

Deliverables

Michael P. Fourman

October 29, 2006

Introduction

The **final deadline** for this project is **6.00pm, Friday, 27th May**. This document gives details of how you should prepare your code for assessment.

Deliverables

The project contains stages `Stage1`, `Stage2`, `Stage3`, `Stage4`, `Stage5`, `Stage6a`, `Stage6b`. To complete each stage you must write a functor. Some of your functors may take parameters of your own choosing (such as implementations of sets and dictionaries). Before the deadline, you must provide, as *deliverables*, versions of these functors that conform to a common interface. Fortunately this is easy; it requires minor changes to your `ml_bind` file.

As an example, consider my implementation of the functor `DATAOPT` for `Stage6b`. It has the following header:

```
functor DATAOPT(  
    structure F: FlowSig  
    structure S: SetSig  
    structure D: DictSig  
    structure Basic: BasicSig  
    sharing type S.Item = string  
        and type D.Key  = F.G.Vertex = int  
        and type D.Item = S.Set) :  
sig  
    structure OF: FlowSig  
    val dataOpt : F.G.Graph -> OF.G.Graph  
end
```

The parameters include a set, `S`, a dictionary, `D`, and a collection, `Basic` of basic functions, which I have implemented using separate functors, `ORDSET`, `ASSOCLIST`, and `BASIC`.

To make a deliverable functor `D_DATAOPT` with these implementations “built-in”, I place the following lines in my `ml_bind` file

```
functor D_DATAOPT( structure F : FlowSig ) =
  DATAOPT(
    structure F = F
    structure S = ORDSET( type Item = string
                          val < = op < : string * string -> bool)
    structure D = ASSOCLIST( type Key = int
                             type Item = S.Set)
    structure Basic = BASIC()
  );
```

To give another example, for stage 2 you should make a declaration similar to the following

```
functor D_BLOCKTOCODE( structure C : CodeSig
                       structure B : BlockSig ) =
  let structure DB = BLOCKASDAG( structure B = B )
  in BLOCKTOCODE (
    structure C = C
    structure B = B
    structure S = TOPSORT( structure G = DB.G )
    structure D = ASSOCLIST( type Key = B.Expn
                             type Item = string)
  ) end;
```

Here, the `let...in...end` construction is necessary because we want to access a substructure of the structure `DB`.

At the deadline, your `ml_bind` file should contain **only**

- A declaration for a deliverable functor corresponding to each of the stages you have completed. The header for each of these functors is given below.
- A declaration of a list, `completed`, of the stages you have completed. If you have completed all stages the declaration would read

```
val completed = [Stage1,Stage2,Stage3,Stage4,Stage5,Stage6a,Stage6b]
```

If you have only completed stages 1, 2, and 6a, it would read

```
val completed = [Stage1,Stage2,Stage6a];
```

All the code for your project should be contained in the directory `~/CS201/Project` and should compile using `PolyML.make ". "` Bring a printout of your `ml_bind` file with you to the demonstration.

Testing

From Tuesday 24th May, an ML database `projtest` will be available. You can use this to “test” your solutions by

- running `ml` with the command
`ml projtest`
- Running
`PolyML.make ".";`
- Running
`projtest completed;`

This “test” will only ensure that you have got the interfaces correct; it will not test the functionality of your code—that is your responsibility. To help you do this, running `projtest` will produce, for each of the stages you name, a function from `Code` to `Code` using your implementation. For example, running

```
projtest [Stage2];
```

will declare a function

`myStage2 : Code -> Code` that will use my code to produce a flowgraph, and then use your implementation of stage 2 to produce code from that flowgraph.

Demonstrations

Marking of your project will include a live demonstration of your system. You will be expected to attend a short demonstration, of the code collected for your project, during your scheduled lab session in the final week of term.

You should have received mail confirming your lab registration, contact Mark Messenger immediately, to resolve any problems.

You should bring the following documents, each clearly labelled with your name, login name, and student ID number, with you to the demonstration:

- a list of known bugs in your implementation,
- a printout of your `ml.bind` file.

Specification of Deliverables

```
Stage1 functor D_CODETOBLOCK(  
    structure C : CodeSig  
    structure B : BlockSig  
): CodeToBlockSig  
  
Stage2 functor D_BLOCKTOCODE(  
    structure C : CodeSig  
    structure B : BlockSig  
): BlockToCodeSig  
  
Stage3 functor D_FLOWGRAPH (structure C2B : CodeToBlockSig):  
    sig  
        include FlowSig  
        val codeToFlow : C.Code -> G.Graph  
    end  
  
Stage4 functor D_LOOPOPT(F : FlowSig):  
    sig  
        structure OF : FlowSig  
        val loopOpt   : F.G.Graph -> OF.G.Graph  
    end  
  
Stage5 functor D_DATAOPT(F : FlowSig):  
    sig  
        structure OF : FlowSig  
        val dataOpt   : F.G.Graph -> OF.G.Graph  
    end  
  
Stage6a functor D_DAGOPT(B : BlockSig):  
    sig  
        structure B   : BlockSig  
        val blockOpt  : B.Block -> B.Block  
    end  
  
Stage6b functor D_OPTBLOCKTOCODE(  
    structure C : CodeSig  
    structure B : BlockSig ) : BlockToCodeSig
```