

Study Formulæ

Philip Gross

May 5, 2003

1 Chapter 1

1.1 Relations

R is *reflexive* if for every $a \in A$, aRa .

R is *symmetric* if for every a and b in A , if aRb , then bRa .

R is *transitive* if for every a , b , and c in A , if aRb and bRc , then aRc .

Equivalence relation defines an *equivalence class* (think modulo).

$$[a]_R = \{x \in A \mid xRa\}$$

1.2 Languages

A *language* is simply a set of strings involving symbols from some alphabet.

The set of all strings over an alphabet Σ is Σ^* , and so all languages over Σ are subsets of Σ^* .

Concatenation of languages: if $L_1, L_2 \subseteq \Sigma^*$, then

$$L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

Kleene *:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

2 Chapter 2

2.1 Induction

Suppose $P(n)$ is a statement involving an integer n . Then to prove that $P(n)$ is true for every $n \geq n_0$, it is sufficient to show:

1. $P(n_0)$ is true.
2. For any $k \geq n_0$, if $P(k)$ is true, then $P(k+1)$ is true.

Strong principle: Suppose $P(n)$ is a statement involving an integer n . Then to prove that $P(n)$ is true for every $n \geq n_0$, it is sufficient to show:

1. $P(n_0)$ is true.
2. For any $k \geq n_0$, if $P(k)$ is true for every n satisfying $n_0 \leq n \leq k$, then $P(k+1)$ is true.

3 Chapter 3

3.1 Regular Languages

A *regular language* over an alphabet Σ is one that can be obtained from the very simplest languages over Σ , those containing a single string of length 0 or 1, using only the operations of union, concatenation, and Kleene $*$.

The class R of regular languages over Σ :

1. \emptyset is an element of R , and the corresponding regular expression is \emptyset .
2. $\{\Lambda\}$ is an element of R , and the corresponding regex is Λ .
3. For each $a \in \Sigma$, $\{a\}$ is an element of R , and the corresponding regex is a .
4. if L_1 and L_2 are any elements of R , and r_1 and r_2 are the corresponding regexes, then
 - (a) $L_1 \cup L_2$ is an element of R , and the regex is $(r_1 + r_2)$
 - (b) $L_1 L_2$ is an element of R , and the regex is $(r_1 r_2)$
 - (c) L_1^* is an element of R and the regex is (r_1^*)

3.2 Finite Automata

A *finite automaton* is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where

- Q is a finite set of *states*
- Σ is a finite alphabet of *input symbols*
- $q_0 \in Q$ is the *initial state*
- δ is a function from $Q \times \Sigma \rightarrow Q$
- $A \subseteq Q$ is the set of *accepting states*

For any element q of Q , and any symbol $a \in \Sigma$, we interpret $\delta(q, a)$ as the state to which the FA moves, if it is in state q and receives the input a .

Let $M = (Q, \Sigma, q_0, A, \delta)$ be an FA. We define the function

$$\delta^* : Q \times \Sigma^* \rightarrow Q$$

as follows:

1. For any $q \in Q$, $\delta^*(q, \Lambda) = q$

2. For any $y \in \Sigma^*$, $a \in \Sigma$ and $q \in Q$,

$$\delta^*(q, ya) = \delta(\delta^*(q, y), a)$$

“The state in which the FA ends up, if it begins in state q and receives the string x of input symbols = $\delta^*(q, x)$.”

Let $M = (Q, \Sigma, q_0, A, \delta)$ be an FA. A string $x \in \Sigma^*$ is *accepted* by M if $\delta^*(q_0, x) \in A$.

$$L(M) = \{x \in \Sigma^* \mid x \text{ is accepted by } M\}$$

A language L over Σ is regular iff \exists an FA that accepts L .

3.3 Distinguishability

Let L be a language in Σ^* . Two strings x and y in Σ^* are *distinguishable with respect to L* if \exists a string $z \in \Sigma^*$ so that exactly one of the strings xz and yz is in L . The string z is said to distinguish x and y with respect to L .

