

Programming Language Concepts, CS2104 (12th Nov 2007)

Tutorial 10 (Please prepare in advance.)

Exercise 1. (WaitOr and WaitSome) One problem that occurs quite often in practice is to wait until at least one out of two variables becomes bound. For this purpose, Oz provides the procedure `{WaitOr X Y}`. It suspends until `X` or `Y` becomes bound. Write an Oz procedure able to simulate `{WaitOr ...}`.

For instance, Mozart provides the procedure `{Record.waitOr R ?LI}` which blocks until at least one field of `R` is determined. It returns the feature `LI` of a determined field and it raises an exception if `R` is not a proper record, that is, if `R` is a literal. For example,

```
{Browse {Record.waitOr a(_ b:1)}} displays b
{Browse {Record.waitOr a(2 b:_)}} displays 1
{Browse {Record.waitOr a(_ b:_)}} blocks.
```

Moreover, write a procedure `{WaitSome Xs}` that suspends the executing thread until at least one variable from the list `Xs` becomes bound.

Exercise 2. (cells – reference and value) Explain what and why the following Oz program will display:

```
declare
X = {NewCell 0}
{Assign X 5}
Y = X
{Assign Y 10}
{Browse {Access X} == 10}
{Browse X == Y}
Z = {NewCell 10}
{Browse Z == Y}
{Browse @X == @Y}
```

Exercise 3. (arrays) Write an Oz function which takes `N` as the input and returns the array `<1!, 2!, 3!, ..., N!>`, where `N!` means ‘factorial of `N`’ (that is, $N! = 1 * 2 * \dots * N$).

Exercise 4. (call by value and call by reference) Explain what and why the following Oz program will display:

```
declare
proc {F A}
  A:=@A+1
  A:=@A*@A
end
proc {G A}
  E={NewCell A}
in
  E:=@E+1
  E:=@E*@E
```

```

end
local
  C={NewCell 0}
  D={NewCell 1}
in
  C:=5
  D:=6
  {Browse @C#@D}
  {F C}
  {G @D}
  {Browse @C#@D}
end

```

Exercise 5. Consider the following Oz procedures that can be used to capture relationships between people:

```

proc {Male X}
  choice X=richard | X=john | ... end
end
proc {Female X}
  choice X=susan | X=amy | ... end
end
proc {Parent X Y} // X is the parent of Y
  choice X=susan Y=john | X=richard Y=john | ... end
end

```

Based on the above relations, we can define a new procedure which determines if X is a son of Y, as follows:

```

proc {Son X Y} // X is the son of Y
  {Parent Y X} {Male X}
end

```

In a similar fashion, write new non-deterministic procedures for the following relationships.

```

proc {Mother X Y} // X is the mother of Y

proc {GrandPa X Y} // X is the grandfather of Y

proc {Brother X Y} // X is a brother of Y

proc {Uncle X Y} // X is a uncle of Y

proc {Descendant X Y} // X is descendant of Y

```