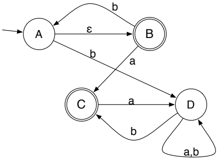
Introduction to Computability: Languages and Strings

October 29, 2012

CS 81: Computability and Logic

Selected Topics in Computability

Finite State Machines



Regular Expressions

a(a|b)*a

Context-Free Grammars

E → n
E → E + n

Turing Machines

Partial Recursive Functions

h(μx.[f(g(x))=0])

Lambda Calculus & Combinatory Logic (CS 131)

λf.λb.f(f(b))

S(SII)(SII)KS

If these are the answers, what is the question?

Describing the Problems to be Solved

Assumption 1: Inputs are *finite strings from some finite alphabet of characters.*

- ✓ `"37"`
- ✓ `"3*7=21"`
- ✓ `"sort([3,1,4,1,2], [1,2,3,4])."`

Assumption 2: We care about *decision problems* (i.e., answer is just yes or no)

- ✓ Is 37 prime?
- ✓ Does $3 \times 7 = 21$?
- ✓ Is $[1, 2, 3, 4]$ the result of sorting $[3, 1, 4, 1, 2]$?
- ✓ $\vdots$

How can we justify making these assumptions?

Justifying the Assumptions

Assumption 1: Inputs are *finite strings from a finite alphabet of characters*.

- ✓ Generalizes "everything's just a string of bits"

Assumption 2: We care about *decision problems* (i.e., answer is just yes or no)

- ✓ We can solve other problems by asking enough questions.
- ✓ E.g., how could I sort $[3, 1, 4, 1, 2]$ by asking yes/no questions?

Combining the Assumptions

Any *decision problem* is *equivalent* to asking

"Is the input a member of the set L?"

for a suitable set L.

- ✓ For primality testing, $L = \{2, 3, 5, 7, 11, 13, 17, 19, \ldots\}$.
- ✓ For multiplication-correctness,

$$L = \{ \begin{array}{lll} 0{\times}0{=}0, & 0{\times}1{=}0, & \ldots \\ 1{\times}0{=}1, & 1{\times}1{=}1, & \ldots \\ 2{\times}0{=}1, & 2{\times}1{=}2, & \ldots \end{array} \}$$

Conclusion

A one-to-one correspondance:

(computational) problems to solve

$\updownarrow$

sets of finite strings.

These sets are called “formal languages”

Alphabet

An *alphabet* Σ is a nonempty set.
The elements of Σ are called *letters* or *symbols*.

- ✓ We will always assume a finite alphabet.
- ✓ Examples?

Strings

A *string over* Σ is a sequence

$$c_1 c_2 \cdots c_n$$

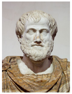
where $n \geqslant 0$ and each $c_i \in \Sigma$.

- ✓ Every string is *finite*
- ✓ We will write strings without quotation marks

  xyzzy

- ✓ We write the empty string as ε

  Note: $\varepsilon \notin \Sigma$!



What about my alphabet?

Languages

A *language* L over Σ is a set of strings over Σ.

- ✓ Σ^* (the set of all strings over Σ) is a language.
- ✓ The empty set $\emptyset$ is a language.
- ✓ The singleton set ε is a different language.
- ✓ Languages may be finite or infinite, but they contain only finite strings!
- ✓ Other examples?

Strings (more formally)

The set Σ^* can be defined inductively:

- ✓ $\varepsilon \in \Sigma^*$
- ✓ If $a \in \Sigma$ and $x \in \Sigma^*$ then $a \bullet x \in \Sigma^*$

The set Σ^* can be defined inductively:

- ✓ $\varepsilon \in \Sigma^*$
- ✓ If $a \in \Sigma$ and $x \in \Sigma^*$ then $ax \in \Sigma^*$

Resulting Induction Principles

- ✓ **Structural induction on strings:** If $P(\varepsilon)$ and

$$\forall x \in \Sigma^*. \forall a \in \Sigma. \quad P(x) \to P(ax)$$

then $\forall w \in \Sigma^*. P(w)$.

- ✓ **Induction by length:** If $P(\varepsilon)$ and

$$\begin{aligned} \forall n > 0. \quad & (\forall w \in \Sigma^*.\mathtt{length}(w) = n{-}1 \to P(w)) \\ & \to (\forall w \in \Sigma^*.\mathtt{length}(w) = n \to P(w)) \end{aligned}$$

then $\forall w \in \Sigma^*. P(w)$.**Strong induction by length:** If

$$\begin{aligned} \forall n \geqslant 0. \quad & (\forall w \in \Sigma^*.\ \mathtt{length}(w) < n \to P(w)) \\ & \to (\forall w \in \Sigma^*.\ \mathtt{length}(w) = n \to P(w)) \end{aligned}$$

then $\forall w \in \Sigma^*. P(w)$.

The inductive definition of strings also justifies defining *functions* by induction/recursion.

Operations on Strings

```
append(ε,y)   := y                       εy     := y
append(ax,y) := a•append(x,y)            (ax)y  := a(xy)

length(ε)   := 0                         |ε|   := 0
length(ax) := 1 + length(x)              |ax|  := 1 + |x|

rev(ε)   := ε                            ε^R     := ε
rev(ax) := append(rev(x),a)              (ax)^R  := x^R a
```

Prove

- ✓ $\forall w \in \Sigma^*. (w\varepsilon = w)$
- ✓ $\forall w \in \Sigma^*. \forall y, z \in \Sigma^*. (wy)z = w(yz)$
- ✓ $\forall w \in \Sigma^*. \forall z \in \Sigma^*. \texttt{length}(wz) = \texttt{length}(w) + \texttt{length}(z)$
- ✓ $\forall w \in \Sigma^*. \forall z \in \Sigma^*. (wz)^R = z^R w^R$

Operations on Languages

$$L \cup M$$

$$L \cap M$$

$$L \setminus M$$

$$LM := \{\, xy \mid x \in L,\ y \in M \,\}$$

$$L^n \qquad L^0 := \{\varepsilon\}$$
$$L^{n+1} := LL^n$$

$$L^* := L^0 \cup L^1 \cup L^2 \cup \cdots$$

$$L^+ := L^1 \cup L^2 \cup \cdots$$

Example

Assume $L = \{ \mathsf{ba}, \mathsf{da} \}$ and $M = \{ \mathsf{da}, \mathsf{rk}, \varepsilon \}$.

$$
\begin{aligned}
L \cup M &= \\
L \cap M &= \\
L \setminus M &= \\
LM &= \\
L^3 &= \\
L^* &= \\
L^+ &=
\end{aligned}
$$

Language Equivalences (not the same L)

- ✓ $L\emptyset =$
- ✓ $L\{\varepsilon\} =$
- ✓ $\{\varepsilon\}^* =$
- ✓ $\{\varepsilon\}^+ =$
- ✓ $\emptyset^* =$
- ✓ $\emptyset^+ =$
- ✓ $(L \cup M)N =$
- ✓ $(L^*)^* =$