Programming Languages and Compilers (CS 421)

Sasa Misailovic

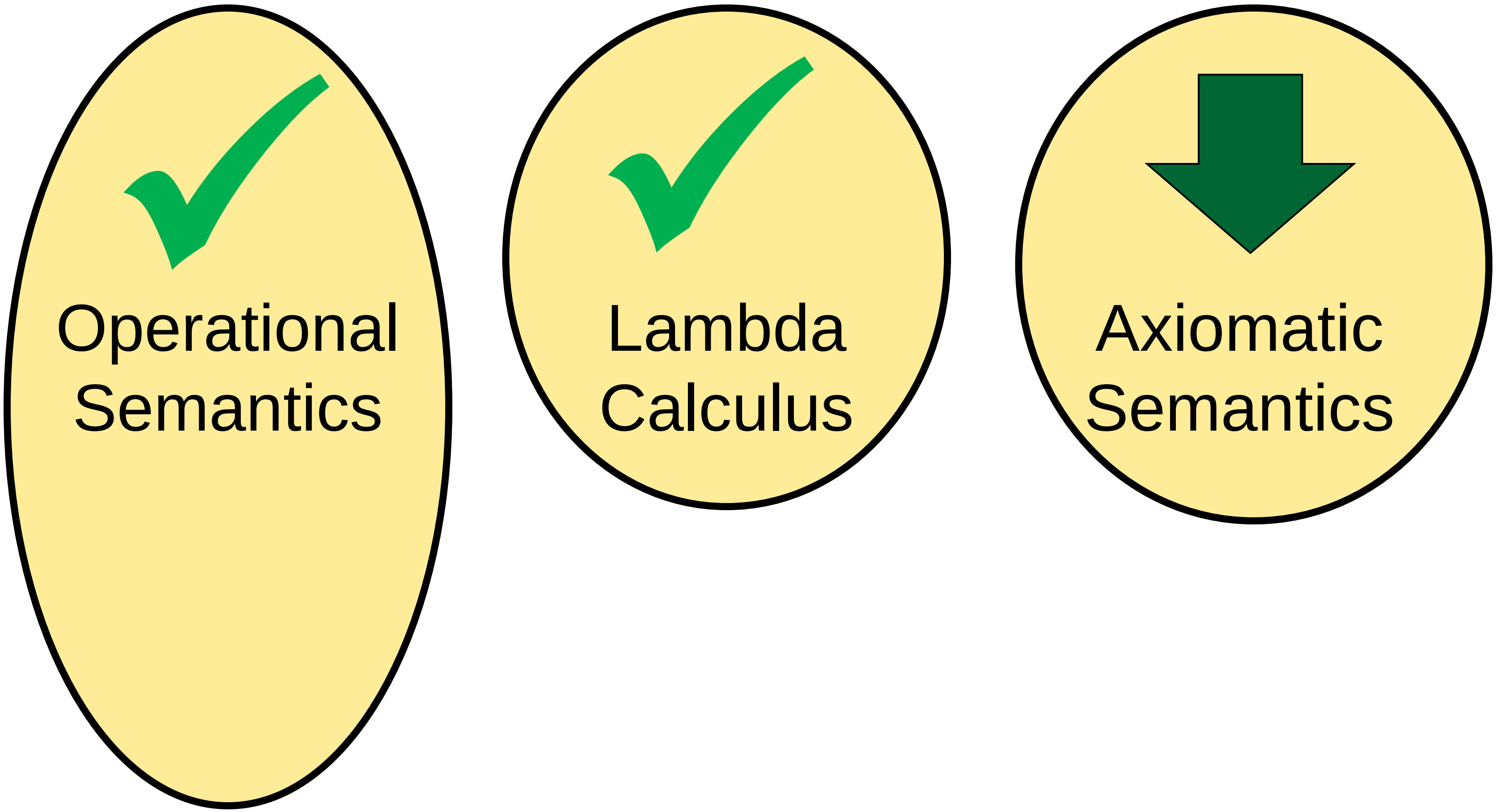
4110 SC, UIUC

https://courses.engr.illinois.edu/cs421/fa2024/CS421C

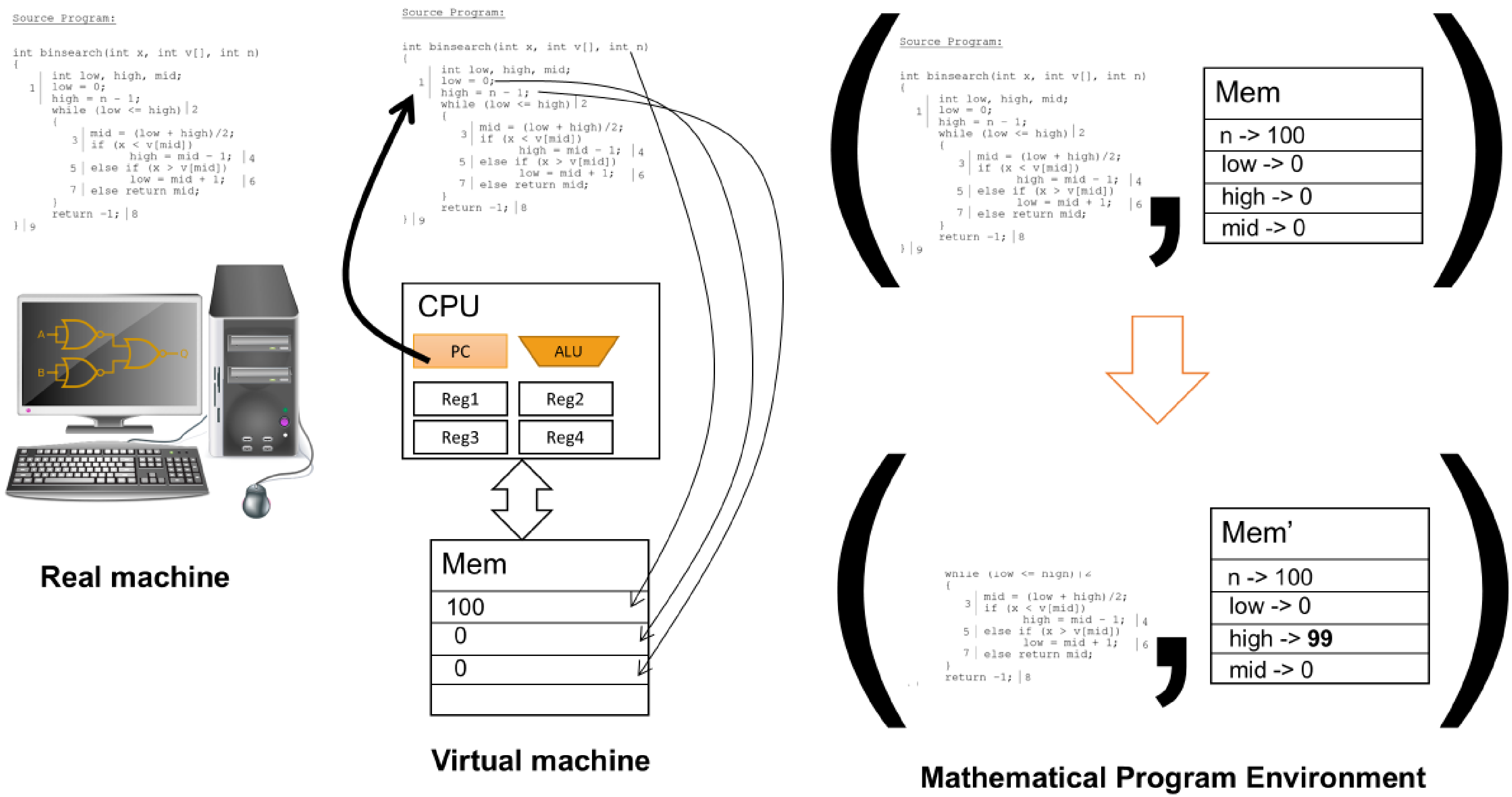
Based on slides by Elsa Gunter, which are based in part on previous slides by Mattox Beckman and updated by Vikram Adve and Gul Agha

