Q&A Session for Programming Languages Exam 2 Topics

Session Number: 1200880572
Date: 2020-11-6
Starting time: 14:29

ANON - 14:36
Q: could you type any announcements in the chat again? I just logged on
Priority: N/A
	Konstantin Kuzmin - 14:38
	A: Just did. :)

ANON - 15:01
Q: What is the difference between shallow recursion and deep recursion? Thank you.
Priority: N/A
	Steven Haussmann - 15:01
	A: Shallow recursion operates on each element in the list, but will not recurse through those elements as well.
	Steven Haussmann - 15:02
	A: A shallow "reverse list" function would turn ((1, 2, 3), (4, 5, 6)) into ((4, 5, 6), (1, 2, 3)). A deep "reverse list" function would produce ((6, 5, 4), (3, 2, 1))

ANON - 15:03
Q: is pre-recorded video a review, not a lecture?
Priority: N/A
	Ana L. Milanova - 15:04
	A: Yes. Review and problem solving, the same as what we had before Exam 1.
	Ana L. Milanova - 15:05
	A: What I intended for the live lecture. I recorded this in an emergency this morning, given how unreliable my internet has been for 3 days.

ANON - 15:05
Q: Will this pre-recorded video be posted on mediasite? I don't see it as of now
Priority: N/A
	Steven Haussmann - 15:06
	A: It's marked as Lec18
	Steven Haussmann - 15:06
	A: You can sort by "Most Recent" to find the latest videos.

ANON - 15:07
Q: In static scoping with functions as first-class values, is there still the notion of shallow/deep binding?
Priority: N/A
	Ana L. Milanova - 15:08
	A: No. Shallow and deep binding come up only in dynamic scoping.
	Ana L. Milanova - 15:08
	A: Shallow is the default, when the function value carries no reference environement, so non-local bindings are resolved in the current dynamic environment following the dynamic links.
	Ana L. Milanova - 15:10
	A: Deep binding happens when one function, say A passes another, say B, as an argument. So A passes its own ref. environment with B, and when B is called, it searches for non-local bindings in that enviornment.

ANON - 15:08
Q: How do we determine whether a lambda term reducible quickly?
Priority: N/A
	Steven Haussmann - 15:09
	A: An expression is reducible if and only if it has an application where the left side is an abstraction.

ANON - 15:12
Q: For homework 5, if we are working in a team, do we still have late days? I assume it would be if both group members have late days then the group has at least one late day at the expense of all group members, but i just wanted to make sure
Priority: N/A
	Konstantin Kuzmin - 15:13
	A: Yes, you are correct. I made a post on this about 20 minutes ago: https://submitty.cs.rpi.edu/courses/f20/csci4430/forum/threads/417

ANON - 15:12
Q: how can we quickly check if a funciton is tail-recursive? l
Priority: N/A
	Steven Haussmann - 15:13
	A: A function is tail-recursive if its last action is to make the recursive call.
	Steven Haussmann - 15:13
	A: Every path must lead to a recursive call with no further work done. If even one path exists that does work after the call, it's not tail-recursive.

ANON - 15:12
Q: So in static scoping, if functions are first class values, what reference environment is used? static binding?
Priority: N/A
	Ana L. Milanova - 15:14
	A: Yes, the static environment, i.e., the enclosing environment.

ANON - 15:13
Q: how do we know if it returns the immediately recursive call?
Priority: N/A
	Steven Haussmann - 15:14
	A: An expression returns whatever is at its end -- in the tail position.
	Steven Haussmann - 15:16
	A: More intuitively, you return whatever is at the end of the path you follow through the function. So, (if some_condition 1 2) can return either 1 or 2; both of those would be valid places for a recursive call to get tail-call optimization

---

ANON - 15:16
Q: is there a solution key for hw 3?
Priority: N/A
	Ana L. Milanova - 15:20
	A: No, I don't have one. With a class this big there are students with extensions still working on those hw sets.
	Ana L. Milanova - 15:21
	A: You can send a regrade request, or come to office hours to check your solutions.

ANON - 15:19
Q: When you recursively pass a function around, would the local reference binding?
Priority: N/A
	Steven Haussmann - 15:20
	A: The reference environment is set when the function is first passed.

