CSCI 340: Computational Models

Languages

Chapter 2

Department of Computer Science
What is a Language?

- English: “letters”, “words”, “sentences”
- Programming: “keywords”, “variables”, “numbers”, “symbols”
- General: *language structure* – decision of whether a given string of units is “matched” or *valid*
Important Terms

- *alphabet* – finite set of fundamental units out of which we build structures.
- *language* – a certain specified set of strings of characters from the alphabet
- *words* – strings which are permissible in the language
- *empty string or null string* – a string which has no letters ($\lambda$)
- *null set* – denoted as $\emptyset$

Question

Is there a difference between empty string and an empty language?
## An Aside on Set Theory

### Assume

- $L$ is a language
- $+$ is “union of sets” operator
- $\emptyset$ is empty set
- $\lambda$ is empty string

### Claim 1

$L + \{\lambda\} \neq L$

### Claim 2

$L + \emptyset = L$

This implies that $\emptyset$ is a valid definition for a language
The English Languages

Alphabet
\[ \Sigma = \{a, b, c, d, e, \ldots, z', -\} \]

Words

\textit{ENGLISH-WORDS} = \{all the words in a standard dictionary\}

Problem: How can we represent sentences?
The Real English Languages

Alphabet

\[ \Gamma = \text{entries of } ENGLISH-WORDS + \{ \text{space} \} + \{ \text{punctuation} \} \]

Words (a.k.a. English Sentences)

- Must rely on grammatical rules of English
- There are infinitely many
  - I ate one apple.
  - I ate two apples.
  - I ate three apples.
  - ...........

We can list all rules of the grammar to give a finite description for an infinite language. This will make “I ate three Tuesdays” valid!
Defining a Language

Language Defining Rules

1. Tell us how to test a string of alphabet letters that we are presented with
2. Tell us how to construct all of the words in the language by some clear procedure

Example

\( \Sigma = \{x\} \)

\( L_1 = \{x \ xx \ xxx \ xxxx \ \ldots\} \)

Alternatively,

\( L_1 = \{x^n \text{ for } n = 1 \ 2 \ 3 \ \ldots\} \)
Working with a Language

Null String?
A language does not need to accept \( \lambda \). \( L_1 \) doesn’t

Concatenation

- Two strings written side by side yield a new string
- \( x^n \) concatenated with \( x^m \) is \( x^n + m \)

Symbols

- We can designate a word in a given language by a new symbol
  - Let \( a = xx \) and \( b = xxx \)
  - Therefore, \( ab = xxxxxx \)
- Two words of \( L \) concatenated are not guaranteed to produce another word in \( L \)
Example: Numbers

Example

Σ = \{0 1 2 3 4 5 6 7 8 9\}

\( L_3 = \{\text{any finite string of } \Sigma \text{ letters that doesn’t start with } 0\} \)

A subset of \( L_3 \) might look like:

\( L_3 = \{1 2 3 4 5 6 7 8 9 10 11 12 \ldots\} \)

If we want to allow the string (word) 0, we could say:

\( L_3 = \{\text{any finite string of } \Sigma \text{ letters that, if it starts with } 0, \text{ has no more letters after the first }\} \)
Example: Length

We define the function **length** of a string to be the number of letters in the string. We write this function using the word “length”. For example, if $a = xxxx$ in the language $L_1$, then

$$\text{length}(a) = 4$$

Or we could write directly that in a language, such as $L_3$,

$$\text{length}(428) = 3$$

In any language which includes $\lambda$ we have

$$\text{length}(\lambda) = 0$$

Corollary: For any word $w$ in a language, if $\text{length}(w) = 0$, then $w = \lambda$
Redefining Number with **length**

We can present another definition for $L_3$

$$L_3 = \{ \text{any finite string of } \Sigma \text{ letters that, if it has}$$

$$\text{length more than 1, does not start with a 0 } \}$$

This isn’t necessarily a better definition, but it illustrates equivalent languages can be defined in multiple ways.
Adding \( \lambda \) to a finite language

If we look back to \( L_1 \), which described one or more “x” characters defining valid words, we may want to expand the language to include *empty string*

\[
L_4 = \{ \lambda \ x \ xx \ xxx \ xxxx \ \ldots \} 
\]

Alternatively,

\[
L_4 = \{ x^n \text{ for } n = 0 \ 1 \ 2 \ 3 \ \ldots \} 
\]

**Notice:** \( x^0 = \lambda \)
Example: Reverse

Definition
Let us introduce the function `reverse`. If \( a \) is a word in some language, \( L \), then `reverse(a)` is the same string of letters spelled backward even if this backwards string is not a word in \( L \).

Example

\[
\begin{align*}
\text{reverse}(xxx) &= xxx \\
\text{reverse}(xxxxx) &= xxxxx \\
\text{reverse}(145) &= 541 \\
\end{align*}
\]

But let us also note that in \( L_1 \),

\[
\begin{align*}
\text{reverse}(140) &= 041 \\
\end{align*}
\]

which is not a word in \( L_1 \)
Example: Palindrome Language

**Definition**

PALINDROME ($P$) is a new language over the alphabet

$$\Sigma = \{a \ b\}$$

$$P = \{\lambda, \text{ and all strings } x \mid \text{reverse}(x) = x\}$$

$$\therefore P = \{\lambda \ a \ b \ aa \ bb \ aaa \ aba \ bab \ bbb \ aaaa \ abba \ \ldots\}$$

**Interesting Properties**

1. *concatenating* two words from $P$ sometimes produces a word within $P$. e.g. $abba + abba = abbaabba$