Suppose that $L \subseteq \Sigma^*$ and $M = (Q, \Sigma, q_0, A, \delta)$ is any FA recognizing L . If x and y are two strings in Σ^* for which $\delta^*(q_0, x) = \delta^*(q_0, y)$, then x and y are indistinguishable with respect to L .

Suppose that $L \subseteq \Sigma^*$ and, for some positive integer n , there are n strings in Σ^* , any two of which are distinguishable with respect to L . Then there can be no FA recognizing L with fewer than n states. (pigeonhole principle).

3.4 Union, Intersection, Subtraction

To produce a new language from two others, let $Q = Q_1 \times Q_2$, $q_0 = (q_1, q_2)$, and $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$, then if

- $A = \{(p, q) \mid p \in A_1 \text{ or } q \in A_2, M \text{ accepts } L_1 \cup L_2\}$.
- $A = \{(p, q) \mid p \in A_1 \text{ and } q \in A_2, M \text{ accepts } L_1 \cap L_2\}$.
- $A = \{(p, q) \mid p \in A_1 \text{ and } q \notin A_2, M \text{ accepts } L_1 - L_2\}$.

4 Chapter 4

4.1 NFAs

An NFA is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where Q and Σ are nonempty finite sets, $q_0 \in Q$, $A \subseteq Q$, and

$$\delta : Q \times \Sigma \rightarrow 2^Q$$

4.2 Nonrecursive definition of NFA δ^*

For an arbitrary NFA $M = (Q, \Sigma, q_0, A, \delta)$ and for any $p \in Q$, $\delta^*(p, \Lambda) = \{p\}$. For any $p \in Q$ and for any $x = a_1 a_2 \cdots a_n \in \Sigma^*$, $\delta^*(p, x)$ is the set of all states q for which there is a sequence of states $p = p_1 p_2 \cdots p_n = q$ satisfying $p_i \in \delta(p_{i-1}, a_i)$ for each i with $1 \leq i \leq n$.

4.3 Recursive definition of NFA δ^*

Let $M = (Q, \Sigma, q_0, A, \delta)$ be an NFA. The function $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$ is defined as follows:

1. For any $q \in Q$, $\delta^*(q, \Lambda) = \{q\}$.
2. For any $y \in \Sigma^*$, $a \in \Sigma$, and $q \in Q$

$$\delta^*(q, ya) = \bigcup_{p \in \delta^*(q, y)} \delta(p, a)$$

4.4 The big theorem

For any NFA $M = (Q, \Sigma, q_0, A, \delta)$ accepting a language $L \subseteq \Sigma^*$, there is an FA $(Q_1, \Sigma, q_1, A_1, \delta_1)$ that also accepts L .

M_1 is defined as: $Q_1 = 2^Q$, $q_1 = \{q_0\}$, $q \in Q_1$, $a \in \Sigma$, $\delta_1(q, a) = \bigcup_{p \in q} \delta(p, a)$, $A_1 = \{q \in Q_1 \mid q \cap A \neq \emptyset\}$

4.5 Λ closure

For an NFA- Λ , $\delta : Q \times (\Sigma \cup \{\Lambda\}) \rightarrow 2^Q$.

The Λ -closure of S is the set $\Lambda(S)$:

1. Every element of S is an element of $\Lambda(S)$
2. For any $q \in \Lambda(S)$, every element of $\delta(q, \Lambda)$ is in $\Lambda(S)$
3. No other elements of Q are in $\Lambda(S)$

For NFA- Λ , $\delta^*(q, \Lambda) = \Lambda(\{q\})$.

4.6 NFA- Λ equivalency

If $L \subseteq \Sigma^*$ is a language accepted by NFA- Λ $M = (Q, \Sigma, q_0, A, \delta)$, then there is an NFA $(Q_1, \Sigma, q_1, A_1, \delta_1)$ that also accepts L .

