, Englewood Cliffs, NJ, 1972.

[3] ——, *The Theory of Parsing, Translation, and Compiling, Vol. II: Compiling*, Prentice Hall, Englewood Cliffs, NJ, 1973.

[4] ——, *Principles of Compiler Design*, Addison-Wesley, Reading, MA, 1977.

[5] M.A. ARBIB AND Y. GIVE'ON, Algebra automata I: Parallel programming as a prolegomenon to the categorical approach, *Information and Control*, 12 (1968), pp. 331–345.

[6] R.C. BACKHOUSE, *Closure Algorithms and the Star-Height Problem of Regular Languages*, Ph.D. thesis, Imperial College, London, 1975.

[7] J.W. BACKUS, The syntax and semantics of the proposed international algebraic language of the Zürich ACM-GAMM conference, in *Proc. Intl. Conf. Information Processing*, UNESCO, 1959, pp. 125–132.

[8] Y. BAR-HILLEL, M. PERLES, AND E. SHAMIR, On formal properties of simple phrase structure grammars, *Z. Phonetik. Sprachwiss. Kommunikationsforsch.*, 14 (1961), pp. 143–172.

[9] H.P. BARENDREGT, *The Lambda Calculus*, North-Holland, Amsterdam, 1984.

[10] S.L. BLOOM AND Z. ÉSIK, Equational axioms for regular sets, *Mathematical Structures in Computer Science*, 3 (1993), pp. 1–24.

[11] M. BOFFA, Une remarque sur les systèmes complets d'identités rationnelles, *Informatique Théoretique et Applications/Theoretical Informatics and Applications*, 24 (1990), pp. 419–423.

[12] ———, Une condition impliquant toutes les identités rationnelles, *Informatique Théoretique et Applications/Theoretical Informatics and Applications*, 29 (1995), pp. 515–518.

[13] W.S. BRAINERD, *Tree Generating Systems and Tree Automata*, Ph.D. thesis, Purdue University, Indiana, 1967.

[14] ———, The minimalization of tree automata, *Information and Control*, 13 (1968), pp. 484–491.

[15] W.S. BRAINERD AND L.H. LANDWEBER, *Theory of Computation*, John Wiley, New York, 1974.

[16] G. CANTOR, Über eine Eigenschaft des Inbegriffes aller reellen algebraischen Zahlen, *J. für die reine und angewandte Mathematik*, 77 (1874), pp. 258–262. Reprinted in *Georg Cantor Gesammelte Abhandlungen*, Berlin, Springer-Verlag, 1932, pp. 115–118.

[17] N. CHOMSKY, Three models for the description of languages, *IRE Trans. Information Theory*, 2 (1956), pp. 113–124.

[18] ———, On certain formal properties of grammars, *Information and Control*, 2 (1959), pp. 137–167.

[19] ———, Context-free grammars and pushdown storage, Tech. Rep., MIT Research Lab. in Electronics, Cambridge, MA, 1962.

[20] ———, Formal properties of grammars, *Handbook of Math. Psych.*, 2 (1963), pp. 323–418.

[21] N. CHOMSKY AND G.A. MILLER, Finite state languages, *Information and Control*, 1 (1958), pp. 91–112.

[22] N. CHOMSKY AND M.P. SCHÜTZENBERGER, The algebraic theory of context free languages, in *Computer Programming and Formal Systems*, P. Braffort and D. Hirschberg, eds., North-Holland, Amsterdam, 1963, pp. 118–161.

[23] A. CHURCH, A set of postulates for the foundation of logic, *Ann. Math.*, 33–34 (1933), pp. 346–366, 839–864.

[24] ———, A note on the Entscheidungsproblem, *J. Symbolic Logic*, 58 (1936), pp. 345–363.

[25] ———, An unsolvable problem of elementary number theory, *Amer. J. Math.*, 58 (1936), pp. 345–363.

[26] ———, The calculi of lambda-conversion, *Ann. Math. Studies*, 6 (1941).

[27] J.H. CONWAY, *Regular Algebra and Finite Machines*, Chapman and Hall, London, 1971.

[28] S.A. COOK, The complexity of theorem proving procedures, in *Proc. Third Symp. Theory of Computing*, Assoc. Comput. Mach., New York, 1971, pp. 151–158.

[29] H.B. CURRY, An analysis of logical substitution, *Amer. J. Math.*, 51 (1929), pp. 363–384.

[30] N.J. CUTLAND, *Computability*, Cambridge University, Cambridge, 1980.

[31] M. DAVIS, *Computability and Unsolvability*, McGraw-Hill, New York, 1958.

[32] ———, *The Undecidable*, Raven Press, Hewlitt, NY, 1965.

[33] A. EHRENFEUCHT, R. PARIKH, AND G. ROZENBERG, Pumping lemmas and regular sets, *SIAM J. Computing*, 10 (1981), pp. 536–541.

[34] S. EILENBERG AND J.B. WRIGHT, Automata in general algebra, *Information and Control*, 11 (1967), pp. 452–470.

[35] J. ENGELFRIET, Tree automata and tree grammars, Tech. Rep. DAIMI FN-10, Aarhus University, Aarhus, Denmark, 1975.

[36] J. EVEY, Application of pushdown store machines, in *Proc. Fall Joint Computer Conf.*, AFIPS Press, Montvale, NJ, 1963, pp. 215–227.

[37] P.C. FISCHER, On computability by certain classes of restricted Turing machines, in *Proc. Fourth Symp. Switching Circuit Theory and Logical Design*, 1963, pp. 23–32.

[38] ———, Turing machines with restricted memory access, *Information and Control*, 9 (1966), pp. 364–379.

[39] P.C. FISCHER, A.R. MEYER, AND A.L. ROSENBERG, Counter machines and counter languages, *Math. Systems Theory*, 2 (1968), pp. 265–283.

[40] M.R. GAREY AND D.S. JOHNSON, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.

[41] F. GÉCSEG AND I. PEÁK, *Algebraic Theory of Automata*, Akadémiai Kiadó, Budapest, 1972.

[42] F. GÉCSEG AND M. STEINBY, *Tree Automata*, Akadémiai Kiadó, Budapest, 1984.

[43] S. GINSBURG, *The Mathematical Theory of Context-Free Languages*, McGraw-Hill, New York, 1966.

[44] S. GINSBURG AND S.A. GREIBACH, Deterministic context-free languages, *Information and Control*, 9 (1966), pp. 563–582.

[45] S. GINSBURG AND H.G. RICE, Two families of languages related to ALGOL, *J. Assoc. Comput. Mach.*, 9 (1962), pp. 350–371.

[46] S. GINSBURG AND G.F. ROSE, Operations which preserve definability in languages, *J. Assoc. Comput. Mach.*, 10 (1963), pp. 175–195.

[47] ———, Some recursively unsolvable problems in ALGOL-like languages, *J. Assoc. Comput. Mach.*, 10 (1963), pp. 29–47.

[48] ———, Preservation of languages by transducers, *Information and Control*, 9 (1966), pp. 153–176.

