# Recursively Defining Languages

## Homework

Assignment 1: Due Tuesday, Sept 17th

- Exercises in Textbook:
  - 1.38
  - 1.39a,b
  - 2.10
  - 2.36
  - 2.39a,b,c

## Today's plan

- Recursive Definition of Languages
- Mathematical Proofs

## Computation Hall of Fame

- Steven Cole Kleene
  - 1909-1994
  - b. Hartford, Conn.
  - PhD – Princeton (1934)
  - Prof at U of Wisc at Madison (1935 – 1979)
  - Introduced Kleene Star op
  - Defined regular expressions

## Specifying Languages

- Recall:
  - What is a language?
  - A language is nothing more than a set of strings.

## Specifying Languages

- How do we specify languages?
  - If language is finite, you can list all of its strings.
    - L = {a, aa, aba, aca}
  - Using basic Language operations
    - L= $\{aa, ab\}^* \cup \{b\}\{bb\}^*$
  - Descriptive:
    - L = $\{x \mid n_a(x) = n_b(x)\}$

## Specifying Languages

- Today we will learn how to specify languages recursively.

## Recursive Definitions

- Definition is given in terms of itself
  - Example (factorial)

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n * (n-1)! & \text{otherwise} \end{cases}$$

$4! = 4 * 3!$
$= 4 * (3 * 2!)$
$= 4 * (3 * (2 * 1!))$
$= 4 * (3 * (2 * (1 * 0!)))$
$= 4 * (3 * (2 * (1 * 1))))$
$= 24$

## Recursive Definitions and Languages

- Languages can also be described by using a recursive definition
  1. Initial elements are added to your set
  2. Additional elements are added to your set by applying a rule(s) to the elements already in your set
  3. Complete language is obtained by applying step 2 infinitely

## Recursive Definitions and Languages

- Example:
  - Recursive definition of $\Sigma^*$

1. $\Lambda \in \Sigma^*$
2. For all $x \in \Sigma^*$ and all $a \in \Sigma$, $xa \in \Sigma^*$
3. Nothing else is in $\Sigma^*$ unless it can be obtained by a finite number of applications of rules 1 and 2.

## Recursive Definitions and Languages

- Let's iterate through the rules for $\Sigma = \{a,b\}$
  - i=0 $\Sigma^* = \{\Lambda\}$
  - i=1 $\Sigma^* = \{\Lambda, a, b\}$
  - i=2 $\Sigma^* = \{\Lambda, a, b, aa, ab, ba, bb\}$
  - i=3 $\Sigma^* = \{\Lambda, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb\}$
  - …and so on

## Recursive Definitions and Languages

- Example (in honor of Kleene):
  - Recursive definition of $L^*$

1. $\Lambda \in \Sigma^*$
2. For all $x \in L$ and all $y \in L$, $xy \in L^*$
3. Nothing else is in $\Sigma^*$ unless it can be obtained by a finite number of applications of rules 1 and 2.

## Recursive Definitions and Languages

- Let's iterate through the rules for L = {aa,bb}
  - i=0 $L^* = \{\Lambda\}$
  - i=1 $L^* = \{\Lambda, aa, bb\}$
  - i=2 $L^* = \{\Lambda, aa, bb, aaaa, aabb, bbbb, bbaa\}$
  - i=3 $L^* = \{\Lambda, aa, bb, aaaa, aabb, bbbb, bbaa, aaaaaa, aaaabb, aabbaa, aabbbb, bbbbaa, bbbbbb, \ldots\}$
  - …and so on

## Recursive Definitions – another Example

- Example: Palindromes
  - A palindrome is a string that is the same read left to right or right to left
  - First half of a palindrome is a "mirror image" of the second half
  - Examples:
    - a, b, aba, abba, babbab.

