Ocaml expressions and types

Jeffrey Heinz
heinz@udel.edu

University of Delaware

Feb. 8, 2012
As you learn OCaml, you will initially spend a lot of time getting the OCaml type checker to accept your programs. Be patient, you will eventually find that the type checker is one of your best friends. It will help you figure out where errors may be lurking in your programs. If you make a change, the type checker will help track down the parts of your program that are affected. In the meantime, here are some rules about type checking.

1. Every expression has exactly one type.

2. When an expression is evaluated, one of four things may happen:
   2.1 it may evaluate to a value of the same type as the expression,
   2.2 it may raise an exception (well discuss exceptions in Chapter 9),
   2.3 it may not terminate,
   2.4 it may exit.

One of the important points here is that there are no “pure commands.” Even assignments produce a value although the value has the trivial `unit` type.
## Basic Types

<table>
<thead>
<tr>
<th>type</th>
<th>what it is</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>integer</td>
<td>0, 5, 42, -17, 0x00ff</td>
</tr>
<tr>
<td>float</td>
<td>floating point number</td>
<td>0.0, -27.33, 1e-10</td>
</tr>
<tr>
<td>bool</td>
<td>truth value</td>
<td>true, false</td>
</tr>
<tr>
<td>char</td>
<td>character</td>
<td>‘a’, ‘\ 101’, ‘\ n’</td>
</tr>
<tr>
<td>string</td>
<td>character string</td>
<td>&quot;&quot;, &quot;lo, how a rose&quot;</td>
</tr>
<tr>
<td>unit</td>
<td>the unit type</td>
<td>()</td>
</tr>
<tr>
<td>&lt;fun&gt;</td>
<td>the function type</td>
<td>(+), (=), fun x -&gt; x+1</td>
</tr>
</tbody>
</table>

**Table:** Types in Ocaml
## Less Basic Types

<table>
<thead>
<tr>
<th>type</th>
<th>what it is</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘a list</td>
<td>list of things of type ‘a</td>
<td>[1;2], [‘a‘;‘A‘;‘b‘]</td>
</tr>
<tr>
<td>‘a * ‘b</td>
<td>pairing (tuple) of types</td>
<td>(‘X‘,‘Y‘), (&quot;hello&quot;,5)</td>
</tr>
<tr>
<td>‘a and ‘b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** More Types in Ocaml