UCLA CS 131 Midterm, Fall 2015
100 minutes total, open book, open notes

Name: Marek Pokorny Student ID: 204477064

	1	2	3	4	5	6	7
	11	10	10	8	17	6	11

total 83/100

1. What are the types of the functions f, g, h defined by the following OCaml declarations? For each function, give an example call to that function.

a (6 minutes).

```
let f = fun fact oid ->
  if oid <= 0 then 1 else oid * fact (oid - 1)
```

b (6 minutes).

```
let g a b c = c a b
```

c (10 minutes).

```
let rec h m n = function
  |[] -> n
  |x::t -> h m (m n x) t
```

n: 'a
hidden = 'c list
m: 'a -> 'c -> 'a

2 (10 minutes). Is the following definition equivalent in type and behavior to the definition in (1c)? If so, explain why; if not, explain why not.

```
let h m =
  let rec h1 n ls = match ls with
    |[] -> n
    |x::t -> h1 (m x n) t
  in h1
```

3a (6 minutes). The OCaml expression '(fun x -> x) = (fun x -> x)' causes the interpreter to report 'Exception: Invalid_argument "equal: functional value".' What does this diagnostic mean? Why is this error caught at run-time rather than at compile-time?

3b (6 minutes). Suppose 'f' is as given in problem (1a), suppose 'id' the identity function 'fun x -> x', and suppose you built a working solution to Homework 1. What, if anything, is the type of the expression 'computed_fixed_point (=) f id'? What, if anything, is the value of the expression? Briefly explain.

if f = (fun x->x) (fun oid -> int)

4. Js_of_ocaml <http://ocsigen.org/js_of_ocaml/> is a compiler from OCaml bytecode programs to JavaScript. The idea is to run OCaml programs on Web browsers that support JavaScript. Js_of_ocaml works with an existing installation of OCaml, with no need to recompile any library, and comes with bindings for a large part of the browser APIs. According to its authors' benchmarks, the generated programs typically run faster than with the OCaml bytecode interpreter.

4a (6 minutes). Draw a diagram explaining how Js_of_ocaml works. Use circles for programs, boxes for data, and arrows for transfer of information.

4b (6 minutes). JavaScript does type checking at run-time, and doesn't really support multiple threads with shared memory. Which of these two things is more likely to be a significant problem for the Js_of_ocaml implementation, and why? Briefly explain.

4c (6 minutes). Assuming Js_of_ocaml works as advertised, would it make sense to do something similar for running Java programs on Web browsers that support JavaScript? If so, explain the biggest problems you would run into, and how you would address them. If not, explain why not.

5. Consider the following grammar for msg-id, modified from Internet RFC 5322:

```
msg-id          = "<" id-left "@" id-right ">"
id-left         = dot-atom-text
id-right        = dot-atom-text / no-fold-literal
no-fold-literal = msg-id
dot-atom-text   = 1*atext *("." 1*atext)
atext           = "a" / "b" / "c" / "d" / "@"
```

5a (6 minutes). Prove that this modified grammar is ambiguous.

5b (6 minutes). Modify the grammar to make it unambiguous. Make as few changes to the grammar as you can.

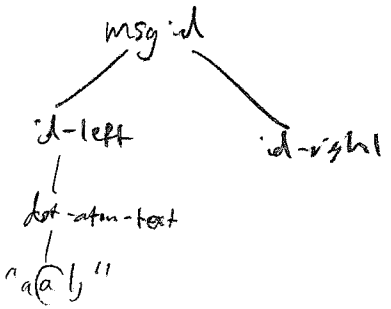
5c (6 minutes). Give a minimal syntax diagram for the unmodified grammar.

6 (8 minutes). When doing type checking, does OCaml use structural equivalence or name equivalence? Briefly explain why its choice (structural or name equivalence) is the better one for typical OCaml programs.

7. Suppose I design a new language Volatile-Java. Volatile-Java is just like Java, except that all class and instance variables behave as if they were declared with the 'volatile' keyword, regardless of whether the source code says 'volatile'. Volatile-Java's local variables are the same as Java's.

7a (8 minutes). Show that every data-race-free Java program is also a data-race-free Volatile-Java program. Also, show that the converse is not true, i.e., show that not every DRF Volatile-Java program is a DRF Java program.

7b (4 minutes). Give a significant advantage of Java over Volatile-Java. Give a significant disadvantage.



