Formal Languages

Let Σ be an alphabet, i.e., a finite set. The set of all strings over Σ is denoted by Σ^*.

By a *(formal) language* over Σ we mean any subset of Σ^*.

For example, the empty set $\emptyset$ and the set Σ^* itself are formal languages in this sense.

Other examples of a formal languages are the sets

$L_1 = \{w \in \{0,1\}^* \,:\, w \text{ has an equal number of 0's and 1's}\}$

and

$$L_2 = \{w \in \{0,1\}^* \,:\, w \text{ begins with a 1}\}.$$

The last sets have been defined via comprehension:

$$L = \{w \in \Sigma^* \,:\, P(w)\}.$$

For computational purposes comprehension is too powerful a formalism. For instance, the problem of determining whether a string is an element of a specified language can not be solved algorithmically for arbitrary languages specified via comprehension.

We will briefly discuss other specification formalisms that are useful in the definition of (the syntax of) programming languages.

Arithmetic Expressions

An important example of a formal language are arithmetic expressions, which are typically defined by recursion:

1. Variables, integers, and real numbers are arithmetic expressions.
2. If E_1 and E_2 are arithmetic expressions, then the following are also arithmetic expressions:
   (a) $(E_1 + E_2)$
   (b) $(E_1 - E_2)$
   (c) $(E_1 \times E_2)$
   (d) (E_1/E_2)
   (e) $(-E_1)$

For example,

$$\begin{array}{l} x \\ 5 \\ (x \times 5) \\ (-(x \times 5) + y) \end{array}$$

are arithmetic expressions according to this definition.

The above definition is hierarchical in that it presupposes definitions of formal languages representing variables, integers, and (certain) real numbers.

Balanced Parentheses

If we focus on the parenthesis structure of arithmetic expressions we obtain a simpler formal language S that can also be defined recursively:

1. The empty string λ is an element of S.
2. If s_1 and s_2 are elements of S, then the following are also elements of S:
   (a) $(s_1 s_2)$
   (b) (s_1)

The set S is formal language as it is a subset of Σ^* where Σ is the two-element set $\{(,)\}$.

The elements of S are examples of strings of *balanced parentheses* (though there are also balanced strings that are not elements of S).

Arithmetic Expressions (cont.)

Arithmetic expressions are usually written in a simplified form by dropping redundant parentheses.

Let A denote the set of arithmetic expressions as defined above. By A' we denote the set of all expressions that can be obtained from elements of A by possibly omitting outer parentheses from an expression and/or parentheses that are implied by:

1. Precedence (highest to lowest): unary minus; multiplication and division; addition and subtraction
2. Associativity (say, to the left)

For example,

$$-5 + x + (10 \times y)$$

and

$$-5 + x + 10 \times y$$

are elements of A' as each can be obtained from the fully parenthesized expression

$$(((-5) + x) + (10 \times y)).$$

Note that the omission of (matching pairs) of parentheses is optional. If one wishes one can keep all parentheses and consequently A is a subset of A'.

Balanced Parentheses (cont.)

The set of strings S corresponds to the set of expressions A, but not to A'.

For example, the expression $(x + y) \times (x - y)$ contains the string of parentheses ()(), which is not an element of S.

Consider the set S_1 defined by:

1. The empty string λ is an element of S_1.
2. If x and y are elements of S_1, then the following are also elements of S_1:
   (a) xy
   (b) (x)

Do the strings of parentheses in S_1 correspond to the arithmetic expressions of A'?

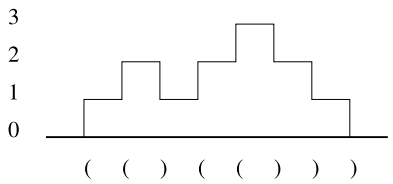
What about the set S_2, defined by:

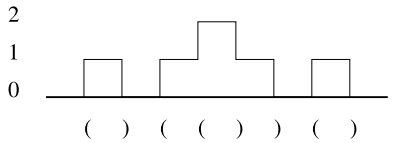
1. The empty string λ is an element of S_2.
2. If x and y are elements of S_2, then $(x)y$ is also an element of S_2.

