ECS 140A Programming Languages

Aditya Thakur

https://thakur.cs.ucdavis.edu

ECS 140A Programming Languages

$\boxed{\text{Aditya}}$ Thakur

ECS 140A Programming Languages

Prof. Aditya Thakur

ECS 140A Programming Languages

$\boxed{\text{Prof.}}$ Aditya Thakur

Course Objectives

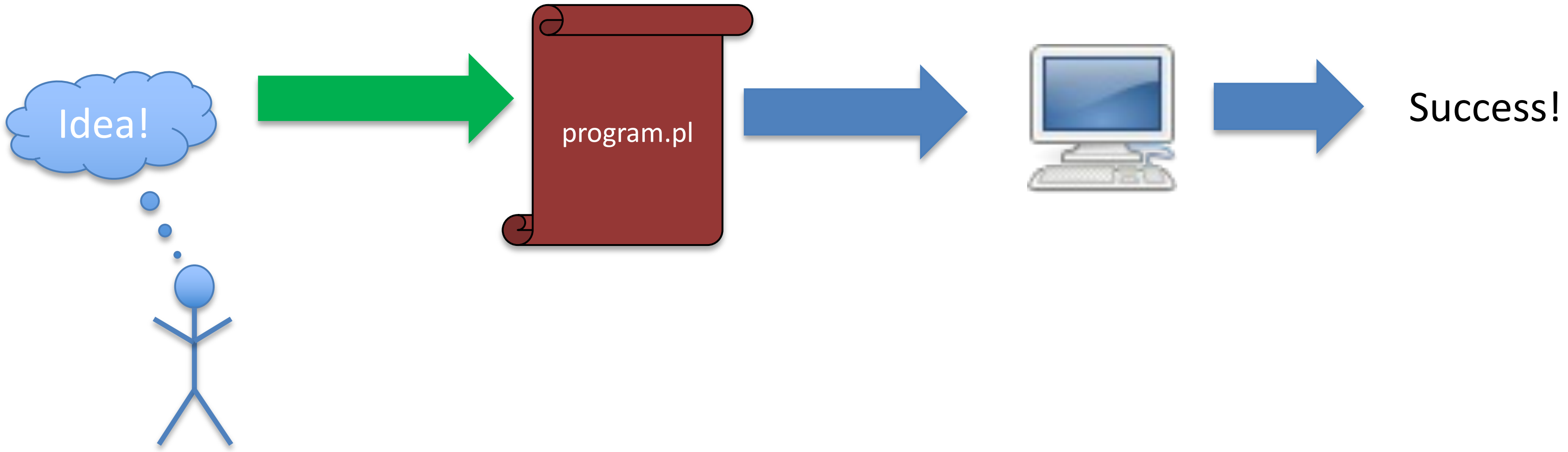
- Learn the fundamental principles of modern computer programming languages
- Learn about different programming languages and what each is specifically good for
- Gain programming experience in the selected languages: Go, Lisp, and Prolog.

What is a Programming Language?

"A language for programming"

Set of rules specifying valid **syntax** and its associated **semantics**

Make a computer perform a particular task



What is a Programming Language?

"A language for programming"

Set of rules specifying valid **syntax** and its associated **semantics**

Make a computer perform a particular task

- The *syntax* of a programming language defines *how* programs look: their form and structure
- The *semantics* defines *what* programs do: their behavior and meaning

What is a Programming Language?

"A language for programming"