[49] S. GINSBURG AND E.H. SPANIER, Quotients of context-free languages, *J. Assoc. Comput. Mach.*, 10 (1963), pp. 487–492.

[50] K. GÖDEL, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I, *Monatshefte für Mathematik und Physik*, 38 (1931), pp. 173–198.

[51] ———, On undecidable propositions of formal mathematical systems, in *The Undecidable*, M. Davis, ed., Raven Press, Hewlitt, NY, 1965, pp. 5–38.

[52] J. GOLDSTINE, A simplified proof of Parikh's theorem, *Discrete Math.*, 19 (1977), pp. 235–240.

[53] S.A. GREIBACH, A new normal form theorem for context-free phrase structure grammars, *J. Assoc. Comput. Mach.*, 12 (1965), pp. 42–52.

[54] L. HAINES, *Generation and recognition of formal languages*, Ph.D. thesis, MIT, Cambridge, MA, 1965.

[55] M.A. HARRISON, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, MA, 1978.

[56] J. HARTMANIS AND J.E. HOPCROFT, Structure of undecidable problems in automata theory, in *Proc. Ninth Symp. Switching and Automata Theory*, IEEE, 1968, pp. 327–333.

[57] J. HARTMANIS AND R.E. STEARNS, On the complexity of algorithms, *Trans. Amer. Math. Soc.*, 117 (1965), pp. 285–306.

[58] F.C. HENNIE, *Introduction to Computability*, Addison-Wesley, Reading, MA, 1977.

[59] J.E. HOPCROFT, An $n \log n$ algorithm for minimizing the states in a finite automaton, in *The Theory of Machines and Computation*, Z. Kohavi, ed., Academic Press, New York, 1971, pp. 189–196.

[60] J.E. HOPCROFT AND J.D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[61] D.A. HUFFMAN, The synthesis of sequential switching circuits, *J. Franklin Institute*, 257 (1954), pp. 161–190, 275–303.

[62] J. JAFFE, A necessary and sufficient pumping lemma for regular languages, *SIGACT News*, 10 (1978), pp. 48–49.

[63] N.D. JONES, *Computability Theory: An Introduction*, Academic Press, New York, 1973.

[64] R.M. KARP, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–103.

[65] T. KASAMI, An efficient recognition and syntax algorithm for context-free languages, Tech. Rep. AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA, 1965.

[66] S.C. KLEENE, A theory of positive integers in formal logic, *Amer. J. Math.*, 57 (1935), pp. 153–173, 219–244.

[67] ———, General recursive functions of natural numbers, *Math. Annalen*, 112 (1936), pp. 727–742.

[68] ———, Recursive predicates and quantifiers, *Trans. Amer. Math. Soc.*, 53 (1943), pp. 41–74.

[69] ———, *Introduction to Metamathematics*, D. van Nostrand, Princeton, NJ, 1952.

[70] ———, Representation of events in nerve nets and finite automata, in *Automata Studies*, C.E. Shannon and J. McCarthy, eds., Princeton University Press, Princeton, NJ, 1956, pp. 3–41.

[71] D.E. KNUTH, On the translation of languages from left to right, *Information and Control*, 8 (1965), pp. 607–639.

[72] D.C. KOZEN, *The Design and Analysis of Algorithms*, Springer-Verlag, New York, 1991.

[73] ———, A completeness theorem for Kleene algebras and the algebra of regular events, *Infor. and Comput.*, 110 (1994), pp. 366–390.

[74] D. KROB, A complete system of B-rational identities, *Theoretical Computer Science*, 89 (1991), pp. 207–343.

[75] W. KUICH, The Kleene and Parikh theorem in complete semirings, in *Proc. 14th Colloq. Automata, Languages, and Programming*, T. Ottmann, ed., vol. 267 of *Lect. Notes in Comput. Sci.*, EATCS, Springer-Verlag, New York, 1987, pp. 212–225.

[76] W. KUICH AND A. SALOMAA, *Semirings, Automata, and Languages*, Springer-Verlag, Berlin, 1986.

[77] S.Y. KURODA, Classes of languages and linear bounded automata, *Information and Control*, 7 (1964), pp. 207–223.

[78] P.S. LANDWEBER, Three theorems on phrase structure grammars of type 1, *Information and Control*, 6 (1963), pp. 131–136.

[79] H.R. LEWIS AND C.H. PAPADIMITRIOU, *Elements of the Theory of Computation*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[80] P.M. LEWIS, D.J. ROSENKRANTZ, AND R.E. STEARNS, *Compiler Design Theory*, Addison-Wesley, Reading, MA, 1976.

[81] M. MACHTEY AND P. YOUNG, *An Introduction to the General Theory of Algorithms*, North-Holland, Amsterdam, 1978.

[82] Z. MANNA, *Mathematical Theory of Computation*, McGraw-Hill, New York, 1974.

[83] A.A. MARKOV, *The Theory of Algorithms*, vol. 42, Trudy Math. Steklov Inst., 1954. English translation, National Science Foundation, Washington, DC, 1961.

[84] W.S. MCCULLOCH AND W. PITTS, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophysics*, 5 (1943), pp. 115–143.

[85] R. MCNAUGHTON AND H. YAMADA, Regular expressions and state graphs for automata, *IEEE Trans. Electronic Computers*, 9 (1960), pp. 39–47.

[86] A.R. MEYER AND D.M. RITCHIE, The complexity of loop programs, in *Proc. ACM Natl. Meeting*, 1967, pp. 465–469.

[87] R. MILNER, Operational and algebraic semantics of concurrent processes, in *Handbook of Theoretical Computer Science*, J. van Leeuwen, ed., vol. B, North-Holland, Amsterdam, 1990, pp. 1201–1242.

[88] M.L. MINSKY, Recursive unsolvability of Post's problem of 'tag' and other topics in the theory of Turing machines, *Ann. Math.*, 74 (1961), pp. 437–455.

[89] M.L. MINSKY AND S. PAPERT, Unrecognizable sets of numbers, *J. Assoc. Comput. Mach.*, 13 (1966), pp. 281–286.

[90] E.F. MOORE, Gedanken experiments on sequential machines, *Automata Studies*, (1956), pp. 129–153.

[91] J. MYHILL, Finite automata and the representation of events, Technical Note WADD 57-624, Wright Patterson AFB, Dayton, Ohio, 1957.

[92] ———, Linear bounded automata, Technical Note WADD 60-165, Wright Patterson AFB, Dayton, Ohio, 1960.

[93] P. NAUR, Revised report on the algorithmic language ALGOL 60, *Comm. Assoc. Comput. Mach.*, 6 (1963), pp. 1–17.

[94] A. NERODE, Linear automaton transformations, *Proc. Amer. Math. Soc.*, 9 (1958), pp. 541–544.

[95] A.G. OETTINGER, Automatic syntactic analysis and the pushdown store, *Proc. Symposia on Applied Math.*, 12 (1961).

[96] W. OGDEN, A helpful result for proving inherent ambiguity, *Math. Systems Theory*, 2 (1968), pp. 191–194.