2. More often than not, *concatenating* two words from $P$ does not yield a word within $P$. e.g. $aa + aba = aaaba$
Kleene Closure (or the Kleene Star)

**Definition**

- Given an alphabet $\Sigma$, we wish to define a language in which any string of letters from $\Sigma$ is a word, even the null string $\lambda$.
- This language shall be known as the **closure** of the alphabet.
- Symbolically denoted as: $\Sigma^*$

**Example**

If $\Sigma = \{x\}$, then $\Sigma^* = \{\lambda x xx xxx xxxx \ldots\}$

**Example**

If $\Sigma = \{0, 1\}$, then $\Sigma^* = \{\lambda 0 1 00 01 10 11 000 001 \ldots\}$

**Example**

If $\Sigma = \{a, b, c\}$, then $\Sigma^* = \{\lambda a b c aa ab ac ba bb bc ca cb cc aaa \ldots\}$
Kleene Closure

• an operation that makes an infinite language or strings of letters out of an alphabet
• infinitely many words, each of a finite length
• often ordered by size first, then lexicographically

Definition

If $S$ is a set of words, then $S^*$ means the set of all finite strings formed by concatenating words from $S$. Any word may be used as often as we like, and $\lambda$ is also included.

Problem

Compare:

ENGLISH-WORDS* and ENGLISH-SENTENCES
Kleene Closure

Example

\[ S = \{aa \ b\} \]
\[ S^* = ? \]

Example

\[ S = \{a \ ab\} \]
\[ S^* = ? \]

To prove that a certain word is in the closure language \( S^* \), we must show how it can be written as a concatenation of words from the base set \( S \).
The **concatenation** of words from a base set $S$ can be viewed as a **factor** of a word from closure set $S^*$

**Example**

$S = \{xx \ xxx\}$

$S^* = \{x^n \text{ for } n = 0 \ 2 \ 3 \ 4 \ \ldots\}$

Notice how the word $x$ is the only word not in the language $S^*$

There is also ambiguity in factoring certain strings e.g. $xxxxxxx$

$$(xx)(xx)(xxx) \text{ or } (xx)(xxx)(xx) \text{ or } (xxx)(xx)(xx)$$

How can we **prove** that $S$ only contains repetitions of letter $x$ not equal to size of 1?
Proving $S^*$ contains all $x^n \mid n \neq 1$

Example

$S = \{xx \ xxxx\}$
$S^* = \{x^n \text{ for } n = 0 \ 2 \ 3 \ 4 \ldots\}$

Proof (by constructive algorithm).

**Base:** $x^0 = \lambda$
**Base:** $x^2 = xx$
**Base:** $x^3 = xxx$

**Factor:** $x^4 = x^2 + x^2$
**Factor:** $x^5 = x^3 + x^2$

$x^{n+2} = x^n + x^2$
Kleene Closure

The Kleene closure of two sets can end up being the **same language**

**Example**

\[
S = \{a \ b \ ab\} \\
T = \{a \ b \ bb\}
\]

- Both \(S^*\) and \(T^*\) define languages of all strings of \(a\)'s and \(b\)'s.
- Any string of \(a\)'s and \(b\)'s can be factored into syllables (\(a\)) and (\(b\)).

Consider \(ababbbabba\) and \(abababbb\)
+ Notation

If for some reason we wish to modify the concept of closure to refer to only the concatenation of some *non-zero* strings from a set $S$, we use the notation $^+$ instead of $^*$

**Example**

If $\Sigma = \{x\}$, then $\Sigma^+ = \{x \, xx \, xxx \, \ldots\}$

- This is often referred to as *positive closure* (“one-or-more”)
- If $S$ is a language which contains $\lambda$, then $S^+ = S^*$
- If $S$ is a language which doesn’t contain $\lambda$, then $S^+ = S^* - \{\lambda\}$
Double Closure

Given $S^*$, apply its closure: $(S^*)^*$

- If $S$ is not $\emptyset$ or $\{\lambda\}$, then $S^*$ is infinite
- We will be taking the closure of an infinite set
- Arbitrary concatenation of the alphabet, applied twice

Proving $S^* = S^{**}$ (by construction).

$S = \{a \ b\}$

$s = aababaaaaaba$  \hspace{4cm} [arbitrary string]

$s = (aaba)(baaa)(aaba)$  \hspace{1cm} [constructed from $S^*$]

$s = [(a)(a)(b)(a)][(b)(a)(a)(a)][(a)(a)(b)(a)]$  \hspace{1cm} [constructed from $S^{**}$]

$s = (a)(a)(b)(a)(b)(a)(a)(a)(a)(a)(b)(a)$  \hspace{1cm} [converted from $S^{**}$ to $S^*$]

$S^{**} \subset S^*$  \hspace{1cm} $[\forall e \in S^{**}, e \in S^*]$

$S^* \subset S^{**}$  \hspace{1cm} $[\forall e \in S^*, e \in S^{**}]$

$S^* = S^{**}$

□
Homework 1a

1. Consider the language $S^*$, where $S = \{aa \ b\}$. How many words does this language have of length 4? of length 5? of length 6? What can be said in general?

2. Consider the language $S^*$, where $S = \{aa \ aba \ baa\}$. Show that the words $aabaa$, $baaabaa$, and $baaaaababaaaaa$ are all in this language. Can any word in this language be interpreted as a string of elements from $S$ in two different ways? Can any word in this language have an odd total number of $a$’s?

3. Prove that for all sets $S$,
   
   1. $(S^+)^* = (S^*)^*$
   2. $(S^+)^+ = S^+$
   3. Is $(S^*)^+ = (S^+)^*$ for all sets $S$?