Little Languages

Lecture 32:

The Big Picture

We will be on the VM today. Go ahead and pull.

Also run: **sudo apt install jq**

Curious what you installed? **man jq**

What is *fundamental* versus *fad*?

- Time is a fine filter.
- The list of programs and software tools produced over the past 50 years is unbelievably long. The majority have not stood the test of time.
- Tools whose abstractions have not improved since invention, but are still used widely in the real world today, illuminate fundamentals.
- The tools and ideas we've spent time with in this course were first conceived 50+ years ago. If you go to any major technology company and ask "is anyone here making use of X", the answer is either "yes" or "no, but we use Y which is the X of our platform".

Structured Language is a Foundation of CS

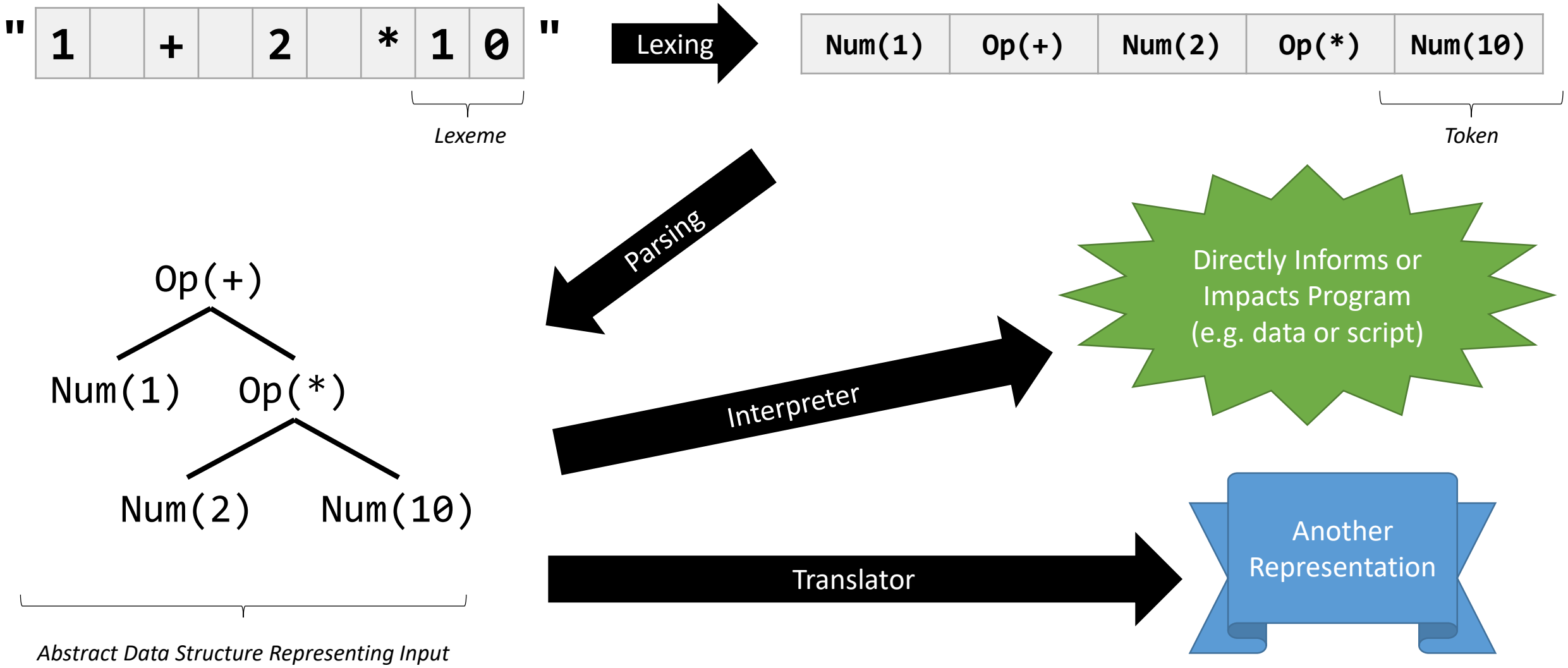
- Structured languages are the most effective means humanity has to:

1. Express algorithms interpretable by both humans and machines

2. Convey data between humans and machines

- Knowing how to work with and think in terms of structured language has unlimited applications.

