Inductions and Strings

Lecture Outline

Mathematical background for the “Theory of Computing”

- Induction
- Strings
- An Example

Axioms for the Natural Numbers

Axiom 0: 0 is a natural number.

Axiom 1: if x is a natural number, so is $succ(x)$

Axiom 2: if x is a natural number, $succ(x) > x$.

Axiom 3: if x and y are natural numbers and $x > y$, then $succ(x) > y$.

Axiom 4: if x and y are natural numbers and $x > y$, then $x \neq y$.

We write $\mathbb{N}$ to denote the set of natural numbers.

Operations on the Natural Numbers

- Addition:

$$\begin{aligned} x + 0 &= x, \\ x + succ(y) &= succ(x + y). \end{aligned}$$

- Multiplication:

$$\begin{aligned} x * 0 &= 0, \\ x * succ(y) &= (x * y) + x. \end{aligned}$$

Assume
x=5 , y=5

Two More Operations

- Division:

$$(x/y) = q \quad \Leftrightarrow \quad y * q = x.$$

- Exponentiation:

$$\begin{aligned} x^0 &= succ(0), \\ x^{succ(y)} &= (x^y) * x. \end{aligned}$$

Abbreviations

- Decimal digits:

$$1 = succ(0), \quad 2 = succ(1), \quad 3 = succ(2), \quad 4 = succ(3),$$
$$5 = succ(4), \quad 6 = succ(5), \quad 7 = succ(6), \quad 8 = succ(7),$$
$$9 = succ(8), \quad 10 = succ(9).$$

- Multidigit numbers:

$$1437 = 1*10^3 + 4*10^2 + 3*10^1 + 7*10^0$$
$$= \underbrace{succ(succ(succ(\ldots(succ(}_{1437 \text{ ``succ(''s}}0\underbrace{))\ldots)))}_{1437 \text{ ``)''s}}$$

0 is the primitive element for the naturals.

Strings

Let Σ be a finite set of "symbols".

- Informal definition: a string is a sequence of zero or more elements from Σ.
- Inductive definition: $s \in \Sigma^*$ iff
  - $s = \epsilon$, the empty string.
  - There is a $w \in \Sigma^*$ and a $c \in \Sigma$ such that $s = w \cdot c$.
- Note: The operator $\cdot$ represents concatenation, and we often omit writing it, just like skipping the $*$ for multiplication.

Tuple-Terror

In this class, we will often get definitions along the lines of:

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

1. Q is a finite set called the states.
2. $\ldots$

(From *Sipser*, Def. 1.5, p. 35)

"Tuples" are the mathematicians way of describing things that resemble what programmers call "data structures."

Regular Languages

Regular Languages

- Definition of regular languages
  - Regular languages are recognized by finite automata
  - Examples
- Closure properties

Languages (review)

A language is a set of strings.

- Let Σ be a finite set, called an alphabet.
- Σ^* is the set of all strings of Σ, i.e. sequences of zero or more symbols from Σ.
- A language is a subset of Σ^*. Examples:
  - Example, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, and L_1 is the set of all strings that of length at most two:

$$L_1 \quad = \quad \{\epsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\}$$

  - With Σ as above, let L_2 be the set of all strings where every $\mathtt{a}$ is followed immediately by a $\mathtt{b}$:

$$L_2 \quad = \quad \{\epsilon, \mathtt{b}, \mathtt{ab}, \mathtt{bb}, \mathtt{abb}, \mathtt{bab}, \mathtt{bbb}, \ldots\}$$

  - With Σ as above, let L_3 be the set of all strings that have more $\mathtt{a}$'s than $\mathtt{b}$'s:

$$L_3 \quad = \quad \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aab}, \mathtt{aba}, \mathtt{aab}, \ldots\}$$

Deterministic Finite Automata (review)

- A deterministic finite automaton (DFA) is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$ where:

  Q is a finite set of states.

  Σ is a finite set of symbols.

  $\delta : Q \times \Sigma \rightarrow Q$ is the next state function.

  q_0 is the initial state.

  F is the set of accepting states.