Programming Languages & Compilers

III : Language Semantics



Reminder: Intuition about Operational Semantics



Sources: https://www.researchgate.net/figure/Example-of-Control-Flow-Graph_fig5_4065402 and https://freesvg.org/computer-station-vector-graphics

Three Flavors of Semantics we studied

Operational semantics: models the virtual machine

X	Y
3	0

```
31:   ...
32:   y = x + 2
33:   ...
```

X	Y
3	5

The statements transform the program state

We represent machine as (pure) mathematical model

Lambda calculus: models execution as term rewriting

$$(\lambda\, x.\, P)\ N \quad -\beta\!\to \quad P\,[N / x]$$

The expression itself is directly simplified

We can represent math as computation

Axiomatic semantics: program transforms logic formulas

$$\{\, x \geq 3 \,\}$$

$$y := x+2$$

$$\{\, y \geq 5 \,\}$$

The statements transform the formulas

We can turn computation into math formula manipulation

Axiomatic Semantics

- Also called Floyd-Hoare Logic
- Based on formal logic (first order predicate calculus)
- Axiomatic Semantics is a logical system built from *axioms* and *inference rules*
- Mainly suited to simple imperative programming languages

Axiomatic Semantics

- Used to formally prove a property (*post-condition*) of the *state* (the values of the program variables) after the execution of program

- To do so, it assumes another property (*pre-condition*) of the state holds before execution

Axiomatic Semantics

- Goal: Derive statements of form

  $$\{P\}\ C\ \{Q\}$$

  - P , Q logical statements about state,
    P precondition,
    Q postcondition,
    C program

- Example: $\{x > 1\}\ x := x + 1\ \{x > 2\}$

Axiomatic Semantics

- *Approach:* For each kind of language statement, give an axiom or inference rule stating how to derive assertions of form

  {P} C {Q}

  where C is a statement of that kind

- Compose axioms and inference rules to build proofs for complex programs

Axiomatic Semantics

- An expression {P} C {Q} is a *partial correctness* statement
- For *total correctness* must also prove that C terminates (i.e. doesn't run forever)
  - Written: [P] C [Q]
- Will only consider partial correctness here

Language

- We will give rules for simple imperative language (assignment, sequence, conditionals, loops)

```
<command> ::=
     <variable> := <term>
   | <command>; … ;<command>
   | if <expression> then <command>
     else <command> fi
   | while <expression> do <command> od
```

- Could add more features, like for-loops

Substitution

- Notation: P[e/v] (sometimes P[v <- e])
- Meaning: Replace every v in P by e
- Example:

$$(x + 2)\,[y-1/x] = ((y - 1) + 2)$$

The Assignment Rule

$$\frac{}{\{P\,[e/x]\}\; x := e\; \{P\}}$$

Example:

$$\frac{}{\{\quad ?\quad \}\; x := y\; \{x = 2\}}$$

The Assignment Rule

$$\frac{}{\{P\,[e/x]\}\; x := e\; \{P\}}$$

Example:

$$\frac{}{\{\,\_ = 2\,\}\; x := y\; \{\,x = 2\,\}}$$

The Assignment Rule

$$\frac{}{\{P\,[e/x]\}\; x := e\; \{P\}}$$

Example:

$$\frac{}{\{y = 2\}\; x := y\; \{x = 2\}}$$

The Assignment Rule

$$\frac{}{\{P\,[e/x]\,\}\ x := e\ \{P\}}$$

Examples:

$$\frac{}{\{y = 2\}\ x := y\ \{x = 2\}}$$

$$\frac{}{\{y = 2\}\ x := 2\ \{y = x\}}$$

$$\frac{}{\{x + 1 = n + 1\}\ x := x + 1\ \ \{x = n + 1\}}$$

$$\frac{}{\{2 = 2\}\ x := 2\ \{x = 2\}}$$

The Assignment Rule – Your Turn

- What is a valid precondition here?