[97] C.H. PAPADIMITRIOU, *Computational Complexity*, Addison-Wesley, Reading, MA, 1994.

[98] R. PARIKH, On context-free languages, *J. Assoc. Comput. Mach.*, 13 (1966), pp. 570–581.

[99] E. POST, Finite combinatory processes-formulation, I, *J. Symbolic Logic*, 1 (1936), pp. 103–105.

[100] ———, Formal reductions of the general combinatorial decision problem, *Amer. J. Math.*, 65 (1943), pp. 197–215.

[101] ———, Recursively enumerable sets of positive natural numbers and their decision problems, *Bull. Amer. Math. Soc.*, 50 (1944), pp. 284–316.

[102] M.O. RABIN AND D.S. SCOTT, Finite automata and their decision problems, *IBM J. Res. Develop.*, 3 (1959), pp. 115–125.

[103] V.N. REDKO, On defining relations for the algebra of regular events, *Ukrain. Mat. Z.*, 16 (1964), pp. 120–126. In Russian.

[104] H.G. RICE, Classes of recursively enumerable sets and their decision problems, *Trans. Amer. Math. Soc.*, 89 (1953), pp. 25–59.

[105] ———, On completely recursively enumerable classes and their key arrays, *J. Symbolic Logic*, 21 (1956), pp. 304–341.

[106] H. ROGERS, JR., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.

[107] J.B. ROSSER, A mathematical logic without variables, *Ann. Math.*, 36 (1935), pp. 127–150.

[108] A. SALOMAA, Two complete axiom systems for the algebra of regular events, *J. Assoc. Comput. Mach.*, 13 (1966), pp. 158–169.

[109] A. SALOMAA AND M. SOITTOLA, *Automata Theoretic Aspects of Formal Power Series*, Springer-Verlag, New York, 1978.

[110] S. SCHEINBERG, Note on the Boolean properties of context-free languages, *Information and Control*, 3 (1960), pp. 372–375.

[111] M. SCHÖNFINKEL, Über die Bausteine der mathematischen Logik, *Math. Annalen*, 92 (1924), pp. 305–316.

[112] M.P. SCHÜTZENBERGER, On context-free languages and pushdown automata, *Information and Control*, 6 (1963), pp. 246–264.

[113] J.I. SEIFERAS AND R. MCNAUGHTON, Regularity-preserving relations, *Theor. Comput. Sci.*, 2 (1976), pp. 147–154.

[114] J.C. SHEPHERDSON, The reduction of two-way automata to one-way automata, *IBM J. Res. Develop.*, 3 (1959), pp. 198–200.

[115] J.R. SHOENFIELD, *Degrees of Unsolvability*, North-Holland, Amsterdam, 1971.

[116] R.I. SOARE, *Recursively Enumerable Sets and Degrees*, Springer-Verlag, Berlin, 1987.

[117] D. STANAT AND S. WEISS, A pumping theorem for regular languages, *SIGACT News*, 14 (1982), pp. 36–37.

[118] L.J. STOCKMEYER, The polynomial-time hierarchy, *Theor. Comput. Sci.*, 3 (1976), pp. 1–22.

[119] J.W. THATCHER AND J.B. WRIGHT, Generalized finite automata theory with an application to a decision problem of second order logic, *Math. Syst. Theory*, 2 (1968), pp. 57–81.

[120] A.M. TURING, On computable numbers with an application to the Entscheidungsproblem, *Proc. London Math. Soc.*, 42 (1936), pp. 230–265. Erratum: Ibid., 43 (1937), pp. 544–546.

[121] ——, Systems of logic based on ordinals, *Proc. London Math. Soc.*, 42 (1939), pp. 230–265.

[122] M.Y. VARDI, A note on the reduction of two-way automata to one-way automata, *Information Processing Letters*, 30 (1989), pp. 261–264.

[123] A.N. WHITEHEAD AND B. RUSSELL, *Principia Mathematica*, Cambridge University Press, Cambridge, 1910–1913. Three volumes.

[124] A. YASUHARA, *Recursive Function Theory and Logic*, Academic Press, New York, 1971.

[125] D.H. YOUNGER, Recognition and parsing of context-free languages in time n^3, *Information and Control*, 10 (1967), pp. 189–208.

Notation and Abbreviations

$\subseteq$	subset	7
N	natural numbers	8
Σ	finite alphabet	8
$a, b, \ldots$	letters	8
$x, y, \ldots$	strings	8
$\|x\|$	length of string x	8
ϵ	null string	8
$\in$	set containment	8
a^n	string of n a's	8
Σ^*	set of strings over Σ	8
$\{x \mid P(x)\}$	set of all x satisfying property P	8
$\varnothing$	empty set	8
x^n	concatenation of n x's	9
$\#a(x)$	number of a's in x	9
$A, B, \ldots$	sets of strings	10
$\|A\|$	cardinality of set A	10
$\cup$	set union	10

iff	if and only if	10
$\cap$	set intersection	10
$\sim$	set complement	10
A^n	nth power of A	11
A^*	asterate of A	11
A^+	$= AA^*$	11
DFA	deterministic finite automaton	15
δ	transition function of DFA	15
$\times$	Cartesian product	15
$\widehat{\delta}$	extended transition function of DFA	16
$L(M)$	language accepted by M	17
$\#x$	number represented by binary string x	20
$a \bmod n$	remainder upon dividing a by n	20
$a \equiv b \bmod n$	a congruent to b mod n	20
NFA	nondeterministic finite automaton	26
S	start states of NFA	32
Δ	transition function of NFA	32
2^Q	power set of Q	32
$p \xrightarrow{a} q$	nondeterministic transition	32
$\widehat{\Delta}$	extended transition function of NFA	33
$p \xrightarrow{\epsilon} q$	ϵ-transition	36
$\alpha, \beta, \ldots$	patterns, regular expressions	40
$L(\alpha)$	set of strings matching pattern α	40
ϵ	atomic pattern	41
$\emptyset$	atomic pattern	41
#	atomic pattern	41
@	atomic pattern	41
+	pattern operator	41

$\cap$	pattern operator	41
$\cdot$	pattern operator	41
$+$	pattern operator	41
$*$	pattern operator	41
$\sim$	pattern operator	41
$\leq$	inclusion (regular expressions)	50
α_{uv}^{X}	regular expression for paths from u to v through X	51
$\leq$	natural order in a Kleene algebra	56
$\mathcal{M}(n, \mathcal{K})$	$n \times n$ matrices over Kleene algebra $\mathcal{K}$	58
$h(A)$	image of A under h	62
$h^{-1}(B)$	preimage of B under h	62
$\approx$	collapsing relation	80
$[p]$	equivalence class of p	80
$M/\approx$	quotient automaton	80
$\binom{n}{k}$	n choose k	85
$\equiv_M$	Myhill–Nerode relation induced by M	90
$M_{\equiv}$	DFA constructed from Myhill–Nerode relation $\equiv$	91
$\equiv_{M_{\equiv}}$	Myhill–Nerode relation constructed from $M_{\equiv}$	92
$M_{\equiv_M}$	automaton constructed from $\equiv_M$	92
$\equiv_R$	maximal Myhill–Nerode relation for R	96
$C_{\approx}(B)$	states related to B under a bisimulation	101
$A \approx B$	bisimulation extended to sets	101
$\circ$	relational composition	102
$\equiv_M$	maximal autobisimulation	103
$\gtrsim$	bisimulation between an NFA and its quotient	104
Σ	signature	108
T_Σ	ground terms over Σ	109
$f^{\mathcal{A}}$	interpretation of function symbol f	110

