# EECS 321
# Programming Languages

Fall 2015

Instructor: **Robby Findler**

# Course Details

**`http://www.eecs.northwestern.edu/~robby/courses/321-2015-fall/`**

(or google "findler" and follow the links)

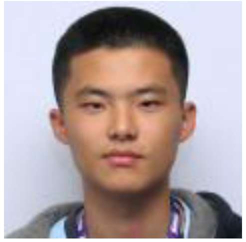
# TA & Office Hours

Your TAs:

**Zavier Henry**

**Anuj Iravane**

**Adrien Tateno**



**Josh Xu**

Office Hours in Wilkenson (see website for details)

# Registration

Last day for registration is Friday

If you’re not registered and want to be after you do the first assignment, send me email.

**`robby@eecs.northwestern.edu`**

# Programming Language Concepts

This course teaches concepts in two ways:

By implementing **interpreters**

- new concept $\Rightarrow$ new interpreter

By using **Racket** and variants

- we don’t assume that you already know Racket

# Interpreters vs Compilers

An ***interpreter*** takes a program and produces a result

- DrRacket
- x86 processor
- desktop calculator
- **`bash`**
- Algebra student

# Interpreters vs Compilers

An ***interpreter*** takes a program and produces a result

- DrRacket
- x86 processor
- desktop calculator
- **`bash`**
- Algebra student

A ***compiler*** takes a program and produces a program

- DrRacket
- x86 processor
- **`gcc`**
- **`javac`**

# Interpreters vs Compilers

An ***interpreter*** takes a program and produces a result

- DrRacket
- x86 processor
- desktop calculator
- **`bash`**
- Algebra student

Good for understanding program behavior, easy to implement

A ***compiler*** takes a program and produces a program

- DrRacket
- x86 processor
- **`gcc`**
- **`javac`**

Good for speed, more complex (take 322)

# Interpreters vs Compilers

An ***interpreter*** takes a program and produces a result

- DrRacket
- x86 processor
- desktop calculator
- `bash`
- Algebra student

Good for understanding program behavior, easy to implement

A ***compiler*** takes a program and produces a program

- DrRacket
- x86 processor
- `gcc`
- `javac`

Good for speed, more complex (take 322)

So, what's a ***program***?

# A Grammar for Algebra Programs

A grammar of Algebra in ***BNF*** (Backus-Naur Form):

$$
\begin{array}{lcl}
\langle \text{prog} \rangle & ::= & \langle \text{defn} \rangle^{*}\ \langle \text{expr} \rangle \\
\langle \text{defn} \rangle & ::= & \langle \text{id} \rangle(\langle \text{id} \rangle) = \langle \text{expr} \rangle \\
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
 & | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
 & | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\
 & | & \langle \text{id} \rangle \\
 & | & \langle \text{num} \rangle \\
\langle \text{id} \rangle & ::= & \text{a variable name: } \mathbf{f}, \mathbf{x}, \mathbf{y}, \mathbf{z}, \ldots \\
\langle \text{num} \rangle & ::= & \text{a number: } 1, 42, 17, \ldots
\end{array}
$$

# A Grammar for Algebra Programs

A grammar of Algebra in ***BNF*** (Backus-Naur Form):

| | | |
|---|---|---|
| ⟨prog⟩ | ::= | ⟨defn⟩* ⟨expr⟩ |
| ⟨defn⟩ | ::= | ⟨id⟩(⟨id⟩) = ⟨expr⟩ |
| ⟨expr⟩ | ::= | (⟨expr⟩ + ⟨expr⟩) |
| | \| | (⟨expr⟩ - ⟨expr⟩) |
| | \| | ⟨id⟩(⟨expr⟩) |
| | \| | ⟨id⟩ |
| | \| | ⟨num⟩ |
| ⟨id⟩ | ::= | a variable name: **f**, **x**, **y**, **z**, ... |
| ⟨num⟩ | ::= | a number: 1, 42, 17, ... |

Each ***meta-variable***, such as ⟨prog⟩, defines a set

# Using a BNF Grammar

$\langle$id$\rangle$ ::= a variable name: **f**, **x**, **y**, **z**, ...

$\langle$num$\rangle$ ::= a number: 1, 42, 17, ...

