Strings, Languages, and Regular expressions

Lecture 2

Strings

Definitions for strings

- **alphabet** Σ = finite set of symbols
- **string** = finite sequence of symbols of Σ
- **length** of a string w is denoted $|w|$.
- **empty string** is denoted "ε".

e.g., $\Sigma = \{0,1\}$, $\Sigma = \{\alpha, \beta, \ldots, \omega\}$, Σ = set of ascii characters

Could formalize as a function $w: [n] \rightarrow \Sigma$ where $|w| = n$

$|\varepsilon| = 0$

$|\text{cat}|=3$

Variable conventions (for this lecture)

$a, b, c, \ldots$ elements of Σ (i.e., strings of length 1)

$w, x, y, z, \ldots$ strings of length 0 or more

$A, B, C, \ldots$ sets of strings

Much ado about nothing

- ε is a **string** containing no symbols. It is not a set.
- $\{\varepsilon\}$ is a **set** containing one string: the empty string ε. It is a set, not a string.
- Ø is the **empty set**. It contains no strings.

Concatenation & its properties

- xy denotes the **concatenation** of strings x and y (sometimes written $x \cdot y$)
- Associative: $(uv)w = u(vw)$ and we write uvw.
- Identity element ε : $\varepsilon w = w \varepsilon = w$
- Can be used to *define* strings (set of all strings Σ^*) inductively
- NOT commutative: $ab \neq ba$

If $|x|=m$, $|y|=n$
$xy : [m+n] \rightarrow \Sigma$
such that
$xy(i) = x(i)$ if $i \leq m$
$xy(i) = y(i-m)$ else

Substring, Prefix, Suffix, Exponents

- v is a **substring** of w iff there exist strings x, y, such that $w = xvy$.
  - If $x = \varepsilon$ $(w = vy)$ then v is a **prefix** of w.
  - If $y = \varepsilon$ $(w = xv)$ then v is a **suffix** of w.
- If w is a string, then w^n is defined inductively by:
  - $w^n = \varepsilon$ if $n = 0$
  - $w^n = ww^{n-1}$ if $n > 0$

(blah)^4 =
blahblahblahblah

Set Concatenation

- If X and Y are sets of strings, then

  $XY = \{xy \mid x \in X, y \in Y\}$

  e.g. X = { fido, rover, spot }, Y = { fluffy, tabby }

  then XY ={ fidofluffy, fidotabby, roverfluffy, ...}

$|XY| = 6$

A = {a,aa}, B = {ε,a}
$|AB|$ = 3

A = {a,aa}, $B = \emptyset$
$AB = \emptyset$

Σ^n, Σ^*, and Σ^+

- Σ^n is the set of all strings over Σ of length exactly n. Defined inductively as:
  - $\Sigma^0 = \{\varepsilon\}$
  - $\Sigma^n = \Sigma\Sigma^{n-1}$ if $n > 0$
- Σ^* is the set of all finite length strings:

$$\Sigma^* = \bigcup_{n\geq 0} \Sigma^n$$

- Σ^+ is the set of all <u>nonempty</u> finite length strings:

$$\Sigma^+ = \bigcup_{n\geq 1} \Sigma^n$$

Σ^n, Σ^*, and Σ^+

- $|\Sigma^n| = |\Sigma|^n$
- $|\emptyset^n| = ?$
  - $\emptyset^0 = \{\varepsilon\}$
  - $\emptyset^n = \emptyset\emptyset^{n-1} = \emptyset$ if $n > 0$
- $|\emptyset^n| = 1$ if $n = 0$
  $|\emptyset^n| = 0$ if $n > 0$

Σ^n, Σ^*, and Σ^+

- Σ^* is the set of all finite length strings:

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$$

This can be the formal definition of a "string"

- x is a string iff $x=\varepsilon$ or $x=au$ where $|u|=|x|-1$
- $|\Sigma^*| = ?$
  - Infinity. More precisely, $\aleph_0$
  - $|\Sigma^*| = |\Sigma^+| = |\mathbb{N}| = \aleph_0$
- How long is the longest string in Σ^*?

no longest string!

