Q&A Session for Programming Languages Lecture 8

Session Number: 1209524436
Date: 2020-9-29
Starting time: 14:25

ANON - 14:31
Q: My Webex cut out, what announcements did you make at the beginning?
Priority: N/A
        Ana L. Milanova - 14:31
        A: None.
        Ana L. Milanova - 14:32
        A: Just intro to Lecture 8, we'll do Q&A in between and at the end of the stream.

ANON - 14:39
Q: I am getting a weird effect in prolog when using logical or (';'). In an instance where one statement has one condition true and one condition false, it returns true then false. Splitting it didn't work. What should I do?
Priority: N/A
        Ana L. Milanova - 14:41
        A: We can take some Prolog questions, but I'd like to do this at the end of class. (For your question here, just quickly, one common mistake is that the precedence of AND is higher than precedence of OR. So C,A;B is (C,A);B not C,(A;B).)
        Steven Haussmann - 14:47
        A: you may be thinking of ; as an operator that combines two booleans, like "true or false == true". It doesn't; it offers two choices. Prolog will try the first one, and then, if it has to backtrack and try something else, it will try the second one.
        Ana L. Milanova - 16:25
        A: Ok, now on reading the question again, I think this is the expected behavior, no? Assuming you have A;B and A is true but B is false, Prolog tries one A, issues true, then upon ; it goes the other branch and issues false. Adding A,!;B will exit with true.
        Ana L. Milanova - 16:26
        A: And won't try the B branch. (If this is what you want to do.)

ANON - 14:42
Q: I didn't use any and statement. It is just an or, no ands in the function whatsoever.
Priority: N/A
        Ana L. Milanova - 14:43
        A: The , is an AND.

```
____________________________________________________________

ANON - 14:43
Q: I didn't use a ',' , I used ';' only
Priority: N/A
        Ana L. Milanova - 14:44
        A: Then I don't know what the issue is, but like I said we'll
take Prolog questions later
____________________________________________________________

ANON - 14:44
Q: What do you mean by dispatch?
Priority: N/A
        Ana L. Milanova - 14:45
        A: When you have a Virtual Call, then dynamic dispatch
happens. E.g., in Java if you have a.m(), and say "a" can be an A, B,
or a C object, each declaring a method m(), then the actual m()
(A.m(), B.m() or C.m()) is dispatched dynamically, based on the
        Ana L. Milanova - 14:45
        A: runtime type of the receiver object that a refers to at
runtime.
____________________________________________________________

ANON - 14:44
Q: What did the 13.7% Percent Represent? I heard that the 5.2% is the
Time for Dispatch in C++ Code, but I couldn't make out what the 13.7%
was.
Priority: N/A
        Ana L. Milanova - 14:46
        A: That is essentially what happens in Java. In Java all
instance calls are virtual. In C++ calls can be direct, unless
declared virtual, so we have a lot less dynamic dispatch.
____________________________________________________________

ANON - 14:46
Q: In stactic scoping, why isn't there a search for declaration of a
variable in other functions? Can it only search within itself and
within main?
Priority: N/A
        Ana L. Milanova - 14:47
        A: It searches from innermost to outermost enclosing blcoks.
More on this later in lecture today!
        Steven Haussmann - 14:49
        A: You wouldn't want to have declarations in distant parts of
your program interfering with your code!
____________________________________________________________

ANON - 14:46
Q: Does type erasure in Java generics introduce dispatch overhead in
classes without any special inheritance (other than Object)
```

Priority: N/A
        Steven Haussmann - 14:50
        A: Type erasure has no runtime overhead, according to Java's documentation.
        Ana L. Milanova - 14:53
        A: Yes, what Steven says. Generics were added to Java much later in Java 5 and type erasure was needed to for backward compatibility (if I remember correctly). I think it may introduce some overhead indirectly, due to lost opportunity for optimization.

ANON - 14:50
Q: Why is the sp Pointing to Lower Memory if we said that Stack goes from Higher to Lower Memory?
Priority: N/A
        Steven Haussmann - 14:54
        A: The stack pointer points to the top of the stack -- the thing that was pushed onto the stack most recently. It's in lower memory because the stack grows downward from high memory.

