Alphabets, strings, and languages

Stefan D. Bruda

CS 310, Winter 2025

Alphabets and Strings

- Alphabet Σ: a finite set of symbols
- Strings (not sets!) over an alphabet
- The set of all strings over Σ: Σ^*; Empty string: ε
- Operations on strings:
  - Length ($|w|$), concatenation ($\cdot$ or juxtaposition), substring, suffix, prefix
    - ε is the identity for concatenation ($\varepsilon w = w\varepsilon = w$)
    - a string w is a trivial prefix, suffix and substring of itself
    - ε is a trivial prefix, suffix and substring of any string
  - Length over a set A: $|w|_A$ is the length of the string w from which all the symbols not in A have been erased
    - Abuse of notation: $|w|_a$ is a shorthand for $|w|_{\{a\}}$
  - Exponentiation: w^n = n copies of w concatenated together
    - Recursive definition: $w^0 = \varepsilon$; $w^{i+1} = w^i w$
  - Reversal: $w = \varepsilon \Rightarrow w^{\mathbb{R}} = \varepsilon$; for $a \in \Sigma$: $w = ua \Rightarrow w^{\mathbb{R}} = au^{\mathbb{R}}$

Formal Languages

- (Formal) language: well defined set of strings; example: Σ^*
  - Abuse of notation: For $a \in \Sigma$ we simply write a instead of $\{a\}$
- Operations: union ($\cup$, $+$), intersection ($\cap$), difference ($\setminus$), complement ($\overline{A} = \Sigma^* \setminus A$)
  - Union is commutative, associative, and idempotent, with $\emptyset$ as identity
- Concatenation: $L_1 L_2 = \{w_1 w_2 : w_1 \in L_1 \land w_2 \in L_2\}$
  - Associative, distributive to $+$, identity: ε, zero: $\emptyset$
- Exponentiation: $L^n = \{w_1 w_2 \ldots w_n : \forall 1 \leq i \leq n : w_i \in L\}$
- Closure (Kleene closure): $L^* = \{w_1 w_2 \cdots w_n : n \geq 0 \land \forall 1 \leq i \leq n : w_i \in L\} = \sum_{i \geq 0} L^i$ ($L^* = \varepsilon + L^1 + L^2 + L^3 + \cdots$)
- Recursion: $L = f(L)$
  - $B = \varepsilon + 0B1$
    - The empty string is an element of B
    - For any $b \in B$, $0b1$ is also a string in B
    - These are the only elements of B

Formal Languages (cont'd)

- Recursion (cont'd):
  - Another example: $P = \varepsilon + 0 + 1 + 0P0 + 1P1$
  - How to solve a recursive equation $L = f(L)$ defined using only ε, symbols in Σ, union, and concatenation:
    - $L_0 = \emptyset$
    - For $j = 0, 1, 2, \ldots$: $L_{j+1} = f(L_j)$
    - $L = \bigcup_{j \geq 0} L_j$
  - Note that L above is the smallest solution of the equation $L = f(L)$
    - $L' = f(L') \Rightarrow L \subseteq L'$
  - The scheme above is also generalizable to k mutually recursive equations
    - Such as $L = f(L, M)$ and $M = g(L, M)$ (for $k = 2$)

Key Problems in Formal Languages

- Specification: formalized notations for rigorous definitions
- Realization: systematic methods for programming recognizers
- Classification: organize the space of formal languages into a hierarchy of classes
  - Languages can be classified by the set of operations used to define them
  - All the languages defined using only symbols from Σ, union, and concatenation are finite languages; we can define every finite language if we add ε and $\emptyset$
  - Languages defined using only union, concatenation, and closure from symbols from Σ, ε, and $\emptyset$ are called regular languages
    - A language description as above is called a regular expression
  - Languages defined using only union, concatenation, and recursion from symbols from Σ, ε, and $\emptyset$ are called context-free languages
    - A language description as above is called a context-free grammar
  - finite $\subset$ regular $\subset$ context-free $\subset$ unrestricted

Practical Examples of Regular Expressions

$$([+-]+\varepsilon)[0-9][0-9]^*.[0-9][0-9]^*(E([+-]+\varepsilon)[0-9][0-9]^*+\varepsilon)$$

$$\begin{aligned}
\text{letter_} &= [A-Za-z_] \\
\text{digit} &= [0-9] \\
\text{id} &= \text{letter_ (letter_} + \text{digit)}^* \\
\text{digits} &= \text{digit digit}^* \\
\text{fraction} &= \text{. digits} \\
\text{exp} &= E\ ([+-]+\varepsilon)\ \text{digits} \\
\text{number} &= \text{digits fraction? exp?} \\
\text{if} &= i\,f \\
\text{then} &= t\,h\,e\,n \\
\text{else} &= e\,l\,s\,e \\
\text{rel_op} &= < + > + <= + >= + == + !=
\end{aligned}$$

```
  ^\(.*\), \(.*\):00\([0-9]*\)$
⇒ \2 \1:\,(downcase (concat (substring \2 0 1) \1)):CS\3
```