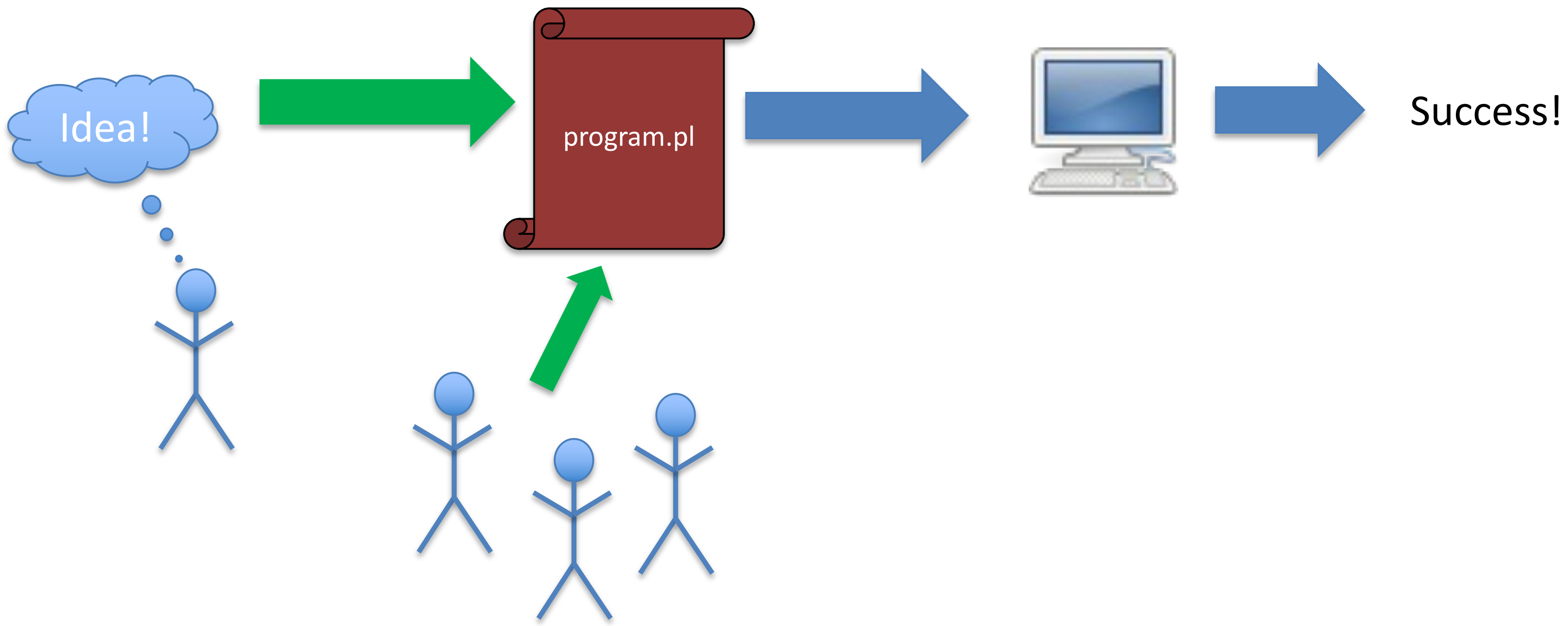
Set of rules specifying valid **syntax** and its associated **semantics**

Syntax and semantics should be formally defined; specification has to be precise and unambiguous.

Make a computer perform a particular task

Efficiently execute the program on the computer hardware.

What is missing in this picture?



What are the properties of a “good” program?

Language Design Criteria

- Readability
- Writability
- Simplicity
- Expressivity
- Efficiency of execution
- Safety/Correctness
- …

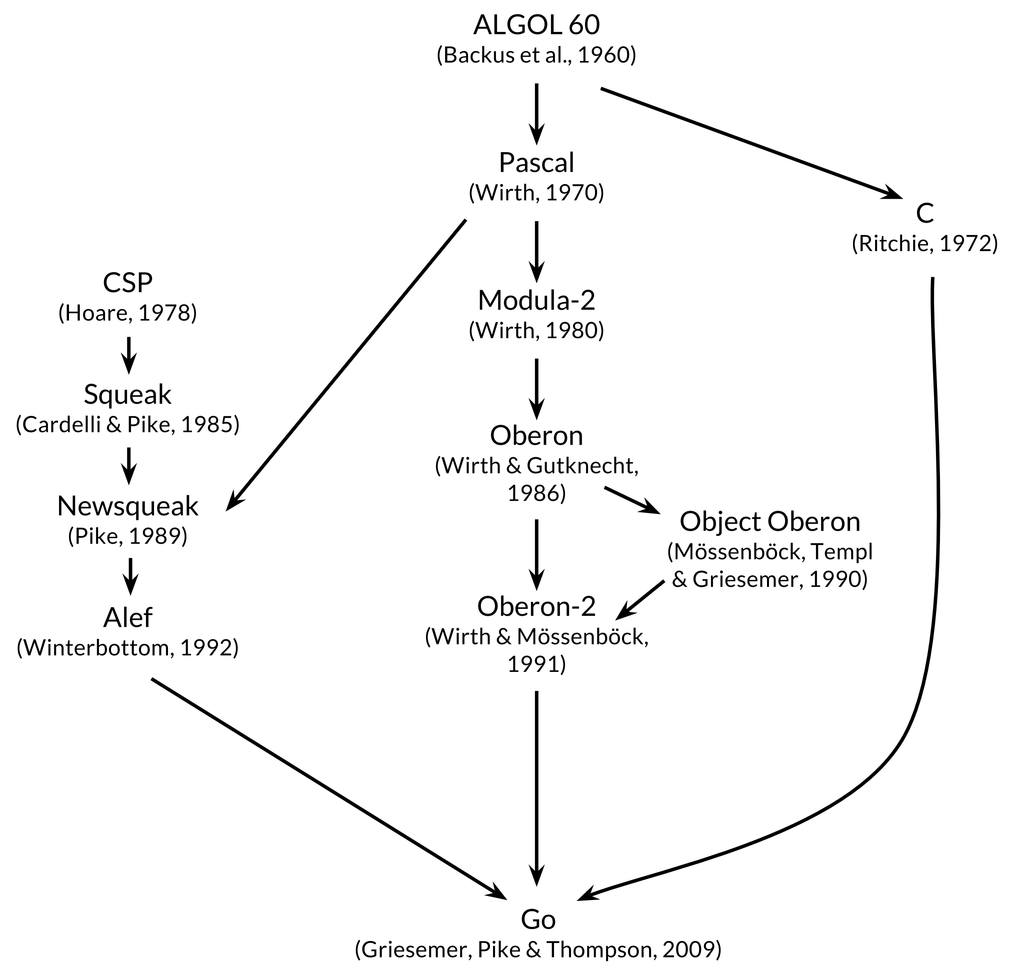
Often these criteria conflict with each other.

