```
Q&A Session for Programming Languages Lecture 17

Session Number: 1209867609
Date: 2020-11-3
Starting time: 14:25
```

```
ANON - 14:36
Q: was having difficulty accessing email. what does that say on the
right side of  (lmbda x.E)M) ? E subscript 2?
Priority: N/A
        Konstantin Kuzmin - 14:39
        A: Suscript 1, I belive.
        Ana L. Milanova - 15:16
        A: Yes, it was E_1. I will post the pdf of notes.
```

```
ANON - 14:37
Q: lambda y . w
Priority: N/A
```

```
ANON - 14:38
Q: ooh yes.. parameter y is not used.
Priority: N/A
        Konstantin Kuzmin - 14:41
        A: Yep!
```

```
ANON - 14:38
Q: Z.  Maybe to make this more interactive we can have the chat
channel open to all instead of private
Priority: N/A
        Konstantin Kuzmin - 14:40
        A: Sure, let's give it a shot, and if any students complain
for whatever reason, we can go back to the previous configuration.
```

```
ANON - 14:42
Q: so from the above (lambda x.x) (lambda z.z) -> beta redux to
(lambda z.z)
Priority: N/A
        Ana L. Milanova - 15:17
        A: Yes, that is correct.
```

```
ANON - 14:42
Q: will these notes be posted?
Priority: N/A
```

Konstantin Kuzmin - 14:43
A: You mean the Q&A or chat?
Ana L. Milanova - 15:18
A: Yes, I figured out how to save those notes into a pdf! I will post the notes along with the Lecture17.pptx

---

ANON - 14:43
Q: I mean the notepad Prof. Milanova is drawing on
Priority: N/A
Konstantin Kuzmin - 14:44
A: I need to check with her. At a minimum, you wil have access to the video that I'm recording now...
Ana L. Milanova - 15:18
A: Yes, I will.

ANON - 14:44
Q: Correct me if I'm wrong, but Scheme automatically renames formal parameters, but on the exam and HW, if we're asked to perform substitution or beta reduction, we only rename if necessary (on name classes)?
Priority: N/A
Ana L. Milanova - 15:19
A: Yes, that is correct. This is what we do in all examples of reduction and evaluation in lecture as well.

ANON - 14:44
Q: So in the exam, we can skip the renaming step if it's not needed?
Priority: N/A
Ana L. Milanova - 15:20
A: Correct.

ANON - 14:44
Q: So E[m|x] means replace every occurance of x in E with m? and not Expected value of m given x?
Priority: N/A
Ana L. Milanova - 15:21
A: Correct :). This is one of the existing notations for substitution...

ANON - 14:46
Q: Great, thank you, I just don't remember any subscripts as mentioned in a previous question
Priority: N/A
Ana L. Milanova - 15:22
A: Which subscripts are you referring to?

ANON - 14:48
Q: (i had this question on an example further up) so from the above (lambda x.x) (lambda z.z) -> beta redux to (lambda z.z)
Priority: N/A

Ana L. Milanova - 15:23
A: Yes, that is correct. lambda x.x takes an argument and returns that argument. Argument can be a variable, an application expression, or an abstraction expression.

ANON - 14:50
Q: But in the exam, if we choose to use the aggressive renaming, that will be accepted?
Priority: N/A

Ana L. Milanova - 15:24
A: Yes. But that is a lot more complicated to carry out by hand than just rename when necessary (but it is easy to code, and have the code do the renaming).

ANON - 15:04
Q: whats an example of something not in WHNF?
Priority: N/A

Steven Haussmann - 15:08
A: (λx.x) y would not be in WHNF. In general, you're not in WHNF if you have a top-level application where the left side is an absraction.

ANON - 15:13
Q: So long story short, SKSK = K?
Priority: N/A

Steven Haussmann - 15:18
A: Yes -- it'll become KK(SK) on evaluating the first S, and then K is applied with K and (SK) to produce K

ANON - 15:18
Q: what are the delta rules?
Priority: N/A

Steven Haussmann - 15:22
A: It's a reduction where we apply one of these primitive functions -- iszero, pred, if, etc

---

ANON - 15:23
Q: what is language ML on slide 4?
Priority: N/A

Ana L. Milanova - 15:26
A: ML is one of the most important functional programming languages. There are important dialects that have actually found production use, like OCaml.
Steven Haussmann - 15:27
A: It's short for "Meta Language". The modern flavor is called Standard ML.