Marek Pokorny
20447064

CS 131 Midterm

+11

1. a) (int → int) → int → int +3
   b) 'a → 'b → ('a → 'b → 'c) → 'c +3
   c) ('a → 'b → 'a) → 'a → 'b list → 'a +5

examples.... :(

2. It is not equivalent in type because the order of the arguments passed to m have been swapped, giving the new function h the type +5
   ('b → 'a → 'a) → 'a → 'b list → 'a 0 +5.

+10

This makes the behavior different too, assuming that the same m was given to both versions of h. For the same behavior to be achieved, the arguments of m would need to be swapped.

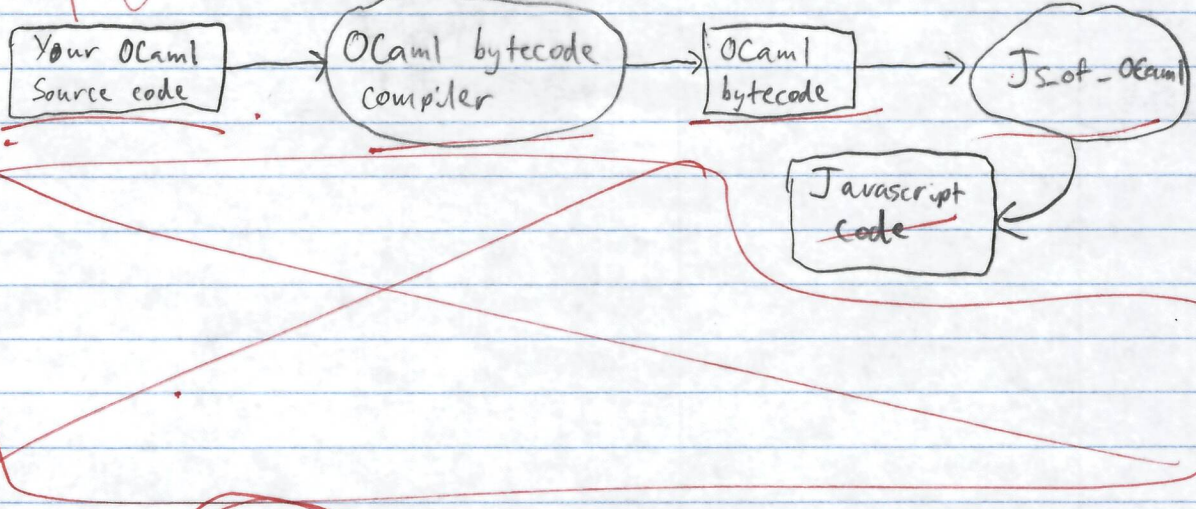
3. a) This diagnostic means that the OCaml = operator cannot compare two functions with each other. This is caught only at run time because the = operator has type 'a → 'a → bool, and the 'a type parameter is bound only at runtime.

+4

b) The type of the expression is
   int → int
   because it returns a curried version of f with only the first argument given. The expression will have no value, because computed_fixed_point would have to compare two functions in order to return a value of type 'int → int' which we know from before will fail.

+6

4. a)



+6

b) Javascript's run-time type checking will be the bigger problem because OCaml has compile time type checking. This becomes an issue when "native" Javascript and Js_of_Ocaml code interact; the "native" Javascript may attempt to call a Js_of_Ocaml function with the wrong type, or vice versa, potentially causing run time errors. Both OCaml and Javascript are typically single threaded, so that's not an issue.

+6 plausible

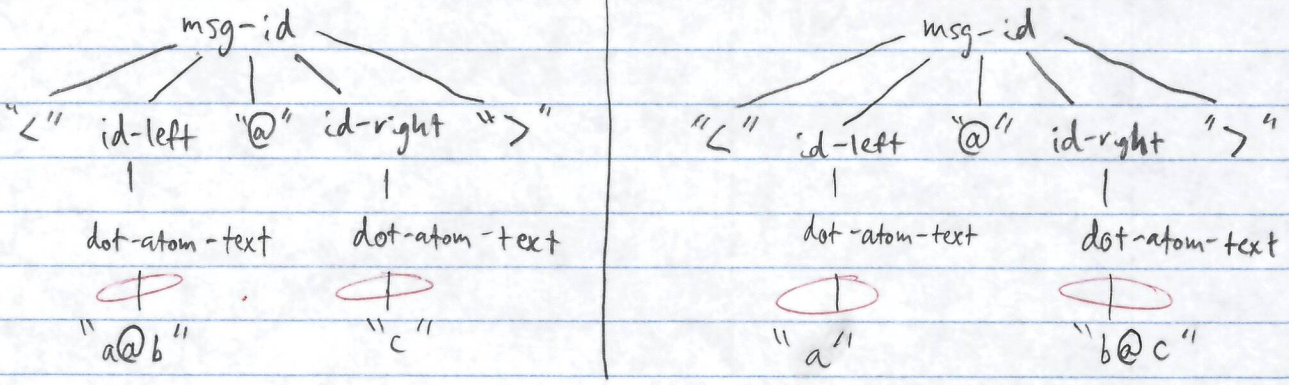
+6 c) It would almost make sense to do this, although there would be a giant list of problems. Firstly, you would have to port the entire standard library, and some classes, such as Thread, as well as the language features regarding syncronization, would need to be disabled because JavaScript doesn't support multiple threads with shared memory. Also, Javascript doesn't technically have classes, and especially not abstract classes or interfaces, so this would have to be worked around in Javascript. Similar for generics.