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

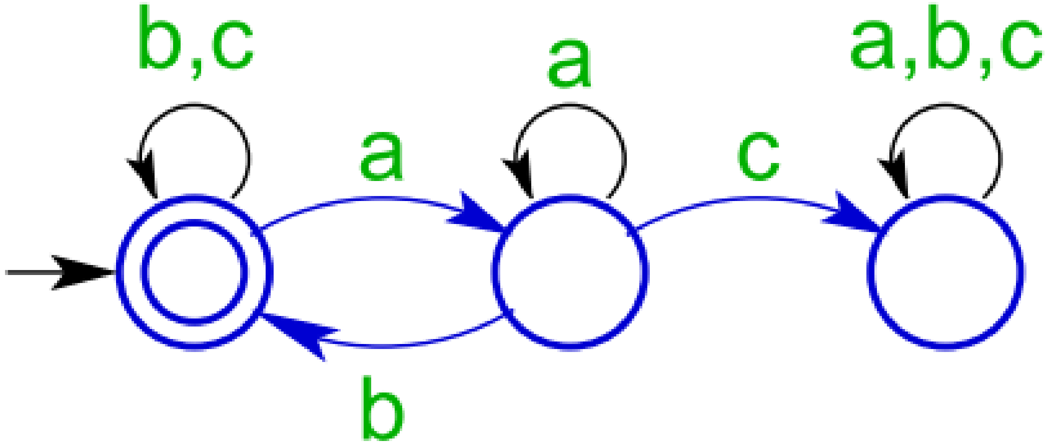
  For $s \in \Sigma^*$,

$$
\begin{aligned}
\delta(q, s) &= q, & \text{if } s = \epsilon \\
&= \delta(\delta(q, x), c), & \text{if } s = x \cdot c \text{ for } c \in \Sigma
\end{aligned}
$$

  The language accepted by M is

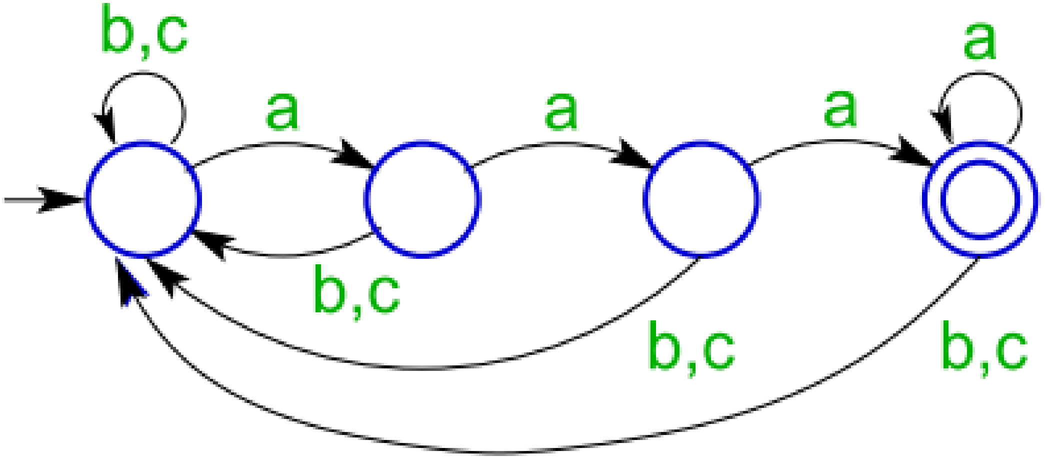
$$
L(M) = \{ s \in \Sigma^* \mid \delta(q_0, s) \in F \}
$$

DFA examples



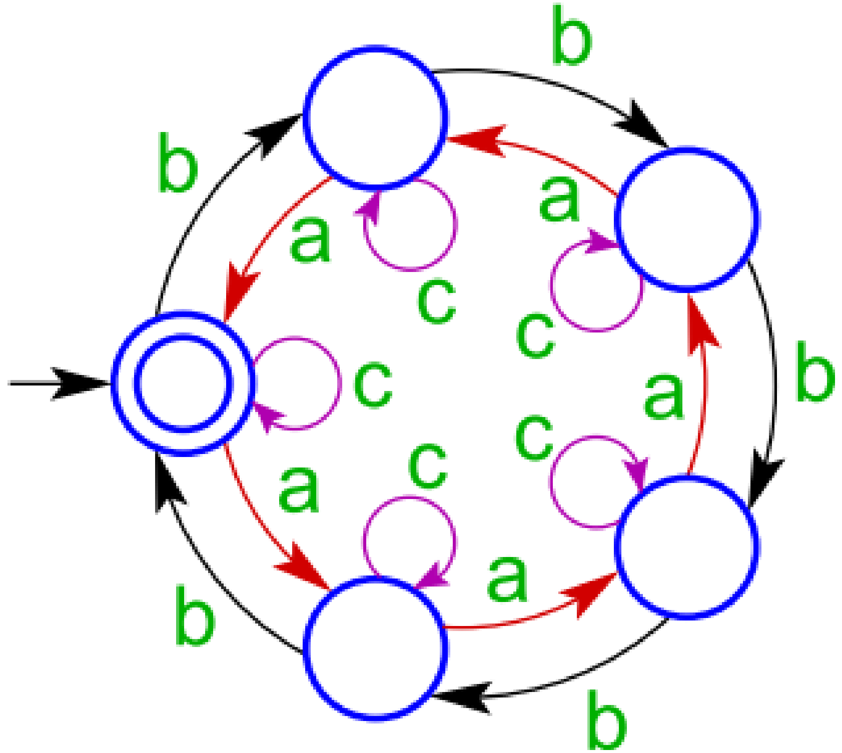
$$L(M_1) = \left\{ s \in \Sigma^* \;\middle|\; \text{Every a in } s \text{ is followed by a b without an intervening c.} \right\}$$

DFA examples



$$L(M_2) \quad = \quad \{s \in \Sigma^* \mid s \text{ ends with three consecutive a's.}\}$$

DFA examples



$$L(M_3) = \left\{ s \in \Sigma^* \;\middle|\; \text{the difference between the number of a's in } s \text{ and the number of b's is a multiple of 5.} \right\}$$

Regular Languages (Definition)

A language, B, is a regular language iff there is some DFA M such that $L(M) = B$.