ANON - 15:19
Q: For deep binding ^
Priority: N/A

ANON - 15:20
Q: Could we quickly review the differences between binding, scoping,

and typing at a high-level?
Priority: N/A
        Steven Haussmann - 15:21
        A: "binding" is the process of associating a variable with something -- in the λ-calculus, we say a variable is bound if it's inside an abstraction with its name
        Steven Haussmann - 15:22
        A: "scoping" is how you resolve a name. It is somewhat related to binding, since scoping determines to what a variable gets bound.
        Steven Haussmann - 15:23
        A: "typing" is how we declare and reason about what kind of data our program is using. It can be done statically, to eliminate some categories of errors (e.g. adding non-numbers), as well as dynamically (to halt execution for invalid states)

ANON - 15:21
Q: what is an application expression versus a normal expression?
Priority: N/A
        Ana L. Milanova - 15:22
        A: A lambda expressions can be one of 3 kinds: variable expression, e.g., x; application expression, E1 E2, or an abstraction expression, \lambda x. E.
        Steven Haussmann - 15:23
        A: This gives us a recursive definition of the language! An abstraction, for example, is an expression made up of an identifier and another expression.

ANON - 15:22
Q: to determine if something is a free variable: should we check 1) if it comes from outer scope 2) if there's a Lz where z is the variable we are trying to determine if it's free
Priority: N/A
        Ana L. Milanova - 15:24
        A: You should check whether it has a surrounding \lambda x. binding. E.g., \lambda x. x, x is bound because it is surrounded by the \lambda x. binding.

ANON - 15:23
Q: What does it mean when a variable "shadows" another?
Priority: N/A
        Steven Haussmann - 15:24
        A: That variable has the same name as a variable from some outer context; we find it instead of the original variable when doing resolution.
        Steven Haussmann - 15:25
        A: (let ((x 1)) (let ((x 2)) x)) produces 2, not 1, beacuse the inner let's x=2 shadows the outer let's x=1

ANON - 15:28
Q: what is the checklist we should have in our head to check if an expression is in NF?
Priority: N/A
Steven Haussmann - 15:28
A: To be in normal form, there must be no expressions ANYWHERE in the expression. So, it's an exhaustive search.
Steven Haussmann - 15:29
A: er, no reducible expressions anywhere :)
Ana L. Milanova - 15:30
A: Slide 13 in Lecture 16 gives the definitions of the different normal forms.

ANON - 15:29
Q: Lecture Notes 17, HNF.. Why is x ((λx.x) (λy.y)) in HNF? Can't isn't (λx.x) (λy.y) reducible? That would mean it
Priority: N/A
Ana L. Milanova - 15:31
A: HNF is a lazy normal form. Since this is an application expression "headed" by an x, it is in HNF. HNF is lazy in the sense that it does not evaluate the argument until it's needed.

ANON - 15:31
Q: Oh my apologies. It is in HNF.... x E1 E2 is in HNF and WHNF. i was confusing it with the abstraction rule
Priority: N/A
Ana L. Milanova - 15:32
A: Yes, that is correct.

ANON - 15:32
Q: how do we know that \lambda x x z can be reduced? i don't understand how we came to that conclusion from the parse tree
Priority: N/A
Ana L. Milanova - 15:33
A: (\lambda x. x) z -> z. We apply the \lamda x.x function on z, and we get z.
Steven Haussmann - 15:34
A: On the other hand, λx. x z would, indeed, be in normal form, because application takes precedence over abstraction. It's equivalent to λx.(x z), which obviously lacks a reducible expression

---

ANON - 15:34
Q: what is the purpose for identifying if an expression is in NF, HNF, or WHNF?

Priority: N/A
Ana L. Milanova - 15:36
A: In our case, it solidifies understanding of the lambda calculus... The normal form is the meaning of the term.
Ana L. Milanova - 15:37
A: HNFs are used in "lazy" languages like Haskell, which use normal order reduction and keep parts of the expression unevaluated.

ANON - 15:38
Q: For the exam, would we be allowed to work out the lambda problems and upload the pics? I think it would be easier to write down and work out the expressions as opposed to trying to do that in the answer boxes in submitty.
Priority: N/A
Ana L. Milanova - 15:40
A: We are planning for as few uploads as possible. We haven't finalized everything yet, but for now we plan on shorter reduction sequences and typing in a Submitty text box.
Ana L. Milanova - 15:40
A: Terms will be less convoluted compared to ones on paper tests.