yes

5 a) Consider the following message id:
<a@b@c>

+6

This can have more than one parse tree:



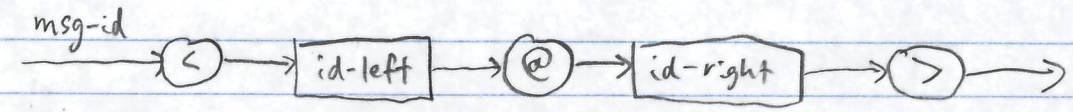
leaving out atext

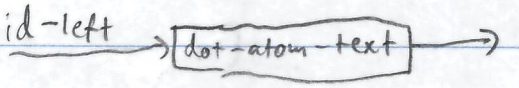
b) Change the atext rule to:
atext = "a" / "b" / "c" / "d"
(eliminate the "@" from atext)

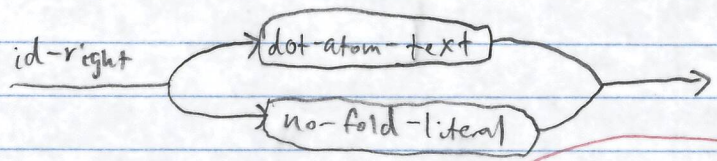
+6

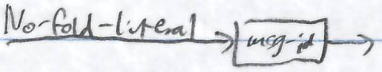
c)

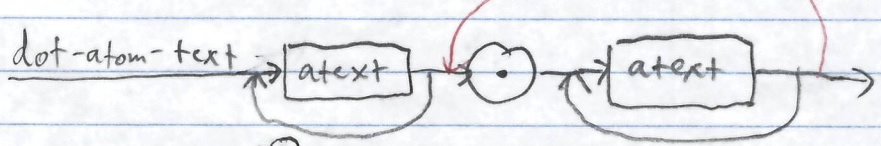
+5



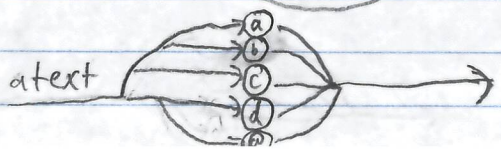








not minimal



6. OCaml uses structural equivalence for type checking. This is better for typical OCaml programmers because of OCaml's type inference system and the fact that OCaml is used primarily for functional programming. For example, when passing in a curried function as an argument to another function, the programmer shouldn't have to deal with type names; instead, OCaml uses structural equivalence to figure out weather or not that function had the correct type, which can be fairly complicated when considering type parameters, so this system is unsuitable for named types in most cases.

6

example; the arg is not entirely clear...

7. a) All DRF Java programs are also DRF Volatile-Java programs because all the volatile keyword does is ensure that values are loaded and stored when we think they are; if the program is already DRF in Java, then those additional loads and stores while updating a value do not matter, as the value should not be changed during the update if the program was already DRF. The converse is not true; due to the optimizations the compiler might make to reduce loads and stores, a value may get loaded in thread 1, changed by thread 2, and would have been loaded again in Volatile-Java, but not in Java. Thus, in Java the incorrect value might be used, while the volatile keyword would've solved the data race.

7

specific example would be clearer

b) An advantage of Java over Volatile-Java is performance; using volatile hurts single threaded performance by forcing additional memory accesses. An advantage of Volatile-Java over Java is that it makes some programs DRF "for free," as seen by part a. Switching from Java to Volatile-Java can only help a program in terms of correctness in multithreaded code.

4