3.4 RECURSIVE DEFINITIONS

Functions can be defined recursively. The simplest form of recursive definition of a function f on the natural numbers specifies a basis rule

(B) the value $f(0)$

and a recursion rule

(R) how to obtain $f(n)$ from $f(n-1)$, $\forall n \geq 1$

Example 3.4.1: n-factorial $n!$

(B) $$0! = 1$$

(R) $$(n+1)! = (n+1) \cdot n!$$

However, recursive definitions often take somewhat more general forms.

Example 3.4.2: mergesort ($A[1 \ldots 2^n]$: real)

if n = 0

    return(A)

otherwise

    return(merge (m'sort(1st half), m'sort(2nd half)))

Since a sequence is defined to be a special kind of a function, some sequences can be specified recursively.

Example 3.4.3: Hanoi sequence
$0, 1, 3, 7, 15, 31, \ldots$
$h_0 = 0$
$h_n = 2h_{n-1} + 1$ for $n \geq 1$

Example 3.4.4: Fibonacci seq
$1, 1, 2, 3, 5, 8, 13, \ldots$
$f_0 = 1$
$f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$

Example 3.4.5: partial sums of sequences

$$\sum_{j=0}^{n} a_j = \begin{cases} a_0 & \text{if } n = 0 \\ \sum_{j=0}^{n-1} a_j + a_n & \text{otherwise} \end{cases}$$

Example 3.4.6: Catalan sequence
$1, 1, 2, 5, 14, 42, \ldots$
$c_0 = 1$
$c_n = c_0 c_{n-1} + c_1 c_{n-2} + \cdots + c_{n-1} c_0$ for $n \geq 1$

RECURSIVE DEFINITION of SETS

DEF: A ***recursive definition of a set*** S comprises the following:

(B) a ***basis clause*** that specifies a set of ***primitive elements***;

(R) a ***recursive clause*** that specifies how elements of the set may be constructed from elements already known to be in set S; there may be several recursive subclauses;

(E) an ***implicit exclusion clause*** that anything not in the set as a result of the basis clause or the recursive clause is not in set S.

Backus Normal Form (BNF) is an example of a context-free grammar that is useful for giving resursive definitions of sets. In W3261, you will learn that context-free languages are recognizable by pushdown automata.

Example 3.4.7: a rec. def. set of integers

(B) $7, 10 \in S$

(R) if $r \in S$ then $r + 7, r + 10 \in S$

This reminds us of the postage stamp problem.

Claim $(\forall n \geq 54)[n \in S]$

Basis: $54 = 2 \cdot 7 + 4 \cdot 10$

Ind Hyp: Assume $n = r \cdot 7 + s \cdot 10$ with $n \geq 54$.

Ind Step: Two cases.

Case 1: $r \geq 7$. Then $n+1 = (r-7) \cdot 7 + (s+5) \cdot 10$.

Case 2: $r < 7 \Rightarrow r \cdot 7 \leq 42 \Rightarrow s \geq 2$.

Then $n + 1 = (r + 3) \cdot 7 + (s - 2) \cdot 10$.

In computer science, we often use recursive definitions of sets of strings.

RECURSIVE DEFINITION of STRINGS

NOTATION: The set of all strings in the alphabet Σ is generally denoted Σ^*.

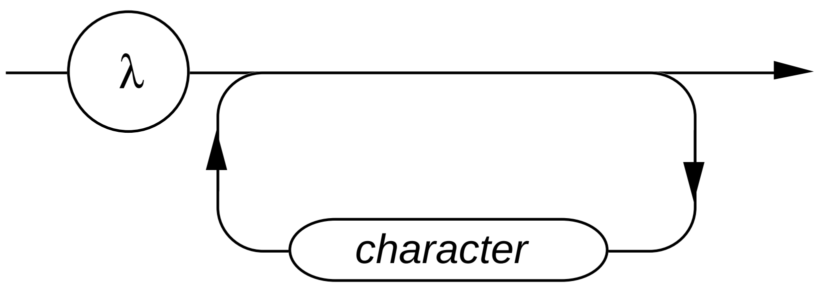
Example 3.4.8: $\{0, 1\}^*$ denotes the set of all binary strings.

DEF: ***string in an alphabet*** Σ

(B) (empty string) λ is a string;

(R) If s is a string and $b \in \Sigma$, then sb is a string.

