Numerical Representation of Strings

We will now look at several **primitive recursive functions on strings**.

These functions will be useful for our further discussion of calculation on strings.

It will be helpful to remember the functions g and h:

$g(0, n, x) = x$

$g(m + 1, n, x) = Q^+(g(m, n, x), n)$,

then $g(m, n, x) = u_m$.

$h(m, n, x) = R^+(g(m, n, x), n)$,

then $i_m = h(m, n, x)$, $m = 0, \ldots, k$.

Numerical Representation of Strings

1.) f(u) = |u|

This "length" function is defined on A* and yields a natural number.

How can we compute f on the **number** associated with A*?

For each t, the number $\sum_{j=0}^{t} n^j$ has the base n representation $s_1^{[t+1]}$.

(The expression $s^{[m]}$ stands for the string that consists of m times the symbol s.)

Numerical Representation of Strings

For example, given the alphabet
$A = \{s_1, s_2, s_3\}$ and t = 2:

The string representation of $\sum_{j=0}^{t} 3^j = s_1s_1s_1$.

This number is the smallest number whose base 3 representation contains t + 1 symbols.

If we subtract 1 from the string $s_1s_1s_1$, it becomes s_3s_3.

So the length |u| can be defined as follows:

$|u| = \min_{t \le u} (\sum_{j=0}^{t} n^j > u)$.

Obviously, this function is primitive recursive.

Numerical Representation of Strings

2.) g(u, v) = $\text{CONCAT}_n(u, v)$

This function is primitive recursive because it is defined by the following equation:

$\text{CONCAT}_n(u, v) = u \cdot n^{|v|} + v$

Example: Alphabet A = {1, 2, 3, 4, 5, 6, 7, 8, 9, X}:

$\text{CONCAT}_{10}(13, 478) = 13 \cdot 1000 + 478 = 13478$

Numerical Representation of Strings

3.) $\text{CONCAT}_n^{(m)}(u_1, \ldots, u_m)$

This function is primitive recursive for each m, n ≥ 1.

It follows at once from the previous example using composition.

Example:

Alphabet A = {1, 2, 3, 4, 5, 6, 7, 8, 9, X}, m = 3:

$\text{CONCAT}_n(u, v, w) = u \cdot n^{|v| + |w|} + v \cdot n^{|w|} + w$

$\text{CONCAT}_{10}(13, 478, 9) = 13 \cdot 10000 + 478 \cdot 10 + 9 = 134789$

Numerical Representation of Strings

4.) $\text{RTEND}_n(w) = h(0, n, w)$

Remember:

$i_m = h(m, n, x)$, $m = 0, \ldots, k$.

RTEND_n gives the rightmost symbol of a given word.

We know that h is primitive recursive, so RTEND_n is also primitive recursive.

Numerical Representation of Strings

5.) $LTEND_n(w) = h(|w| \dot{-} 1, n, w)$

Corresponding to $RTEND_n$, $LTEND_n$ gives the leftmost symbol of a given word.

Numerical Representation of Strings

6.) $RTRUNC_n(w) = g(1, n, w)$

Remember:

$g(m, n, x) = u_m$

$u_0 = i_k \cdot n^k + i_{k-1} \cdot n^{k-1} + \ldots + i_1 \cdot n^1 + i_0$

$u_1 = i_k \cdot n^{k-1} + i_{k-1} \cdot n^{k-2} + \ldots + i_1$

$RTRUNC_n$ gives the result of removing the rightmost symbol from a given nonempty string.

When we can omit the reference to the base n, we often write w^- for $RTRUNC_n(w)$.

Note that $0^- = 0$.

Numerical Representation of Strings

7.) $LTRUNC_n(w) = w \dot{-} (LTEND_n(w) \cdot n^{|w| \dot{-} 1})$

$LTRUNC_n$ gives the result of removing the leftmost symbol from a given nonempty string.

Example: Alphabet A = {1, 2, 3, 4, 5, 6, 7, 8, 9, X}:

$LTRUNC_{10}(3478) = 3478 - 3 \cdot 1000 = 478$.

Numerical Representation of Strings

We will use these newly introduced primitive recursive functions to prove the computability of a pair of functions that can be used in **changing base**.

Let $1 \leq n < l$.

Let $A \subset \tilde{A}$, where A is an alphabet of n symbols and $\tilde{A}$ is an alphabet of l symbols.

So whenever a string belongs to A^*, it also belongs to $\tilde{A}^*$.

Numerical Representation of Strings

For any $x \in N$, let w be the word in A^* that represents x in base n.

Then we write $UPCHANGE_{n,l}(x)$ for the number which w represents in base l.

Examples:

$UPCHANGE_{2,6}(5) = 13$

The representation of 5 in **base 2** is s_2s_1. In **base 6**, s_2s_1 represents the number 13.

$UPCHANGE_{1,5}(3) = 31$

The representation of 3 in **base 1** is $s_1s_1s_1$. In **base 5**, $s_1s_1s_1$ represents the number 31.