CS 205
Notes on Strings of Symbols

Magnus M. Halldorsson   Ann Yasuhara

September 6, 1988

Let Σ be a finite, non-empty set of symbols.

1. Definition of Σ^*

   Intuitive idea: We want to define the set which will contain all possible strings of symbols from Σ. The notion of string itself is also being defined - intuitively it is just writing various members of Σ next to each other (but only finitely many). This may include writing nothing at all, which is known as the *empty* string and denoted by λ.

   For example, if $\Sigma = \{a, b, c\}$ some of the strings on Σ are λ, a, b, c, aa, ab, ac, ba, bb, bc, ..., aaabcbbacca, ... All of these are to be members of Σ^*.

   Formal inductive definition (also see text pp.162-3):

   $\lambda \in \Sigma^*$ and,
   if $\sigma \in \Sigma$ and $x \in \Sigma^*$ then $\sigma x \in \Sigma^*$
   further, Σ^* is the intersection of all sets satisfying these properties.

   Example: $\Sigma = \{a, b, c\}$

   $\lambda \in \Sigma^*$.
   Since $a \in \Sigma, a\lambda = a \in \Sigma^*$.
   Since $b \in \Sigma$ and $a \in \Sigma^*, ba \in \Sigma^*$.
   Since $a \in \Sigma$ and $ba \in \Sigma^*, aba \in \Sigma^*$.
   Since $c \in \Sigma$ and $aba \in \Sigma^*, caba \in \Sigma^*$.

2. Equality

   Intuitive idea: We'd like to know what it means to say that two strings x and y in Σ^* are equal. Since we are dealing just with symbols, it seems that two strings should be equal if they look exactly alike, symbol by symbol, reading, say, from left to right. Example, for $\Sigma = \{a, b, c\}$

does $abca$ equal $abca$?
does $abca$ equal $abba$?

Formal definition, based on inductive definition of Σ^*.

If $x = \lambda$ and $y = \lambda$ then $x = y$.
If $x = \sigma x\prime$ and $y = \sigma\prime y\prime$, for $\sigma, \sigma\prime \in \Sigma$ and $x\prime, y\prime \in \Sigma^*$,
then $x = y$ iff $\sigma = \sigma\prime$ and $x\prime = y\prime$.

Example:

(a) $x = abca$ and $y = abca$
In the definition, $\sigma = a$ and $x\prime = bca$ and $\sigma\prime = a$ and $y\prime = bca$.
So $x = y$ iff $a = a$ and $bca = bca$.
Now, since a does equal a, we have to apply the definition to bca and bca to see if they are equal, etc.

(b) $x = abca$ and $y = abba$
Begin as in example i) but $y\prime = bba$.
So $x = y$ iff $a = a$ and $bca = bba$. Since $a = a$, must check to see if $bca = bba$.
So $bca = bba$ iff $b = b$ and $ca = ba$. Since $b = b$, must check to see if $ca = ba$.
So $ca = ba$ iff $c = b$ and $a = a$.
But $c \neq b$, so $ca \neq ba$, hence $bca \neq bba$ hence $abca \neq abba$.

3. Concatenation

The most basic function associated with Σ^* is called *concatenation.* Intuitively it just means take two strings from Σ^* and from them obtain one string by just writing them next to each other.

Example:

For $\Sigma = \{a, b, c\}$, $cbab \in \Sigma^*$ and $aba \in \Sigma^*$, then $cbababa$ is the result of concatenating $cbcb$ with aba and certainly $cbababa \in \Sigma^*$. Notice that concatenation in the other order, aba with $cbab$ is $abacbab$ and is **not** equal to $cbababa$.

Formal inductive definition (follows the inductive definition of Σ^*).

Define $\textsc{Concat} : \Sigma^* \times \Sigma^* \to \Sigma^*$ by, for any $y \in \Sigma^*$,

$$\textsc{Concat}(\lambda, y) = y,$$

and, for $\sigma \in \Sigma$ and $x \in \Sigma^*$

$$\textsc{Concat}(\sigma x, y) = \sigma\textsc{Concat}(x, y).$$

Example: For $\Sigma = \{a, b, c\}$

$$\begin{aligned}
\textsc{Concat}(\lambda, aba) &= aba \\
\textsc{Concat}(cbab, aba) &= c\textsc{Concat}(bab, aba) \\
&= cb\textsc{Concat}(ab, aba) \\
&= cba\textsc{Concat}(b, aba) \\
&= cbab\textsc{Concat}(\lambda, aba) \\
&= cbababa
\end{aligned}$$

