CSCI 334:
Principles of Programming Languages

Lecture 14: Program interpretation

Instructor: Dan Barowy

Williams

Your to-dos

1. Read *Parsing* if you have not already done so.
2. Lab 8, **due Wednesday, April 16** (partner lab).

Announcements

- **Ellie Pavlick, Brown University**
- **Not-Your-Mother's-Connectionism: LLMs as Cognitive Models**, **Fri, Apr 18 @ 2:35pm in Wege Auditorium**

Recent advances in AI have led to large neural network models which exhibit human-like behavior across a range of language and reasoning tasks. This (re-)opens important theoretical questions about the nature of the structure that is required to support such behaviors, leading to debates reminiscent of long-running arguments that pit neural network models against explicitly structured symbolic models of the mind. In this talk, I will describe a series of experiments which highlight the ways in which LLMs today appear importantly different from the connectionist systems that inspired these debates originally. I will argue for a more nuanced stance which does not assume neural networks to be diametrically opposed to traditional models of the mind, but still acknowledges the potential of LLMs to teach us something fundamentally new about the structures that govern language and cognition in humans.

Ellie Pavlick is an Assistant Professor of Computer Science and Linguistics at Brown University, and a Research Scientist at Google Deepmind.

Announcements

Please **consider being a TA** next semester (especially for this class!)

Applications **due Friday, April 18**.

https://csci.williams.edu/tatutor-application/

Announcements

Please **consider providing TA feedback**

Feedback **due Friday, April 18**.

https://forms.gle/sbqCGVLAFnhUQ4i39

Announcements

- **Midterm exam**, in class, Thursday, May 8.

Final project timeline

1. ~~Brainstorm (Lab 7), due Wed 4/9~~
2. Project Proposal (Lab 9), **due Wed 4/23**
3. Minimally working version (Lab 10), **due Wed 4/30**
4. Final project + video presentation (Lab 11), **due Wed 5/14**

Student final projects in this class are routinely nominated for the **Ward Prize.**

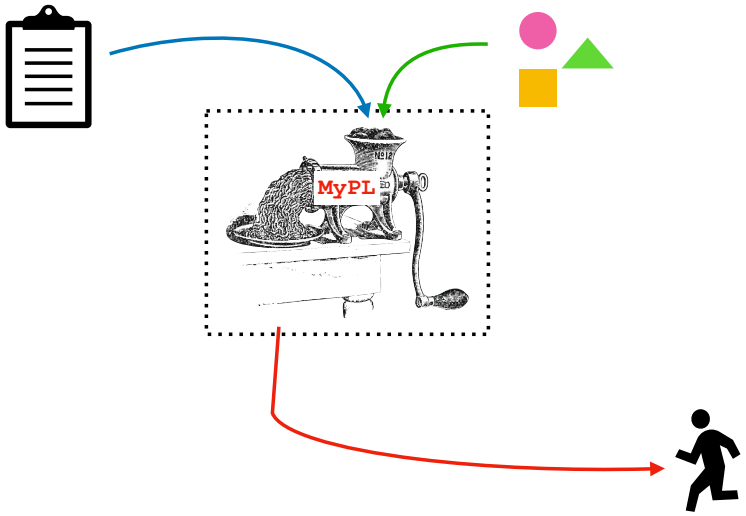
Topics

Demo

Program interpretation

What is a programming language?

What is a programming language?

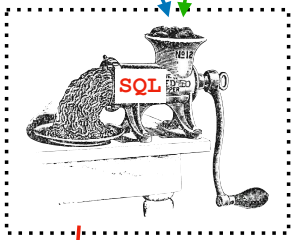


What is a programming language?

```
SELECT * FROM Employee
WHERE EmpId > 3000
```

Employee

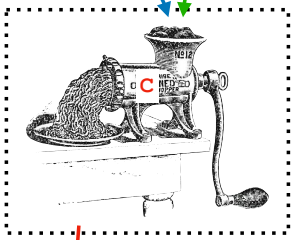
Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales

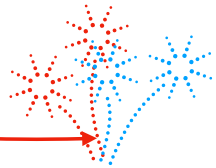


Name	EmpId	DeptName
Harry	3415	Finance
George	3401	Finance

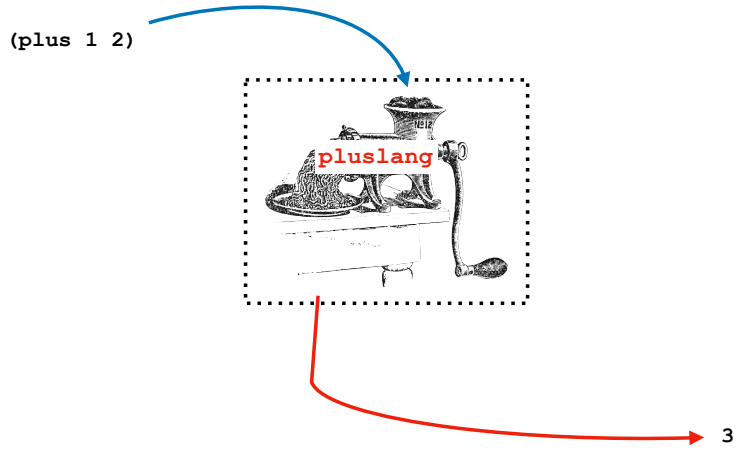
What is a programming language?

```
int main() {…}
```