The Big Picture



Old Ideas, Modern Context: **nodejs** and **TypeScript**

- Let's first look at an interpreter: nodejs is a JavaScript interpreter

```
$ node
> function f(i, j, k) { return i + j * k; }
> f(1, 2, 3)
7
```

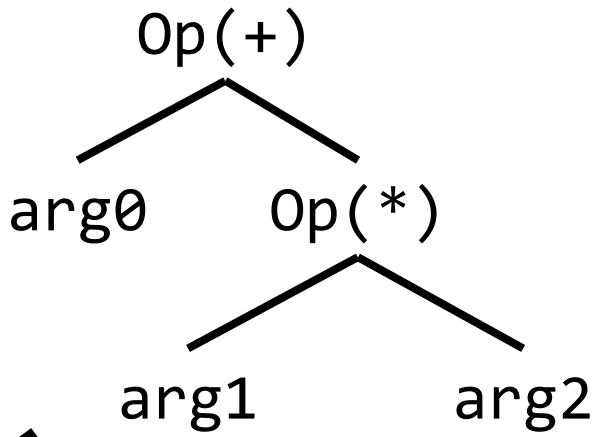
- As we input JavaScript, node is directly interpreting it and executing it.

node's interpreter (V8) is really a JIT compiler

- The interpretation in nodejs is non-trivial. It's JavaScript engine, V8, actually compiles (translates) your code into "bytecode" then machine code, *then* finally executes it.
- This is called Just-in-Time Compilation.

```
$ node --print-bytecode
... press enter, keys, and backspace a few times ...
> function f(i, j, k) { return i + j * k; }
> f(1, 2, 3)
...
[generating bytecode for function: f]
Parameter count 4
Frame size 0
   10 E> 0x2acbc93b21ea @    0 : 91             StackCheck
   22 S> 0x2acbc93b21eb @    1 : 1d 02          Ldar a2
   35 E> 0x2acbc93b21ed @    3 : 2d 03 00       Mul a1, [0]
   31 E> 0x2acbc93b21f0 @    6 : 2b 04 01       Add a0, [1]
   39 S> 0x2acbc93b21f3 @    9 : 95             Return
...
7
```

Lexing, Parsing



Translation

TypeScript is a Translator

- JavaScript projects are messy at scale because of dynamic typing
  - Goal: Add strong type checking to JavaScript.
  - Challenge: The de facto language of web browsers is JavaScript. It is standardized. Improvements to JavaScript itself take an absurdly long time to "land" and making such a fundamental change verges on politically impossible.
  - Solution: Invent a language that compiles to JavaScript
    - TypeScript's "compiler" emits JavaScript
- This is not a new idea. Stroustrup's first implementation of C++ converted C++ to C.
  - In fact, many programming language's first implementations translate to C first.
- This is not a new idea. Lorinda Cherry's **bc** language translated to **dc**.
- This is not a new idea. Every compiler translates source code to some other kind of code "closer" to the machine.

Translating TypeScript - 1/2

- Open a new TypeScript file named example.ts, and add:

```
function f(i: number, j: number, k: number): number {
  return i + j * k;
}
```

- Compile this program to JavaScript with the TypeScript Compiler tsc

$ tsc example.ts

- Open up the resulting JavaScript file, example.js, for inspection.

Translating TypeScript - 2/2

- Add the following class to the TypeScript file:

```
class Dog {
  private name: string;
  constructor(name: string) { this.name = name; }
  getName(): string { return this.name; }
}
```

- Compile this program to JavaScript with the TypeScript Compiler tsc

$ tsc example.ts

- Open up the resulting JavaScript file, example.js, for inspection.
- Check-in on PollEv.com/compunc once you've checked it out.

JSON and jq (1 / 2)

- If you didn't at start of lecture: **sudo apt install jq**

- What is jq? It's *sed* for *JSON*.
  - In industry you'll see the it's <Old Utility> for <New Technology> pattern often.
  - Maven is *make* for *Java*.

- Let's grab a JSON file:

```
$ curl 'https://api.github.com/repos/comp590-19s/590-material/commits' > commits.json
```

- Open it to take a look…

JSON and jq (2 / 2)

- The authors of jq built a little language for extracting and transforming data from JSON.

- Some examples to try on this data…

- $ jq '.[] | [.commit.message, .commit.author.date]' commits.json

- $ jq '[.[] | {message: .commit.message, date:.commit.author.date}]' commits.json

- For reference on the discussion in class: https://stedolan.github.io/jq/tutorial/


```
[
  {
    "sha": "06003993faa45db7d0461ff0e3e1aaa175a74be0",
    "commit": {
      "author": {
        "name": "Kris Jordan",
        "email": "kris@cs.unc.edu",
        "date": "2019-04-15T16:40:05Z"
      },
      "committer": {
        "name": "Kris Jordan",
        "email": "kris@cs.unc.edu",
        "date": "2019-04-15T16:40:05Z"
      },
      "message": "rename 31",
 ...
  }, ...
]
```

Strengthened Foundational Skills

As a computer scientist, data scientist, software engineer, and so on, having developed these skills this semester are invaluable:

- Command-line and Text Editor Comfort
- Understanding of the Process Model
- Test-driven Development
- Dependencies and Build Tools
- Regular Expression Fluency
- Grammar Comprehension