ANON - 15:42
Q: Can we call \L z.z as I because technically it is the renamed Idenity Combinator?
Priority: N/A
Ana L. Milanova - 15:42
A: Yes.

ANON - 15:46
Q: will non-normal form expressions not reduce to a final normal form?
Priority: N/A
Steven Haussmann - 15:46
A: An expression that has a normal form can be reduced to it via normal-order evaluation.
Steven Haussmann - 15:47
A: However, it is possible for an experssion to not have a normal form. Consider (λx. x x) (λx. x x) - this will reduce to the same thing, forever, and has no normal form. It will never be in normal form.

ANON - 15:47
Q: do normal-order evaluation apply to HNF and WHNF?
Priority: N/A
Steven Haussmann - 15:48
A: They are unrelated concepts; NF/HNF/WHNF means that an

expression is either fully or partially non-reducible. Normal/ applicative order decides in which order you perform reductions.

ANON - 15:50
Q: is "tru" is a variable?
Priority: N/A
Steven Haussmann - 15:51
A: "tru" and "fls" are just shorthand for the true and false Church booleans -- λx.λy.x and λx.λy.y

---

ANON - 15:50
Q: oh it's a function. got you.
Priority: N/A

ANON - 15:51
Q: in this example, are we defining logical OR and logical AND in terms of church booleans?
Priority: N/A
Steven Haussmann - 15:53
A: Yes, we're creating an expression that accepts two Church booleans and produces a Church boolean

---

ANON - 15:55
Q: Random Queston. For the Extra Credit on the Quiz, if we have an Extra Trailing Space, will we lose points?
Priority: N/A
Ana L. Milanova - 15:55
A: I'm back, but don't know for how long... It just goes on and off. Keep asking questions here.
Ana L. Milanova - 15:56
A: No, you shouldn't loose any points for that. Prof. Kuzmin's fuzzy string comparison works very well.
Konstantin Kuzmin - 16:06
A: You shouldn't, unless the custom autograder that I wrote for Submitty is buggy. If it is the case, just let me know, and I will get it fixed. :)

ANON - 15:55
Q: When you recursively pass a function around, would the local reference binding ever change in the case of dynamic deep binding, or does it always use the reference bindings from when it was first passed, when it gets evaluated down the stack?
Priority: N/A
Steven Haussmann - 15:56
A: No, it only gets bound once.

Steven Haussmann - 15:56
A: (and that binding happens when it is first passed)

ANON - 15:55
Q: Will answers be posted for the problem sets one day before the exam? Just like last time?
Priority: N/A
Ana L. Milanova - 15:56
A: Yes, I should post this earlier, today or tomorrow.

ANON - 15:58
Q: this is a very basic question, but how do we check for equality in scheme?
Priority: N/A
Steven Haussmann - 15:59
A: equal? does the job. eq? checks if its two arguments are the *exact same thing in memory*, which usually is not what you want.
Steven Haussmann - 16:00
A: (eq? '(1 2) '(1 2)) is false, because those are two separate things in memory. (let ((x '(1 2))) (eq? x x)) is true, since the two arguments are the same thing.

ANON - 16:00
Q: if we have a nested list in scheme, do we only need to add ' to the outermost list or every innermost list as well?
Priority: N/A
Steven Haussmann - 16:00
A: Only once. Quoting means "do not evaluate any of this!"
Steven Haussmann - 16:00
A: Putting extra quotes in the inside gets very weird -- you wind up with a quotechar inside of the resulting list

---

ANON - 16:01
Q: if we forget a quote on a list like '(1 2 3), how will it be evaluated?
Priority: N/A
Steven Haussmann - 16:01
A: Racket will attempt to call a function named 1 with arguments 2 and 3. This will fail.

ANON - 16:02
Q: what is a reference envrionment? is it referring to the stack frame?
Priority: N/A
Steven Haussmann - 16:02

A: It's the bindings that exist for a particular function during execution. So, yes, it's basically a stack frame, since each function gets a frame to put its variables in.

ANON - 16:04
Q: i don't really understanding how shallow and deep binding actually impact the program?
Priority: N/A
Steven Haussmann - 16:05
A: Deep binding means that you start from where your function was passed, not from where it was called.
Steven Haussmann - 16:05
A: If a function is not passed as an argument, and is just used directly, it makes no difference.