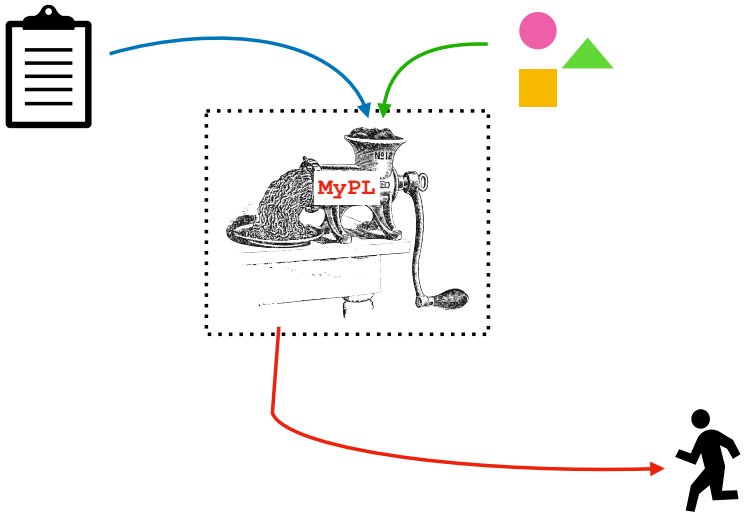




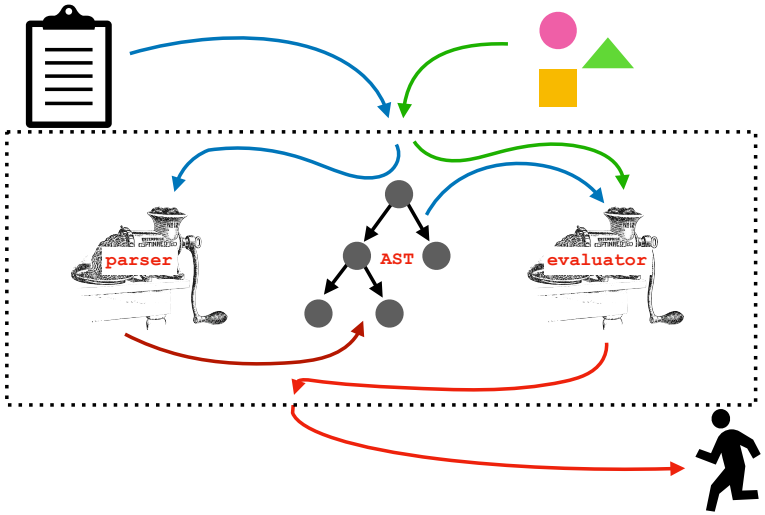
What is a programming language?



What is a programming language?



What is a programming language?



Program Interpreter

Program Interpreter

A **program interpreter** is a computer program that "interprets" given statements or expressions in a programming language. Unlike a compiler, an interpreter **directly** carries out the instructions implied by user code, usually by traversing an **abstract syntax tree** and carrying out the sequence of operations discovered during the traversal.

pluslang

```
<expr> ::= (plus <expr> <expr>)
         | n ∈ ℕ
```

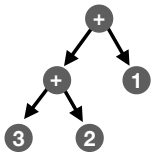
```
(plus 1 2)

(plus (plus 1 2) 3)

(plus (plus 1 2) (plus 3 4))
```

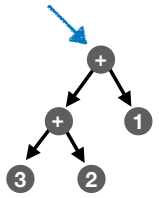
Example

```
(plus (plus 3 2) 1)
```



Example

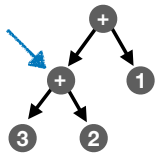
```
(plus (plus 3 2) 1)
```



Eager evaluation: usually a **post-order traversal** of an AST.

Example

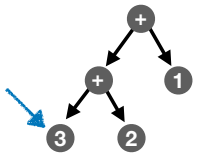
```
(plus (plus 3 2) 1)
```



Eager evaluation: usually a **post-order traversal** of an AST.

Example

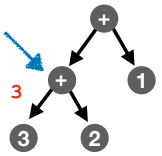
```
(plus (plus 3 2) 1)
```

Eager evaluation: usually a **post-order traversal** of an AST.

Example

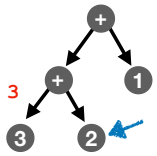
```
(plus (plus 3 2) 1)
```

Eager evaluation: usually a **post-order traversal** of an AST.

Example

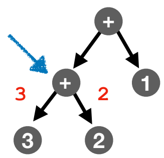
```
(plus (plus 3 2) 1)
```

Eager evaluation: usually a **post-order traversal** of an AST.

Example

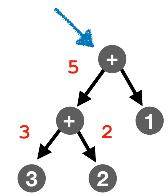
`(plus (plus 3 2) 1)`



Eager evaluation: usually a **post-order traversal** of an AST.

Example

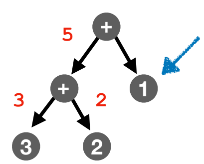
`(plus (plus 3 2) 1)`



Eager evaluation: usually a **post-order traversal** of an AST.

Example

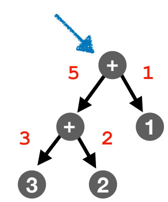
`(plus (plus 3 2) 1)`



Eager evaluation: usually a **post-order traversal** of an AST.

Example

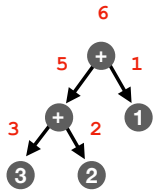
`(plus (plus 3 2) 1)`



Eager evaluation: usually a **post-order traversal** of an AST.

Example

```
(plus (plus 3 2) 1)
```



Eager evaluation: usually a **post-order traversal** of an AST.

This traversal is conveniently written as a **recursive function**.

Let's write the interpreter

But we will write the parser first

Recap & Next Class

Today:

Program interpretation

Next class:

A graphical language