Strings and Languages

“It is always best to start at the beginning”

-- Glynda, the good witch of the North

What is a Language?

• A language is a set of strings made of symbols from a given alphabet.
• An alphabet is a finite set of symbols (usually denoted by \( \Sigma \)).

  – Examples of alphabets:
    • \([0, 1]\)
    • \(\{a, \beta, \gamma, \delta, \epsilon, \zeta, \eta\}\)
    • \(\{a, b, c, d, e, \phi, \chi, \mu, n, o, \rho, q, r, s, t, u, v, w, x, y, z\}\)
    • \(\{a\}\)

What is a string?

• A string over \(\Sigma\) is a finite sequence (possibly empty) of elements of \(\Sigma\).
• \(\epsilon\) denotes the null string, the string with no symbols.

  – Example strings over \(\{a, b\}\)
    • \(\epsilon, a, b, aa, bb, aba, abba\)
  – NOT strings over \(\{a, b\}\)
    • \(aaa\ldots, abca\)

The length of a string

• The length of a string \(x\), denoted \(|x|\), is the number of symbols in the string

  – Example:
    • \(|abba\beta| = 5\)
    • \(|a| = 1\)
    • \(|b| = 7\)
    • \(|\epsilon| = 0\)

Strings and languages

• For any alphabet \(\Sigma\), the set of all strings over \(\Sigma\) is denoted as \(\Sigma^*\).
• A language over \(\Sigma\) is a subset of \(\Sigma^*\).

  – Example
    • \(\{a, b\}^* = \{\epsilon, a, b, aa, bb, ab, ba, aaa, bbb, baa, \ldots\}\)
  – Example Languages over \(\{a, b\}\)
    • \(\{a, b, aa, bb\}\)
    • \(\{x \in \{a,b\}^* \mid |x| = 8\}\)
    • \(\{x \in \{a,b\}^* \mid |x| \text{ is odd}\}\)
    • \(\{x \in \{a,b\}^* \mid n_a(x) = n_b(x)\}\)
    • \(\{x \in \{a,b\}^* \mid n_a(x) + 2 \text{ and } x \text{ starts with } b\}\)
Concatenation of String

• For $x, y \in \Sigma^*$
  – $xy$ is the concatenation of $x$ and $y$.
    • $x = aba$, $y = bbb$, $xy = ababbb$
    • For all $x, \varepsilon x = x$
  – $x^i$ for an integer $i$, indicates concatenation of $x$, $i$ times
    • $x = aba$, $x^3 = abababa$
    • For all $x, x^0 = \varepsilon$

Some string related definitions

• $x$ is a substring of $y$ if there exists $w, z \in \Sigma^*$ (possibly $\varepsilon$) such that $y = wxz$.
  – $car$ is a substring of $carnage, descartes, vicar, car$, but not a substring of charity.
• $x$ is a suffix of $y$ if there exists $w \in \Sigma^*$ such that $y = wx$.
• $x$ is a prefix of $y$ if there exists $z \in \Sigma^*$ such that $y = xz$.

Operations on Languages

• Since languages are simply sets of strings, regular set operations can be applied:
  – For languages $L_1$ and $L_2$ over $\Sigma^*$
    • $L_1 \cup L_2 = \{xy \mid x \in L_1 \text{ or } y \in L_2\}$
    • $L_1 \cap L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$
    • $L_1 - L_2 = \{xy \mid x \in L_1 \text{ that are not in } L_2\}$
    • $L^* = \Sigma^* - L$

Concatenation of Languages

• If $L_1$ and $L_2$ are languages over $\Sigma^*$
  – $L_1L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$
  – Example:
    • $L_1 = \{\text{hope, fear}\}$
    • $L_2 = \{\text{less, fully}\}$
    • $L_1L_2 = \{\text{hopeless, hopefully, fearless, fearfully}\}$

Concatenation of Languages

• If $L$ is a language over $\Sigma^*$
  – $L^k$ is the set of strings formed by concatenating elements of $L$, $k$ times.
  – Example:
    • $L = \{aa, bb\}$
    • $L^3 = \{aaaaaa, aaaaab, aabbaa, aabbbb, bbbbb, bbbbaa, bbaaab, bbaaaa\}$
    • $L^0 = \{\varepsilon\}$

Kleene Star Operation

• The set of strings that can be obtained by concatenating any number of elements of a language $L$ is called the Kleene Star, $L^*$

\[ \bigcup_{n \geq 0} L^n = L^* \]

Note that since, $L^*$ contains $L^0$, $\varepsilon$ is an element of $L^*$
Kleene Star Operation

- The set of strings that can be obtained by concatenating one or more elements of a language $L$ is denoted $L^+$

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

- Next we will define how to specify languages recursively

  - In future classes, we will describe how to specify languages by defining a mechanism for generating the language

  - Any questions?