The set $\langle$id$\rangle$ is the set of all variable names

The set $\langle$num$\rangle$ is the set of all numbers

# Using a BNF Grammar

$\langle \mathrm{id} \rangle$ ::= a variable name: **f**, **x**, **y**, **z**, ...

$\langle \mathrm{num} \rangle$ ::= a number: 1, 42, 17, ...

The set $\langle \mathrm{id} \rangle$ is the set of all variable names

The set $\langle \mathrm{num} \rangle$ is the set of all numbers

To make an example member of $\langle \mathrm{num} \rangle$, simply pick an element from the set

# Using a BNF Grammar

| | | |
|---|---|---|
| $\langle \text{id} \rangle$ | ::= | a variable name: **f**, **x**, **y**, **z**, ... |
| $\langle \text{num} \rangle$ | ::= | a number: 1, 42, 17, ... |

The set $\langle \text{id} \rangle$ is the set of all variable names

The set $\langle \text{num} \rangle$ is the set of all numbers

To make an example member of $\langle \text{num} \rangle$, simply pick an element from the set

$$2 \in \langle \text{num} \rangle$$

$$298 \in \langle \text{num} \rangle$$

# Using a BNF Grammar

$$
\begin{array}{rcl}
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
 & | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
 & | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\
 & | & \langle \text{id} \rangle \\
 & | & \langle \text{num} \rangle
\end{array}
$$

The set $\langle \text{expr} \rangle$ is defined in terms of other sets

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
& | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle \\
& | & \langle \text{num} \rangle
\end{array}
$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable
- combine the examples with literal text

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle \mathrm{expr} \rangle & ::= & (\langle \mathrm{expr} \rangle + \langle \mathrm{expr} \rangle) \\
 & | & (\langle \mathrm{expr} \rangle - \langle \mathrm{expr} \rangle) \\
 & | & \langle \mathrm{id} \rangle(\langle \mathrm{expr} \rangle) \\
 & | & \langle \mathrm{id} \rangle \\
 & | & \langle \mathrm{num} \rangle
\end{array}
$$

To make an example ⟨expr⟩:

- choose one case in the grammar
- pick an example for each meta-variable
- combine the examples with literal text

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
& | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle \\
& | & \langle \text{num} \rangle
\end{array}
$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable

$$7 \in \langle \text{num} \rangle$$

- combine the examples with literal text

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle\text{expr}\rangle & ::= & (\langle\text{expr}\rangle + \langle\text{expr}\rangle) \\
 & | & (\langle\text{expr}\rangle - \langle\text{expr}\rangle) \\
 & | & \langle\text{id}\rangle(\langle\text{expr}\rangle) \\
 & | & \langle\text{id}\rangle \\
 & | & \langle\text{num}\rangle
\end{array}
$$

To make an example $\langle\text{expr}\rangle$:

- choose one case in the grammar
- pick an example for each meta-variable

$$7 \in \langle\text{num}\rangle$$

- combine the examples with literal text

$$7 \in \langle\text{expr}\rangle$$

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
& | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle \\
& | & \langle \text{num} \rangle
\end{array}
$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable
- combine the examples with literal text

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
& | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \quad \Leftarrow \\
& | & \langle \text{id} \rangle \\
& | & \langle \text{num} \rangle
\end{array}
$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable

$$\mathbf{f} \in \langle \text{id} \rangle$$

- combine the examples with literal text

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
 & | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
 & | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\
 & | & \langle \text{id} \rangle \\
 & | & \langle \text{num} \rangle
\end{array}
$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable

$$\mathbf{f} \in \langle \text{id} \rangle \qquad\qquad 7 \in \langle \text{expr} \rangle$$

- combine the examples with literal text

# Using a BNF Grammar

$$\begin{array}{lcl} \langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\ & | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\ & | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\ & | & \langle \text{id} \rangle \\ & | & \langle \text{num} \rangle \end{array}$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable

$$\mathbf{f} \in \langle \text{id} \rangle \qquad\qquad 7 \in \langle \text{expr} \rangle$$

