PAGE 11

ML has operators that apply to Boolean values. These operators are similar to the operators AND, OR, and NOT found in Pascal (&&, ||, and ! in C), but with some important differences.

andalso, orelse

NOT has the usual Pascal meaning, but must be lowercase, as `not`. AND and OR of Pascal are replaced in ML by `andalso` and `orelse`. Like C but not Pascal, the right operand of `andalso` and `orelse` is evaluated only when the left operand does not determine a value (i.e., only when the left operand of `andalso` is true, or the left operand of `orelse` is false).

The precedence of the logical operators in ML agrees with C; `not` is highest, then `andalso`, and then `orelse`. Operators `andalso` and `orelse` are of lower precedence than the comparison or arithmetic operators, but `not` is higher than any of these.

PAGE 13

e)* `3>4 orelse 5<6 andalso not (7<>8)`

PAGE 138

```
datatype ('label) tree =
    Node of 'label * 'label tree list
```

datatype 'a tree
*con Node : 'a * 'a tree list → 'a tree*

Fig. 13.6. Datatype for a general rooted tree.

PAGE 102

```
fun map(F,nil) = nil
|   map(F,x::xs) = F(x)::map(F,xs);
```

*val map = fn : ('a → 'b) * 'a list → 'b list*

PAGE 280

1.1e: *val it = false : bool.* The `andalso` groups its operands before `orelse`. Thus, the whole expression is grouped

```
3>4 orelse (5<6 andalso (not (7<>8)))
```

We begin by evaluating the left operand of the outermost operator, `orelse`. That operand `3>4` is false, so we must evaluate the right operand,

```
5<6 andalso (not (7<>8))
```

The left operand `5<6` is true, so we consider the right operand: `not (7<>8)`. Since `7<>8` is true, `not (7<>8)` is false, and the result of the `andalso` is false. Thus, both operands of the `orelse` are false and the entire expression is false.