$R^{\mathcal{A}}$	interpretation of relation symbol R 110
$c^{\mathcal{A}}$	interpretation of constant c 110
$t^{\mathcal{A}}$	interpretation of ground term t 110
$L(\mathcal{A})$	set accepted by term automaton $\mathcal{A}$ 111
$T_\Sigma(A)$	term algebra with relation R interpreted as A 113
$\equiv_\sigma$	kernel of homomorphism σ 114
2DFA	two-way deterministic finite automaton 119
2NFA	two-way nondeterministic finite automaton 119
$\vdash$	left endmarker 119
$\dashv$	right endmarker 119
δ	transition function of 2DFA 120
$\xrightarrow[x]{1}$	next configuration relation of 2DFA on input x 122
$\xrightarrow[x]{n}$	nth iterate of $\xrightarrow[x]{1}$ 122
$\xrightarrow[x]{*}$	reflexive transitive closure of $\xrightarrow[x]{1}$ 122
$L(M)$	set accepted by a 2DFA 122
$\bullet$	used in 2DFA tables 124
$\perp$	used in 2DFA tables 124
BNF	Backus–Naur form 129
|	for separating alternatives in a CFG 130
$L(G)$	language generated by CFG G 131
CFL	context-free language 131
PDA	pushdown automaton 131
LIFO	last-in-first-out 132
CFG	context-free grammar 132
$A, B, \ldots$	nonterminals of a CFG 132
$a, b, \ldots$	terminals of a CFG 132
$\alpha, \beta, \ldots$	strings of terminals and nonterminals of a CFG 132
$A \to \alpha$	production of a CFG 132

$\xrightarrow[G]{1}$	derivation step of a CFG 133
$\xrightarrow[G]{*}$	reflexive transitive closure of $\xrightarrow[G]{1}$ 133
$L(G)$	language generated by CFG G 133
PAREN	balanced strings of parentheses 135
$L(x)$	number of left parentheses in x 135
$R(x)$	number of right parentheses in x 135
CNF	Chomsky normal form 140
GNF	Greibach normal form 140
$\xrightarrow[G]{L}$	derivation used in Greibach normal form proof 144
NPDA	nondeterministic pushdown automaton 157
$\perp$	initial stack symbol 158
$\xrightarrow[M]{1}$	next configuration relation of PDA M 159
$\xrightarrow[M]{n}$	nth iterate of $\xrightarrow[M]{1}$ 160
$\xrightarrow[M]{*}$	reflexive transitive closure of $\xrightarrow[M]{1}$ 160
DPDA	deterministic pushdown automaton 176
$\dashv$	right endmarker of a DPDA 176
DCFL	deterministic context-free language 177
$\bot$	falsity 181
$\top$	truth 181
$\wedge$	and 181
$\vee$	or 181
$\rightarrow$	implies 181
$\leftrightarrow$	if and only if 181
$\neg$	negation 181
CKY	Cocke–Kasami–Younger algorithm 191
PAREN_n	balanced strings of parentheses of n types 198
PAREN_Γ	parenthesis language on parentheses Γ 199
ψ	Parikh map 201

TM	Turing machine	206
PA	Peano arithmetic	209
$\sqcup$	blank symbol	210
$\sqcup^\omega$	semi-infinite string of blanks	212
ω	smallest infinite ordinal	212
$\alpha, \beta, \ldots$	Turing machine configurations	212
$\xrightarrow[M]{1}$	next configuration relation of Turing machine M	213
$\xrightarrow[M]{*}$	reflexive transitive closure of $\xrightarrow[M]{1}$	213
$L(M)$	strings accepted by Turing machine M	213
r.e.	recursively enumerable	213
co-r.e.	complement of a recursively enumerable set	213
$L(E)$	strings enumerated by enumeration machine E	226
U	universal Turing machine	229
HP	halting problem	230
MP	membership problem	230
$\leq_m$	many-one reducibility	240
FIN	finiteness problem	241
VALCOMPS	valid computation histories	250
CSG	context-sensitive grammar	258
LBA	linear bounded automaton	258
s	successor function	258
z	zero function	258
π_k^n	projection function	258
$\circ$	functional composition	258
μ	minimization operator	259
$\mathbf{const}_n$	constant function with value n	259
$\dot{-}$	proper subtraction	260
$\rightarrow$	reduction in the λ-calculus	263

$\xrightarrow{\alpha}$	α-reduction	264
$\xrightarrow{\beta}$	β-reduction	264
$\overline{n}$	Church numeral for n	265
f^n	n-fold composition of f	266
S	primitive combinator	266
K	primitive combinator	266
$\rightarrow$	reduction in combinatory logic	267
Env	set of all environments	270
$\sigma[x \leftarrow a]$	function σ with value of x changed to a	271
$\leq_T$	Turing reducibility	275
Σ_1^0	r.e. sets	276
Δ_1^0	recursive sets	276
Π_1^0	co-r.e. sets	276
Σ_n^0	class in arithmetic hierarchy	276
Π_n^0	class in arithmetic hierarchy	276
Δ_n^0	class in arithmetic hierarchy	276
EMPTY	set of TMs accepting $\varnothing$	277
TOTAL	set of total TMs	277
FIN	set of TMs accepting finite sets	278
COF	set of TMs accepting cofinite sets	278
Δ_1^1	hyperarithmetic sets	281
Π_1^1	inductive sets	281
L	language of number theory	282
$\forall$	for all	283
$\exists$	there exists	283
Th(N)	first-order number theory	284
$\vdash$	provability in PA	292
$\models$	truth	292

$\ulcorner\varphi\urcorner$	integer code of formula φ	292
ZF	Zermelo–Fraenkel set theory	297
$\|\|$	shuffle operator	304
PAREN_2	balanced strings of parentheses of two types	306
$\leftarrow$	residuation operator	310
$C_\epsilon(A)$	ϵ-closure of A	318
$\overset{\infty}{\forall}$	for all but finitely many	324

