

CS20a: Computation, Computers, Programs



- Instructor: Jason Hickey
 - Email: jyh@cs.caltech.edu
 - Office hours: TR 10-11am
- TAs:
 - Nathan Gray (n8gray@cs.caltech.edu)
 - Brian Aydemir (emre@cs.caltech.edu)
 - Jason Frantz (frantz@its.caltech.edu)
 - Robert Li (robertli@its.caltech.edu)



Outline

- Alphabets and strings
- DFAs




Decision problems

- A *decision problem* is a set A , together with a subset $B \subseteq A$ of problem instances that are true.
- We could consider problems that have an output (for example, "compute the shortest path in a graph"), but the decision problems are simpler and already have the kind of behavior we want to study.



Alphabets

- An *alphabet* Σ is a *finite* set. The specific alphabet does not matter much, and we'll use different alphabets at different times.
 - $\{a, b\}$
 - ASCII
 - $\{\}$
 - $\{0, 1, 2, 3, 4, 5, 6, 7, 9, 8\}$



Computation, Computers, and Programs
<http://www.cs.caltech.edu/~cs20/a>

Course Introduction
 October 3, 2002


4

Strings

- A *string* over Σ is any finite sequence of symbols from Σ .

Σ	a string over Σ
$\{a, b\}$	<i>abbabbbaaaba</i>
ASCII	<i>Hello world</i>
$\{\}$	ϵ
$\{0, 2, 1, 6, 3, 4, 7, 9, 5, 8\}$	31415926

- The ϵ string is the string containing no symbols; it is a string in any alphabet.




Computation, Computers, and Programs
<http://www.cs.caltech.edu/~cs20/a>

Course Introduction
 October 3, 2002

5

Notation

- The length of a string s is $|s|$
 - $|31415926| = 8$
 - $|\epsilon| = 0$
- The notation a^n means the concatenation of n a symbols
 - $a^7 = aaaaaaa$
 - $a^0 = \epsilon$
- The set of strings over an alphabet Σ is Σ^*
 - $\{a\}^* = \{\epsilon, a, aa, aaa, \dots\}$
 - $\{\}$ $^* = \{\epsilon\}$



Computation, Computers, and Programs
<http://www.cs.caltech.edu/~cs20/a>

Course Introduction
 October 3, 2002

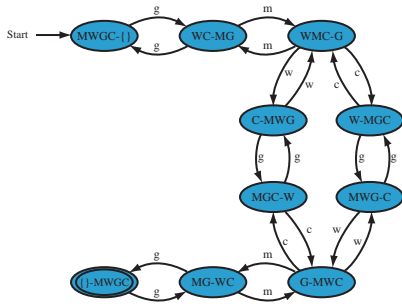
6

Grouping

- We will often use parentheses indiscriminately:
 - $(ab)^4 = abababab$
 - $((ab)^2c)^2 = ababcababc$
- We'll try to avoid this:
 - $\Sigma = \{ \}, \{ \}$
 - $()()^2 =)()()$



Wolves, Goats, and cabbages



Deterministic Finite Automata

- A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
 - Q is a *finite* set of *states*
 - Σ is an *alphabet*
 - $\delta : Q \times \Sigma \rightarrow Q$ is a *transition function*
 - $q_0 \in Q$ is the *initial state*
 - $F \subseteq Q$ is a set of *final* or *accepting* states



Definitions and conventions

- The lowercase letters a, b, c represent symbols,
- The lowercase letters x, y, z, w represent strings.
- A string x is *accepted* by an automaton $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) \in F$.

- The *language* $L(M)$ is defined as

$$L(M) = \{x \mid \delta(q_0, x) \in F\}$$

- A language is *regular* iff it is accepted by some finite automaton.



Some properties of regular languages

- Suppose two languages L_1 and L_2 are regular.
 - Is $L_1 \cup L_2$ regular?
 - Is $L_1 \cap L_2$ regular?
 - Is $\Sigma^* - L_1$ regular?



Complement is regular

- If L is regular, there is some machine M that accepts it.
- $M = (Q, \Sigma, \delta, q_0, F)$
- To build $\Sigma^* - L$, construct the following machine:

$$M' = (Q, \Sigma, \delta, q_0, Q - F)$$



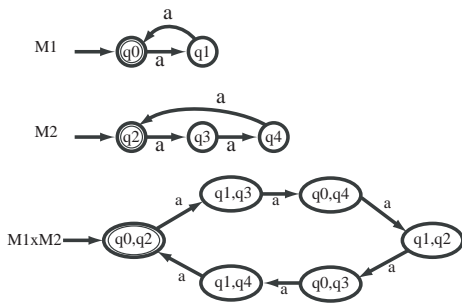
Intersection is regular

- If L_1 and L_2 are regular, there are FA s.t. $L_1 = L(M_1)$ and $L_2 = L(M_2)$.
- Build a product machine:

$$\begin{array}{l}
 M_1 \cap M_2 = (Q, \Sigma, \delta, s, F) \\
 \hline
 Q = Q_1 \times Q_2 \\
 \Sigma = \Sigma_1 = \Sigma_2 \\
 \delta((q_1, q_2), c) = (\delta_1(q_1, c), \delta_2(q_2, c)) \\
 s = (s_1, s_2) \\
 F = F_1 \times F_2
 \end{array}$$