{ ? }

`x := x + y`

$\{ x + y = w - x \}$

The Assignment Rule – Your Turn

- What is a valid precondition here?

$$\{ (x + y) + y = w - (x + y) \}$$

```
x := x + y
```

$$\{ x + y = w - x \}$$

Precondition Strengthening

$$\frac{P \Rightarrow P' \qquad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}}$$

- Meaning: If we can show that P implies P' ($P \Rightarrow P'$) and we can show that $\{P'\}\ C\ \{Q\}$, then we know that $\{P\}\ C\ \{Q\}$

- P is *stronger* than P' means $P \Rightarrow P'$

Precondition Strengthening

- Examples:

$$\frac{x = 3 \Rightarrow x < 7 \quad \{x < 7\}\ \texttt{x := x + 3}\ \{x < 10\}}{\{x = 3\}\ \texttt{x := x + 3}\ \{x < 10\}}$$

$$\frac{True \Rightarrow 2 = 2 \quad \{2 = 2\}\ \texttt{x := 2}\ \{x = 2\}}{\{True\}\ \texttt{x := 2}\ \{x = 2\}}$$

$$\frac{x=n \Rightarrow x+1=n+1 \quad \{x+1=n+1\}\ \texttt{x:=x+1}\ \{x=n+1\}}{\{x=n\}\ \texttt{x:=x+1}\ \{x=n+1\}}$$

Which Inferences Are Correct?

$$\frac{\{x > 0 \wedge x < 5\}\ \texttt{x := x * x}\ \{x < 25\}}{\{x = 3\}\ \texttt{x := x * x}\ \{x < 25\}}$$

$$\frac{\{x = 3\}\ \texttt{x := x * x}\ \{x < 25\}}{\{x > 0 \wedge x < 5\}\ \texttt{x := x * x}\ \{x < 25\}}$$

$$\frac{\{x * x < 25\}\ \texttt{x := x * x}\ \{x < 25\}}{\{x > 0 \wedge x < 5\}\ \texttt{x := x * x}\ \{x < 25\}}$$

Which Inferences Are Correct?

$$\frac{\{x > 0 \wedge x < 5\}\ \texttt{x := x * x}\ \{x < 25\}}{\{x = 3\}\ \texttt{x := x * x}\ \{x < 25\}}$$ ✓

$$\frac{\{x = 3\}\ \texttt{x := x * x}\ \{x < 25\}}{\{x > 0 \wedge x < 5\}\ \texttt{x := x * x}\ \{x < 25\}}$$ ✗

$$\frac{\{x * x < 25\}\ \texttt{x := x * x}\ \{x < 25\}}{\{x > 0 \wedge x < 5\}\ \texttt{x := x * x}\ \{x < 25\}}$$ ✓

Sequencing

$$\frac{\{P\}\ C_1\ \{Q\} \qquad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1;\ C_2\ \{R\}}$$

- Example:

$$\frac{\begin{array}{c}\{z = z \land z = z\}\ \texttt{x := z}\ \{x = z \land z = z\} \\ \{x = z \land z = z\}\ \texttt{y := z}\ \{x = z \land y = z\}\end{array}}{\{z = z \land z = z\}\ \texttt{x := z; y := z}\ \{x = z \land y = z\}}$$

Sequencing

$$\frac{\{P\}\ C_1\ \{Q\} \qquad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1;\ C_2\ \{R\}}$$

- Example:

$$\frac{\{z = z \land z = z\}\ \texttt{x := z}\ \{x = z \land z = z\} \qquad \{x = z \land z = z\}\ \texttt{y := z}\ \{x = z \land y = z\}}{\{z = z \land z = z\}\ \texttt{x := z; y := z}\ \{x = z \land y = z\}}$$

Postcondition Weakening

$$\frac{\{P\}\ C\ \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\ C\ \{Q\}}$$

Example:

$$\frac{\{z = z \wedge z = z\}\ \texttt{x := z; y := z}\ \{x = z \wedge y = z\} \qquad \mathbf{(x = z \wedge y = z) \Rightarrow (x = y)}}{\{z = z \wedge z = z\}\ \texttt{x := z; y := z}\ \mathbf{\{x = y\}}}$$

Rule of Consequence

$$\frac{P \Rightarrow P' \qquad \{P'\}\ C\ \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\ C\ \{Q\}}$$

- Logically equivalent to the **combination of** Precondition Strengthening and Postcondition Weakening
- Uses $P \Rightarrow P'$ and $Q' \Rightarrow Q$

If Then Else

$$\frac{\{P \text{ and } B\}\ C_1\ \{Q\} \quad \{P \text{ and } (\text{not } B)\}\ C_2\ \{Q\}}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ C_1\ \textbf{else}\ C_2\ \textbf{fi}\ \{Q\}}$$

- Example: Want

$\{y=a\}$

```
if x < 0 then y:= y-x else y:= y+x fi
```

$\{y=a+|x|\}$

Suffices to show:

(1) $\{y=a \land x<0\}$ `y:=y-x` $\{y=a+|x|\}$ and

(4) $\{y=a \land \text{not}(x<0)\}$ `y:=y+x` $\{y=a+|x|\}$

$\{y=a \land x<0\}$ `y:=y-x` $\{y=a+|x|\}$

(3) $(y=a \land x<0) \Rightarrow y-x=a+|x|$

(2) $\{y-x=a+|x|\}$ `y:=y-x` $\{y=a+|x|\}$

(1) $\{y=a \land x<0\}$ `y:=y-x` $\{y=a+|x|\}$

(1) Reduces to (2) and (3) by ***Precondition Strengthening***

(2) Follows from ***assignment*** axiom

(3) Because from algebra: $x<0 \Rightarrow |x| = -x$

{y=a ∧ not(x<0)} y:=y+x {y=a+|x|}

(6) $(y=a \wedge \text{not}(x<0)) \Rightarrow (y+x=a+|x|)$

(5) $\{y+x=a+|x|\}$ `y:=y+x` $\{y=a+|x\}\}$

---

(4) $\{y=a \wedge \text{not}(x<0)\}$ `y:=y+x` $\{y=a+|x|\}$

(4) Reduces to (5) and (6) by **Precondition Strengthening**

(5) Follows from **assignment** axiom

(6) Because $\text{not}(x<0) \Rightarrow |x| = x$

If Then Else

(1) $\{y=a \wedge x<0\}$ `y:=y-x` $\{y=a+|x|\}$

(4) $\{y=a \wedge \text{not}(x<0)\}$ `y:=y+x` $\{y=a+|x|\}$

$\{y=a\}$

```
if x < 0 then y:= y-x else y:= y+x
```

$\{y=a+|x|\}$

By the **IfThenElse rule**

While

- We need a rule to be able to make assertions about **while** loops.
  - Inference rule because we can only draw conclusions if we know something about the body
  - Let's start with:

$$\frac{\{\quad ?\quad\}\ \ C\ \ \{\quad ?\quad\}}{\{\quad ?\quad\}\ \ \textbf{while}\ \ B\ \ \textbf{do}\ \ C\ \textbf{od}\ \ \{\ P\ \}}$$

