Concepts of Programming Languages

Tutorial 05, CS2104 (2003-09-22)

1 Slow and Fast Addition

Take the two procedures SADD and FADD from Lecture 06. Execute in some more detail the following statements, where you can start from an environment and store that already contain the appropriate identifiers and values for SADD and FADD.

```
local X Y Z in X=2 Y=3 {SADD X Y Z} end
local X Y Z in X=2 Y=3 {FADD X Y Z} end
```

2 Is Append Tail-Recursive?

Rewrite the following definition of Append into kernel language:

```
fun {Append Xs Ys}
   case Xs
   of nil  then Ys
   [] X|Xr then X|{Append Xr Ys}
   end
end
```

Remember that nested value construction is always moved before nested procedure application.

Can you give a reason why nested value construction is given preference over procedure call?

Execute with the abstract machine

```
local Xs Ys Zs in Xs=[1 2] Ys=[3] {Append Xs Ys Zs}
```

where you can again assume that environment and store contain the necessary identifiers and values for Append.

Is Append tail-recursive? If yes, why? Which role do single-assignment variables play here?

Homework

The following exercises are designed to be done at home.

3 Procedures Can Create Procedures

What is the value for Z after execution of the following statement:

```
local X Y Z M P Q B in
   M = proc {$ X MX}
          MX = proc {$ Y} Y=X end
       end
   {M X P}
   {M Y Q}
   B=true
   if B then R=P else R=Q end
   {R Z}
   X=2 Y=3
end
```

Execute with the abstract machine to find the anwser.

4 Odd and Even

Due to Dragan Havelka. Consider the following statement

```
local Odd Even N B in
   fun {Odd N}
      if N==0 then false else {Even N-1} end
   end
   fun {Even N}
      if N==0 then true else {Odd N-1} end
   end
   N=3
   B={Odd N}
end
```

Rewrite the statement to kernel language and then execute it.

5 Different local definitions

Given the following defintions F1 and F2:

```
declare F1
local
  fun {H X Y}
        X+Y
  end
in
  fun {F1 X}
      {H X 1}
  end
end
```

and

```
declare F2
fun {F1 X}
  fun {H X Y}
        X+Y
```

```
    end
  in
    {H X 1}
  end
```

F1 and F2 computer the same function, i.e. they are equivalent. Also both encapsulate the auxiliary function H. Still they will behave differenly in terms of the way they are executed. Please explain the difference. Translate to the kernel language and try to execute, for example: {F1 3} and {F2 3}.