ANON - 15:25
Q: This plus would only work if x and y are non-negative, correct?
Priority: N/A
Ana L. Milanova - 15:27
A: Yes, that is correct.

ANON - 15:27
Q: what is the purpose of applied lambda calculus? I'm not sure how it's different beside the notation
Priority: N/A
Ana L. Milanova - 15:28
A: Just a transition step from the pure lambda calculus to a working functional programming langauge. There is a famos saying that (roughly, paraphrasing) "a functional programming langauge is just syntactically-sugared lambda calculus".
Ana L. Milanova - 15:30
A: You can think of the applied lambda calculus as introducing "syntacting sugar" on top of the pure lambda calculus. The friendlier, "sugared" syntax makes the lambda calculus more friendly to program with.

ANON - 15:28
Q: What is meant by the predecessor of x and successor of y?
Priority: N/A
Steven Haussmann - 15:31
A: The predecessor of a number is the number that comes before it (and the successor is the one that follows it)
Ana L. Milanova - 15:31
A: pred(x) is x-1. succ(x) is x+1.
Steven Haussmann - 15:31
A: It's a recursive definition of the integers, starting from 0. you can get every other integer by applying succ or pred

---

ANON - 15:29
Q: I'm confused on what we did for fix M and why we had to how that for the Plus Question.
Priority: N/A
Steven Haussmann - 15:33

A: The key point of fix is to allow us to pass something to itself (repeatedly).
Steven Haussmann - 15:34
A: That lets us have a function that can recurse by calling itself -- and to let that function pass itself a copy of itself, too. This lets it recurse forever.

ANON - 15:31
Q: Are we allowed to copy a bubble sort function from a website if we reference it?
Priority: N/A
Ana L. Milanova - 15:36
A: If I remember correctly, according to the rules we specified in the syllabus, this will not be allowed. I will go back after class and reread those rules.

ANON - 15:35
Q: is type system the way to define types?
Priority: N/A
Steven Haussmann - 15:36
A: A type system is, at a high level, something that decides if your program is well-formed or not.
Steven Haussmann - 15:38
A: Most type systems do have some way to define new types. But there's much more to it than just creating the types.

ANON - 15:42
Q: What is a Type Judgment? Are those the Specified Input Types + Output Types like we saw w/ E1 Sigma -> T?
Priority: N/A
Steven Haussmann - 15:48
A: A judgement produces some result (e.g. that the type of (E1 E2) is τ) given some premises (e.g. that E1 is σ->τ and E2 is σ)
Steven Haussmann - 15:48
A: So, if we had a function that took an int and gave a string, and applied it with an int, the result would be a string

---

ANON - 15:42
Q: I don't completely understand how type systems are used?
Priority: N/A
Steven Haussmann - 15:51
A: A type system decides if a program is valid or not by constraining what you're allowed to do. A very strict type system wouldn't let you add two variables if they weren't known in advance to be numbers, for example.
Steven Haussmann - 15:52

        A: This includes both static and dynamic behavior -- Python has no static type checks, but it will error out if you try to add a string and an integer

---

ANON - 15:46
Q: What is an example of type soundness vs type completeness in practice?
Priority: N/A
        Steven Haussmann - 15:48
        A: Rejecting every program would be type sound; accepting every program would be type complete
        Ana L. Milanova - 15:54
        A: This is more advanced material that we typically cover in more advanced courses, in more detail. It is very interesting and there are many research questions! But you don't in any way have to worry about this being on any exam.
        Ana L. Milanova - 15:55
        A: Sorry, the above answer is the answer to a different question!

ANON - 15:46
Q: Even if lecture 17 is not on Exam 2, could it appear on the Final exam?
Priority: N/A
        Ana L. Milanova - 15:49
        A: No, it won't.
        Ana L. Milanova - 15:56
        A: Sorry, I typed the answer onto a different question. This is more advanced material, and you don't have to worry in any way about any of these being on any exam.