ANON - 14:52
Q: On slide 5 part 3 when you were defining the scope of the variables. For S, are you saying that S can call itself? Why would S need to know where S is defined?
Priority: N/A
        Ana L. Milanova - 14:55
        A: Yes, but that's Part 3...

ANON - 14:53
Q: The second photo she drew talks about dyanmic memory yes?
Priority: N/A
        Ana L. Milanova - 14:56
        A: On slide 16, I drew just the runtime stack. It's often drawn in two different ways: to grow from the bottom towards the top of the page, and the other way around, from the top towards teh bottom of the page.

ANON - 14:54
Q: Why do we need both the frame pointer and the stack pointer?
Priority: N/A
        Steven Haussmann - 14:56
        A: The frame pointer tells us where the stack pointer was before a function was called. You need that to find things like arguments on the stack.
        Steven Haussmann - 14:56
        A: I believe it's called the base pointer in x86 - ebp or rbp.

        Ana L. Milanova - 14:57
        A: Yes, rbp in x86 64 is the base pointer, that's the beginning of the frame (roughly) and rsp, that's end end of the frame (again roughly speaking).

ANON - 15:00
Q: ok no problem. Thank you.
Priority: N/A

ANON - 15:01
Q: So both the frame and stack pointer grow towards lower memory, but the difference is that one starts at a higher memory location then the other? I'm just trying to understand the distinction.
Priority: N/A
        Steven Haussmann - 15:01
        A: When you call a function, the old frame pointer is pushed onto the stack, and then the frame pointer is set to be the current stack pointer.
        Steven Haussmann - 15:02
        A: Then, as you make space on the stack for your function, your stack pointer grows downward. So, yes, both of them will move towards lower memory as functions are called.

ANON - 15:04
Q: i dont remember it
Priority: N/A

ANON - 15:13
Q: That is interesting that the man pages has that warning about gets(). What is the point of having the function if it is telling us not to use it?
Priority: N/A
        Steven Haussmann - 15:15
        A: It's kept around so that old programs continue to work. It has been deprecated for quite a while.
        Ana L. Milanova - 15:16
        A: Yes, it was removed from the standard but most impelmentations still supporte it. For compatibility reasons.
        Ana L. Milanova - 15:18
        A: gets is very useful if you do binary exploitation :).

ANON - 15:16
Q: Is it correct to say that the new fp gets the old sp when additional frames are added to the stack?
Priority: N/A

Steven Haussmann - 15:16
A: Yes, you store the old frame pointer, then set the frame pointer to the current stack pointer.

ANON - 15:22
Q: what is NBE
Priority: N/A
Ana L. Milanova - 15:25
A: MBE = Modern Binary Exploitation. This is an RPI class that was developed by RPISEC.

ANON - 15:25
Q: with gets(), what are we overriding when we go over the buffer? The memory on the stack with larger memory locations (up until the fp)?
Priority: N/A
Ana L. Milanova - 15:28
A: Yes, if our buffer is at address A, then we can override as many bytes after A as we want. Most importantly, we can get to the return address, and hijack execution by giving the address of some code of ours.
Steven Haussmann - 15:28
A: You don't have to stop at the frame pointer, either. That's just a point in memory that the CPU is keeping track of; you can keep clobbering past that.
Ana L. Milanova - 15:28
A: And we can go past the old fp, and corrupt other frames that are on the stack.

ANON - 15:33
Q: Regarding gets(), doens't C automatically cuts off any additional characters (data) that is outside of the buffer size?
Priority: N/A
Ana L. Milanova - 15:34
A: Nope. That's C and gets() doesn't cut and that's why it's so unsafe.
Steven Haussmann - 15:35
A: More generally, the size of a buffer isn't stored anywhere. gets() just sees a pointer.