Managing these trade-offs leads to different programming languages.

Why so many languages?

- Languages developed in industry
  - Fill a business need
  - Apple → Objective-C, Swift; Mozilla → Rust; Facebook→ Hack; Google→Go; Microsoft→C#
- Languages developed to fit an application domain
  - Artificial intelligence: symbolic computation (Lisp, Prolog)
  - Scientific Computing: high performance (Fortran)
  - Business: report generation (COBOL)
  - Systems programming: low-level access (C)
- Languages combining features of previous languages

Origins of Go



Why so many languages?

- Languages developed in industry
  - Fill a business need
  - Apple → Objective-C, Swift; Mozilla → Rust; Facebook→ Hack; Google→Go; Microsoft→C#,F#,JScript
- Languages developed to fit an application domain
  - Artificial intelligence: symbolic computation (Lisp, Prolog)
  - Scientific Computing: high performance (Fortran)
  - Business: report generation (COBOL)
  - Systems programming: low-level access (C)
- Languages that combine features of previous languages
- Languages are still evolving
  - Java 18, C++20, go1.19

Programming Language Paradigms

Imperative

- Syntax: sequence of commands
- Semantics: updates to program state
- Examples: C, C++, Go, Cobol, Fortran, Java

Functional

- Syntax: composition of functions
- Semantics: evaluation of mathematical functions
- Examples: Lisp, Haskell, Ocaml

Logic

- Syntax: set of constraints or rules
- Semantics: constraint satisfaction via search
- Examples: Prolog

Programming Language Paradigms

Go

```
func foo(n int) int {
   f := 1
   for i:=1; i<=n; i++ {
      f *= i
   }
   return f
}
```

Programming Language Paradigms

Go

```
func factorial(n int) int {
   f := 1
   for i:=1; i<=n; i++ {
      f *= i
   }
   return f
}
```

Lisp

```
(defun fact (x)
   (if (<= x 0) 1 (* x (fact (- x 1)))))
```

Prolog

```
fact(X,1) :-   X =:= 1.
fact(X,Fact) :-
   X > 1,
   NewX is X - 1,
   fact(NewX, NF),
   Fact is X * NF.
```

Programming Language Paradigms

Object-oriented

- Based on the concept of *objects* that encapsulate the data and the methods acting on that data.
- Examples: Smalltalk, Java, C++, Go (sort of)

```
public class MyInt {
   private int value;

   public MyInt(int value) {
      this.value = value;
   }

   public int getValue() {
      return value;
   }
```

```
public MyInt getFact() {
      return new MyInt(fact(value));
}

private int fact(int n) {
      int sofar = 1;
      while (n > 1) sofar *= n--;
      return sofar;
}
```

Java definition for objects that hold an integer value and know how to report both that value and its factorial

Programming Language Paradigms

Object-oriented

- Based on the concept of *objects* that encapsulate the data and the methods acting on that data.
- Examples: Smalltalk, Java, C++, Go (sort of)

```
type MyInt int;

func (n MyInt) fact() int {
   f := 1
   for i:=1; i<=int(n); i++ {
      f *= i
   }
   return f
}
```