Index

Abstraction, 5
  functional, 263
  λ-, 263
Acceptance, 17
  2DFA, 120, 122
  by empty stack, 160, 163, 164
  by final state, 160, 164
  DFA, 16
  DPDA, 177
  NFA, 26, 33
  NFA with ϵ-transitions, 65, 318
  term automaton, 111
  Turing machine, 210, 213, 216
Accept state, 15, 111, 121, 158, 210, 211
Accessible, 29, 75, 77, 78, 84, 94, 95, 103, 116
Ackermann, W., 273
Ackermann's function, 261, 273
Aho, A.V., 190
Algorithm
  CKY, 191–195, 249, 338
  Cocke–Kasami–Younger, *see* CKY
  collapsing, 84–88, 106–107, 327, 365
  Euclidean, 340, 364
  maximal bisimulation, 106–107
  minimization, *see* collapsing
  parsing, 181–190
  primality, 217–219
  recursive, 214
Alphabet, 8
  single letter, 73, 74, 318, 326, 335, 346
  stack, 158
α-reduction, 264
Alternating
  finite automaton, 330, 366
  quantifiers, *see* quantifier
Ambiguous, 131, 182, 186, 339
Analytic hierarchy, *see* hierarchy
Annihilator, 12, 56
Antimatter, 354
Antisymmetry, 56, 96
Application operator, 267
Arbib, M.A., 118
Arithmetic hierarchy, *see* hierarchy
Arity, 108
Assignment statement, 130
Associativity, 9, 11, 42, 45, 55, 264, 266, 285
Asterate, 11, 56
  closure of CFLs under, 195
  closure of regular sets under, 38
Atomic pattern, 40, 41
Autobisimulation, 103
  maximal, 106
Automaton
  counter, 224–225, 227, 341

Automaton (*cont.*)
finite, 4, 14
deterministic, 15, 24, 54
nondeterministic, 26, 32, 39
two-way, 119–128, 331
with ϵ-transitions, 36–37, 65, 318
linear bounded, 4, 258, 268, 309, 313
product, 22
pushdown, 4, 131, 175
deterministic, 163, 176–177, 191, 196
nondeterministic, 157
one-state, 172
quotient, 80
term, 108–118
two-stack, 224
Axioms
Kleene algebra, 55–56
Peano arithmetic, 284
Axiom scheme, 285

Backhouse, R.C., 60
Backus, J.W., 129, 134
Backus–Naur form, 129, 134
Balanced parentheses, 131, 135–140, 143, 145, 161, 198, 304, 306, 307, 329
Bar-Hillel, Y., 71, 147, 156, 197, 255
Basic pump, 203
Basis, 21
Begin statement, 130
β-reduction, 264
Biconditional, 181
Binary
relation, 110
representation, 19, 207, 224, 284, 301, 307, 323, 329
symbol, 108
Bisimilar, 101
Bisimilarity class, 101
Bisimulation, 100, 101, 331
strong, 100
Blank symbol, 210, 211
Bloom, S.L., 60
BNF, *see* Backus–Naur form
Boffa, M., 60
Brainerd, W.S., 118, 268

Cantor, G., 230, 243, 297, 298
Capture of free variable, 264
Cardinality, 10, 208
Carrier, 110
Cell, 119, 157
CFL, *see* context-free language
Characteristic function, 275, 330
Chomsky
hierarchy, 4, 257, 268
normal form, *see* normal form
Chomsky, N., 4, 54, 134, 147, 175, 200, 268
Chomsky–Schützenberger theorem, 198–200
Church
–Rosser property, 265
–Turing thesis, *see* Church's thesis
numerals, 265
Church, A., 4, 206, 207, 214, 265, 268
Church's thesis, 207–208, 214
CKY algorithm, 191–195, 197, 249, 338
Closure
properties
of CFLs, 154, 155, 195–197, 308, 335, 367
of DCFLs, 177–180, 196–197, 338
of r.e. sets, 340
of regular sets, 21–24, 37–39, 47, 62, 302
reflexive transitive, 56, 110, 160, 213, 315, 322
CNF, *see* normal form
Coarser, 96
Coarsest, 97
Cocke, J., 197
Cocke–Kasami–Younger algorithm, *see* CKY algorithm
Cofinite, 278
Collapsing, 77, 81
algorithm, 84–88, 98, 106–107, 327, 365
DFA, 77–99
NFA, 100–107
relation, 80, 84, 98, 100, 101
Combinator, 207, 266
Combinatorial completeness, 267

Combinatory logic, 4, 206, 207, 256, 266, 268
Commutative image, 202
Commutativity, 12, 55, 202
Compiler compiler, 181
Complement, 10
  CFLs not closed under, 155, 196
  closure of DCFLs under, 177–180, 196
  closure of recursive sets under, 219
  closure of regular sets under, 23
  r.e. sets not closed under, 219
Completeness
  for a class of decision problems, 278–281, 349
  combinatorial, 267
  deductive, 286
Complete partial order, 339
Complexity theory, 281
Composition
  functional, 240, 258, 263, 271
  relational, 57, 101, 110, 122
  sequential, 269
Compound pattern, 40, 41
Computability, 5, 206, 207, 256
Computable
  function, 347
    total, 273, 347, 350
  reduction, 240
Concatenation
  closure of CFLs under, 195
  closure of regular sets under, 37
  set, 10, 37
  string, 9
Concurrency, 100
Conditional, 269
Configuration
  2DFA, 122
  DPDA, 177
  PDA, 158
  queue machine, 368
  start, 122, 159, 212, 216, 250
  Turing machine, 212, 216, 250
Congruence, 114, 115, 117, 202
  class, 116
  right, 90, 95, 97, 116
Consistency, 296, 297
Constant, 108
  propositional, 181
Context, 116
Context-free
  grammar, 4, 129, 131, 132, 134
  language, 4, 131, 133, 134
    deterministic, 177, 191, 196
Context-sensitive
  grammar, 4, 258, 268, 313
  language, 4
Contradiction, 68, 74
Contrapositive, 70, 153
Conway, J.H., 60
Cook, S.A., 281
Co-r.e., 213
Coset, 202
Counter, 163, 224
  automaton, 224–225, 227, 341
Curry, 263
Curry, H.B., 4, 206, 263, 268
Cycle, 337

Davis, M., 268
DCFL, *see* deterministic context-free language
Dead code, 310
Decidability, 220, 235–236
Decision problem, 7, 284
Demon game
  for CFLs, 153–155
  for regular sets, 71–73
De Morgan laws, 12, 24, 45
Denesting rule, 57
Depth, 150, 203, 330
Derivation
  leftmost, 149, 168
  rightmost, 149
  tree, *see* parse tree
Determinism, 176
Deterministic
  context-free language, 177, 191, 196
  finite automaton, 54
  pushdown automaton, 163, 175–177, 191, 196
DFA, *see* finite automaton
Diagonalization, 230–234, 239, 243
Direct product, 108
Disjoint
  pumps, 205

Disjoint (*cont.*)
union, 48
Distributivity, 12, 56, 285
DPDA, *see* deterministic pushdown automaton
Dyck language, *see* parenthesis language
DYLAN, 262
Dynamic programming, 191