In other words, the regular languages are the languages that are recognized by DFAs.

- To show that a language is regular, we can construct a DFA that recognizes is.
- Conversely, we can show that a language is not regular by proving that there can be no DFA that accepts it.

Regular Languages (Properties)

The regular languages are closed under:

Complement: If B is a regular language, then so is $\overline{B}$.

- A string is in $\overline{B}$ iff it is not in B.

Intersection: If B_1 and B_2 are regular languages, then so is $B_1 \cap B_2$.

- A string is in $B_1 \cap B_2$ iff it is in both B_1 and B_2.
- Because we have complement and intersection, we can conclude that the union, difference, symmetric difference, etc. of regular langauges is regular.

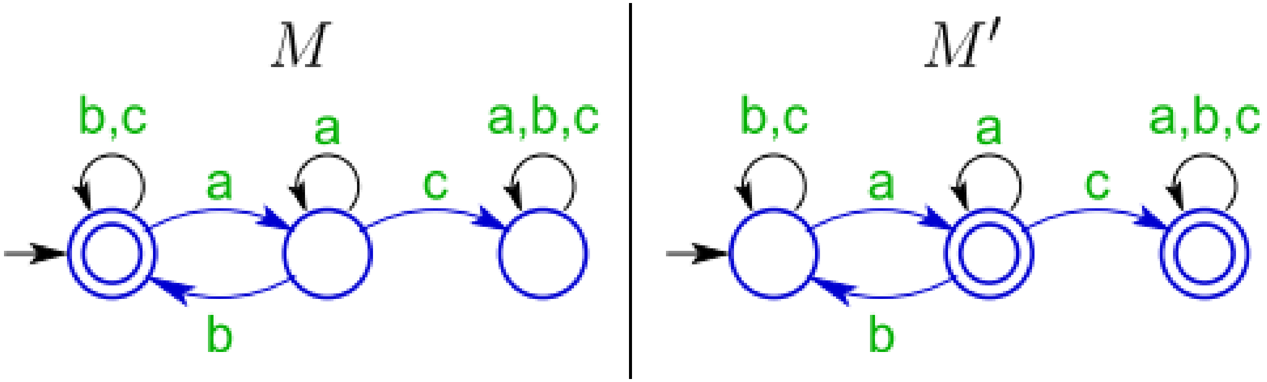
Concatenation: If B_1 and B_2 are regular languages, then so is $B_1 \cdot B_2$.

- A string, s, is in $B_1 \cdot B_2$ iff there are strings x and y such that $x \in B_1$, $y \in B_2$, and $s = x \cdot y$. Note that x and/or y may be ϵ.

Kleene star: B is a regular language, then so is B^*.

- A string, s, is in B^* iff $s = \epsilon$ or there are strings x and y such that $x \in B^*$, $y \in B$, and $s = x \cdot y$.
- Note that even if $B = \emptyset$, $\epsilon \in B^*$. Thus, for any language B, $B^* \neq \emptyset$.

Complement example



$$L(M') = \left\{ s \in \Sigma^* \;\middle|\; \begin{array}{l} s \text{ ends with an a or has an a followed} \\ \text{immediately by a } c. \end{array} \right\}$$

Closure under Complement

Let $B \subseteq \Sigma^*$ be a regular language.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes B.

Let $M' = (Q, \Sigma, \delta, q_0, \overline{F})$. M' recognizes $\overline{B}$.

Proof: let $s \in \Sigma^*$ be a string.

- If $s \in B$, then $\delta(q_0, s) \in F$.
  Thus, $\delta(q_0, s) \notin \overline{F}$.
  Thus $s \notin L(M')$.
- If $s \notin B$, then $\delta(q_0, s) \notin F$.
  Thus, $\delta(q_0, s) \in \overline{F}$.
  Thus $s \in L(M')$.

$\overline{B}$ is recognized by a DFA;
therefore, $\overline{B}$ is regular.

□

Closure under Intersection

- Let $B_1, B_2 \subseteq \Sigma^*$ be regular languages.
- Let $M_1 = (Q_1, \Sigma, \delta_1, q_{1,0}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{2,0}, F_2)$ be DFAs that recognize B_1 and B_2 respectively.
- Let $M^\cap = (Q_1 \times Q_2, \Sigma, \delta, q_0, F_1 \times F_2)$ where

$$\begin{aligned} q_0 &= (q_{1,0}, q_{2,0}) \\ \delta((q_1, q_2), c) &= (\delta_1(q_1, c), \delta_2(q_2, c)) \end{aligned}$$

for any $q_1 \in Q_1$, $q_2 \in Q_2$ and $c \in \Sigma$.
$M^\cap$ recognizes $B^1 \cap B^2$.