4. Length

   Another basic function on Σ^* is the **length**, call it $\mathcal{L}$. Intuitively we want a function that takes a string from Σ^* and tells exactly how many symbols occur in it, including duplications. Example for $\Sigma = \{a, b, c\}$, it should be clear that $\mathcal{L}$ says:
   $\mathcal{L}(\lambda) = 0$, $\mathcal{L}(aa) = \mathcal{L}(ab) = 2$, $\mathcal{L}(baabccbbbaac) = 12$.

   Formal inductive definition of $\mathcal{L} : \Sigma^* \rightarrow \mathbf{N}$.

   $\mathcal{L}(\lambda) = 0$, and
   $\mathcal{L}(\sigma x) = 1 + \mathcal{L}(x)$, for $\sigma \in \Sigma$ and $x \in \Sigma^*$.

   Example:

$$\begin{aligned}
\mathcal{L}(cbab) &= 1 + \mathcal{L}(bab) = 1 + 1 + \mathcal{L}(ab) = 1 + 1 + 1 + \mathcal{L}(b) \\
&= 1 + 1 + 1 + 1 + \mathcal{L}(\lambda) = 1 + 1 + 1 + 1 + 0 = 4
\end{aligned}$$

5. $\mathcal{L}$ and $\textsc{Concat}$ in combo

   Now we would like to see how $\mathcal{L}$ works with respet to $\textsc{Concat}$. Intuitively, if we know the length of two strings and concatenate the two strings the length of the resulting string ought to be the sum of the lengths of the two given strings.
   Example for $\Sigma = \{a, b, c\}$:

   $\mathcal{L}(cbab) = 4$ and $\mathcal{L}(aba) = 3$ and $\mathcal{L}(cbababa) = 7$,
   so $\mathcal{L}(\textsc{Concat}(cbab, aba)) = \mathcal{L}(cbab) + \mathcal{L}(aba)$.

   The following theorem shows that this is true in general.

   Theorem 1 *: For any Σ and any $x, y \in \Sigma^*$,*

$$\mathcal{L}(\textsc{Concat}(x, y)) = \mathcal{L}(x) + \mathcal{L}(y)$$

Proof: Follow the inductive definition of Σ^* and use the definitions of CONCAT and $\mathcal{L}$.

$$\mathcal{L}(\text{CONCAT}(\lambda, y)) = \mathcal{L}(y) = 0 + \mathcal{L}(y) = \mathcal{L}(\lambda) + \mathcal{L}(y)$$

Induction hypothesis: Assume $\mathcal{L}(\text{CONCAT}(x, y)) = \mathcal{L}(x) + \mathcal{L}(y)$.

$$\begin{array}{lcll}
\mathcal{L}(\text{CONCAT}(\sigma x, y)) & = & \mathcal{L}(\sigma\text{CONCAT}(x, y)) & \text{by def. of Concat} \\
& = & 1 + \mathcal{L}(\text{CONCAT}(x, y)) & \text{by def. of } \mathcal{L} \\
& = & 1 + \mathcal{L}(x) + \mathcal{L}(y) & \text{by Ind. Hypoth.} \\
& = & \mathcal{L}(\sigma x) + \mathcal{L}(y) & \text{by def of } \mathcal{L}
\end{array}$$

Example:

$$\begin{array}{lcl}
\mathcal{L}(\text{CONCAT}(cbab, aba)) & = & \mathcal{L}(c\text{CONCAT}(bab, aba)) \\
& = & 1 + \mathcal{L}(\text{CONCAT}(bab, aba)) \\
& = & 1 + \mathcal{L}(b\text{CONCAT}(ab, aba)) \\
& = & 1 + 1 + \mathcal{L}(\text{CONCAT}(ab, aba)) \\
& = & 1 + 1 + \mathcal{L}(a\text{CONCAT}(b, aba)) \\
& = & 1 + 1 + 1 + \mathcal{L}(\text{CONCAT}(b, aba) \\
& = & 1 + 1 + 1 + \mathcal{L}(b\text{CONCAT}(\lambda, aba)) \\
& = & 1 + 1 + 1 + 1 + \mathcal{L}(\text{CONCAT}(\lambda, aba)) \\
& = & 1 + 1 + 1 + 1 + \mathcal{L}(aba) \\
& = & \mathcal{L}(cbab) + \mathcal{L}(aba)
\end{array}$$

