8.4

Structural Induction

P. Danziger

1 Recursive Definitions

Sequences of numbers are not the only type of objects that can be defined recursively, many objects may be defined in this way. One particularly useful application of recursion is to define certain certain sets (classes or families) of strings.
When we are defining a set of strings we usually start with an *alphabet* Σ. The members of Σ are called *characters*, and all members of our set will be strings of characters from Σ. The set of all strings is denoted Σ^* and we will denote the empty string by ϵ
Generally recursive definitions have three parts, a set of *base cases* which are given strings that are in the set and a set of (recursive) rules which tell us how to make new strings from old. Finally there is a rule which closes the recursion.

Example 1

1. We define a *Fully Parenthesised Arithmetic Expression* ($FPAE$) as follows over the alphabet $\Sigma = \{+, \times, -, (,), a, b, c\}$:

   Base Case $a \in FPAE,\ b \in FPAE, c \in FPAE$

   Recursion Given two strings $x, y \in FPAE$ then the following are also strings in FPAE.

   (a) $(x + y)$
   (b) $(x \times y)$
   (c) $(-x)$

   Closure No other string is an FPAE other than those obtained by repeated applications of a to c.

   Thus the following strings are in $FPAE$:

   $$a, \quad b, \quad (a + b), \quad ((a + b) \times a), \quad (-((a + b) \times a))$$

   The following strings are **not** in $FPAE$:

   $$(a), \quad a \times b, \quad ab, \quad a + b, \quad (a + b, \quad a \times b)$$

2. We define *Revers Polish Notation* (RPN) as follows over the alphabet $\Gamma = \{+, \times, -, (,), a, b, c, d\}$:

   Base Case $a \in RPN,\ b \in RPN, c \in RPN, d \in RPN$

   Recursion Given two strings $x, y \in RPN$ then the following are true:

   (a) $(+xy) \in RPN$
   (b) $(\times xy) \in RPN$

(c) $(-x) \in RPN$

Closure No other string is an RPN strring other than those obtained by repeated applications of a to c.

Thus the following strings are in RPN:

$$a, \quad b, \quad (+ab), \quad (-(+ab)), \quad (\times(-(+ab))a)$$

The following strings are **not** in RPN:

$$(a), \quad +ab, \quad (a \times b), \quad (-ab), \quad (+abc)$$

2 Structural Induction

We may wish to prove theorems about recursively defined objects, but we cannot use induction (strong or weak) as defined so far, since there are no numbers associated with the objects. The solution is to use *Structural Induction*.
In structural induction we first show that the assertion holds for each base case of the recursion. (Base Case)
In the Inductive step we assume that x is an object defined by the recursion, but it is **not** one of the base cases. Now x must have been formed from one of the recursive rules, which build up x from previously defined strings. We thus take each rule in turn and show that if the ingredients have the property (this is the inductive hypothesis) then so does the result. We may then conclude that x has the property no matter which rule was used to create it.

Definition 2 *Given any alphabet Σ, a string $x \in \Sigma^*$ and a character $a \in \Sigma$ we define*

$$|x|_a = \textit{the number of times the character } a \textit{ appears in the string } x.$$

Example 3

1. **Theorem 4** *For any string $x \in FPAE$, $|x|_(= |x|_) = |x|_+ + |x|_- + |x|_\times$.*
2. **Definition 5** *Given a string $x \in \Gamma^*$ we define the weight function $w : \Gamma^* \to \mathbb{Z}$ as follows:*

$$\begin{aligned} &w(a) = w(b) = w(c) = w(d) = 1, \\ &w(+) = w(\times) = -1, \\ &w(-) = w(\epsilon) = w(\ )\ ) = w(\ (\ ) = 0 \end{aligned}$$

Theorem 6 *If $x \in RPN$ then $w(x) = 1$*

Theorem 7 *If $x \in RPN$ and $x = yz$ (ie x is the concatenation of the string y with the string z), where z is non empty, then $w(y) < 1$.*

Theorem 8 *If $y \in \Gamma^*$ and $w(y) < 1$ then there exists $z \in \Gamma^*$ such that $x = yz \in RPN$.*

Note that these theorems can be used to check syntax as the string is entered.