Three equivalent statements: a language is recognizable by an { FA | NFA | NFA- Λ }

4.7 Kleene's theorem, pt. 1

Any regular language can be accepted by a finite automaton.

Proof: Must be true over union, concatenation, and Kleene*, starting with basic languages \emptyset , $\{\Lambda\}$, and $\{a\} (a \in \Sigma)$.

Construct $M_u = (Q_u, \Sigma, q_u, A_u, \delta_u)$: Create a fork. Split the two machines into two disjoint sets of states, define a q_u such that $\delta(q_u, \Lambda) = \{q_1, q_2\}$, and $\delta(q_u, a) = \emptyset$. For all other q , use the 'parent' δ .

Construct $M_c = (Q_c, \Sigma, q_c, A_c, \delta_c)$: Create a chain. $q_c = q_1$ and $A_c = A_2$. Add Λ -transitions from A_1 to q_2 .

Construct $M_k = (Q_k, \Sigma, q_k, A_k, \delta_k)$: Create a loop.

4.8 Kleene's theorem, pt. 2

The language accepted by any finite automaton is regular.

We say a path "goes through" state s if there are nonnull strings y and z , such that $x = yz$, $\delta^*(p, y) = s$, $\delta^*(s, z) = q$. Let $L(p, q, j)$ be the set of strings representing paths from p to q that go through no state numbered higher than j . $L(p, q, n) = L(p, q)$.

Induction basis: $L(p, q, 0) \subseteq \Sigma \cup \{\Lambda\}$. This is regular, since every finite language is regular.

Induction step:

$$L(p, q, k + 1) = L(p, q, k) \cup L(p, k + 1, k)L(k + 1, k + 1, k)^*L(k + 1, q, k)$$

5 Chapter 5: regular/nonregular languages

5.1 Defining Regularity

Def: Let L be any language in Σ^* . The relation I_L on Σ^* (the indistinguishability relation) is defined as follows: For any two strings $x, y \in \Sigma^*$, $x I_L y$ if, for any $z \in \Sigma^*$, either xz and yz are both in L , or xz and yz are both in L' .

I_L is an equivalence relation on Σ^* .

Thm 5.1 Let $L \subseteq \Sigma^*$, and let Q_L be the set of equivalence classes of the relation I_L on Σ^* . If Q_L is a finite set, then $M_L = (Q_L, \Sigma, q_L, A_L, \delta_L)$ is a finite automaton accepting L , where $q_L = [\Lambda]$, $A_L = \{q \in Q_L \mid q \cap L \neq \emptyset\}$, and $\delta_L : Q_L \times \Sigma \rightarrow Q_L$ is defined by the formula $\delta([x], a) = [xa]$. Furthermore, M_L has the fewest states of any FA accepting L .

Corollary 5.1 (Myhill-Nerode theorem): L is a regular language iff the set of equivalence classes of I_L is finite.

5.2 Minimal FAs

Alg 5.1: List all (unordered) pairs of states (p, q) for which $p \neq q$. Make a sequence of passes through the pairs. On the first pass, mark each pair for which exactly one element is in A . On each subsequent pass, mark any pair (r, s) if there is an $a \in \Sigma$ for which $\delta(r, a) = p$, $\delta(s, a) = q$, and (p, q) is already marked. After a pass in which no new pairs are marked, stop. The marked pairs (p, q) are precisely those for which $p \neq q$.

5.3 pumping lemma

Suppose L is a regular language. Then \exists an integer n such that $\forall x \in L$ with $|x| \geq n$, \exists strings u, v , and w such that

- $x = uvw$
- $|uv| \leq n$
- $|v| > 0$
- for any $m \geq 0$, $uv^m w \in L$

Thm 5.4 Suppose L is an infinite regular language. There are integers p and q , with $q > 0$, so that for every $m \geq 0$, L contains a string of length $p + mq$.

6 Context-Free Grammars

6.1 Definition

Def 6.1 A *context-free grammar* (CFG) is a 4-tuple $G = (V, \Sigma, S, P)$, where V and Σ are disjoint finite sets, $S \in V$, and P is a finite set of formulas of the form $A \rightarrow \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$.