Profiles

By the *profile* of a string of parentheses we mean the running total, as we move from left to right, of the number of left parentheses minus the number of right parentheses.

Profiles can be represented graphically as shown below for the strings (()(())) and ()(())(), respectively.





Profile-Balanced Strings

We say that a string x is *profile-balanced* if its profile begins and ends at 0 and never turns negative.

Let S_3 be the set of all profile-balanced strings of parentheses.

Lemma

$$S_1 = S_2 = S_3.$$

Proof

We show that S_3 is a subset of both S_1 and S_2, and vice versa.

1. $S_2 \subseteq S_3$

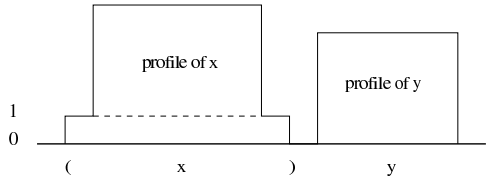
   We have to show that every string w in S_2 is profile-balanced, i.e., an element of S_3. This assertion can be proved by induction on the number of applications of recursion needed to produce w according to the definition of S_2.

   Induction basis. The only string $w \in S_2$ that can be obtained without any application of the recursive rule is the empty string λ, which is indeed profile-balanced.

   Next suppose $n > 0$. We assume, as *induction hypothesis*, that any string in S_2 that can be obtained by fewer than n applications of the recursive rule is profile-balanced. We need to show that each string in S_2 that can be obtained by n applications of the recursive rule is profile-balanced. Let w be any arbitrary such string.

   Since w requires at least one application of recursion, it must be a string of the form $(x)y$, where x and y require fewer than n applications of the recursive rule. By the induction hypothesis, x and y are profile-balanced.

   An inspection of the profile of w indicates that w is also profile-balanced, as shown below.



2. $S_1 \subseteq S_3$

   This part can be proved in a similar way as the preceding part.

3. $S_3 \subseteq S_2$

   We have to show that every profile-balanced string w is an element of S_2. We prove this assertion by induction on the length of w.

Induction basis. If $|w| = 0$, then w must be the empty string, which is an element of S_2 by the definition of S_2.

Next suppose $n > 0$. We assume, as *induction hypothesis*, that any profile-balanced string of length less than n is an element of S_2. We need to show that each profile-balanced string of length n is an element of S_2. Let w be be an arbitrary such string.

We first break the string w into substrings x and y, such that $w = xy$ and x is the shortest non-empty profile-balanced prefix of w.

For example, if $x = ()(())$, then $y = ()$ and $z = (())$.

Note that x may be identical to w, in which case $y = \lambda$. The strings w and x both begin with a left parenthesis and end with a right parenthesis. Thus there exists a string v such that $x = (v)$.

Since v and y are both profile-balanced strings strictly shorter than w, we may apply the induction hypothesis to infer that they are elements of S_2.

Since w is identical to $(v)y$, we conclude that it is an element of S_2.

4. $S_3 \subseteq S_1$

   This part can be proved using similar ideas as above.

Summary

We have characterized strings of (matching pairs of) parentheses in different, but equivalent ways.

Let S' be the set S_1 (or S_2 or S_3). The elements of S' are called *balanced strings of parentheses*.

These strings (i.e., elements of S') correspond to simplified arithmetic expressions (elements of A').

In particular, any string of parentheses obtained from an expression in the set A' is balanced. And conversely, every balanced string can be expanded to an arithmetic expression.

We will not give a formal proof of this correspondence, but the key observation is that in a redundant pair of parentheses the left and right parenthesis are at the same level in the profile of the string. Their omission will change the profile but keep it in balance.