ANON - 15:39
Q: How is the gets() Function able to corrupt the values that are below (i.e., High memory) since the Stack grows from Higher to Lower Memory so then how is it able to reach those Higher Data when it grows in the other direction?
Priority: N/A

```
        Steven Haussmann - 15:40
        A: If you read n bytes of data in with gets() starting at
location p, then you will write between addresses (p) and (p + n). So,
you will clobber thing sthat are in higher memory -- thus, things that
are deeper in the stack
```

```
ANON - 15:40
Q: I'm confused by what's meant by a hole in scope
Priority: N/A
        Ana L. Milanova - 15:42
        A: I'll get back to this question afterwards. But quickly,
suppose you have an enclosing procedure P that declares a variable
"a". Then "a" is "in scope" in P, and in the procedures enclosed in P.
        Ana L. Milanova - 15:43
        A: Now suppose we have a procedure Q that's nested in P, which
declares it's own variable "a". Then We say that Q introduces a hole
in the scope of P's "a", because P's "a" is shadowed by Q
```

```
ANON - 15:48
Q: Why is it not S.c for the last Dynamic Scoping Question?
Priority: N/A
        Ana L. Milanova - 15:57
        A: Yes, as you point out later in the Q&A box. Because dynamic
scoping looks at the current stack, starting from the most recently
added frame.
```

```
ANON - 15:49
Q: So the hw2 format we submit should be zip file, right?
Priority: N/A
        Steven Haussmann - 15:55
        A: Yes, you'll bundle everything up into a single zip file and
submit that.
        Ana L. Milanova - 16:37
        A: You can definitely zip, but it is not necessary. You can
just drag the individual files into the box and Submitty will grade
fine. The only hard constraint is that you use the file names as
specified, foxes_and_hens.pl and parser.pl.
```

```
ANON - 15:49
Q: Any tips on finding the right locations to cut? I'm having issues
getting rid of repeated solutions on Problem 1 of the HW
Priority: N/A
        Steven Haussmann - 15:52
        A: You should cut if you're in a situation where more than one
option can be satisfied with the same inputs. For example, if you have
`(A < B ; B == 0)`, then A = -1 and B = 0 will satisfy twice. Changing
```

it to `(A < B, ! ; B == 0)` would fix that.

ANON - 15:52
Q: Never mind to the Above Question. It seems to because Dynamic Scoping looks for the Closest/Most Recently Invoked Predecessor so P() would be the first to have Variable C, not S.
Priority: N/A

ANON - 15:56
Q: on the hw first question. to find if a generated configuration is valid of course, we only need to search one move back. is this something where ! would be used?
Priority: N/A

Steven Haussmann - 15:58
A: Cutting prevents prolog from backtracking past whatever was affected by the cut operator. You don't want that -- you want to be able to go back all the way to the beginning to look for other solutions, if necessary.

Steven Haussmann - 15:59
A: maybe I'm misunderstanding you, though, since you mentioned checking if it's valid, not finding solutions. Validity should already be enforced by the predicates that build up the list of moves.

ANON - 15:56
Q: For question 2, what should we do to differentiate the type checking between '-' and '*'?
Priority: N/A

Ana L. Milanova - 16:01
A: I'm assuming you mean in "transform". Something like this should work: transform([-|T],[term(minus,_)|T1]) :- ... % here compute the new tail T1 from the old tail T. Similarly for *: transform([*|T], [term(times,_)|T1]) :- ...

Ana L. Milanova - 16:38
A: I might be misunderstanding you, if so, post on Submitty.

ANON - 15:58
Q: Would you need to use cut if you have (A < B, B==0) and your inputs satisfy both? (same as steven's example for cut above but with and instead of or). Or will it not do the second one since it's an and instead of an or
Priority: N/A

Steven Haussmann - 15:59
A: No, since there aren't multiple ways for this to be satisfied.

ANON - 16:00
Q: I am a bit confused about the attribute function and like parse and solve? I have parseLL but I am a bit confused about what attribute is supposed even do?
Priority: N/A
        Ana L. Milanova - 16:39
        A: There is a long post on Submitty that guides you through. But high-level picture, "attribute" adds some meaning/interpetation to the string. As opposed to parsing, which just checks syntactic well-formedness.

ANON - 16:03
Q: Would it be possible if the homework be due at 11:59:59 pm Friday opposed to 1:59:59 ?
Priority: N/A

ANON - 16:04
Q: So there is no end of file in our grammar correct?
Priority: N/A
        Ana L. Milanova - 16:40
        A: There is no explicit "$$" in the grammar. But $$ is usually implicit, and we'll have to add it in the implementation of the parser.