Recursive Definitions

- Definition is given in terms of itself
  - Example (factorial)
    
    $$
    4! = 4 \cdot 3!,
    = 4 \cdot (3 \cdot 2!),
    = 4 \cdot (3 \cdot (2 \cdot 1!)),
    = 4 \cdot (3 \cdot (2 \cdot (1 \cdot 0!))),
    = 24
    $$

Recursive Definitions and Languages

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

Recursive Definitions and Languages

- Example:
  - Recursive definition of $\Sigma^*$
    1. $\epsilon \in \Sigma^*$
    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 $\Sigma = \{a,b\}$
  - $i=0$ $\Sigma^* = \{\epsilon\}$
  - $i=1$ $\Sigma^* = \{\epsilon, a, b\}$
  - $i=2$ $\Sigma^* = \{\epsilon, a, b, aa, ab, ba, bb\}$
  - $i=3$ $\Sigma^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb\}$
  - …and so on

Recursive Definitions and Languages

• Example:
  - Recursive definition of $L^*$
    1. $\epsilon \in L^*$
    2. For all $x \in L$ and all $y \in L$, $xy \in L^*$
    3. Nothing else is in $L^*$ unless it can be obtained by a finite number of applications of rules 1 and 2.

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. $\epsilon \in \text{pal}$
  2. For any $a \in \Sigma$, $a \in \text{pal}$
  3. For any $x \in \text{pal}$ and $a \in \Sigma$, $axa \in \text{pal}$
  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 $\Sigma = \{a,b\}$
  - $i=0$ $\text{pal} = \{\epsilon, a, b\}$
  - $i=1$ $\text{pal} = \{\epsilon, a, b, aaa, aba, bab, bbb\}$
  - $i=2$ $\text{pal} = \{\epsilon, a, b, aaa, aba, bab, bbb, aaaaa, aabaa, ababa, abbaa, baaba, ababa, babba, bbabb, bbabb\}$
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 \( AE = \{ a, (a+a), (a-a), (a + (a + a)), (a - (a + a)), (a + (a - a)), (a - (a + a)), (a + a) + a), (a + a) - a), \ldots \} \)

Recursive Definitions – a final Example

• \( L = \{ x \in \{0,1\}^* | x = 0^i1^j \text{ and } i \geq j \geq 0 \} \)
  – 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 and Languages

• Questions on Recursive Definition?

• Functions on strings and languages can also be defined recursively.
Structural Induction

• When dealing with languages, it is sometime cumbersome to restate the problems in terms of an integer.

• For languages described using a recursive definition, another type of induction, structural induction, is useful.

Structural Induction

• Principles
  – Suppose
    • U is a set,
    • I is a subset of U (BASIS),
    • Op is a set of operations on U (INDUCTION).
    • L is a subset of U defined recursively as follows:
      – I ⊆ L
      – L is closed under each operation in Op
      – L is the smallest set satisfying 1 & 2

Then

  – To prove that every element of L has some property P, it is sufficient to show:
    1. Every element of I has property P
    2. The set of elements of L having property P is closed under Op

#2: If x ∈ L has property P, Op(x) also must have property P

Structural Induction

• To prove that every element of L has some property P:
  – Our property is:
    A = {x ∈ {0,1}⁺ | x = 0ⁱ1ʲ and i ≥ j ≥ 0}

P(x) is true if x ∈ A.
Structural Induction

– To prove that every element of L has some property P, it is sufficient to show:
  1. Every element of I has property P
     In our case, must show that $\varepsilon$ has property P, i.e. $\varepsilon \in A$, $\varepsilon = 0^i : i \geq 0$
     Once again, this is the case where $i=j=0$

2. The set of elements of L having property P is closed under $Op$
   If $x \in L$ has property P, $Op(x)$ also must have property P
   Assume $x$ has property P,
   $x \in A$, $x = 0^i : i \geq 0$
   $Op(x) = 0^x$, which is an element of A
   $Op(0^x) = 0^x1$ which is an element of A
   Similar proof to induction with no mention of an integer

Questions?

• Any questions?

• Next Time:
  – Our first machine: The Finite Automata!