Exercise #1: Let $\Sigma = \{a, b, c\}$. Give a careful inductive definition of a function *Pong* : $\Sigma^* \to \mathbf{N}$, such that, intuitively, for any $x \in \Sigma^*$, Pong(x) says exactly how many occurrences of a there are in x. For example, we want $Pong(abacca) = 3$. Then, using your definition, prove carefully that

$$Pong(\text{CONCAT}(x, y)) = Pong(x) + Pong(y)$$

6. Head and Tail

Two further well-known functions on Σ^* are HEAD and TAIL. The idea is that HEAD should pick off the left-most symbol of a string and TAIL gives the string with its HEAD removed.

HEAD : $\Sigma^* \to \Sigma \cup \{\lambda\}$ is defined ***intuitively*** by

if $x = \lambda$ then $\text{HEAD}(x) = \lambda$

else $\textsc{Head}(x)$ = leftmost symbol of x

and is defined formally by *induction*:

$\textsc{Head}(\lambda) = \lambda$, and
$\textsc{Head}(\sigma x) = \sigma$.

Similarily,

$\textsc{Tail} : \Sigma^* \to \Sigma^*$ is defined *intuitively* by

```
if $x = \lambda$ then $\textsc{Tail}(x) = \lambda$
             else $\textsc{Tail}(x) =$ the string obtained from x
                                       by erasing the leftmost symbol
```

and is defined formally by *induction*:

$\textsc{Tail}(\lambda) = \lambda$, and
$\textsc{Tail}(\sigma x) = x$.

Examples: Let $\Sigma = \{a, b, c\}$.

$\textsc{Head}(a) = a$ and $\textsc{Tail}(a) = \lambda$
$\textsc{Head}(caba) = c$ and $\textsc{Tail}(caba) = aba$.

Exercise: Show by using the definition that if $x \in \Sigma^*$ then

$$\mathcal{L}(\textsc{Tail}(x)) = \begin{cases} 0 & \text{if } x = \lambda \\ \mathcal{L}(x) - 1 & \text{if } x \neq \lambda \end{cases}$$

Exercise: Show by using the definitions that

$$\textsc{Concat}(\textsc{Head}(x), \textsc{Tail}(x)) = x$$

Refer back to the definition of CONCAT on p.2. If we use the HEAD and TAIL functions we can rewrite the definition by

$$\textsc{Concat}(\lambda, y) = y$$

and

$$\textsc{Concat}(x, y) = \textsc{Head}(x)\textsc{Concat}(\textsc{Tail}(x), y)$$

Be sure you understand what the latter right hand side above means:

Take the tail of x and concatenate it with y and take the head of x and write it next to, on the left of that concatenation.

We can use these ideas to write a recursive program for concat if our programming language include HEAD and TAIL and write a symbol on the left of a string. We call the function our program computes C and then prove $C = $ CONCAT. C takes pairs of string in Σ^* as input and returns a string in Σ^*.

```
C($x$,$y$) := if $x = \lambda$ then $y$
                          else HEAD($x$)C(TAIL($x$),$y$)
```

Note that the **if-then-else** above is not a statement, as in most programming languages, but is an operator, or a polymorphic function. The **if-then-else** operator is a function

$$If : \mathbf{B} \times \tau \times \tau \to \tau$$

where **B** is the set of Boolean values {False,True} and τ is any type that can be assigned. As before, the condition is a predicate that evaluates to either True or False, in which case either the 'Then' value is returned or the 'Else' value.

Example:

F(x) := if $x = \lambda$ then 0 else $1 + F(tail(x))$
$F(\lambda) = 0$, $F(ab) = 1 + F(b) = 1 + 1 = 2$
If fact, F is the length function $\mathcal{L}$.

Exercise: Prove $C = $ CONCAT by induction on x, keeping y fixed.

In practice we often abbreviate CONCAT(x,y) by $x \cdot y$, or even more simply xy. (*Warning*: Always remember the context - here x and y are *strings*, **not** numbers, so $x \cdot y$ does **not** mean multiplication). We will usually be using xy for CONCAT(x, y) except when we are specifically studying properties of CONCAT.