- combine the examples with literal text

$$\mathbf{f}(7) \in \langle \text{expr} \rangle$$

# Using a BNF Grammar

$$
\begin{array}{lcl}
\langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\
& | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\
& | & \langle \text{id} \rangle \\
& | & \langle \text{num} \rangle
\end{array}
$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable

$$\mathbf{f} \in \langle \text{id} \rangle \qquad\qquad \mathbf{f}(7) \in \langle \text{expr} \rangle$$

- combine the examples with literal text

# Using a BNF Grammar

$$\begin{array}{lcl} \langle \text{expr} \rangle & ::= & (\langle \text{expr} \rangle + \langle \text{expr} \rangle) \\ & | & (\langle \text{expr} \rangle - \langle \text{expr} \rangle) \\ & | & \langle \text{id} \rangle(\langle \text{expr} \rangle) \\ & | & \langle \text{id} \rangle \\ & | & \langle \text{num} \rangle \end{array}$$

To make an example $\langle \text{expr} \rangle$:

- choose one case in the grammar
- pick an example for each meta-variable

$$\mathbf{f} \in \langle \text{id} \rangle \qquad \mathbf{f}(7) \in \langle \text{expr} \rangle$$

- combine the examples with literal text

$$\mathbf{f}(\mathbf{f}(7)) \in \langle \text{expr} \rangle$$

# Using a BNF Grammar

$$\langle \text{prog} \rangle \quad ::= \quad \langle \text{defn} \rangle^* \; \langle \text{expr} \rangle$$

$$\langle \text{defn} \rangle \quad ::= \quad \langle \text{id} \rangle(\langle \text{id} \rangle) = \langle \text{expr} \rangle$$

$$\mathbf{f}(\mathbf{x}) = (\mathbf{x} + 1) \in \langle \text{defn} \rangle$$

# Using a BNF Grammar

$$\langle \text{prog} \rangle \quad ::= \quad \langle \text{defn} \rangle^{*}\ \langle \text{expr} \rangle$$

$$\langle \text{defn} \rangle \quad ::= \quad \langle \text{id} \rangle(\langle \text{id} \rangle) = \langle \text{expr} \rangle$$

$$\mathbf{f}(\mathbf{x}) = (\mathbf{x} + 1) \in \langle \text{defn} \rangle$$

To make a $\langle \text{prog} \rangle$ pick some number of $\langle \text{defn} \rangle$s

$$(\mathbf{x} + \mathbf{y}) \in \langle \text{prog} \rangle$$

$$\left.\begin{array}{l} \mathbf{f}(\mathbf{x}) = (\mathbf{x} + 1) \\ \mathbf{g}(\mathbf{y}) = \mathbf{f}((\mathbf{y} - 2)) \\ \mathbf{g}(7) \end{array}\right. \quad \in \langle \text{prog} \rangle$$

# Programming Language

A ***programming language*** is defined by

- a grammar for programs
- rules for evaluating any program to produce a result

# Programming Language

A ***programming language*** is defined by

- a grammar for programs
- rules for evaluating any program to produce a result

For example, Algebra evaluation is defined in terms of evaluation steps:

$$(2 + (7 - 4)) \quad \rightarrow \quad (2 + 3) \quad \rightarrow \quad 5$$

# Programming Language

A ***programming language*** is defined by

- a grammar for programs
- rules for evaluating any program to produce a result

For example, Algebra evaluation is defined in terms of evaluation steps:

$$\mathbf{f}(\mathbf{x}) = (\mathbf{x} + 1)$$

$$\mathbf{f}(10) \quad \rightarrow \quad (10 + 1) \quad \rightarrow \quad 11$$

# Evaluation

- Evaluation $\rightarrow$ is defined by a set of pattern-matching rules:

$$(2 + (7 - 4)) \quad \rightarrow \quad (2 + 3)$$

due to the pattern rule

$$\ldots (7 - 4) \ldots \quad \rightarrow \quad \ldots 3 \ldots$$

# Evaluation

- Evaluation $\rightarrow$ is defined by a set of pattern-matching rules:

$$\mathbf{f}(\mathbf{x}) = (\mathbf{x} + 1)$$

$$\mathbf{f}(10) \quad \rightarrow \quad (10 + 1)$$

due to the pattern rule

$$\ldots\ \langle \text{id} \rangle_{\mathbf{1}}(\langle \text{id} \rangle_{\mathbf{2}}) = \langle \text{expr} \rangle_{\mathbf{1}}\ \ldots$$