The elements of V are called *variables*, or *nonterminal* symbols, and those of the alphabet Σ are called *terminal* symbols, or *terminals*. S is called the *start symbol*; and the elements of P are called *grammar rules*, or *productions*.

Def 6.2 Let $G = (V, \Sigma, S, P)$ be a CFG. The language generated by G is

$$L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}$$

A language L is a *context-free language* (CFL) if there is a CFG G so that $L = L(G)$.

6.3 Union, Concatination, and *'s of CFLs

Thm 6.1 If L_1 and L_2 are context-free languages, then the languages $L_1 \cup L_2$, $L_1 L_2$, and L_1^* are also CFLs.

A grammar $G_u = (V_u, \Sigma, S_u, P_u)$ generating $L_1 \cup L_2$. First we rename the elements of V_2 if necessary so that $V_1 \cap V_2 = \emptyset$. Let $V_u = V_1 \cup V_2 \cup \{S_u\}$, where $S_u \notin \{V_1 \cup V_2\}$. Finally, let $P_u = P_1 \cup P_2 \cup \{S_u \rightarrow S_1 \mid S_2\}$.

A grammar $G_c = (V_c, \Sigma, S_c, P_c)$ generating $L_1 L_2$. Same as above, but let $P_c = P_1 \cup P_2 \cup \{S_c \rightarrow S_1 S_2\}$.

A grammar $G^* = (V, \Sigma, S, P)$ generating L_1^* . Let $V = V_1 \cup \{S\}$, $S \notin V_1$. $P = P_1 \cup \{S \rightarrow S_1 S \mid \Lambda\}$.

6.4 Ambiguity

Def 6.3 A context-free grammar G is *ambiguous* if there is at least one string in $L(G)$ having two or more distinct derivation trees (or, equivalently, two or more distinct leftmost derivations).

6.5 Unambiguous Basic Algebra Grammar

$$\begin{aligned} S &\rightarrow S + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (S) \mid a \end{aligned}$$

6.6 Simplified and Normal Forms

Def 6.5 A *nullable* variable in a CFG $G = (V, \Sigma, S, P)$ is defined as follows.

1. Any variable A for which P contains the production $A \rightarrow \Lambda$ is nullable.

2. If P contains the production $A \rightarrow B_1B_2 \cdots B_n$ and $B_1B_2 \cdots B_n$ are nullable variables, then A is nullable.
3. No other variables in V are nullable.

Def 6.6 A CFG is in *Chomsky normal form* (CNF) if every production is of one of these two types:

$$A \rightarrow BC$$

$$A \rightarrow a$$

where A , B , and C are variables and a is a terminal symbol.

7 Pushdown Automata

7.2 definition

Def 7.1 A *pushdown automaton* (PDA) is a 7-tuple $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$, where

- Q is a finite set of states
- Σ and Γ are finite sets (input and stack alphabets)
- $q_0 \in Q$ is the initial state
- $Z_0 \in \Gamma$ is the initial stack symbol
- $A \subseteq Q$ is the set of accepting states
- $\delta : Q \times (\Sigma \cup \{\Lambda\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$.

Def 7.2 if $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$, and $x \in \Sigma^*$, x is *accepted* by M if

$$(q_0, x, Z_0) \vdash_M^* (q, \Lambda, \alpha)$$

for some $\alpha \in \Gamma^*$ and some $q \in A$.

7.3 Deterministic PDAs

Def 7.3 Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$, and $x \in \Sigma^*$. M is *deterministic* if:

1. For any $q \in Q$, $a \in \Sigma \cup \{\Lambda\}$, and $X \in \Gamma$, the set $\delta(q, a, X)$ has at most one element.
2. For any $q \in Q$ and $X \in \Gamma$, if $\delta(q, \Lambda, X) \neq \emptyset$, then $\delta(q, a, X) = \emptyset$ for every $a \in \Sigma$.

