UCLA CS 131 Midterm, Fall 2011
100 minutes total, open book, open notes

Name: Prasanth Veeriya Student ID: 803765086

1	2	3	4	5	6	total
13	7	12	12	7	11	62

1. Consider the following syntax, written in ISO EBNF. The start symbol is "syntax". Assume that the "letter" token stands for any ASCII letter and the "digit" token for any ASCII digit.

```
syntax = syntax rule, {syntax rule};
syntax rule = meta id, '=', defns list, ';';
defns list = defn, {'|', defn};
defn = term, {',', term};
factor = [integer, '*'], term;
term = repeated sequence | meta id | empty;
repeated sequence = '{', defns list, '}';
optional sequence = '[', defns list, ']';
meta id = letter, {letter | digit};
empty = ;
```

1a (3 minutes). Which rules in the above grammar are unreachable?

1b (8 minutes). Translate the reachable part of the grammar into an equivalent grammar that uses the OCaml-based notation of Homeworks 1 and 2. If this is not possible for some reason, explain why not, and go as far in the translation as you can.

1c (5 minutes). If we assume that the above grammar is the entire syntax for ISO EBNF, Which parts of the above grammar are not written in valid ISO EBNF?

2. For each of the following OCaml expressions, give its type and value.

2a (3 minutes). `let i x = x in i i i`

2b (5 minutes).

```
let expr = "3 + 4"
in let lvalue = "a"
   in [expr, ["("; expr; ")"];
       expr, [lvalue];
       lvalue, ["$"; expr];
       lvalue, ["self"]]
```

2c (7 minutes).

```
let accept_all derivation string =
      Some (derivation, string)
in let accept_empty_suffix derivation =
      function
        | [] -> Some (derivation, [])
        | _ -> None
   in accept_all accept_empty_suffix "aoooogah!"
```

3 (12 minutes). Write a curried Ocaml function eqf that compares two functions for equality. It should accept two functions, and a list of values, and return true if and only if the functions return equal results for each argument value in the list. As a trivial case, if the list of values is empty, eqf should return true. Use only standard OCaml functions, defined in the Pervasives module, and do not use side effects.

4 (12 minutes). Suppose I design a new language There-Is-No-Try-Java. This new language is just like Java, except that there's no "try" keyword; one simply writes exactly the same code as in Java, but without the "try". For example, instead of this:

```
try { s = buf.readLine(); }
catch (IOException e)
  { System.out.println (e); }
finally { discard (s); }
```

you write this:

```
{ s = buf.readLine(); }
catch (IOException e)
  { System.out.println (e); }
finally { discard (s); }
```

Is the syntax of There-Is-No-Try-Java ambiguous? If so, illustrate the ambiguity with a program that can be parsed in two different ways. If not, explain why not. Repeat the exercise for There-Is-No-Finally-Java, a language that is just like Java except without a "finally" keyword.

5 (15 minutes). Typically, the C memory allocator is mostly written in C, though there may be a few small parts of it that are written in machine code. Could the OCaml memory allocator be done similarly? That is, could it be mostly written in OCaml? If so, give an outline of how it would work; if not, explain why not.

6 (30 minutes). The documentation for Java's availableProcessors() method says "Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately." This is to support virtualized machines where physical processors can be given to (or taken away from) the virtual machine, on the fly.

Suppose your solution to Homework 3 is working, and suppose you want to modify it so that if the -p option is not used, your implementation dynamically alters the number of worker threads so that it equals the number of processors currently available. If the number of processors shrinks, you'll wait for the currently-running threads to finish the blocks they're working on, before you delete the excess threads.

Explain how you would implement this modification. Focus on the specific Java language and library features that you will use as part of your implementation, and why you'll use them. Also, mention any Java features we covered in class that are *not* relevant to your modification, and why. For example, will your solution involve exception handling? How about generic classes? Why or why not?

Prasanth Veerina
8037650866

CS131 Midterm

3 1.a) factor and optional sequence are unreachable

b) (syntax, [Syntax , [N syntax_rule];
Syntax , [N Syntax_rule; N Syntax];
Syntax_rule, [N meta_id; T "="; N defn_list; T ";"];
defn_list, [N defn];
defn_list, [N defn; T "|", N defn_list];
defn ;= [N Term];
defn, [N Term; T","; N defn];
Term, [N repeated_sequence];
Term, [N meta_id];
Term, [N empty];
repeated_sequence, [T "{"; N defn_list; T "}"];
meta_id =

We cannot translate this as written, we would need to define more NTs

empty = [];

5

c) factor is invalid because optional sequence is not a term.

2.a) val ; : 'a -> 'a
= 'a -> 'a -> 'a

b) (string* string list) list =
[("3+4", ["("; "3+4"; ")"]); ("3+4", ["a"]); ("a", ["$"; "3+4"]);
("a", ["self"])]

c) val accept_all : 'a -> 'b -> ('a * 'b) option = <fun>
val accept_empty_suffix -> 'a -> 'b list -> ('a * 'b list) option = <fun>
Evaluates to Some (<fun>, string) ✓

5

2

3. let rec eqf f g l = match l with
| [] -> true
| h::t -> if((g h) = (f h)) then
eqf f g t
else false

12

4. if { Stmt } catch { Stmt }

This would not produce ambiguity, it may produce many situations where syntax is invalid however it won't produce multiple valid syntax trees.

12

try { Stmt } catch { Stmt } { }

~~in the finally catch~~ we ~~are also safe~~ since we always want the finally ~~to execute~~ it is

try { } catch { } { }

is this try-catch-finally or try-catch-random code block.
it is unclear ~~that~~ that finally should be applied here or if it is a regular block of code.

Prasan[illegible]
Veera[illegible]
803765086

7

5. Since Ocaml has no side effects this could be somewhat difficult, we wouldn't be able to make assignment statements, or save state globally.

yes, but...
you have asm,
(also, full Ocaml has side effects...)

~~I would implement this by using having my main thread poll~~

L

In my HW 3 implementation the main thread starts n # threads and lets them continue to process until all input blocks are processed. to implement this new functionality I would have main periodically check the number of processors and update a member variable of numThreads. next I would check to see in numThreads is < the number of threads in my thread array. on each of my worker threads I would implement a function called cleanup which will prevent the thread from taking a new block after it finishes its current one. from the main thread I would call cleanup on the number of threads I need to remove and then poll their state until I see that the thread is terminated at which point I delete it. My implementation would work fine with generics, Exceptions would work the same way as well. This is thread safe since we are not try to access any shared memory the main thread simply directs the workers to stop

11

race possible here, how
bottleneck!