A3: up tomorrow
T2Q2: focus of tutorial

CSC236 fall 2016

languages: definitions and proofs

Danny Heap
heap@cs.toronto.edu / BA4270 (behind elevators)
http://www.cdf.toronto.edu/~csc236h/fall/
416-978-5899

Using Introduction to the Theory of Computation, Chapter 7

Outline

FSAs formally

formal languages

notes

build an automaton with formalities...

quintuple: $(Q, \Sigma, q_0, F, \delta)$ $\Sigma = \{0, 1\}$, e.g

Q is set of states, Σ is finite, non-empty alphabet, q_0 is start state

F is set of accepting states, and $\delta : Q \times \Sigma \mapsto Q$ is transition function

We can extend $\delta : Q \times \Sigma \mapsto Q$ to a transition function that tells us what state a **string** s takes the automaton to:

extended transition function

"" (empty string)

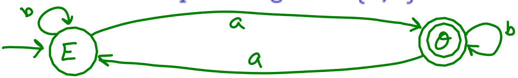
$$\delta^* : Q \times \Sigma^* \mapsto Q \qquad \delta^*(q, s) = \begin{cases} q & \text{if } s = \varepsilon \\ \delta(\delta^*(q, s'), x) & \text{if } s' \in \Sigma^*, \\ & \quad x \in \Sigma, s = s'x \end{cases}$$

all strings over

String s is accepted **if and only if** $\delta^*(q_0, s) \in F$, it is rejected otherwise.

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as



Formal proof requires inductive proof of state invariant:

* Only prove "only if" at each step. "if" follows because we exhaust all possibilities

P(s):
$$\delta^*(E, s) = \begin{cases} E & \text{only if } s \text{ has even number of } a\text{s} \\ O & \text{only if } s \text{ has odd number of } a\text{s} \end{cases}$$

<u>Proof – structural induction:</u>

def Σ^*

1. $\varepsilon \in \Sigma^*$
2. $x \in \Sigma^* \Rightarrow xa, xb \in \Sigma^*$

<u>Basis</u>

$$\delta^*(E, \varepsilon) = \begin{cases} E \Rightarrow \varepsilon \text{ has even \# of } a\text{s} & \text{(true antecedent and consequent)} \\ O \Rightarrow \varepsilon \text{ has odd \# of } a\text{s} & \\ \text{False} \Rightarrow \text{anything} & \text{(vacuous truth)} \end{cases}$$

<u>Basis holds</u>

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as

Formal proof requires inductive proof of state invariant:

$P(s)$:

$$\delta^*(E, s) = \begin{cases} E & \text{only if } s \text{ has even number of } a\text{s} \\ O & \text{only if } s \text{ has odd number of } a\text{s} \end{cases}$$

Induction step: Let $s' \in \Sigma^*$ and assume $P(s')$. Must show $P(s)$ when $s = s'a$ or $s = s'b$.

Case $s = s'a$

$$\delta^*(E, s) = \delta^*(E, s'a) = \delta(\delta^*(E, s'), a) = \begin{cases} \delta(E, a) & \Rightarrow s' \text{ has even } \#a\text{s} \text{ By } P(s') \\ \delta(O, a) & \Rightarrow s' \text{ has odd } \#a\text{s} \text{ By } P(s') \end{cases}$$

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as

To show converse, notice that contrapositive of:

$\delta^*(E,s) = E \Rightarrow s$ has even # as is s has odd # as

$\Rightarrow \neg(\delta^*(E,s) = E)$

* next page

so we have if and only if

Formal proof requires inductive proof of state invariant:

$P(s)$:

$$\delta^*(E, s) = \begin{cases} E & \text{only if } s \text{ has even number of } as \\ O & \text{only if } s \text{ has odd number of } as \end{cases}$$

$= \begin{cases} O \Rightarrow s'a \text{ has odd \# } a\text{s (1 more } a) \\ \\ E \Rightarrow s'a \text{ has even \# } a\text{s (1 more } a) \end{cases}$

exercise Case $s = s'b$

Result we prove $\delta^*(E,s) = O \Rightarrow s$ has odd number of as

example — an odd machine

devise a machine that accepts strings over $\{a, b\}$ with an odd number of as

Formal proof requires inductive proof of state invariant:

$$\delta^*(E, s) = \begin{cases} E & \text{only if } s \text{ has even number of } a\text{s} \\ O & \text{only if } s \text{ has odd number of } a\text{s} \end{cases}$$