Corresponding code in Go

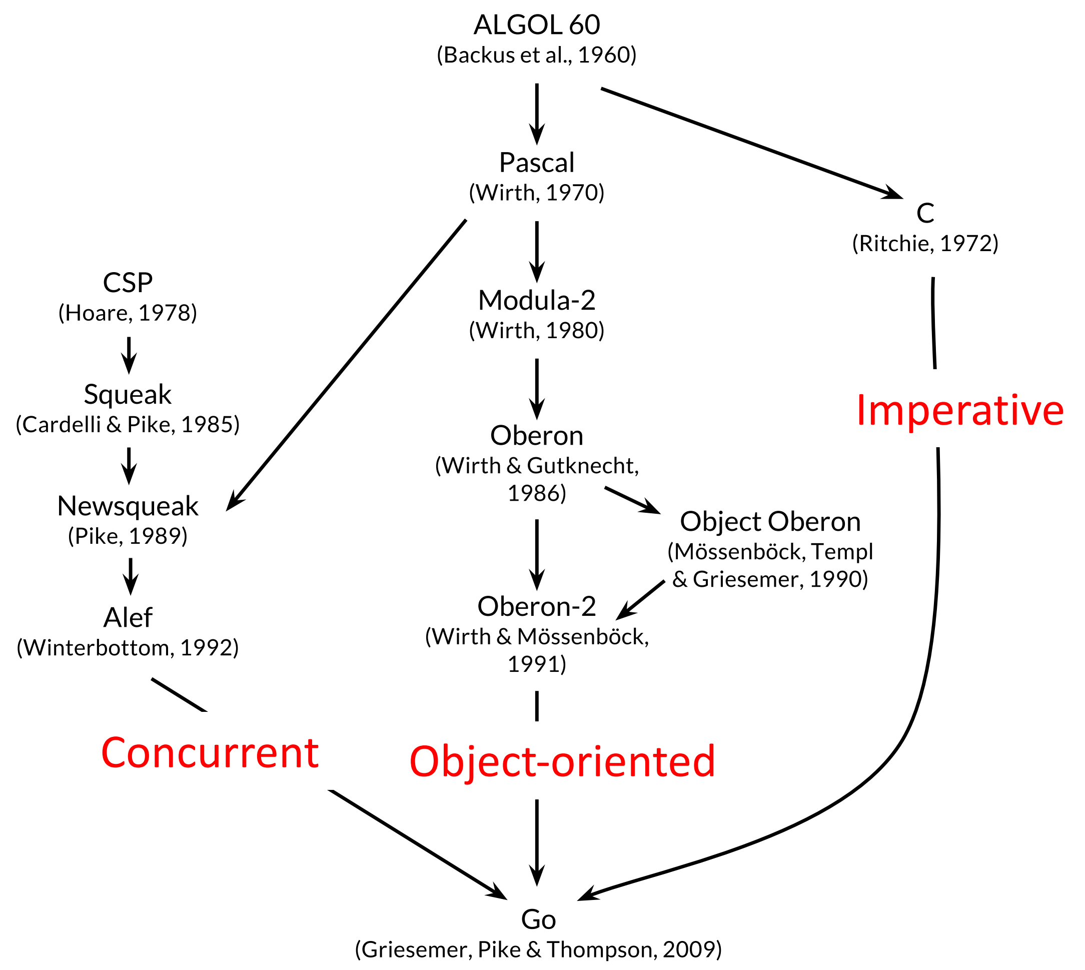
Programming Language Paradigms

Concurrent

- Allows for simultaneous execution of tasks in the program
- Examples: Go, Rust, Erlang

```go
func main() {
    for i:=0; i<10; i++ {
        go fact(i)
    }
}
```