Effective, *see* computability
Ehrenfeucht, A., 71
Eilenberg, S., 118
Emptiness problem
for CFLs, 249, 348
for LBAs, 310
Empty string, 8
Encoding, 207, 292
of configurations, 250
of natural numbers in the λ-calculus, 265
of natural numbers in ZF, 297
Endmarker, 120, 163, 176, 196, 211
Engelfriet, J., 118
Enumeration
machine, 225–227
state, 226
Environment, 270
Epimorphism, 113, 116
ϵ-closure, 37, 65, 318
ϵ-production, 141–142, 145, 147, 334
ϵ-transition, 36–37, 65, 158, 179, 318
Equational logic, 50
Equivalence
among variations of TMs, 221–227
class, 80
letter-, 203
of acceptance by final state and empty stack, 164–166
of bisimilar automata, 103
of CSGs and LBAs, 258, 268
of DFAs and
2DFAs, 124–128
NFAs, 28–36
NFAs with ϵ-transitions, 318
regular expressions, 46–54, 59
right-linear grammars, 54, 306
of μ-recursive functions and the λ-calculus, 265
of NPDAs and
CFGs, 167–175
one-state NPDAs, 172
of primitive recursive functions and **for** programs, 273
of TMs and
enumeration machines, 225–227
four-counter automata, 224–225
λ-calculus, 268
Post systems, 257
two-counter automata, 225
two-stack automata, 224
type 0 grammars, 258, 268, 343
of **while** programs and μ-recursive functions, 271
problem, 310
relation, 49, 80, 90, 96
Erase, 217
Ésik, Z., 60
Euclid, 363
Euclidean algorithm, 340, 364
Evaluation, 262
Evey, J., 175
Exponentiation, 259
Expression
regular, *see* regular expression
tree, 182

Final state, *see* accept state
Finer, 96
Finite
automaton, 4, 14
alternating, 330, 366
deterministic, 15, 24, 54
nondeterministic, 32, 39
two-way, 119–128, 331
with ϵ-transitions, 36–37, 65, 318
control, 211
index, 90, 95, 116–117
-state transition system, 14, 24
Finitely generated, 202
Finiteness problem
for CFLs, 250, 349
for regular sets, 349
Fischer, P.C., 180, 227
Fixpoint, 293, 339, 346, 347

lemma, 292
For
loop, 269
program, 269, 273
Formalist program, 5, 209, 282
Formal proof system, 282
Free
commutative monoid, 202
monoid, 202
variable, 284, 293
Function
Ackermann's, 261, 273
characteristic, 275, 330
computable, 347
monotone, 339
μ-recursive, 4, 206, 214, 256, 258–261, 268, 269, 345
pairing, 277
partial, 270, 273, 347
primitive recursive, 259, 269, 273, 313
regularity preserving, 76, 324
transition, *see* transition function
weakly regularity preserving, 324
Functional
abstraction, 262–264
application, 262–264
composition, *see* composition

Garey, M., 281
GCD, *see* greatest common divisor
Gécseg, F., 60, 118
Generalization, 285
Generate, 202, 257, 258
Ginsburg, S., 39, 180, 197, 255, 335
Give'on, Y., 118
GNF, *see* normal form
Gödel, K., 4, 5, 206, 207, 209, 214, 258, 268, 269, 273, 282, 287, 292
Gödel numbering, 208
Gödel's
fixpoint lemma, *see* fixpoint lemma
incompleteness theorem, *see* incompleteness theorem
Goldstine, J., 205
Grammar, 4, 257
context-free, 4, 129, 132, 134
context-sensitive, 4, 258, 268, 313
left-linear, 306
right-linear, 4, 54, 257, 306
strongly left-linear, 306
strongly right-linear, 306
type 0, 256, 257, 268
type 1, 257, 313
type 2, 257
type 3, 257
Greatest common divisor, 326, 340, 364
Greibach, S.A., 147, 180, 307, 328
Greibach normal form, *see* normal form
Ground term, 109
Guess and verify, 26

Haines, L., 180
Halting
2DFA, 122
configuration, 251
DPDA, 177, 179–180
LBA, 309
problem, 230, 309
undecidability of, 231–234, 244
Turing machine, 213, 216
Hamming distance, 65, 302
Hardest r.e. set, 278
Hardness, 278
Harrison, M.A., 205
Hartmanis, J., 281
Head
read-only, 119, 157, 224
read/write, 210, 212, 220, 221
write-only, 225
Hennie, F.C., 268
Herbrand, J., 4, 206, 214, 268
Hierarchy
analytic, 281
arithmetic, 276–281
Chomsky, 4, 257, 268
Higman's lemma, 329, 365
Hilbert, D., 5, 209, 282
Historical notes, 24, 39, 54, 60, 71, 76, 99, 117, 128, 134, 147, 156, 175, 180, 190, 197, 200, 205, 214, 227, 243, 248, 255, 268, 273, 281

Homomorphic image, 62, 76, 108, 113, 117, 198, 199
  closure of CFLs under, 196, 335, 367
  closure of regular sets under, 62
  minimal, 117
Homomorphic preimage, 62
  closure of CFLs under, 196, 335, 367
  closure of regular sets under, 62
Homomorphism, 61–66, 74, 76, 108, 112, 114, 198, 319
  Σ-algebra, 112
Hopcroft, J.E., 24, 99, 197
Horn clause, 348
Huffman, D.A., 99

Idempotence, 56
Idempotent semiring, 56
Identity
  element, 9, 12, 56, 285
  function, 267
  relation, 96, 105, 110
If statement, 130
Image, 62
Implication, 181
Inaccessible, *see* accessible
Incompleteness theorem, 5, 209, 282–298
Induction
  axiom, 285
  hypothesis, 20
  step, 21
  structural, 47
Infinite string, 108
Initial stack symbol, 158, 163
Input string, 16
Integer division, 283
Interactive computation, 208
Intersection, 10
  CFLs not closed under, 196
  closure of r.e. sets under, 340
  closure of regular sets under, 22–23
  of CFLs, 311, 312
  of CFLs with regular sets, 195, 308, 367
Intuitionism, 297
Inverse, 260

Isomorphism, 78, 100, 105, 113, 229
  of automata, 30, 89, 92–95

Jaffe, J., 71, 325
Johnson, D., 281
Jones, N.D., 268

Kapur, S., 329
Karp, R.M., 281
Kasami, T., 197
Kernel, 114, 115, 117
Kleene, S.C., 4, 24, 54, 55, 206, 214, 227, 255, 281
Kleene algebra, 49, 55–60, 110, 321, 322, 358
Knaster–Tarski theorem, 339
Knuth, D.E., 190
Kozen, D.C., 60
Krob, D., 60
Kuich, W., 60, 205
Kuroda, S.Y., 268

Labeled tree, 108, 109, 111
λ-abstraction, 263
λ-calculus, 4, 206, 214, 256, 262–268
  pure, 263
λ-notation, 262
λ-term, 207, 262, 263, 345
Landweber, L.H., 268
Landweber, P.S., 268
Language, 4
  accepted by M, 17
  context-free, 4, 131, 133, 134
  context-sensitive, 4
  deterministic context-free, 177, 191, 196
  Dyck, *see* parenthesis language
  generated by G, 133
  of number theory, 209, 282, 288, 293, 295, 297
  parenthesis, 198
LBA, *see* linear bounded automaton
Least upper bound, 56, 339
Leftmost derivation, 149, 168
Length, 8, 203
Letter, 8
  -equivalence, 203
Level, 149