ANON - 16:05
Q: Would it be possible if the homework be due at 11:59:59 pm Friday opposed to 1:59:59 ?
Priority: N/A

ANON - 16:05
Q: Is there a difference between the '=' operator and the 'is' operator in Prolog?
Priority: N/A
        Steven Haussmann - 16:06
        A: Yes. The "is" operator will try to evaluate the mathematical expression on the right side, rather than just unifying the two sides. It only works with math expressions, and ensures that the expression gets evaluated first
        Steven Haussmann - 16:06
        A: X = 2 + 3 will bind 2 + 3 to X; X is 2 + 3 will bind 5 to X

ANON - 16:08
Q: yes it does.
Priority: N/A

ANON - 16:09
Q: So basically we have to use a particular string that signifies the end of input?
Priority: N/A
        Ana L. Milanova - 16:09
        A: Yes, use term(end,_) to denote the end-of-input.

ANON - 16:09
Q: character not string .
Priority: N/A

ANON - 16:09
Q: Any suggestions for debugging in Prolog?
Priority: N/A
        Ana L. Milanova - 16:10
        A: You can use trace.
        Ana L. Milanova - 16:10
        A: Just before you start your function, say solve(P), start trace.
        Steven Haussmann - 16:11
        A: Testing the behavior of individual predicates is useful for chasing down bugs. For example, you can check that part 1's predicate for "is this a valid state to be in?" accepts and rejects things correctly
        Ana L. Milanova - 16:11
        A: But generally, look for where you have unbound variables, e.g., if you do append([1,2],X,Y] where X and Y are both unbound. That's when things usually go wrong, when we ahve unbound variables. trace can help you see if this happens in your code.

ANON - 16:10
Q: just to make sure, for that attribute thing we can run that after parseLL is called we don't have to rewrite parseLL to incorporate attribute in it?
Priority: N/A
        Ana L. Milanova - 16:12
        A: You have to add the call to attribute in your parseLL, at about the same line where your call to prod is... Then rename your parseLL into parseAndSolve and that's it.
        Ana L. Milanova - 16:13
        A: You just have to be careful to bind the arguments of "attribute" and "prod" to the same value and relate to the parsing stack.

ANON - 16:10

Q: So the cut operator won't let it backtrack until it backtracks all the way back to the beginning of the program or will it only stop backtracking where the cut operator is?
Priority: N/A
        Steven Haussmann - 16:12
        A: The cut operator prunes choice points that come before the cut in the current predicate. So, any choices that have been made within that predicate are locked in.

ANON - 16:12
Q: I'm not entirely sure how to append additional lists to P. (i.e. I can initialize a list and make a single move, but now I'm not sure how to proceed).
Priority: N/A
        Steven Haussmann - 16:13
        A: Remember that prolog will find bindings for variables to satisfy whatever constraints you give it. If you want to put a on the head of a list P, then append([a], P, Out) will bind the combined list to Out
        Steven Haussmann - 16:13
        A: You can also say Out = [a | P], of course.

ANON - 16:14
Q: when you use "Solve(P)" in the program, by the end of execution, P will be a list of lists, each list being a configuration?
Priority: N/A
        Steven Haussmann - 16:14
        A: Yes.

ANON - 16:14
Q: So for appending additional lists to P, would I mkake two cases then? One that satisfies the final condition of part 1 and one that doesn't?
Priority: N/A
        Ana L. Milanova - 16:44
        A: You will have two versions of "search", the base case that checks for [0,0,0] and exits when position is [0,0,0], and the recursive case, that continues the search. Slide 28 in Lecture 7 outlines the "search" predicate.

ANON - 16:15
Q: My transform is adding spaces and random numbers to the terms added to the list example: R = [term(num, 3), term(minus, _2462), term(num, 5), term(end, _2486)]. What is happening?
Priority: N/A
        Ana L. Milanova - 16:19

A: Yes, that is correct behavior. Each "term" has space for the attribute value, but we don't care about the attribute values of minus and times, so we have those don't care variables (e.g., _2486). We do care about the attribute values of "num".