Oz Programming: Basic syntax cheat sheets

This document is a non-exhaustive reminder of the syntax of the Oz programming language. It is always possible to improve it and your help is therefore welcome – just submit an issue on the link below and we will modify the document. Source code and the latest version of the pdf can be found at the following address: [https://github.com/some-github/a-wonderful-link](https://github.com/some-github/a-wonderful-link)

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic statements</strong></td>
<td></td>
</tr>
<tr>
<td>Var = ...</td>
<td>variable assignment (only single-assignment)</td>
</tr>
<tr>
<td>declare Var</td>
<td>global declaration of Var</td>
</tr>
<tr>
<td>local Var in ... end</td>
<td>local declaration</td>
</tr>
<tr>
<td>fun {FunName Arg1 ... ArgN} ... end</td>
<td>function definition</td>
</tr>
<tr>
<td>proc {ProcName Arg1 ... ArgN} ... end</td>
<td>procedure definition</td>
</tr>
<tr>
<td>if Condition_1 then ... elseif Condition_2 then ... else ... end</td>
<td>if ...else if ...else ...</td>
</tr>
<tr>
<td>case Var of Pattern_1 then ... [] Pattern_2 then ... else ... end</td>
<td>pattern matching</td>
</tr>
<tr>
<td><strong>Booleans expressions and operators</strong></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false value</td>
</tr>
<tr>
<td>true</td>
<td>true value</td>
</tr>
<tr>
<td>andthen</td>
<td>logical AND</td>
</tr>
</tbody>
</table>

Florian Felten
orelse  
logical OR

==  
logical equality

\=  
logical inequality (be careful it is a backslash)

{Not [Your Expression]}  
logical NOT

## Comparison operators

<  
less than

<=  
less than or equal to (because \(<\)= is an arrow)

>  
greater than

>=  
greater than or equal to

## Arithmetic operators

+  
addition

-  
subtraction

*  
multiplication

/  
division (for floating point numbers)

div  
division (for integers)

mod  
modulo

\{Pow A B\}  
\(A^B\)

\{Abs A\}  
absolute value of A

E = "1  
unary negation (because - is an operator)

## Data structures

S = "A string"  
string declaration

A = hELLO  
atom declaration (with lowercase first letter)

A = 'An atom'  
same (with uppercase first letter and space)

X = label(feature1:Field1 ...
    featureN:FieldN)  
record structure (features and label are atoms)
R. feature
access to the record’s fields

T = 1#2#3
common operator (T = ’#’(1:1 2:2 3:3))

L = ’|’(1:1 2: ’|’(1:2 2: nil))
list structure

L = 1|2|nil
a syntactic sugar to declare a list

L = [1 2]
another syntactic sugar for list declaration

Explicit state

X = {NewCell Y}
cell creation (multiple assignment variable)

@X
access to the cell’s current content

X := Z
changes the content of the cell

for X in L do
...
end

foreach loop (used with lists)

for X in 1..N do
...
end
traditional for loop

Object-oriented programming

class AClass

attr a1 ... an
meth init(Arg) ... end
meth m1 ... end
meth mn(Arg) ... end
end

class definition

X = {New AClass init(’arg’)}
{X m1}
object creation and use

Exceptions handling

raise E end
throws an exception E

try ... catch X then ... end
catches a raised exception

Concurrent programming

thread ... end
thread creation