Lewis, P.M., 190
LIFO, 157
Linear
  bounded automaton, 4, 258, 268, 309, 313
  equations, 59–60
  order, 355
  set, 202
  time, 349, 356
LISP, 262
Logic, 256
  combinatory, 4, 206, 207, 266, 268
  equational, 50
  first-order, 282, 297
  propositional, 181
  second-order, 281
Loop, 75
Looping
  2DFA, 122
  DPDA, 177, 179–180
  LBA, 309
  PDA, 161
  Turing machine, 210, 213, 216

Machtey, M., 268
Many-one reduction, 239
Markov, A.A., 134
Markov system, 134
Match, 40, 41
Matching parentheses, 136
Matrix, 322, 353, 357, 358, 361
  over a Kleene algebra, 58–59
Maximal autobisimulation, 106
McCulloch, W.S., 24
McNaughton, R., 39, 54, 76, 324
Membership problem, 230, 278
  undecidability of, 234
Metastatement, 295
Metasymbol, 229, 295
Metasystem, 295, 297
method, 263
Meyer, A.R., 227, 273
Miller, G.A., 54
Minimal
  $\lhd$-, 203
  $\leq$-, 204
  $\sqsubseteq$-, 366
  derivation, 142
  DFA, 78, 89, 98
  homomorphic image, 117
  NFA, 100–101, 104, 331
  Turing machine, 347
Minimization
  algorithm, *see* collapsing algorithm
  unbounded, 259, 273
Modus ponens, 285
Monoid, 9, 110, 201
  free, 202
  free commutative, 202
Monomorphism, 113
Monotone
  function, 339
  operator, 56
  property, 247, 371
Moore, E.F., 99
Multiple
  of k, 301
  of three, 19
μ-recursive function, 4, 206, 214, 256, 258–261, 268, 269, 345
Myhill, J., 99, 118, 268
Myhill–Nerode
  relation, 90–95, 116, 126, 127, 329
  theorem, 95–99, 119, 126, 353, 364
    for term automata, 116–118, 332
  theory, 100, 108

N-ary symbol, 108
Natural numbers, 207, 230, 282
  encoding
    in the λ-calculus, 265
    in ZF, 297
Naur, P., 129, 134
Nerode, A., 99, 118
Next configuration relation
  2DFA, 122
  PDA, 159
  queue machine, 368
  Turing machine, 213, 216
NFA, *see* nondeterministic finite automaton
Non-CFL, 216
Nondeterminism, 25–26, 162, 177, 227
Nondeterministic
  finite automaton, 26, 32, 39
  pushdown automaton, 157

Nondeterministic (*cont.*)
one-state, 172
Turing machine, 207
Nonmonotone property, 247
Nonregular set, 67, 70, 72–74
Nonterminal symbol
CFG, 130, 132
Post system, 257
Nontrivial property, 245, 246
Normal form
Chomsky, 140, 143–144, 147, 191, 192, 199, 203, 334
Greibach, 140, 144–147, 307, 334
λ-calculus, 265
NPDA, *see* nondeterministic pushdown automaton
Null string, 8
Number theory, 209, 284

Oettinger, A.G., 175
Ogden, W., 156
One-way Turing machine, 355
Operand, 183
Operations
set, 10–13
string, 9–10
Operator
application, 267
Kleene algebra, 55
λ-calculus, 263
monotone, 56
number theory, 282
pattern, 40, 41
precedence, 45, 182, 186–190
propositional, 181, 283
regular expression, 45
shuffle, 304, 308, 337
Oracle, 274, 349
tape, 274
Turing machine, 274
Output tape, 225

P-ary representation, 289, 301
$P = NP$ problem, 25
PA, *see* Peano arithmetic
Pairing function, 277
Palindromes, 131, 134
Papadimitriou, C.H., 281
Paradox, 295
Russell's, 209, 297
Paradoxical combinator, 267
Parentheses, *see* balanced parentheses
Parenthesis language, 198
Parikh, R., 71, 201, 205
Parikh map, 201
Parikh's theorem, 201–205, 354
Parse tree, 148, 149, 181, 203
Parser, 181
Parsing, 181–190
Partial
function, 270, 273, 347
order, 56, 96, 187, 203
complete, 339
Path, 148, 203
Pattern, 40
atomic, 40, 41
compound, 40, 41
matching, 40, 257
operator, 40, 41
PDA, *see* automaton, pushdown
Peák, I., 60
Peano arithmetic, 209, 284, 292, 297, 350
Period, 74
Periodic, 179
Perles, M., 71, 147, 156, 197, 255
Permutation, 337
Pigeonhole principle, 68, 69, 150, 179, 205
Pitts, W., 24
Polynomial, 323, 353, 361
time, 25
Pop, 132, 157
Post, E.L., 4, 134, 206, 227, 244, 256, 268
Postage stamp problem, 326
Post system, 4, 134, 206, 256–257, 268
Power
of a set, 11
of two, 283
set, 32, 231
of N, 230
Precedence
operator, 45, 182, 186–190
relation, 187, 308
Predecessor, 260

Prefix, 9, 135
  proper, 10, 138
Prime, 7, 217, 283, 324, 335, 351, 363
  relatively, 364
Primitive
  recursion, 259
  recursive function, 259, 269, 273, 313
Principia Mathematica, 209
Product
  automaton, 22
  construction, 22–23, 74, 195, 301, 308, 315
Production
  CFG, 132
  ϵ-, 141–142, 147, 334
  Post system, 257
  unit, 141–142, 147, 334
Program that prints itself, 287
Projection, 258
Proof, 285
Proper
  prefix, 10, 138
  subtraction, 260
Property
  monotone, 247, 371
  nonmonotone, 247
  nontrivial, 245, 246
  of r.e. sets, 245
Propositional
  constant, 181
  logic, 181
  operator, 181, 283
  variable, 181
Provability, 293, 296
Pump, 203
  basic, 203
  disjoint, 205
Pumping lemma
  for CFLs, 131, 148–156, 203, 337, 354
  for regular sets, 70–73, 325, 354, 363
Push, 132, 157
Pushdown
  automaton, *see* automaton
  store, 131, 157

Quantifier
  alternation, 71, 276
  first-order, 281, 283
  second-order, 281
Queue, 340
  machine, 257, 340, 354, 368–370
Quotient, 202
  algebra, 108, 116
  automaton, 80, 104
  construction, 80–83, 115

Rabin, M.O., 39, 128
Randomized computation, 208
Ray automaton, 341
R.e., *see* recursively enumerable
  -complete, 278
  -hard, 278
Read head, 119, 157, 224
Read/write head, 210, 212, 220, 221
Recursion theorem, 346–347, 355
Recursive, 216, 219
  algorithm, 214
  in, 275
  relation, 312
  set, 214
Recursively enumerable, 213, 216, 219, 225, 227
  in, 275
Redko, V.N., 60
Reduce, 183, 239
Reducibility relation, 240, 244
Reducible, 240
Reduction, 234, 237, 239
  α-, 264
  β-, 264
  combinatory logic, 266, 267
  many-one, 239, 275
  Turing, 275, 349
Redundancy, 44, 45
Refinement, 90, 95–97, 275
Reflexive transitive closure, *see* closure
Reflexivity, 49, 56, 80, 96
Regular
  expression, 4, 45, 54, 199
    operator, 45
  set, 17, 45, 198
  set of terms, 111

