MIDTERM EXAM

TDT4165 PROGRAMMING LANGUAGES

Tuesday 10 October 2006

Materials: D (No printed or handwritten materials are allowed. An approved pocket calculator is allowed.)

Task 1

Consider the following grammar for statements in a language similar to Oz.

```
<s> ::= skip
      | <s> <s>
      | local <x> in <s>
      | <x> = <x>
      | <x> = <v>
      | if <x> then <s> else <s>
```

<x> is an identifier, as in Oz. <v> is a value expression, as in Oz.

Which of the following programs is not syntactically valid according to the grammar?

a) `local A in if A then A = A else A = A`

b) `local A in if B then B = A else A = B`

c) `local A in local B in A = A B = B if A then A = B else B = A`

d) `local A in local B in A = B B = A if A then B else A`

Solution: d)

Task 2

Is the grammar in 1) ambigous and/or context free?

a) Neither ambiguous or context free.

b) Context free but not ambiguous.

c) Ambiguous but not context free.

d) Both ambiguous and context free.

Solution: d)

Task 3

In this subtask we represent arithmetic expressions with Oz-records defined by the following grammar:

```
<expr> ::= plus( <expr> <expr> )
         | minus( <expr> <expr> )
         | mult( <epxr> <expr> )
         | div( <expr> <expr> )
         | <int>
```

Here `<int>` is an integer in Oz.

Which of the following statements is most correct?

a) The grammar gives mult higher precedence than plus.

b) The grammar has no left-associative operators.

c) The grammar is ambiguous.

d) The grammar is not suitable for definition of records for execution on a non-abstract machine (with finite memory), since it can define expressions of unlimited depth.

Solution: b)

Task 4

Consider the following two programs for evaluating an expression E conforming to the grammar in the previous subtask.

```
fun {Eval1 E}
case E
of plus(E1 E2) then {Eval1 E1} + {Eval1 E2}
[] minus(E1 E2) then {Eval1 E1} - {Eval1 E2}
[] mult(E1 E2) then {Eval1 E1} * {Eval1 E2}
[] div(E1 E2) then {Eval1 E1} / {Eval1 E2}
else {Eval1 E}
end
end

fun {Eval2 E}
case E
of plus(E1 E2) then E1 + E2
[] minus(E1 E2) then E1 - E2
[] mult(E1 E2) then E1 * E2
[] div(E1 E2) then E1 / E2
else E
end
end
```

Which of the programs calculate the correct value of the expression?

a) Neither Eval1 nor Eval2.

b) Eval2 but not Eval1

c) Eval1 but not Eval2

d) Both Eval1 and Eval2.

Solution: a)

Task 5

Consider the following state in the execution of a program in the declarative kernel language on the abstract machine:

```
( [ ( {A A}, {A->a} ) ], {a->(proc{$ A} {A A} end, {})} )
```

What will be the next state in the execution?

a) There will be no next state in the execution because of infinite recursion.

b) `( nil, {a->(proc{$ A} {A A} end, {})} )`

c) `( [ ( {A A}, {A->a} ) ], {a->(proc{$ A} {A A} end, {})} )`

d) `( [ ( {A A}, {A->a} ) ( {A A}, {A->a} ) ], {a->(proc{$ A} {A A} end, {})} )`

Solution: c)

Task 6

Consider the following state in the execution of a program in the declarative kernel language on the abstract machine:

```
( [ ( try skip catch E then skip end, {} ) ], {} )
```

What will be the next state in the execution?

a) `( [(skip, {}), (catch E, {}), (skip, {})], {} )`

b) `( [(catch E, {}), (skip, {})], {} )`

c) `( [(skip, {}), (catch E then skip end, {})], {} )`

d) `( [(catch E then skip end, {})], {} )`

Solution: c)

Task 7

Consider the following Java class:

```
public class Test {

private int x;

public Test(int x) {
    this.x = x;
}

public int m(int y) {
    return x + y;
}

}
```

Is the method m declarative, according to the definition in the textbook?

a) No.

b) Yes, but only if the caller of test is the only one with a reference to the object.

c) Yes, but only if the rest of the program doesn't have any global variables (like e.g. variables declared with the keywords public and static).

d) Yes.

Solution: a) is correct, since the return value of the method depends on state that was set in the constructor. We have to give points for d) also, because of confused clarifications given during the exam.

Task 8

Which of the following statements is not correct?

a) In purely functional programming in the declarative sequential computation model, variables are never unbound.

b) Purely functional programming in the declarative sequential computation model always results in declarative programs.

c) If we extend the declarative sequential computation model with the `thread` statement, the resulting model will still be declarative.

d) If we extend the declarative sequential computation model with exceptions, the resulting model will no longer be declarative.