ANON - 16:06
Q: i feel like i understand these concepts separately, but putting it altogether in an example like this is difficult since everything interacts. are there suggestions for this other than the problem set questions?
Priority: N/A
Steven Haussmann - 16:11
A: I don't have any extra examples to suggest. I would just focus on understanding the idea behind dynamic scoping (start from one point on the call stack and work backwards), and then extending that to deep binding, which just starts higher up on the stack

---

ANON - 16:06
Q: So if a Function returns A as opposed to calling A via (A), it would be considered a Higher-Order Function?
Priority: N/A
Ana L. Milanova - 16:07
A: Yes, that is correct.

ANON - 16:17
Q: should we understand map, foldl and foldr for the exam?
Priority: N/A
Ana L. Milanova - 16:18
A: Yes. There will be questions similar to the ones we're going over from the problem sets.
Steven Haussmann - 16:18
A: You should understand them for programming in general, too! They're very nice for reasoning about lists of data (:

ANON - 16:19

Q: When I did this Problem, I had cons y x, but it seems that this would pre-pend the data and have the reverse order of what we want. How does this ensure we get (3 2 4) and not (4 3 2)?
Priority: N/A
        Ana L. Milanova - 16:20
        A: Will show right now.

ANON - 16:20
Q: Oh I see. I did it the same but did not reverse the list.
Priority: N/A

ANON - 16:20
Q: is the exam going to be more focused on reasoning through examples or writing our own code? will it be similar to exam 1 in length?
Priority: N/A
        Ana L. Milanova - 16:22
        A: A mixture of the two, as in those practice set questins.

ANON - 16:22
Q: instead of reversing, can we replace (cons y x) with (cons x (cons y `()))? or would we not want that because that would produce a list with incorrect structure?
Priority: N/A
        Ana L. Milanova - 16:23
        A: Yes, that will be a list with incorrect structure, so we shouldn't be doing that.
        Ana L. Milanova - 16:23
        A: My intention with this question was to reverse twice.
        Ana L. Milanova - 16:24
        A: But the order of elements in the flat list will be correct.

ANON - 16:25
Q: True if you did (cons x (cons y '())) you could then run flatten and get the expected output, but that's not intended
Priority: N/A
        Ana L. Milanova - 16:25
        A: Yes, that is correct.

ANON - 16:26
Q: So in a flat list using (cons x (cons y `())) instead of reversing would give us the correct elements, but the underlying cons cell structure wouldn't follow the usual head/tail form?
Priority: N/A
        Ana L. Milanova - 16:27
        A: With (cons x (cons y '()) we will have a list with

"incorrect", or more precisely unintended cons-cell structure.
	Ana L. Milanova - 16:28
	A: If we flatten, we will have the correct structure, but we are not allowed to do that. (I haven't thought whether we can code flatten with what is allowed. I think we can't)

ANON - 16:27
Q: For that question, wouldn't it be easier to flatten the list first then return a list of all the symbols?
Priority: N/A
	Ana L. Milanova - 16:30
	A: Yes, that is a solution too.

ANON - 16:27
Q: how do we brainstorm the inductive case? I always struggle to think about these
Priority: N/A
	Steven Haussmann - 16:28
	A: "If I have a solution for the smaller problem, how do I produce a solution for that problem, plus one element?"
	Steven Haussmann - 16:28
	A: So, assume your function already works, then figure out what you have to add onto the smaller result.

ANON - 16:28
Q: For exam 2, will we be able to start anytime between 2:30 to 9, and we will have 1hr 50mins to work?
Priority: N/A
	Ana L. Milanova - 16:31
	A: Yes, the intention is to use the same format of submission window as with Exam 1. We will make an announcement over the weekend.

ANON - 16:30
Q: For the sym function, the last scheme exercize you worked out, wouldn't it be easier to flatten the list first and then call sym to return all the symbols (in a list)?
Priority: N/A
	Ana L. Milanova - 16:33
	A: Yes, I believe that will be a valid solution too. Hard to say if it will be "easier" as you'd need to implement the same kind of foldl/map-based deep recursive "flatten" as we did here. Then have one more "filter" pass.