CS442 Assignment 4

Nissan Pow
20187246
npow

April 4, 2007

Part A

1. Yes there is a type that is a subtype of every other type - it is called Bottom. There is a function type that is a supertype of every other function type, and its type is Top → Bottom
2. The rule says that every polymorphic type is a subtype of it's specialized self. This makes sense, since we can always use the polymorphic type wherever the specialized one is valid.
3.
4.
5.

Parts B-E

FILE: a5.e

```
class A5
  creation make
  feature
    tok : TOKENIZER
    db : DATABASE
    dummy : BOOLEAN
    make is
      local
         s : STRING
         rs : ARRAY[STRING]
         r : RULE
         q : ARRAY[STRING] -- stores the query string
         i : INTEGER
         rr : ARRAY[INTEGER] -- stores the result of the query
      do
        !!tok.make(argument(1))
        !!db.make
        !!rs.make(1,0)
        !!q.make(1,0)
        !!rr.make(1,0)
        from
          s := ""
        until
          s.is_equal("EOF")
        loop
          s := tok.next_token
--          io.put_string(s)
--          io.put_new_line
          if s.is_equal(".") then
             !!r.make(rs) -- make rule
             db.addrule(r) -- add to db
             !!rs.make(1,0) -- clear array for next rule
          else
            if not s.is_equal(":-") then
              rs.add_last(s)
            end
          end
        end
        from
          i := 2
        until
          i > argument_count
        loop
          q.add_last(argument(i))
          i := i + 1
        end
-- display the query string
--        from
--          i := 1
--        until
--          i > q.count
--        loop
--          io.put_string(q.item(i))
--          io.put_string(" ")
--          i := i + 1
--        end
--        io.put_new_line

        dummy := db.query(q,rr)
      end
end -- class A5
```

FILE: rule.e

```
class RULE
  creation make
  feature {}
     rules : ARRAY[STRING]

  feature
    make (r : ARRAY[STRING]) is
      do
         rules := r
      end

    displayrule is
       local
          i : INTEGER
        do
           from
              i := 1
           until
              i > rules.count
           loop
              io.put_string(rules.item(i))
              io.put_string(" ")
              i := i + 1
           end
           io.put_new_line
        end

    rule (i : INTEGER) : STRING is
      do
        Result := rules.item(i)
      end

    upper : INTEGER is
      do
        Result := rules.upper
      end

    lower : INTEGER is
      do
        Result := rules.lower
      end

    count : INTEGER is
      do
        Result := rules.count
      end

end -- class RULE
```

FILE: database.e

```
class DATABASE
creation make

feature {}
  db : ARRAY[RULE]
  cut,dummy : BOOLEAN

feature
  addrule (r : RULE) is
do
  db.add_last(r)
end

  make is
    do
      !!db.make(1,0)
      cut := false
    end

  printdb is
    local
      i,j,k : INTEGER
    do
      from
        i := db.lower
      until
        i > db.upper
      loop
        db.item(i).displayrule
        i := i + 1
      end
    end

  query(q : ARRAY[STRING]; r : ARRAY[INTEGER]) : BOOLEAN is
    do
      dummy := qquery(q,r)
      Result := true
    end

  qquery (q : ARRAY[STRING]; r : ARRAY[INTEGER]) : BOOLEAN is
    local
      qq : ARRAY[STRING]
      rr : ARRAY[INTEGER]
      i,j,k : INTEGER
      matched : BOOLEAN
    do
```

```
      matched := false
--      io.put_string("query")
--      io.put_new_line
--      from
--        i := 1
--      until
--        i > q.count
--      loop
--        io.put_string(q.item(i))
--        io.put_string(" ")
--        i := i + 1
--      end
--      io.put_new_line
--      io.put_string("result")
--      io.put_new_line
--      from
--        i := 1
--      until
--        i > r.count
--      loop
--        io.put_integer(r.item(i))
--        io.put_string(" ")
--        i := i + 1
--      end
--      io.put_new_line
--      io.put_string("end")
--      io.put_new_line


      from
        i := db.lower
      until
        i > db.upper or matched or cut
      loop
         if q.count = 0 or else (q.count = 1 and then
            q.item(q.lower).is_equal("!")) then
          -- yay we have a match, so print out the sequence
          matched := true
          from
            j := r.lower
          until
            j > r.upper
          loop
            io.put_integer(r.item(j))
            io.put_string(" ")
            j := j + 1;
          end
          io.put_new_line
          Result := false
        else
          !!qq.make(1,0)
          !!rr.make(1,0)

          if q.item(q.lower).is_equal("!") then
            q.remove_first
            Result := true
          end

          qq := q.twin
          rr := r.twin
          qq.remove_first

          if db.item(i).rule(db.item(i).lower).is_equal(q.item(q.lower)) then
            -- copy the rest of the rule into the query string
            from
              k := db.item(i).upper
            until
              k = 1
            loop
              qq.add_first(db.item(i).rule(k))
              k := k - 1
            end
            rr.add_last(i)
            if qquery(qq,rr) = true then
              cut := true
            end
          end
        end -- end if
        i := i + 1
      end -- end loop
    end -- end do

end -- class DATABASE
```

Transcript

```
-== 608 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> cat test.pl
a :- !, b, c, d.
e :- f.
g.
abc :- de, fg,hi.

-== 612 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> ./a.out test.pl g
3

-== 615 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> cat blah.pl
a :- b, !, c.
b :- d.
f :- b.
b.
a.
d.
c :- e.
c.
e.
a :- d.
f :- a.

-== 617 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> ./a.out blah.pl f
3 2 6
3 4
11 1 2 6 7 9
11 1 2 6 8

-== 625 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> cat t1.pl
# runs indefinitely
p :- q, r, s.
p :- t, u.
q :- u.
t.
u.
p :- u, p.

-== 624 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> ./a.out t1.pl p t | head
2 4 5
```

```
6 5 2 4 5
6 5 6 5 2 4 5
6 5 6 5 6 5 2 4 5
6 5 6 5 6 5 6 5 2 4 5
6 5 6 5 6 5 6 5 6 5 2 4 5
6 5 6 5 6 5 6 5 6 5 6 5 2 4 5
6 5 6 5 6 5 6 5 6 5 6 5 6 5 2 4 5
6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 2 4 5
6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 2 4 5
Broken Pipe

-- The above runs forever

-== 672 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> cat t2.pl
p :- q, r, s.
p :- t, u.
q :- u.
t.
u.
p :- u.

-== 671 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> ./a.out t2.pl p t
2 4 5 4
6 5 4

-== 692 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> cat t3.pl
a :- !.
b.
c.
d :- a.
e :- f, g, h, a.
f.
g :- a.
h :- c, d.
i :- !.

-== 694 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> ./a.out t3.pl a
1

-== 691 ==- npow@cpu16 -== ~/cs442/a5 ==-
--> ./a.out t3.pl e
5 6 7 1 8 3 4 1 1
```

NOTES:
blah.pl is to test that the cut works correctly
t1.pl is the example from page 168 of the notes, and illustrates a non-terminating search
t2.pl is the example from page 167 of the notes, and just tests that searching works properly
t3.pl tests whether we can handle the cut as the last item in a rule.