Railroad Normal Form for strings

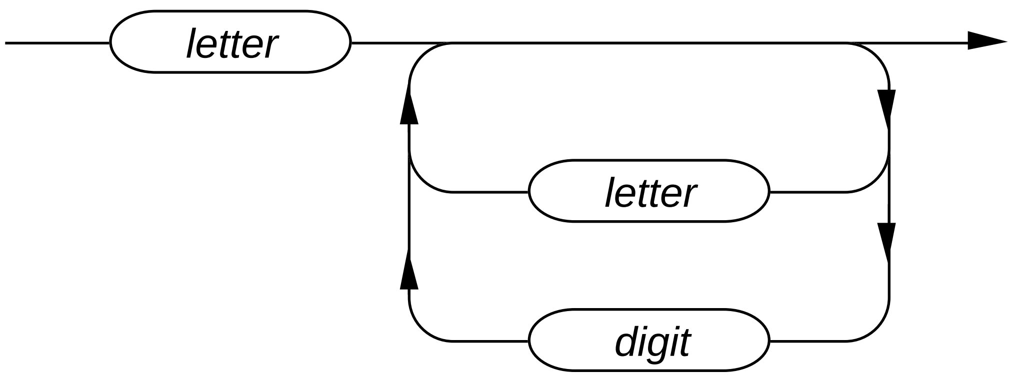


Example 3.4.9: BNF for strings

$\langle\text{string}\rangle ::= \lambda \mid \langle\text{string}\rangle\langle\text{character}\rangle$

RECURSIVE DEFINITION of IDENTIFIERS

DEF: An ***identifier*** is (for some programming languages) either
(B) a letter, or
(R) an identifier followed by a digit or a letter.



Example 3.4.10: BNF for identifiers

$$\langle \text{lowercase\_letter} \rangle ::= a \mid b \mid \cdots \mid z$$
$$\langle \text{uppercase\_letter} \rangle ::= A \mid B \mid \cdots \mid Z$$
$$\langle \text{letter} \rangle ::= \langle \text{lowercase\_letter} \rangle \mid \langle \text{uppercase\_letter} \rangle$$
$$\langle \text{digit} \rangle ::= 0 \mid 1 \mid \cdots \mid 9$$
$$\begin{aligned} \langle \text{identifier} \rangle ::= \langle \text{letter} \rangle &\mid \langle \text{identifier} \rangle \langle \text{letter} \rangle \\ &\mid \langle \text{identifier} \rangle \langle \text{digit} \rangle \end{aligned}$$

ARITHMETIC EXPRESSIONS

DEF: ***arithmetic expressions***

(B) A numeral is an arithmetic expression.

(R) If e_1 and e_2 are arithmetic expressions, then all of the following are arithmetic expressions:

$$e_1 + e_2, e_1 - e_2, e_1 * e_2, e_1/e_2, e_1 * *e_2, (e_1)$$

Example 3.4.11: Backus Normal Form

$$\begin{aligned} \langle\text{expression}\rangle ::= & \langle\text{numeral}\rangle \\ & \mid \langle\text{expression}\rangle + \langle\text{expression}\rangle \\ & \mid \langle\text{expression}\rangle - \langle\text{expression}\rangle \\ & \mid \langle\text{expression}\rangle * \langle\text{expression}\rangle \\ & \mid \langle\text{expression}\rangle / \langle\text{expression}\rangle \\ & \mid \langle\text{expression}\rangle * * \langle\text{expression}\rangle \\ & \mid (\langle\text{expression}\rangle) \end{aligned}$$

SUBCLASSES of STRINGS

Example 3.4.12: binary strings of even length
(B) $\lambda \in S$
(R) If $b \in S$, then $b00, b01, b10, b11 \in S$.

Example 3.4.13: binary strings of even length that start with 1
(B) $10, 11 \in S$
(R) If $b \in S$, then $b00, b01, b10, b11 \in S$.

DEF: A ***strict palindrome*** is a character string that is identical to its reverse. (In natural language, blanks and other punctuation are ignored, as is the distinction between upper and lower case letters.)

Able was I ere I saw Elba.

Madam, I'm Adam.
Eve.

Example 3.4.14: set of binary palindromes
(B) $\lambda, 0, 1 \in S$
(R) If $x \in S$ then $0x0, 1x1 \in S$.

LOGICAL PROPOSITIONS

DEF: ***propositional forms***

(B) p, q, r, s, t, u, v, w are propositional forms

(R) If x and y are propositional forms, then so are $\neg x, x \wedge y, x \vee y, x \rightarrow y, x \leftrightarrow y$ and (x).

Propositional forms under basis clause (B) are called ***atomic***.

Remark: Recursive definition of a set facilitates proofs by induction about properties of its elements.

Proposition 3.4.1. *Every proposition has an even number of parentheses.*

Proof: by induction on the length of the derivation of a proposition.

Basis Step. All the atomic propositions have evenly many parentheses.

Ind Step. Assume that propositions x and y have evenly many parentheses. Then so do propositions $\neg x, x \wedge y, x \vee y, x \rightarrow y, x \leftrightarrow y$ and (x).
$\Diamond$

CIRCULAR DEFINITIONS

DEF: A would-be recursive definition is ***circular*** if the sequence of iterated applications it generates fails to terminate in applications to elements of the basis set.

Example 3.4.15: a circular definition from Index and Glossary of Knuth, Vol 1.

Circular Definition, 260
see Definition, circular
Definition, circular,
see Circular definition