7.4 CFG \rightarrow PDA

Thm 7.2 Let $G = (V, \Sigma, S, P)$. There is a PDA M so that $L(M) = L(G)$.

We define $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ as follows:

$$Q = \{q_0, q_1, q_2\}.$$

$$\Gamma = V \cup \Sigma \cup \{Z_0\}, \text{ where } Z_0 \notin V \cup \Sigma.$$

$$A = \{q_2\}.$$

1. $\delta(q_0, \Lambda, Z_0) = \{(q_1, SZ_0)\}$.
2. For every $A \in V$, $\delta(q_1, \Lambda, A) = \{(q_1, \alpha) \mid A \rightarrow \alpha \text{ is a production in } G\}$.
3. For every $a \in \Sigma$, $\delta(q_1, a, a) = \{(q_1, \Lambda)\}$.
4. $\delta(q_1, \Lambda, Z_0) = \{(q_2, Z_0)\}$.

7.5 PDA \rightarrow CFG

Thm 7.4 Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ accepting L by empty stack. There is a CFG G so that $L(G) = L$.

Productions involving the variable $[p, A, q]$ are thought of as representing any sequence of moves that takes the PDA from state p to q and has the ultimate effect of removing A from the stack.

We define $G = (V, \Sigma, S, P)$ as follows:

$$V = \{S\} \cup \{[p, A, q] \mid A \in \Gamma, p, q \in Q\}$$

P contains precisely:

1. For every $q \in Q$, the production $S \rightarrow [q_0, Z_0, q]$ is in P .
2. For every $q, q_1 \in Q$, $a \in \Sigma \cup \{\Lambda\}$, and $A \in \Gamma$, if $\delta(q, a, A)$ contains (q_1, Λ) , then the production $[q_0, Z_0, q] \rightarrow a$ is in P .
3. For every $q, q_1 \in Q$, $a \in \Sigma \cup \{\Lambda\}$, $A \in \Gamma$, and $m \geq 1$, if $\delta(q, a, A)$ contains $(q_1, B_1 B_2 \cdots B_m)$ for some $B_1, \dots, B_m \in \Gamma$, then for every choice of $q_2, \dots, q_{m+1} \in Q$, the production

$$[q, A, q_{m+1}] \rightarrow a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_m, B_m, q_{m+1}]$$

is in P .

8 CFLs and NCFLs

8.1 Pumping Lemma redux

Lemma 8.1 For any $h \geq 1$ a binary tree having more than 2^{h-1} leaf nodes must have height greater than h .

Thm 8.1a Let L be a CFL. Then there is an integer n so that for any $u \in L$ satisfying $|u| \geq n$, there are strings $v, w, x, y,$ and z satisfying

1. $u = vwx yz$
2. $|wy| > 0$
3. $|wxy| \leq n$
4. for any $m \geq 0, vw^mxy^mz \in L$

Proof: Let p equal the total number of variables in CNF-form CFG G . Let n equal 2^{p+1} . Lemma 8.1 shows that any derivation tree for u must have height at least $p + 2$ (it has more than 2^p leaf nodes, and therefore its height is greater than $p+1$). Consider a path of maximum length and look at the bottom portion, consisting of a leaf node and the $p+1$ nodes above it. Each of these $p + 1$ nodes corresponds to a variable, and because there are only p distinct variables, some variable A must appear twice in this portion of the path. Let x be the portion of u derived from the A closest to the leaf, and let $t = wxy$ be the portion of u derived from the other A . If v and z represent the beginning and ending portions of u , we have $u = vwx yz$.

The A closest to the root in this portion of the path can be considered as the root of a binary derivation tree for wxy . Because we began with a path of maximum length, this tree has height less than or equal to $p + 2$; therefore, by the lemma, $|wxy| \leq 2^{p+1}$. The node containing this A has two children, both corresponding to variables. If we let B denote the one that is not an ancestor of the other A , then since x is derived from that other A , the string of terminals derived from B does not overlap x . It follows that either w or y is nonnull, and therefore $|wy| > 0$.