- How many infinitely long strings in Σ^*?

none

Σ^n, Σ^*, and Σ^+

- Σ^+ is the set of all <u>nonempty</u> finite length strings:

$$\Sigma^+ = \cup_{n\geq 1} \; \Sigma^n$$

- $\Sigma^+ = ?$
  - $\Sigma \, \Sigma^*$
  - $\Sigma^* \, \Sigma$
  - $\Sigma \, \Sigma^* \, \Sigma$
  - $\Sigma \cup \Sigma^2 \, \Sigma^*$

Enumerating Strings

- Canonical (standard) ordering is the lexicographical (dictionary) ordering
  - Order by length (starting with 0)
  - Order the $|\Sigma|^n$ strings of length n by comparing characters left to right

1	ε	0
2	0	1
3	1	1
4	00	2
5	01	2
6	10	2
7	11	2
8	000	3
9	001	3
10	010	3
11	011	3
12	100	3
13	101	3
14	110	3
15	111	3
16	1000	4
17	1001	4
18	1010	4
19	1011	4
20	1100	4

Inductive Definitions

- Often operations on strings are formally defined inductively
  - e.g., w^n in terms of w^{n-1}
  - Another example: w^R (w reversed) inducting on length
    - If $|w| = 0$, $w^R = \varepsilon$
    - If $|w| \geq 1$, $w^R = u^R a$ where $w = au$
      - e.g. $(\text{cat})^R = (\text{c}\cdot\text{at})^R = (\text{at})^R\cdot\text{c} = (\text{a}\cdot\text{t})^R\cdot\text{c}$ $= (\text{t})^R\cdot\text{a}\cdot\text{c} = (\text{t}\cdot\varepsilon)^R\cdot\text{ac} = \varepsilon^R\cdot\text{tac} = \text{tac}$

$$\varepsilon^R = \varepsilon$$
$$(au)^R = u^R a$$

Well-defined: $|u|<|w|$

$a \in \Sigma, u \in \Sigma^*$

Inductive Proofs

- Inductive proofs follow inductive definitions
- **Theorem**: $(uv)^R = v^R u^R$
- **Proof**: By induction

$$\varepsilon^R = \varepsilon$$
$$(au)^R = u^R a$$

But on what? $|u|$, $|v|$, $|u+v|$, double induction on $|u|,|v|$?

$|u|$ (or $|v|$) is good enough:

Base case: $|u| = 0$: i.e., $u = \varepsilon$.

Then: $(uv)^R = v^R$

& $v^R u^R = v^R \varepsilon^R = v^R \varepsilon = v^R$ ✓

Definition of Reversal: base-case

Inductive Proofs

- Inductive proofs follow inductive definitions
- **Theorem**: $(uv)^R = v^R u^R$
- **Proof**: By induction

$$\varepsilon^R = \varepsilon$$
$$(au)^R = u^R a$$

<u>Inductive step:</u> Let $n > 0$. Assume $(wv)^R = v^R w^R \ \ \forall w, |w| < n$

Consider any u with $|u| = n$. So $u = aw$, $a \in \Sigma$, $w \in \Sigma^*$.

$$(uv)^R = (awv)^R = (a(wv))^R = (wv)^R a$$

Definition of Reversal: inductive-case

$$= v^R w^R a$$

Inductive Hypothesis: $|w| < n$

$$= v^R (aw)^R$$

Definition of Reversal: inductive-case

$$= v^R u^R$$

Languages

Recall

Computation

Problem:
To compute a function F that maps each input (a string) to an output bit

Program:
A finitely described process taking a string as input, and outputting a bit (or not halting)

P computes F if for every x, `P(x)` outputs F(x) and halts

Too restrictive?