$$\ldots\ \langle \text{id} \rangle_{\mathbf{1}}(\langle \text{expr} \rangle_{\mathbf{2}})\ \ldots \quad \rightarrow \quad \ldots\ \langle \text{expr} \rangle_{\mathbf{3}}\ \ldots$$

where $\langle \text{expr} \rangle_{\mathbf{3}}$ is $\langle \text{expr} \rangle_{\mathbf{1}}$ with $\langle \text{id} \rangle_{\mathbf{2}}$ replaced by $\langle \text{expr} \rangle_{\mathbf{2}}$

# Rules for Evaluation

- **Rule 1 -** one pattern

$$\ldots\ \langle \text{id} \rangle_{\mathbf{1}}(\langle \text{id} \rangle_{\mathbf{2}}) = \langle \text{expr} \rangle_{\mathbf{1}}\ \ldots$$

$$\ldots\ \langle \text{id} \rangle_{\mathbf{1}}(\langle \text{expr} \rangle_{\mathbf{2}})\ \ldots \qquad \longrightarrow \qquad \ldots\ \langle \text{expr} \rangle_{\mathbf{3}}\ \ldots$$

where $\langle \text{expr} \rangle_{\mathbf{3}}$ is $\langle \text{expr} \rangle_{\mathbf{1}}$ with $\langle \text{id} \rangle_{\mathbf{2}}$ replaced by $\langle \text{expr} \rangle_{\mathbf{2}}$

# Rules for Evaluation

- **Rule 1 -** one pattern

$$\ldots\ \langle\text{id}\rangle_{\mathbf{1}}(\langle\text{id}\rangle_{\mathbf{2}}) = \langle\text{expr}\rangle_{\mathbf{1}}\ \ldots$$

$$\ldots\ \langle\text{id}\rangle_{\mathbf{1}}(\langle\text{expr}\rangle_{\mathbf{2}})\ \ldots \quad \rightarrow \quad \ldots\ \langle\text{expr}\rangle_{\mathbf{3}}\ \ldots$$

where $\langle\text{expr}\rangle_{\mathbf{3}}$ is $\langle\text{expr}\rangle_{\mathbf{1}}$ with $\langle\text{id}\rangle_{\mathbf{2}}$ replaced by $\langle\text{expr}\rangle_{\mathbf{2}}$

- **Rules 2 -** ∞ special cases

| | |
|---|---|
| … (0 + 0) … → … 0 … | … (0 - 0) … → … 0 … |
| … (1 + 0) … → … 1 … | … (1 - 0) … → … 1 … |
| … (2 + 0) … → … 2 … | … (2 - 0) … → … 2 … |
| *etc.* | *etc.* |

# Rules for Evaluation

- **Rule 1 -** one pattern

$$\ldots \langle \mathrm{id} \rangle_1 (\langle \mathrm{id} \rangle_2) = \langle \mathrm{expr} \rangle_1 \ldots$$

$$\ldots \langle \mathrm{id} \rangle_1 (\langle \mathrm{expr} \rangle_2) \ldots \quad \rightarrow \quad \ldots \langle \mathrm{expr} \rangle_3 \ldots$$

where $\langle \mathrm{expr} \rangle_3$ is $\langle \mathrm{expr} \rangle_1$ with $\langle \mathrm{id} \rangle_2$ replaced by $\langle \mathrm{expr} \rangle_2$

- **Rules 2 -** $\infty$ special cases

| | | | | | |
|---|---|---|---|---|---|
| ... (0 + 0) ... | $\rightarrow$ | ... 0 ... | ... (0 - 0) ... | $\rightarrow$ | ... 0 ... |
| ... (1 + 0) ... | $\rightarrow$ | ... 1 ... | ... (1 - 0) ... | $\rightarrow$ | ... 1 ... |
| ... (2 + 0) ... | $\rightarrow$ | ... 2 ... | ... (2 - 0) ... | $\rightarrow$ | ... 2 ... |
| *etc.* | | | *etc.* | | |

When the interpreter is a program instead of an Algebra student, the rules look a little different

# HW 1

On the course web page:

Finger exercises in Racket

Assignment is due **Friday**