While

- The loop may never be executed, so if we want P to hold after, it had better hold before, so let's try:

$$\frac{\{\ \ ?\ \ \}\ \ C\ \ \{\ \ ?\ \ \}}{\{\ P\ \}\ \textbf{while}\ B\ \textbf{do}\ C\ \textbf{od}\ \{\ P\ \}}$$

While

- If all we know is P when we enter the **while** loop, then we all we know when we enter the body is (P and B)
- If we need to know P when we finish the **while** loop, we had better know it when we finish the loop body:

$$\frac{\{\ P \text{ and } B\}\ \ C\ \ \{\ P\ \}}{\{\ P\ \}\ \ \textbf{while}\ \ B\ \ \textbf{do}\ \ C\ \textbf{od}\ \ \{\ P\ \}}$$

While

- We can strengthen the previous rule because we also know that when the loop is finished, **not B** also holds
- Final **while** rule:

$$\frac{\{\ P \text{ and } B\ \}\ \ C\ \ \{\ P\ \}}{\{\ P\ \}\ \textbf{while}\ \ B\ \ \textbf{do}\ \ C\ \textbf{od}\ \ \{\ P \text{ and not } B\ \}}$$

While

$$\frac{\{\ P \text{ and } B\ \}\ \ C\ \ \{\ P\ \}}{\{\ P\ \}\ \textbf{while}\ \ B\ \ \textbf{do}\ \ C\ \textbf{od}\ \ \{\ P \text{ and not } B\ \}}$$

- P satisfying this rule is called a ***loop invariant*** because it must hold **before and after each iteration of the loop**

While

- **While** rule generally needs to be used together with precondition strengthening and postcondition weakening

- There is **NO algorithm for computing the correct P**; it requires intuition and an understanding of why the program works

All Rules on One Slide

The Assignment Rule

$$\frac{}{\{P\,[e/x]\,\}\ x := e\ \{P\}}$$

Sequencing

$$\frac{\{P\}\ C_1\ \{Q\} \qquad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1;\ C_2\ \{R\}}$$

If Then Else

$$\frac{\{P \text{ and } B\}\ C_1\ \{Q\} \qquad \{P \text{ and } (\text{not } B)\}\ C_2\ \{Q\}}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ C_1\ \textbf{else}\ C_2\ \textbf{fi}\ \{Q\}}$$

Precondition Strengthening

$$\frac{P \rightarrow P' \qquad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}}$$

Postcondition Weakening

$$\frac{\{P\}\ C\ \{Q'\} \qquad Q' \rightarrow Q}{\{P\}\ C\ \{Q\}}$$

Rule of Consequence

$$\frac{P \rightarrow P' \qquad \{P'\}\ C\ \{Q'\} \qquad Q' \rightarrow Q}{\{P\}\ C\ \{Q\}}$$

While

$$\frac{\{\ P \text{ and } B\ \}\ \ C\ \ \{\ P\ \}}{\{\ P\ \}\ \textbf{while}\ \ B\ \ \textbf{do}\ \ C\ \textbf{od}\ \ \{\ P \text{ and not } B\ \}}$$

Counting up to n

$$\frac{\{ P \text{ and } B \}\ C\ \{ P \}}{\{ P \}\ \textbf{while}\ B\ \textbf{do}\ C\ \textbf{od}\ \{ P \text{ and not } B \}}$$

- P satisfying this rule is called a ***loop invariant*** because it must hold **before and after the each iteration of the loop**

```
n := 10; x := 0;
while (x < n) {
   x := x + 1
}
```

$P \equiv x \leq n$

Want to show: $x \geq n$

Sum of numbers 1 to n

```
x := 0
y := 0

while y < n {
    y := y + 1;
    x := x + y
}
```

$$P \equiv \quad x = 1 + \ldots + y \;\wedge\; y \leq n \;\wedge\; 0 \leq n$$

Want to show: x = 1 + ... + n

Fibonacci

```
x = 0; y = 1;
z = 1;

while (z < n) {
    y := x + y;
    x := y - x;
    z := z + 1
}
```

$$P \equiv \quad y = \text{fib } z \;\wedge\; x = \text{fib } (z\text{-}1) \;\wedge\; z \leq n \;\wedge\; 1 \leq n$$

```
Want to show: y = fib n
```

List Length

```
x = lst; y = 0

while (x ≠ []) {
    x := tl x;
    y := y + 1
}
```

P ≡ y + len x = len lst

Want to show: `y = len lst`

Example (Use of Loop Invariant in Full Proof)

- Let us prove:

```
{x ≥ 0 and x = a}

fact := 1;
while x > 0 do
   (fact := fact * x; x := x – 1)
od

{fact = a!}
```

Example

- We need to find a condition P that is true both before and after the loop is executed, and such that

$$(P \text{ and not } x > 0) \Rightarrow (fact = a!)$$

Example

- First attempt:

P = {a! = fact * (x!)}

- Motivation:
- What we want to compute: **a!**
- What we have computed: **fact**

  which is the sequential product of **a** down through **(x + 1)**
- What we still need to compute: **x!**

Example

Postcondition Weakening

$$\frac{\{P\}\ C\ \{Q'\} \quad Q' \rightarrow Q}{\{P\}\ C\ \{Q\}}$$

By post-condition weakening suffices to show

1. $\{x \geq 0 \text{ and } x = a\}$

```
fact := 1;
while x > 0 do (fact := fact * x; x := x-1) od
```

$\{a! = fact * (x!) \text{ and not } (x > 0)\}$

And

2. $a! = fact * (x!) \text{ and not } (x > 0) \Rightarrow fact = a!$

Problem!! (Dead End)