Finally,

$$S \Rightarrow^* vAz \Rightarrow^* vwAyz \Rightarrow^* vwx yz$$

(the first A is the one in the higher node, the second the one in the lower), and therefore the third conclusion of the theorem follows from our preliminary discussion.

8.2 Ogden's lemma

Suppose L is a context-free language. Then there is an integer n so that if u is any string in L of length n or greater, and any n or more positions of u are designated as "distinguished," then there are strings $v, w, x, y,$ and z satisfying

1. $u = vwxyz$
2. wy contains at least one distinguished position
3. wxy contains no more than n distinguished positions
4. x contains at least one distinguished position
5. for any $m \geq 0$, $vw^mxy^mz \in L$

8.3 Intersection and Complement of CFLs

CFLs L_1 and L_2 with $L_1 \cap L_2$ not a CFL

$$L_1 = \{a^i b^j c^k \mid i < j\}$$

$$S \rightarrow ABC$$

$$A \rightarrow aAb \mid \Lambda$$

$$B \rightarrow bB \mid b$$

$$C \rightarrow cC \mid \Lambda$$

$$L_2 = \{a^i b^j c^k \mid i < k\}$$

$$S \rightarrow AC$$

$$A \rightarrow aAc \mid B$$

$$B \rightarrow bB \mid \Lambda$$

$$C \rightarrow cC \mid c$$

$$L_1 \cap L_2 = L_1 = \{a^i b^j c^k \mid i < j \text{ and } i < k\}$$

Thm 8.4 If L_1 is a CFL and L_2 is a regular language, then $L_1 \cap L_2$ is a CFL.

Let $M_1 = (Q_1, \Sigma, \Gamma, q_1, Z_0, A_1, \delta_1)$ be a PDA accepting L_1 and $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$ be an FA recognizing L_2 . Define PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ as follows:
 $Q = Q_1 \times Q_2$, $q_0 = (q_1, q_2)$, $A = A_1 \times A_2$
for $p \in Q_1$, $q \in Q_2$, and $Z \in \Gamma$,

$$\delta((p, q), a, Z) = \{(p', q'), \alpha \mid (p', \alpha) \in \delta_1(p, a, Z) \text{ and } \delta_2(q, a) = q'\}$$

for every $a \in \Sigma$; and

$$\delta((p, q), \Lambda, Z) = \{(p', q), \alpha \mid (p', \alpha) \in \delta_1(p, \Lambda, Z)\}$$

9 Turing Machines

9.2 definition

A *Turing machine* (TM) is a 5-tuple $T = (Q, \Sigma, \Gamma, q_0, \delta)$, where

Q is a finite set of states, assumed not to contain h , the *halt* state (the same symbol will be used for the halt state of every TM).

Σ and Γ are finite sets, the *input* and *tape* alphabets, respectively, with $\Sigma \subseteq \Gamma$; Γ is assumed not to contain Δ , the *blank* symbol.

q_0 , the initial state, is an element of Q .

$\delta : Q \times (\Gamma \cup \{\Delta\}) \rightarrow (Q \cup \{h\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$ is a partial function (that is, possibly undefined at certain points).

Def A *configuration* of $T = (Q, \Sigma, \Gamma, q_0, \delta)$ is a pair $(q, x\underline{a}y)$, where q is a state, x and y are strings over $\Gamma \cup \{\Delta\}$, $a \in \Gamma \cup \{\Delta\}$, and the underlined symbol represents the current position of the tape head.

Def A string $x \in \Sigma^*$ is *accepted* by TM $T = (Q, \Sigma, \Gamma, q_0, \delta)$ if there exist $y, z \in (\Gamma \cup \{\Delta\})^*$, and $a \in \Gamma \cup \{\Delta\}$, so that

$$(q_0, \underline{\Delta}x) \vdash_T^* (h, y\underline{a}z)$$

The *language accepted by* T is the set $L(T)$ of input strings on which T halts.