Enough to compute functions with longer outputs too:
`P(x,i)` outputs the i^{th} bit of F(x)

Enough to model *interactive* computation too:
`P*(x,state)` outputs `(y,new_state)`

Language

- A function from Σ^* to $\{0,1\}$ can be identified with the set of strings mapped to 1
- A *language* is a subset of Σ^*
  – Computational problem for a language: given a string in Σ^*, decide if it belongs to the language
- Examples of languages : $\emptyset$, Σ^*, Σ, $\{\varepsilon\}$, set of strings of odd length, set of strings encoding valid C programs, set of strings encoding valid C programs that halt, …
- There are uncountably many languages (but each language has countably many strings)

1	ε	0
2	0	0
3	1	1
4	00	0
5	01	1
6	10	1
7	11	0
8	000	0
9	001	1
10	010	1
11	011	0
12	100	1
13	101	0
14	110	0
15	111	1
16	1000	1
17	1001	0
18	1010	0
19	1011	1
20	1100	0

Operations on Languages

- Already seen concatenation: $L_1L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$
- Set operations:
  - Complement: $\bar{L} = \Sigma^* - L = \{ x \in \Sigma^* \mid x \notin L\}$
  - Union: $L_1 \cup L_2$
  - Intersection, difference (can be based on the above two)
- L^n inductively defined: $L^0 = \{\varepsilon\}, L^n = LL^{n-1}$
- $L^* = \cup_{n \geq 0} L^n$, and $L^+ = LL^*$
- $\{\varepsilon\}^* = ?$ $\emptyset^* = ?$

Complexity of Languages

- How *computable* is a language?
- Singleton languages
  - L such that $|L| = 1$. Example: $L = \{374\}$
  - An algorithm can have the single string hard-coded into it
- More generally, finite languages
  - Algorithm can have all the strings hard-coded into it
- Many interesting languages are uncomputable
- But many others are neither too easy nor impossible…

Regular Languages

Regular Languages

- The set of regular languages over some alphabet Σ is defined inductively by:
- $\emptyset$ is a regular language
- $\{\varepsilon\}$ is a regular language
- $\{a\}$ is a regular language for each $a \in \Sigma$
- If L_1, L_2 are regular, then $L_1 \cup L_2$ is regular
- If L_1, L_2 are regular, then $L_1 L_2$ is regular
- If L is regular, then $L*$ is regular

Regular Languages Examples

- $L = \{w\}$ where $w \in \Sigma^*$ is any fixed string
  - e.g., $L = \{aba\} = \{a\}\{b\}\{a\}$ and $\{a\}\&\{b\}$ are both regular
  - Proof by induction on $|w|$, using concatenation for induction
- L = any finite set of strings
  - e.g., L = set of all strings of length at most 10
  - Proof by induction on $|L|$, using union for induction (and the above)
  - Beware: Induction applicable only for $|L| \in \mathbb{N}$, not $|L| = \aleph_0$

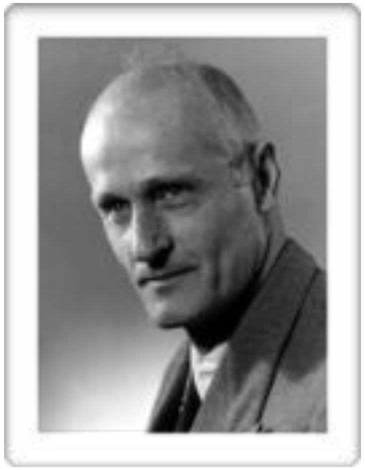
Regular Languages Examples

- Infinite sets, but of strings with "regular" patterns
  - Σ^* (recall: L^* is regular if L is)
  - $\Sigma^+ = \Sigma\Sigma^*$
  - All binary integers, without leading 0's
    - $L = \{1\}\{0,1\}^* \cup \{0\}$
  - All binary integers which are multiples of 37
    - *later*

Regular Expressions

Regular Expressions

- A short-hand to denote a regular language as strings that match a *pattern*
- Useful in
  - text search (editors, Unix/grep)
  - compilers: lexical analysis
- Dates back to 50's: Stephen Kleene, who has a star named after him*



** The star named after him is the Kleene star "*"*

Inductive Definition

A regular expression r over alphabet Σ is one of the following ($L(r)$ is the language it represents):