2. $a! = fact * (x!) \text{ and not } (x > 0) \Rightarrow fact = a!$

- Don't know this if x < 0 !!
  - Need to know that x = 0 when loop terminates
- **Need a new loop invariant**
  - Try adding $x \geq 0$
  - Then will have x = 0 when loop is done

Example

Second try, let us combine the two:

$P \equiv \quad a! = fact * (x!) \quad \text{and} \quad x \geq 0$

We need to show:

1. $\{x \geq 0 \text{ and } x = a\}$

```
fact := 1;
```

$\{P\}$

```
while x > 0 do (fact := fact * x; x := x −1) od
```

$\{P \text{ and not } x > 0\}$

And

2. $P \text{ and not } x > 0 \Rightarrow fact = a!$

Example

```
{x≥ 0 and x = a} (*this was part 1 to prove*)
    fact := 1;
    while x > 0 do (fact := fact * x; x := x -1) od
{a! = fact * (x!) and x ≥ 0 and not (x>0)}
```

- For Part 1, by sequencing rule it suffices to show

3. {x ≥ 0 and x = a}

```
        fact := 1
```

{a! = fact * (x!) and x ≥ 0 }

And

4. {a! = fact * (x!) and x ≥ 0}

```
      while x > 0 do
        (fact := fact * x; x := x -1) od
```

{a! = fact * (x!) and x ≥ 0 and not (x > 0)}

Example

- (Part 3 – Assignment) Suffices to show that

  $a! = fact * (x!)$ and $x \geq 0$

  holds before the while loop is entered

- (Part 4 – While Loop) And that if

  $(a! = fact * (x!))$ and $x \geq 0$ and $x > 0$

  holds before we execute the body of the loop, then

  $(a! = fact * (x!))$ and $x \geq 0$

  holds after we execute the body (part 4)

Example

3. {x>=0 and x = a}
```
fact := 1
```
{a! = fact * (x!) and x >=0 }

Precondition Strengthening

$$\frac{P \rightarrow P' \quad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}}$$

(Part 3) By the assignment rule, we have

{a! = **1** * (x!) and x ≥ 0}
```
fact := 1
```
{a! = **fact** * (x!) and x ≥ 0}

Therefore, to show (3), by precondition strengthening, it suffices to show

$$(x \geq 0 \text{ and } x = a) \Rightarrow (a! = 1 * (x!) \text{ and } x \geq 0)$$

It holds because $x = a \Rightarrow x! = a!$.

- So, we have that a! = fact * (x!) and x ≥ 0 **holds at the start of the while loop!**

Example

To prove (Part 4):

{a! = fact * (x!) and x ≥0}

```
while x > 0 do
  (fact := fact * x; x := x –1)
od
```

{a! = fact * (x!) and x ≥ 0 and not (x > 0)}

we need to **show that** **(a! = fact * (x!)) and x ≥ 0** is **a loop invariant**

- We will use **assignment** rule, **sequencing** rule and **precondition strengthening** rule

Example

Sequencing

$$\frac{\{P\}\ C_1\ \{Q\} \qquad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1;\ C_2\ \{R\}}$$

- We look into the loop body:
  - `(fact := fact * x; x := x - 1)`
- By the sequencing rule, we need to show 2 things:
  - By the ***assignment rule***, show

$$\{(a! = fact * (x!)) \text{ and } x \geq 0 \text{ and } x > 0\}$$

$$fact = fact * x$$

$$\{Q\}$$

  - By the ***assignment rule***, show

$$\{Q\}$$

$$x := x - 1$$

$$\{(a! = fact * (\mathbf{x}!)) \text{ and } \mathbf{x} \geq 0\}$$

Example

The Assignment Rule

$$\frac{}{\{P[e/x]\}\ x := e\ \{P\}}$$

- We look into the loop body:
  - `(fact := fact * x; x := x - 1)`
- By the sequencing rule, we need to show 2 things:
  - By the ***assignment rule***, show

$$\{(a! = fact * (x!)) \text{ and } x \geq 0 \text{ and } x > 0\}$$

$$fact = fact * x$$

$$\{Q\}$$

  - From the ***assignment rule***, we know:

$$\{(a! = fact * (\mathbf{(x - 1)}!)) \text{ and } \mathbf{x - 1} \geq 0\}$$

$$x := x - 1$$

$$\{(a! = fact * (\mathbf{x}!)) \text{ and } \mathbf{x} \geq 0\}$$

Example

- We look into the loop body:
  - `(fact := fact * x; x := x – 1)`
- By the sequencing rule, we need to show 2 things:
  - By the ***assignment rule***, show

$$\{(a! = fact * (x!)) \text{ and } x \geq 0 \text{ and } x > 0\}$$

$$fact = fact * x$$

$$\mathbf{\{(a! = fact * ((x - 1)!)) \text{ and } x - 1 \geq 0\}}$$

  - From the ***assignment rule***, we know:

$$\{(a! = fact * (\mathbf{(x - 1)}!)) \text{ and } \mathbf{x - 1} \geq 0\}$$

$$x := x - 1$$

$$\{(a! = fact * (\mathbf{x}!)) \text{ and } \mathbf{x} \geq 0\}$$

Example

Precondition Strengthening

$$\frac{P \rightarrow P' \quad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}}$$

- By the ***assignment rule***, we have that

  {(a! = **(fact * x)** * ((x-1)!)) and x – 1 ≥ 0}

  ```
  fact = fact * x
  ```

  {(a! = fact * ((x-1)!)) and x – 1 ≥ 0}

- By ***Precondition strengthening***, it suffices to show that

  ((a! = fact * (x!)) and x ≥ 0 and x > 0) ⇒

  ((a! = (fact * x) * (**(x-1)**!)) and **x – 1 ≥ 0**)

From algebra we know that fact * x * (x – 1)! = fact * x!

and (x > 0) ⇒ x – 1 ≥ 0 since x is an integer, so

{(**a! = fact * (x!)**) and x ≥ 0 and x > 0} ⇒

{(**a! = (fact * x) * ((x-1)!)**) and **x – 1 ≥ 0**}

Example

Second try, let us combine the two:

$$P \equiv a! = fact * (x!) \text{ and } x \geq 0$$

We need to show:

1. $\{x \geq 0 \text{ and } x = a\}$

```
fact := 1;
```

$\{P\}$

```
while x > 0 do (fact := fact * x; x := x -1) od
```

$\{P \text{ and not } x > 0\}$

And

2. $P \text{ and not } x > 0 \Rightarrow fact = a!$

Example

- For Part 2, we need

$(a! = fact * (x!) \text{ and } x \geq 0 \text{ and not } (x > 0)) \Rightarrow (fact = a!)$

Since we know $(x \geq 0 \text{ and not } (x > 0)) \Rightarrow (x = 0)$ so

$$fact * (x!) = fact * (0!)$$

And since from algebra we know that 0! = 1,

$$fact * (0)! = fact * 1 = fact$$

- Therefore, we can prove:

$(a! = fact * (x!) \text{ and } x \geq 0 \text{ and not } (x > 0)) \Rightarrow (fact = a!)$

Example

- We proved that (**a! = fact * (x!)**) and x ≥ 0 is the loop invariant
- We proved the sequence rule for the assignment and wile statements
- We applied postcondition weakening to prove the final predicate

This finishes the proof!

```
{x ≥ 0 and x = a}
  fact := 1;
  while x > 0 do (fact := fact * x; x := x – 1) od
{fact = a!}
```

Example

- We proved that (**a! = fact * (x!)**) and $x \geq 0$ is the loop invariant
- We proved the sequence rule for the assignment and wile statements
- We applied postcondition weakening to prove the final predicate

This finishes the proof!

This also finishes all technical material in this class!

Three Flavors of Semantics we studied

Operational semantics: models the virtual machine

X	Y
3	0

```
31:   ...
32:   y = x + 2
33:   ...
```

X	Y
3	5

The statements transform the program state

We represent machine as (pure) mathematical model

Lambda calculus: models execution as term rewriting

$$(\lambda\ x.\ P)\ N \ \text{—}\beta\text{—>}\ P\,[N / x]$$

The expression itself is directly simplified

We can represent math as computation

Axiomatic semantics: program transforms logic formulas

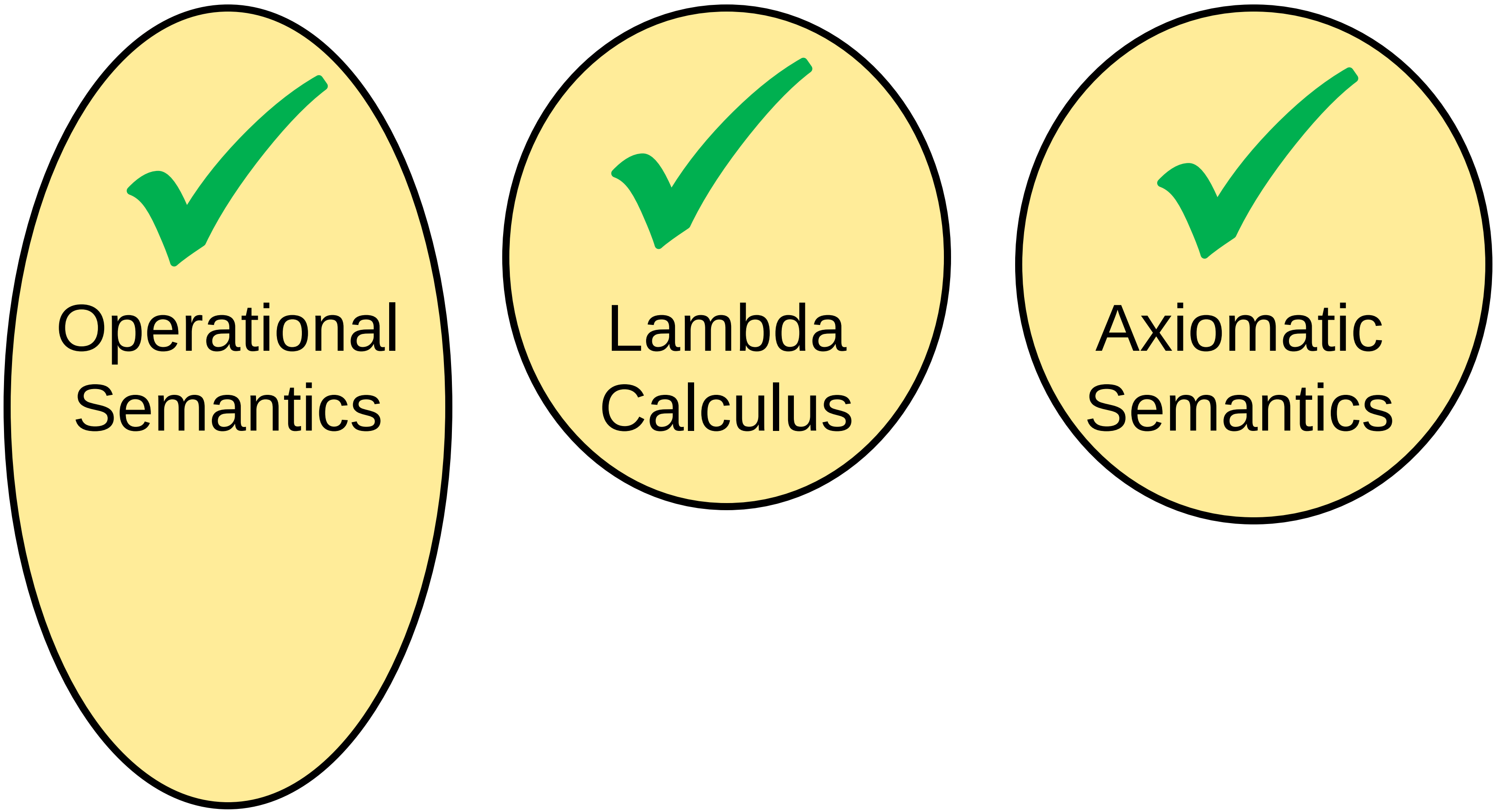
$$\{\ x \geq 3\ \}$$
$$y\ := \ x+2$$
$$\{\ y \geq 5\ \}$$