ANON - 15:55
Q: are type judgements specific to envrionment?
Priority: N/A
        Ana L. Milanova - 15:57
        A: The environment is just a mapping from variables to types, e.g, Gamma = [x:int, y:bool]
        Ana L. Milanova - 15:59
        A: The judgements, i.e., rules, make use of Gamma to determine whether construct is type correct or not.

ANON - 16:02
Q: how can x be an int if we're in the nil envrionment?
Priority: N/A
        Ana L. Milanova - 16:03
        A: x is int when we type the nested abstraction \lambda y. x

```
(you see there is a reference to x)
        Steven Haussmann - 16:03
        A: We explicitly said that x is an int in the outermost
abstraction. That's where the binding comes from.
```

```
ANON - 16:04
Q: Does in the NIL Environment mean the Top-Level is initialized to
[]?
Priority: N/A
        Ana L. Milanova - 16:05
        A: Yes, exactly.
```

```
ANON - 16:05
Q: Would attributed grammars be on the next exam?
Priority: N/A
        Ana L. Milanova - 16:05
        A: There might be a question on attribute grammar over the
lambda calculus. SImilar to the one in one of the practice problem
sets.
        Ana L. Milanova - 16:06
        A: I meant, over the lambda calculus grammar.
```

```
ANON - 16:06
Q: what are these extensions? how do we interpret this?
Priority: N/A
        Ana L. Milanova - 16:07
        A: E1 + E2 is just the rule for addition expressions. E.g.,
x+y.
```

```
ANON - 16:16
Q: what does "stuck state" mean
Priority: N/A
        Ana L. Milanova - 16:22
        A: An erroneous state. A state where we'll apply an operation
on a value of the wrong type. If we have a bool-to-int function, and
program reaches a state where we try to apply that function on an int,
that will be a "stuck state".
```

```
ANON - 16:17
Q: Why was it (int -> int) -> int -> int for one of the Last Examples
and not ((int->int)->int)->int?
Priority: N/A
        Steven Haussmann - 16:18
        A: (int -> int) -> int -> int means "takes an int -> int
function, then an int, and gives an int"
```

        Steven Haussmann - 16:18
        A: ((int -> int) -> int) is a function that takes a function going from int to int and gives you an int

---

ANON - 16:19
Q: Do we have 2 hrs or 2 hrs and 15 minutes for the exam?
Priority: N/A
        Ana L. Milanova - 16:23
        A: It should be 2 hours. If there are Submitty issues at any time, we will extend the time frame, as we did last time.

ANON - 16:19
Q: But I thought (int -> int) -> int -> int would be interpreted Right-Associativly, so it would be interpreted as ((int->int)->(int->int))
Priority: N/A
        Steven Haussmann - 16:21
        A: Being right associative means that we could parenthesize it like (int -> int) -> (int -> int). I missed the exact example you're referring to, though, so I'm not sure what it was
        Ana L. Milanova - 16:24
        A: Yes, that is correct. (int->int) -> int -> int is the same as ((int->int) -> (int->int)).
        Ana L. Milanova - 16:25
        A: We do need the parens around the first (int->int), without those parens, the term will be int-> (int->(int->int))

ANON - 16:19
Q: Is the friday lecture a review session?
Priority: N/A
        Ana L. Milanova - 16:25
        A: Yes, review, and problem solving. We will probably go over some of the problem sets I posted, like we did last time.

ANON - 16:19
Q: when are Prof. Milanova's next OH since the Friday's OH were postponed?
Priority: N/A

ANON - 16:22
Q: So does (int -> int) -> int -> int mean it takes a function value f that takes an int, returns int as input and returns another function value that takes an int and returns an int?
Priority: N/A
        Ana L. Milanova - 16:26

A: This one means the term takes a function value f of type int->int, and returns a function value that takes an int and returns an int.

ANON - 16:22
Q: The Example I was referring to was the \L f. \L x. if x = 1 then x else (f (f (x-1))) : ?
Priority: N/A

Ana L. Milanova - 16:28

A: Yes, I thought so... So this is a function, and it takes a function f, which, we deduced should be of type int->int. It returns another function (the \L x. term). That function takes x of type int, and returns an int.