Atomic expressions (Base cases)	
$\emptyset$	$L(\emptyset) = \emptyset$
ε	$L(\varepsilon) = \{ \varepsilon \}$
a for $a \in \Sigma$	$L(a) = \{ a \}$

Inductively defined expressions	
(r_1+r_2)	$L(r_1+r_2) = L(r_1) \cup L(r_2)$
(r_1r_2)	$L(r_1r_2) = L(r_1)L(r_2)$
$(r)^*$	$L(r^*) = L(r)^*$

alt notation $(r_1|r_2)$ or $(r_1 \cup r_2)$

Any regular language has a regular expression and vice versa

Regular Expressions

- Can omit many parentheses
  - By following precedence rules :
    * before *concatenation* before +
    - e.g. $r^*s + t \equiv ((r^*)\ s) + t$
  - By associativity: $(r+s)+t \equiv r+s+t$, $(rs)t \equiv rst$
- More short-hand notation
  - e.g., $r^+ \equiv rr^*$ (note: $^+$ is in superscript)

Regular Expressions: Examples

- $(0+1)^*001(0+1)^*$
  - All binary strings containing the substring 001
- $0^* + (0^*10^*10^*10^*)^*$
  - All binary strings with #1s ≡ 0 mod 3
- $(01)^* + (10)^* + 1(01)^* + 0(10)^*$
  - Alternating 0s and 1s. Also, $(1+\varepsilon)(01)^*(0+\varepsilon)$
- $(01+1)^*(0+\varepsilon)$
  - All binary strings without two consecutive 0s

Exercise: create regular expressions

- All binary strings with either the pattern 001 or the pattern 100 occurring somewhere

one answer: $(0+1)^*001(0+1)^* + (0+1)^*100(0+1)^*$

- All binary strings with an even number of 1s

one answer: $0^*(10^*10^*)^*$

A non-regular language

An inductively defined language

Define L over $\{0,1\}^*$ by:

- $\varepsilon \in L$
- if $w \in L$, *then* $0w1 \in L$

What do strings in L look like?

Give a characterization of L and prove it correct.

Can you find a regular expression for L ?

will show impossible!

An inductively defined language

Define L over $\{0,1\}^*$ by:

- $\varepsilon \in L$
- if $w \in L$, then $0w1 \in L$

Conjecture: $L = \{\ 0^i 1^i : i \geq 0\ \}$

How can we prove this is correct?

Prove (by induction) that

(a) $L \subseteq \{\ 0^i 1^i : i \geq 0\ \}$

(b) $L \supseteq \{\ 0^i 1^i : i \geq 0\ \}$

$$L \subseteq \{ 0^i 1^i : i \geq 0 \}$$

Show by induction on $|w|$, that if $w \in L$, then w is of the form 0^i1^i.

Base case: $|w| = 0$.

Then $w = \varepsilon = 0^0 1^0$

Inductive Step: Let $n > 0$.

<u>Assume:</u> for all $k < n$,

any w in L with $|w| = k$, is of form 0^i1^i

<u>Prove:</u> Any w in L with $|w| = n$ is of form 0^i1^i

Inductive step

Consider arbitrary $w \in L$, with $|w| = n$.

Then $w = 0u1$ where $u \in L$ has size $n\text{-}2 < n$

(by definition of L)

By induction, u is of form $0^i 1^i$.

Then $w = 0u1 = 00^i1^i1 = 0^{i+1}1^{i+1}$, *the required form*

$L \supseteq \{ 0^i 1^i : i \geq 0 \}$

Show by induction on n, that if w is of the form $0^n 1^n$, then $w \in L$.

Base case: $n = 0$.

Then $w = 0^0 1^0 = \varepsilon$, which is in L by definition

Inductive step:

Let $n > 0$, and assume for all $k < n$ that $0^k 1^k \in L$

$0^n 1^n = 0 0^{n-1} 1^{n-1} 1 = 0 u 1$, with $u \in L$ by induction.

Since $u \in L$, so is $0u1 = 0^n 1^n$ by definition of L