we've shown $\delta^*(E,S) = O$ <u>only if</u> s has odd # as. To show <u>if</u> direction:

if s has odd # as, by contrapositive

$\neg$(s has even # as) $\Rightarrow \neg(\delta^*(E,S) = E)$

$\Rightarrow \delta^*(E,S) = O$ ✓

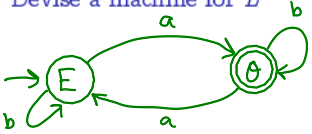
more odd/even: intersection

intersection

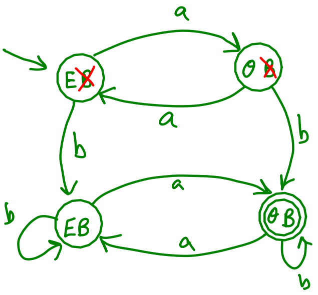
L is the language of binary strings with an odd number of as, and at least one b

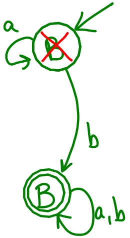
Devise a machine for L

Product of 2 machines — like running both machines



Product of odd as, at least one B





Each state, transition in product machine represents a pair of states/transitions

I chose meaningful labels, eg E - even O - odd

B - at least one b, ~~B~~

more odd/even: union

L is the language of binary strings with an odd number of as, or at least one b

Devise a machine that accepts L,

— same states + transitions <u>but</u> accepting states are <u>now</u> EB, OB, O~~E~~

<u>Exercise</u> devise this machine

<u>Note</u> transitions may be indicated by a table, e.g. odd # as:

	E	O
a	O	E
b	E	O

some definitions

bounds resource for machine

alphabet: finite, non-empty set of symbols, e.g. $\{a, b\}$ or $\{0, 1, -1\}$. Conventionally denoted Σ.

length

string: finite (including empty) sequence of symbols over an alphabet: abba is a string over $\{a, b\}$. Convention: ε is the empty string, never an allowed symbol, Σ^* is set of all strings over Σ.

language: Subset of Σ^* for some alphabet Σ. Possibly empty, possibly infinite subset. E.g. $\{\}$, $\{aa, aaa, aaaa, \dots\}$.

empty language

N.B.: $\{\} \neq \{\varepsilon\}$. $|\{\}| = 0 \neq 1 = |\{\varepsilon\}|$

Many problems can be reduced to languages: logical formulas, identifiers for compilation, natural language processing. Key question is recognition:

Given language L and string s, is $s \in L$?

- Is s accepted by the relevant FSA?
- Is s denoted by the relevant regex?

Languages may be described either by descriptive generators (for example, regular expressions) or procedurally (e.g. finite state automata)

more notation

string length: denoted $|s|$, is the number of symbols in s, e.g. $|bba| = 3$.

$|\varepsilon| = 0$

in Python $s == t$

$s = t$: if and only if $|s| = |t|$, and $s_i = t_i$ for $1 \leq i \leq |s|$.

s^R: reversal of s is obtained by reversing symbols of s, e.g. $1011^R = 1101$.

mostly use

st or $s \circ t$: contcatenation of s and t — all characters of s followed by all those of t, e.g. $bba \circ bb = bbabb$.

s^k: denotes s concatenated with itself k times. E.g., $ab^3 = ababab$, $101^0 = \varepsilon$.

Σ^n: all strings of length n over Σ, Σ^* denotes all strings over Σ.

$\{0,1\}^2 = \{00, 11, 10, 01\}$

language operations

$\overline{L}$: Complement of L, i.e. $\Sigma^* - L$. If L is language of strings over $\{0, 1\}$ that start with 0, then $\overline{L}$ is the language of strings that begin with 1 plus the empty string.

$L \cup L'$: union $= L' \cup L$

$L \cap L'$: intersection $= L' \cap L$

$L - L'$: difference $\neq L' - L$

$\mathrm{Rev}(L)$: $= \{s^R : s \in L\}$

$\neq L'L$

concatenation: LL' or $L \cdot L' = \{rt | r \in L, t \in L'\}$. Special cases $L\{\varepsilon\} = L = \{\varepsilon\}L$, and $L\{\} = \{\} = \{\}L$.

more language operations

exponentiation: L^k is concatenation of L k times. Special case, $L^0 = \{\varepsilon\}$, including $L = \{\}$ (!)

$\{\}^0 = \{\varepsilon\}$ — very strange!

Kleene star: $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$

notes

notes