Regularity preserving function, 76, 324
Rejection, *see* acceptance
Reject state, 121, 210, 211
Relation
  collapsing, 80
  equivalence, 49, 80
  next configuration, *see* next configuration relation
  precedence, 187
  reducibility, 240, 244
  transition, 158
  well-founded, 281
Relational composition, *see* composition
Relative computation, 274–281
Relatively prime, 364
Renaming bound variables, 264
Reversal
  closure of regular sets under, 302
  DCFLs not closed under, 197, 338
Reverse, 302, 353
Rice, H.G., 248, 335
Rice's theorem, 245–248, 345, 347, 355, 371
Right
  congruence, 90, 95, 97, 116
  -linear grammar, *see* grammar
Rightmost derivation, 149
Ritchie, D.M., 273
Robust, 207
Rogers, H., 268, 281
Rose, G.F., 39, 197, 255
Rosenberg, A.L., 227
Rosenkrantz, D.J., 190
Rosser, J.B., 265, 268
Rozenberg, G., 71
Rubik's cube, 14
Rule
  denesting, 57
  of inference, 284, 285
  shifting, 57
Russell, B., 5, 209, 297
Russell's paradox, *see* paradox

Salomaa, A., 60
Scheinberg, S., 197
SCHEME, 262
Schönfinkel, M., 4, 206, 268
Schützenberger, M.P., 175, 180, 200
Scott, D.S., 39, 128
Second incompleteness theorem, 295–297
Seiferas, J.I., 76, 324
Self-reference, 208, 287, 293
Semantics, 270
Semidecidability, 220
Semi-infinite, 212
Semilinear set, 202
Semiring, 56
Sentence, 133
  of first-order logic, 284
Sentential form, 130, 133
Sequential composition, *see* composition
Set
  accepted by M, 17
  complement, *see* complement
  concatenation, *see* concatenation
  intersection, *see* intersection
  operations, 10–13
  regular, *see* regular set
  theory
    Cantor, 297
    Zermelo–Fraenkel, 297
  union, *see* union
Shamir, E., 71, 147, 156, 197, 255
Shepherdson, J.C., 128
Shifting rule, 57
Shift-reduce parser, 196
Shoenfield, J.R., 281
Shuffle operator, 304, 308, 337
Sieve of Eratosthenes, 217–219, 309
Σ-algebra, 110
Signature, 108, 339
  group, 109
  Kleene algebra, 109
  monoid, 108
Simple assignment, 269
Simplification rules for regular expressions, 49
Single letter alphabet, *see* alphabet
Singleton, 35, 42, 46
Soare, R.I., 281
Soittola, M., 60
Soundness, 285, 292, 294, 296

Spanier, E.H., 197
Square root, 261, 349
Stack, 131, 157
  alphabet, 158
Stanat, D., 71, 326
Start
  configuration, *see* configuration
  state, 15, 120, 158, 210, 211
  symbol
    CFG, 132
    Post system, 257
State, 14, 15, 111
  accept, 15, 111, 121, 158, 210, 211
  enumeration, 226
  final, *see* accept
  minimization, *see* collapsing
  of a **while** program, 270
  reject, 121, 210, 211
  start, 15, 120, 158, 210, 211
Statement, 130
Stearns, R.E., 190, 281
Steinby, M., 118
Strategy, 71, 153
String, 8
  concatenation, *see* concatenation
  empty, 8
  infinite, 108
  null, 8
  operations, 9–10
Subset construction, 28, 35–37, 39, 77, 302, 316, 320, 331, 353
Substitution, 262, 264, 293
Successor, 258, 262, 266, 285
Succinctness, 317
Support, 103
Symbol, 8
  table, 183
Symmetry, 49, 80, 101
Syntactic algebra, *see* term algebra
Syntax, 181

Tape, 210
  alphabet, 211
  cell, 213
  output, 225
Term
  algebra, 111
  automaton, 108–118
  ground, 109
  λ-, 207, 262, 263, 345
Terminal symbol
  CFG, 132
  Post system, 257
Ternary symbol, 108
Thatcher, J.W., 117
Theorem, 285
Timesharing, 227, 247
TM, *see* Turing machine
Total, 213
  computable function, 273, 347, 350
  Turing machine, 216, 219
T-predicate, 255
Track, 66, 220, 222
Transition, 14
  diagram, 15
  function
    2DFA, 120
    DFA, 15
    NFA, 33
    Turing machine, 210, 211, 215
  matrix, 16, 317
  relation
    PDA, 158
  system, 14, 24
  table, 15
Transitivity, 49, 56, 80, 96, 101
  of reductions, 240
Trick, 73
Truth, 292, 296
Turing, A.M., 4, 206, 210, 214, 227, 244, 268, 281
Turing
  machine, 4, 206, 207, 210, 214, 227, 268
    deterministic one-tape, 210, 215
    minimal, 347
    multidimensional tape, 207, 342
    multitape, 207, 221
    nondeterministic, 207
    oracle, 274
    two-way infinite tape, 207, 223
    universal, 228–231, 244
  reduction, *see* reduction, Turing
Two-stack automaton, 224
Two-way finite automaton, 119–128, 331

2DFA, *see* two-way finite automaton
2NFA, *see* two-way finite automaton
Type
  n grammar, *see* grammar
  theory, 297

Ullman, J.D., 24, 190, 197
Ultimate periodicity, 74–76, 324, 353, 354
Unambiguous, 131, 182, 186
Unary
  relation, 110
  representation, 323
  symbol, 108
Unbounded minimization, 259, 273
Undecidability, 236–238
  of the halting problem, 231–234, 244
  of the membership problem, 234
Union, 10, 110
  closure of CFLs under, 195
  closure of r.e. sets under, 340
  closure of regular sets under, 24
  DCFLs not closed under, 197
  disjoint, 48
Unit production, 141–142, 147, 334
Universal
  algebra, 108
  machine, 208
  relation, 96
  simulation, 208
  Turing machine, 228–231, 244
UNIX, 40

VALCOMPS, *see* valid computation history
valid computation history, 250, 255, 284, 289, 310, 311
Vardi, M.Y., 128, 331
Variable
  combinatory logic, 266
  free, 284
  Kleene algebra, 59
  λ-calculus, 263
  Post system, 257
  programming language, 269
  propositional, 181

Weiss, S., 71, 326
Well
  -defined, 81
  -founded, 203, 281
  -parenthesized, 182
While
  loop, 269
  program, 256, 269, 335, 348
  statement, 130
Whitehead, A.N., 209
Winning strategy, 71, 153
Work tape, 225
Wright, J.B., 117, 118

Yamada, H., 39, 54
Yasuhara, A., 268
Young, P., 268
Younger, D.H., 197

Zermelo–Fraenkel set theory, 297
ZF, *see* Zermelo–Fraenkel set theory