# Evaluation of Scheme Expressions

Here are the steps for evaluating a Scheme expression.

1. Is the expression a "simple" expression? That is, does it not contain any parentheses? In this case, the value should be immediately available. If the expression is a numeral, its value is predefined in Scheme as the corresponding numeric value. If the expression is a word, or *identifier*, and the word has been *bound* to some value, that's the value returned. Otherwise we have an error: "unbound name" or "undefined name". Certain identifiers are prebound in Scheme, for instance, names of builtin functions.

2. Since the expression has parentheses, we can examine the first item in the parentheses. Is it a special function (define, if, cond, quote, and, or, let, lambda)?

   If so, handle the special function specially.

3. Is the first thing in the parentheses a simple expression? In that case, it must be the name of a function (if not, we have an error: "... not a function"). Suppose the function takes k inputs.

   a. There must be k remaining expressions within the parentheses, otherwise we have an error: "too many inputs" or "not enough inputs". Evaluate these expressions.

   b. Apply the function to the resulting values. This means *binding* the placeholder names to corresponding input values, and then evaluating the body of the function. The effect is that of substituting the input values for the place-holder names throughout the body of the function and then evaluating the body.

4. Otherwise, the first thing in the expression is itself a complicated expression. Evaluate it; the result must be a function. Then count and evaluate the remaining elements of the expression as in step 3.