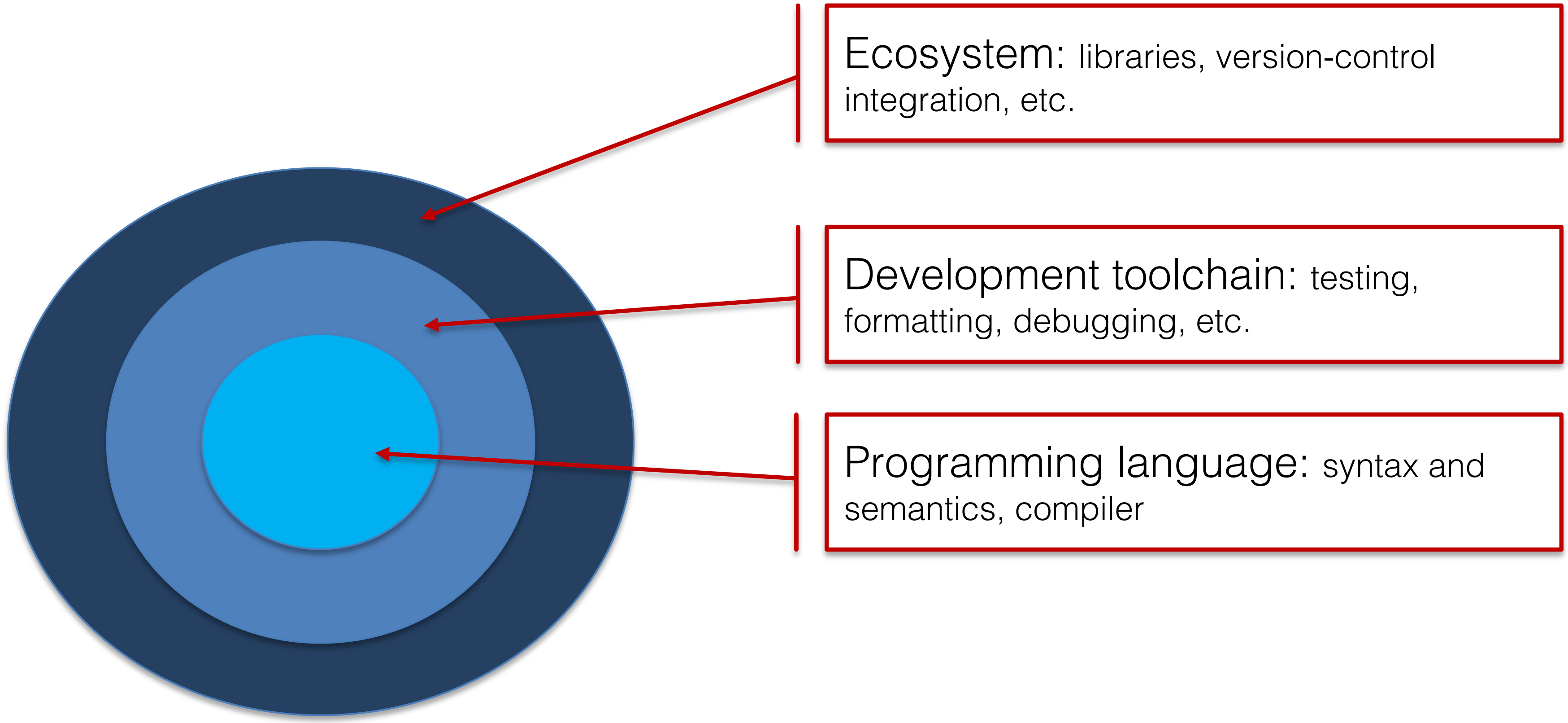
Combining paradigms



ALGOL 60 (Backus et al., 1960)
Pascal (Wirth, 1970)
C (Ritchie, 1972)
CSP (Hoare, 1978)
Modula-2 (Wirth, 1980)
Squeak (Cardelli & Pike, 1985)
Oberon (Wirth & Gutknecht, 1986)
Newsqueak (Pike, 1989)
Object Oberon (Mössenböck, Templ & Griesemer, 1990)
Alef (Winterbottom, 1992)
Oberon-2 (Wirth & Mössenböck, 1991)
Imperative
Concurrent
Object-oriented
Go (Griesemer, Pike & Thompson, 2009)

Choosing a programming language

Factors other than just the syntax and semantics


Ecosystem: libraries, version-control integration, etc.
Development toolchain: testing, formatting, debugging, etc.
Programming language: syntax and semantics, compiler

Choosing a programming language

Factors other than just the syntax and semantics

Choice of language determined by existing code or by someone else

Languages I have had to code in my research and industry experience:
C, C++, Java, Go, Haskell, Ocaml, Scala, Python, C#, F#
(Usually not by choice.)

You need to be able to read and write multiple programming languages and pick up new ones quickly!

Why golang?

- “Modern” language
  - Open source, comes with testing, formatting and other programmer tools
- C-like syntax
  - Focus on readability of code
- Combines multiple paradigms
  - Imperative, Object-oriented, functional, concurrent
- Garbage collected
- Interesting to see the choices made by the language designers
- Used in industry