Solution: d) is clearly incorrect. a) was meant to be correct, but it can be considered to be incorrect since a variable will be unbound in an instant between its creating and its binding.

Task 9

Consider the following definition of the conditional `if` as a boolean function of three parameters:

```
fun {If Condition Consequent Alternative}
   if Condition
      then Consequent
      else Alternative
   end
end
```

We can use this function instead of the `if` statement as follows:

```
if A then B else C end
```

can be rewritten as

```
{If A B C}
```

where A is a boolean expression, and B and C are arbitrary expressions or statements (depending on the context in which the `if` statement is used).

For example,

```
if X>0 then {F1 X} else {F2 X} end
```

would be replaced by

```
{If X>0 {F1 X} {F2 X}}
```

Consider the definition of a function that contains an `if` statement. Given an input to the function, which of the following statements is correct?

a) It is possible to define the function so that the `If`-version will not return at all, even if the `if`-version returns, irrespectively of the computation model used.

b) It is possible to define the function so that the `if`-version may sometimes return a different result than the `If`-version, even if a declarative model is used.

c) The `if`-version of the function will always return the same result as the `If`-version, irrespectively of how the function is defined, but only in the sequential declarative model.

d) The `if`-version of the function will always return the same result as the `If`-version, irrespectively of how the function is defined and irrespectively of the computation model used.

Solution: The intended answer was a), but the expression *irrespectively of the computational model used* implies that we could also think of a computation model (not defined in the book) where all functions are evaluated lazily. Therefore, this all answers are considered to be correct for this task.

Task 10

A double-linked list is a list where each element contains not only the value of the element and a link to the rest of the list (as it is in the case of single-linked lists), but also a link to the previous element of the list.

The following function implements one possibility to encode a double-linked list. It takes as argument a regular (single-linked) list and returns the corresponding double-linked list:

```
fun {MakeDoubleLinked List}
   fun {Recur Current Previous}
      case Current of nil then nil
      [] Head|Tail then '|'(Head {Recur Tail Current} Previous)
      end
   end
in
   {Recur List nil}
end
```

Given this representation of double-linked lists, which is the correct way to access the value of the second element of a double-linked version of the list `[1 2 3]`, that is, the number 2:

a) `{MakeDoubleLinked [1 2 3]}.2.2.1`

b) `{MakeDoubleLinked [1 2 3]}.2.2.2.2.2.1.1`

c) `{MakeDoubleLinked [1 2 3]}.2.2.3.1`

d) `{MakeDoubleLinked [1 2 3]}.3.3.2.2.1`

Solution: c)

Task 11

Consider the following code:

```
fun {Create Plus Minus Eq}
   Ops = ops(plus:Plus minus:Minus eq:Eq) in
   fun {$ Op X Y}
      {Ops.Op X Y}
   end
end
%$
```

This code contains the following elements of higher-order programming:

a) Genericity, but not instantiation and embedding.

b) Embedding and instantiation, but not genericity.

c) Instantiation and genericity, but not embedding.

d) Genericity, embedding and instantiation.

Solution: d)

Task 12

Consider the following two functions:

```
fun {A R L}
   case R#L
   of x(X)#y(Y) then {A X Y}
   else {A L R}
   end
end

fun {B Is}
   case Is
   of nil then nil
   [] I|Ir then Irr = {B Ir} in I+1|Irr
   end
end
```

Which of the functions will run with constant stack size?

a) Neither A nor B.

b) B but not A.

c) A but not B.

d) Both A and B.

Solution: c)

Task 13

Suppose we have a function that is able to test whether a variable is unbound. Let's call it Unbound.

Consider the following program:

```
local X Y in
   thread if {Unbound X} then X=1 end end
   thread if {Unbound X} then X=2 end end
end
```

Which sentence about the program is correct?

a) All executions will lead to a unification failure.

b) Some executions may result in a unification failure.

c) The program is declarative, and X will be bound to 1 in every execution.

d) The program is not declarative, but X will be bound to 1 in every execution.

Solution: b)

Task 14

Consider this concurrent definition of a function that computes Fibonacci numbers:

```
fun {Fibonacci N}
   if N<2 then N
   else thread {Fibonacci N-1} end
      + thread {Fibonacci N-2} end
   end
end
```

Which of the following statements about an application of the function Fibonacci to an argument N is correct:

- if N is 10, then it is not possible that at some time there will be less than 10 threads running concurrently
- if N is 10, then it is possible that at some time there will be more than 100 threads running concurrently
- if N is 2, then the total number of threads created during the execution is 4
- if N is 5, then it is not possible that at some time there will be more than 10 threads running concurrently

Solution: b)