## Recursive Definitions – another Example

- Recursive definition for palindromes (pal) over $\Sigma$
  1. $\Lambda \in$ pal
  2. For any $a \in \Sigma$, $a \in$ pal
  3. For any $x \in$ pal and $a \in \Sigma$, $axa \in$ pal
  4. No string is in pal unless it can be obtained by rules 1-3

## Recursive Definitions – another Example

- Let's iterate through the rules for pal over $\Sigma$ = {a,b}
  - i=0 pal = {Λ, a, b}
  - i=1 pal = {Λ, a, b, aaa, aba, bab, bbb}
  - i=2 pal = {Λ, a, b, aaa, aba, bab, bbb, aaaaa, aabaa, ababa, abbba, baaab, ababa, bbabb, bbbbb}

## Recursive Definitions – yet another Example

- Example: Fully parenthesized algebraic expressions (AE)
  - $\Sigma$ = { a, (, ), +, - }
  - All expressions where the parens match correctly are in the language
  - Examples:
    - a, (a + a), (a + (a – a)), ( (a + a) – (a + a)), etc.

## Recursive Definitions – yet another Example

- Recursive definition for AE
  1. $a \in$ AE
  2. For any x, y $\in$ AE (x + y) and (x – y) $\in$ AE
  3. No string is in pal unless it can be obtained by rules 1-2

## Recursive Definitions – yet another Example

- Let's iterate through the rules for AE
  - i=0 AE = {a}
  - i=1 AE = { a, (a+a), (a-a) }
  - i=2 AE = {a, (a+a), (a-a) , (a + ( a + a)), (a – (a + a)), ( a + ( a – a)), ( a – ( a – a)), ((a + a) + a), ( (a + a) – a), …}

## Recursive Definitions – a final Example

- $L = \{x \in \{0,1\}^* \mid x = 0^i1^j \text{ and } i \geq j \geq 0\}$
  - In English:
    - strings over the alphabet {0,1}
    - each string contains zero or more 0's followed by a zero or more 1's
    - the number of 1's is greater than or equal to the number of 0's

## Recursive Definitions – a final Example

- $L = \{x \in \{0,1\}^* \mid x = 0^i1^j \text{ and } i \geq j \geq 0\}$
- A recursive definition
  1. $\Lambda \in L$
  2. For any $x \in L$, both $0x$ and $0x1 \in L$
  3. No strings are in L unless it can be obtained using rules 1-2.

Later we will prove that this definition does indeed describe L.

## Recursive Definitions and Languages

- Questions on Recursive Definition?
- Functions on strings and languages can also be defined recursively.

## Recursive Function on strings

- Recursive definition for the reverse of a string x (rev(x) or $x^r$)
  - eg rev (abcd) = dcba

1. $\Lambda^r = \Lambda$
2. For any $x \in \Sigma^*$ and $a \in \Sigma$, $(xa)^r = ax^r$

## Recursive Function on strings

$(abca)^r = ((abc)a)^r$
$= a\,(abc)^r$
$= a\,((ab)c)^r$
$= a(c(ab)^r)$
$= ac\,(ab)^r$
$= ac\,((a)(b))^r$
$= ac\,(b(a)^r)$
$= acb(a)^r$
$= acb\,(\Lambda\, a)^r$
$= acb(a(C)^r)$
$= acb(a\,\Lambda) = acba$

$\Lambda^r = \Lambda$
$(xa)^r = ax^r$

## Recursive Function on strings

- Recursive definition for the length of a string x, |x|

1. $|\Lambda| = 0$
2. For any $x \in \Sigma^*$ and $a \in \Sigma$, $|xa| = |x| + 1$

## Recursive Function on strings

$|\Lambda| = 0$

$|xa| = |x| + 1$

$$
\begin{aligned}
|abca| &= |abc| + 1 \\
&= |ab| + 1 + 1 \\
&= |a| + 1 + 1 + 1 \\
&= |\Lambda| + 1 + 1 + 1 + 1 \\
&= 0 + 1 + 1 + 1 + 1 \\
&= 4
\end{aligned}
$$

## Summary

- Languages = set of strings
- Recursive Definition of Languages
- Recursive Definition of Functions

## Questions?

- Any questions?
- Let's take a 10 minute break and then do proofs.