Products



Verifying the product machine

Theorem For any string x ,

$$\delta((s_1, s_2), x) \in F \Leftrightarrow \delta_1(s_1, x) \in F_1 \wedge \delta_2(s_2, x) \in F_2$$



Equational proofs

An *equational* proof is a sequence of (justified) equalities. If

$$\begin{aligned}
 & e_1 \\
 = & \text{(some reason)} \\
 & e_2 \\
 = & \text{(some other reason)} \\
 & \vdots \\
 = & \text{(some final reason)} \\
 & e_n
 \end{aligned}$$

Then, $e_1 = e_n$.



Main lemma

Lemma For any string x

$$\delta((s_1, s_2), x) = (\delta_1(s_1, x), \delta_2(s_2, x))$$

Proof By induction on the length of x .

Base case If $x = \epsilon$, then

$$\delta((s_1, s_2), \epsilon) = (s_1, s_2) = (\delta_1(s_1, \epsilon), \delta_2(s_2, \epsilon))$$



Induction step

Induction step

Assume $\delta((s_1, s_2), x) = (\delta_1(s_1, x), \delta_2(s_2, x))$.

Consider a symbol a .

Prove that $\delta((s_1, s_2), xa) = (\delta_1(s_1, xa), \delta_2(s_2, xa))$.

$$\begin{aligned}
 & \widehat{\delta}((s_1, s_2), xa) \\
 = & \text{(by definition)} \\
 & \widehat{\delta}(\widehat{\delta}((s_1, s_2), x), a) \\
 = & \text{(induction hypothesis)} \\
 & \widehat{\delta}((\widehat{\delta}_1(s_1, x), \widehat{\delta}_2(s_2, x)), a) \\
 = & \text{(definition of } \widehat{\delta}) \\
 & (\widehat{\delta}_1(\widehat{\delta}_1(s_1, x), a), \widehat{\delta}_2(\widehat{\delta}_2(s_2, x), a)) \\
 = & \text{(definition of } \widehat{\delta}) \\
 & (\widehat{\delta}_1(s_1, xa), \widehat{\delta}_2(s_2, xa))
 \end{aligned}$$



Product theorem

Theorem For any string x ,

$$\delta((s_1, s_2), x) \in F \Leftrightarrow \delta_1(s_1, x) \in F_1 \wedge \delta_2(s_2, x) \in F_2$$

Proof

$$\begin{aligned} & \delta((s_1, s_2), x) \in F \\ \Leftrightarrow & \text{(definition of } F) \\ & \delta((s_1, s_2), x) \in F_1 \times F_2 \\ \Leftrightarrow & \text{(Lemma)} \\ & (\delta_1(s_1, x), \delta_2(s_2, x)) \in F_1 \times F_2 \\ \Leftrightarrow & \text{(Definition of } \times) \\ & \delta_1(s_1, x) \in F_1 \wedge \delta_2(s_2, x) \in F_2 \end{aligned}$$



Regular expressions

- The languages accepted by FA are easily expressed using the language of *regular expressions*.
- You have used them before:
 - Shell: `ls -l *.c`
 - Sed/vi: `s/aabb*cc*/abc/g`
 - Shell: `cd ~cs20; rm -rf *`



Definition of regular expressions

- Let Σ be an alphabet. The regular expressions are defined inductively as follows:
 - \emptyset is a regular expression denoting $\{\}$,
 - ϵ is a regular expression denoting $\{\epsilon\}$,
 - for each $a \in \Sigma$, a is a regular expression denoting $\{a\}$,
 - Assume r and s are regular expressions denoting sets R and S , then
 - * rs denotes RS ,
 - * $r + s$ denoted $R \cup S$,
 - * r^* denotes R^*



Examples of regular expressions

- abc denotes $\{abc\}$,
- a^*b denotes $\{b, ab, aab, aaab, \dots\}$,
- $(a + b)^*aa(a + b)^*$: all strings containing at least two consecutive a 's.
- $(0^*(101^*01)^*)^*$: all binary numbers that are a multiple of 3.