The statements transform the formulas

We can turn computation into math formula manipulation

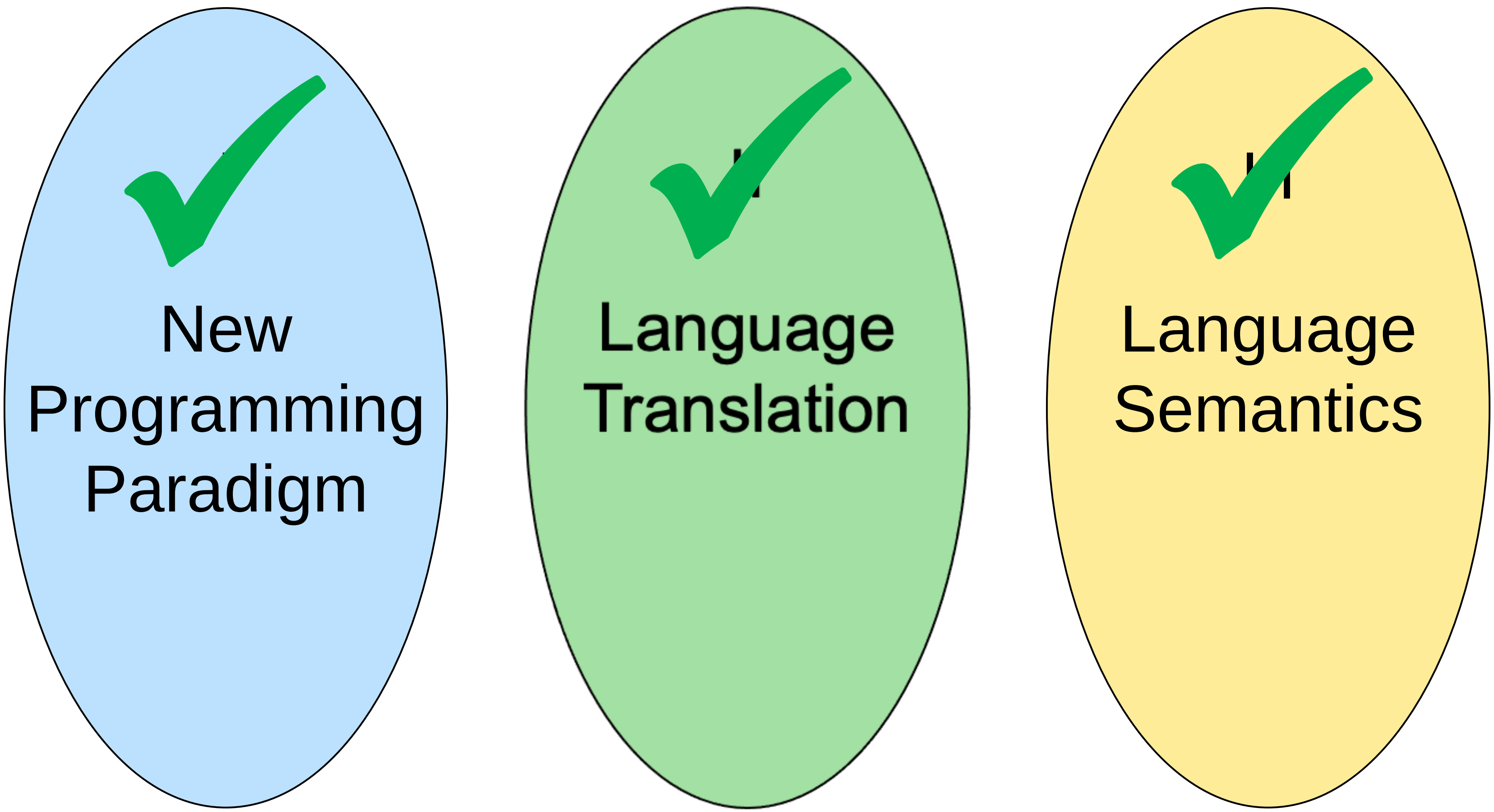
Programming Languages & Compilers

III : Language Semantics



CS 421: Programming Languages & Compilers

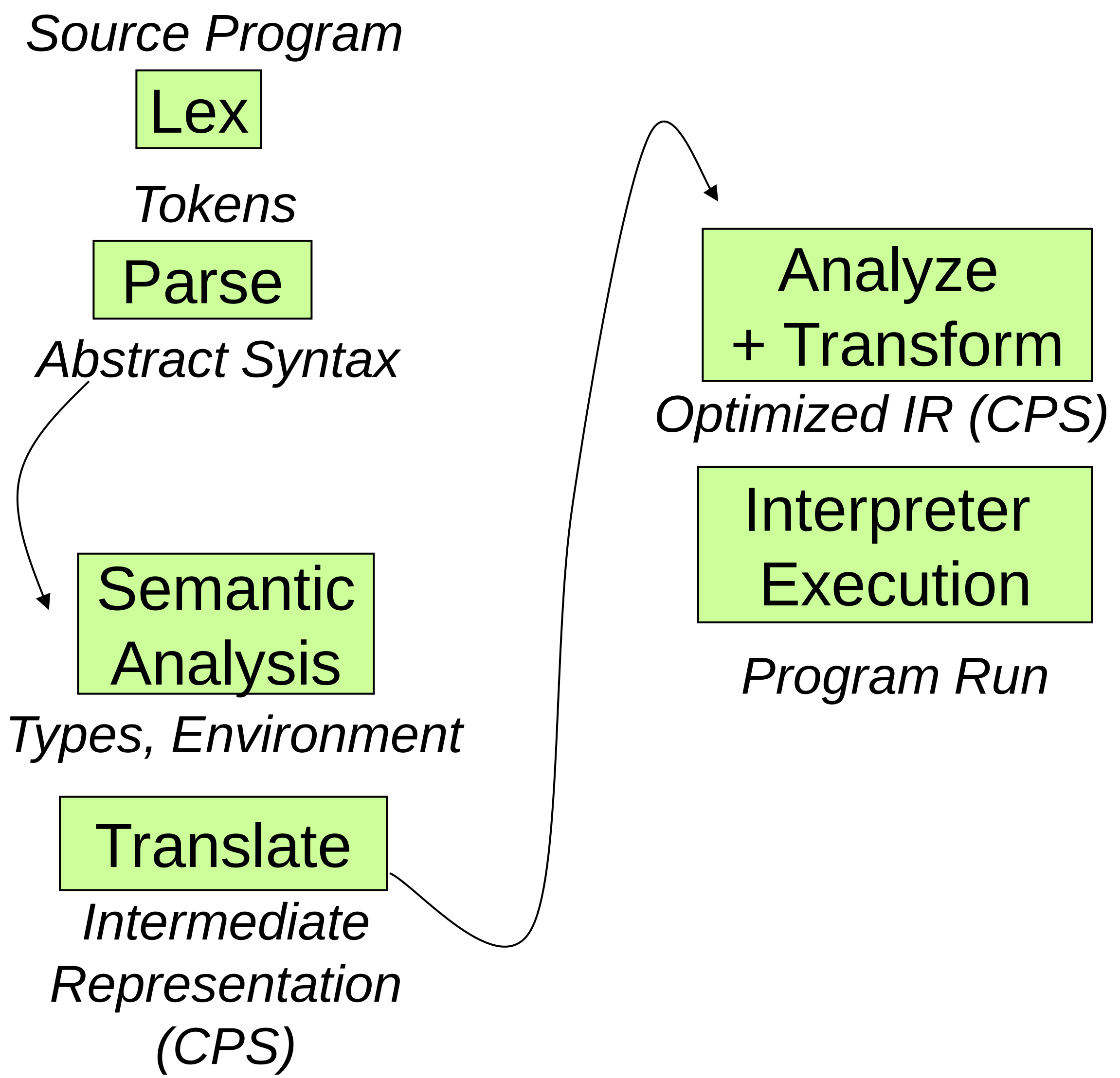
Three Main Topics of the Course



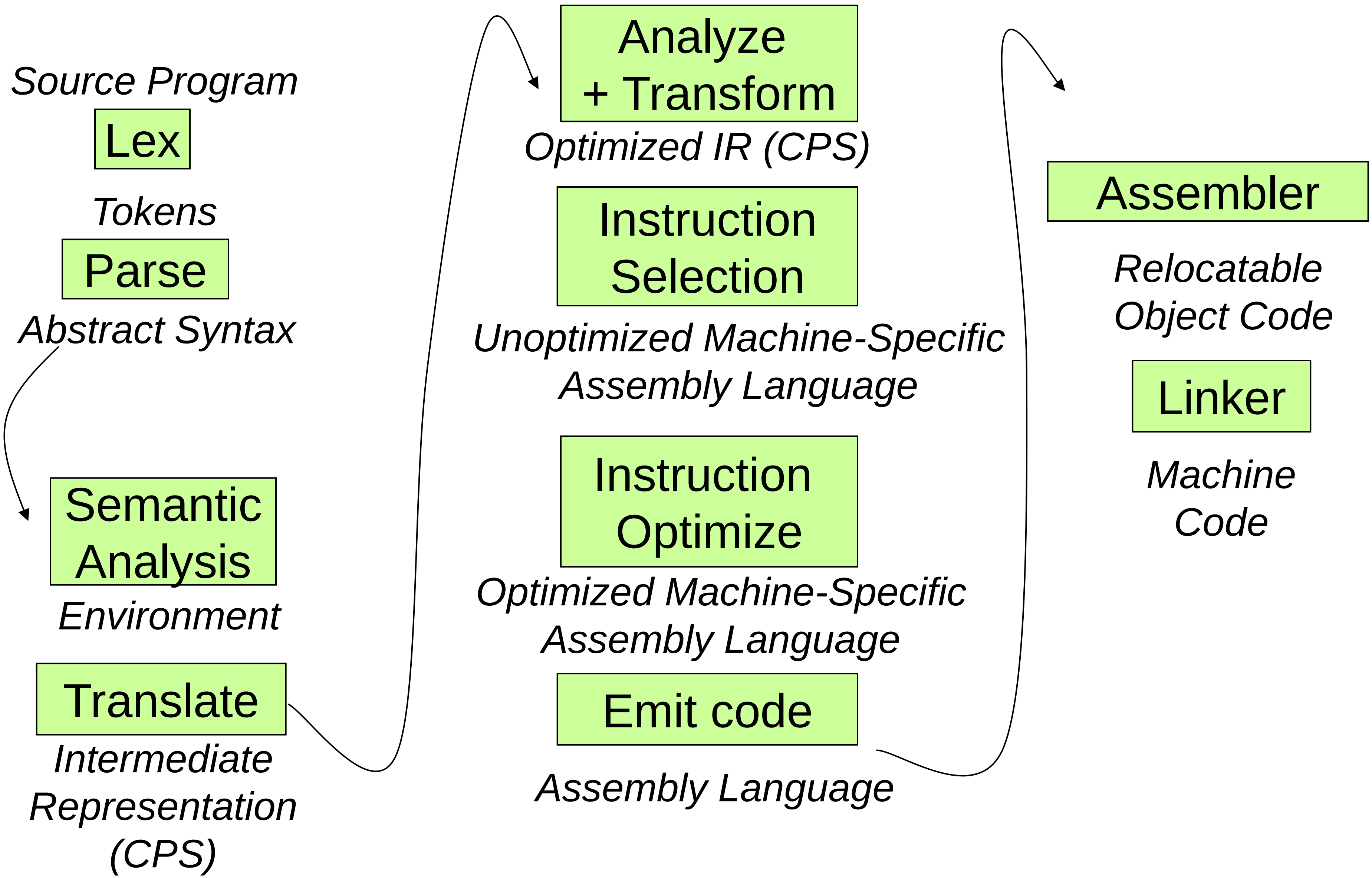
Course Objectives

- New programming paradigm
  - Functional programming
  - Environments and Closures
  - Patterns of Recursion
  - Continuation Passing Style
- Phases of an interpreter / compiler
  - Lexing and parsing
  - Type systems
  - Interpretation
- Programming Language Semantics
  - Lambda Calculus
  - Operational Semantics
  - Axiomatic Semantics

Major Phases of a PicoML Interpreter



Where to go from here?



Modified from “Modern Compiler Implementation in ML”, by Andrew Appel