

EECS 321

Programming Languages

Winter 2013

Instructor: **Robby Findler**

Course Details

`http://www.eecs.northwestern.edu/~robby/courses/321-2013-winter/`

(or google “findler” and follow the links)

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 (come back next quarter)

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 (come back next quarter)

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

A Grammar for Algebra Programs

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

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

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

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$

| $(\langle \text{expr} \rangle - \langle \text{expr} \rangle)$

| $\langle \text{id} \rangle (\langle \text{expr} \rangle)$

| $\langle \text{id} \rangle$

| $\langle \text{num} \rangle$

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

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

A Grammar for Algebra Programs

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

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

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

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$

| $(\langle \text{expr} \rangle - \langle \text{expr} \rangle)$

| $\langle \text{id} \rangle (\langle \text{expr} \rangle)$

| $\langle \text{id} \rangle$

| $\langle \text{num} \rangle$

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

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

Each **meta-variable**, such as $\langle \text{prog} \rangle$, defines a set

Using a BNF Grammar

$\langle \text{id} \rangle ::= \text{a variable name: } \mathbf{f}, \mathbf{x}, \mathbf{y}, \mathbf{z}, \dots$

$\langle \text{num} \rangle ::= \text{a number: } 1, 42, 17, \dots$

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

Using a BNF Grammar

$\langle \text{id} \rangle ::= \text{a variable name: } \mathbf{f}, \mathbf{x}, \mathbf{y}, \mathbf{z}, \dots$

$\langle \text{num} \rangle ::= \text{a number: } 1, 42, 17, \dots$

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

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

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$
 $\quad | (\langle \text{expr} \rangle - \langle \text{expr} \rangle)$
 $\quad | \langle \text{id} \rangle (\langle \text{expr} \rangle)$
 $\quad | \langle \text{id} \rangle$
 $\quad | \langle \text{num} \rangle$

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

Using a BNF Grammar

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

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

```
⟨expr⟩ ::= (⟨expr⟩ + ⟨expr⟩)
         | (⟨expr⟩ - ⟨expr⟩)
         | ⟨id⟩(⟨expr⟩)
         | ⟨id⟩
         | ⟨num⟩
```



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

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$
 $| (\langle \text{expr} \rangle - \langle \text{expr} \rangle)$
 $| \langle \text{id} \rangle (\langle \text{expr} \rangle)$
 $| \langle \text{id} \rangle$
 $| \langle \text{num} \rangle$



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

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


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

```
⟨expr⟩ ::= (⟨expr⟩ + ⟨expr⟩)
          | (⟨expr⟩ - ⟨expr⟩)
          | ⟨id⟩(⟨expr⟩)
          | ⟨id⟩
          | ⟨num⟩
```



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

```
 $\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$   
|  $(\langle \text{expr} \rangle - \langle \text{expr} \rangle)$   
|  $\langle \text{id} \rangle (\langle \text{expr} \rangle)$  ←  
|  $\langle \text{id} \rangle$   
|  $\langle \text{num} \rangle$ 
```

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

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

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

- combine the examples with literal text

Using a BNF Grammar

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$
 $| (\langle \text{expr} \rangle - \langle \text{expr} \rangle)$
 $| \langle \text{id} \rangle (\langle \text{expr} \rangle)$ ←
 $| \langle \text{id} \rangle$
 $| \langle \text{num} \rangle$

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

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

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

- combine the examples with literal text

Using a BNF Grammar

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$
 $\quad | (\langle \text{expr} \rangle - \langle \text{expr} \rangle)$
 $\quad | \langle \text{id} \rangle (\langle \text{expr} \rangle)$ ←
 $\quad | \langle \text{id} \rangle$
 $\quad | \langle \text{num} \rangle$

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

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

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

- combine the examples with literal text

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

Using a BNF Grammar

$\langle \text{expr} \rangle ::= (\langle \text{expr} \rangle + \langle \text{expr} \rangle)$
 $| (\langle \text{expr} \rangle - \langle \text{expr} \rangle)$
 $| \langle \text{id} \rangle (\langle \text{expr} \rangle)$ ←
 $| \langle \text{id} \rangle$
 $| \langle \text{num} \rangle$

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

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

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

- combine the examples with literal text

Using a BNF Grammar

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

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

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

Using a BNF Grammar

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

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

f(x) = (x + 1) ∈ <defn>

Using a BNF Grammar

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

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

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

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

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

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

$\mathbf{g(y) = f((y - 2))} \in \langle \text{prog} \rangle$

$\mathbf{g(7)}$

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:

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

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

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

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

Evaluation

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

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

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

due to the pattern rule

$$\dots \langle \mathbf{id} \rangle_1 (\langle \mathbf{id} \rangle_2) = \langle \mathbf{expr} \rangle_1 \dots$$

$$\dots \langle \mathbf{id} \rangle_1 (\langle \mathbf{expr} \rangle_2) \dots \quad \rightarrow \quad \dots \langle \mathbf{expr} \rangle_3 \dots$$

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

Rules for Evaluation

- **Rule I** - one pattern

... $\langle \text{id} \rangle_1(\langle \text{id} \rangle_2) = \langle \text{expr} \rangle_1$...

... $\langle \text{id} \rangle_1(\langle \text{expr} \rangle_2)$... \rightarrow ... $\langle \text{expr} \rangle_3$...

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

Rules for Evaluation

- **Rule 1** - one pattern

... $\langle \text{id} \rangle_1 (\langle \text{id} \rangle_2) = \langle \text{expr} \rangle_1$...

... $\langle \text{id} \rangle_1 (\langle \text{expr} \rangle_2)$... \rightarrow ... $\langle \text{expr} \rangle_3$...

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

- **Rules 2** - ∞ special cases

... $(0 + 0)$... \rightarrow ... 0 ...

... $(1 + 0)$... \rightarrow ... 1 ...

... $(2 + 0)$... \rightarrow ... 2 ...

etc.

... $(0 - 0)$... \rightarrow ... 0 ...

... $(1 - 0)$... \rightarrow ... 1 ...

... $(2 - 0)$... \rightarrow ... 2 ...

etc.

Rules for Evaluation

- **Rule 1** - one pattern

$$\dots \langle \text{id} \rangle_1 (\langle \text{id} \rangle_2) = \langle \text{expr} \rangle_1 \dots$$

$$\dots \langle \text{id} \rangle_1 (\langle \text{expr} \rangle_2) \dots \quad \rightarrow \quad \dots \langle \text{expr} \rangle_3 \dots$$

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

- **Rules 2** - ∞ special cases

$$\dots (0 + 0) \dots \rightarrow \dots 0 \dots$$

$$\dots (1 + 0) \dots \rightarrow \dots 1 \dots$$

$$\dots (2 + 0) \dots \rightarrow \dots 2 \dots$$

etc.

$$\dots (0 - 0) \dots \rightarrow \dots 0 \dots$$

$$\dots (1 - 0) \dots \rightarrow \dots 1 \dots$$

$$\dots (2 - 0) \dots \rightarrow \dots 2 \dots$$

etc.

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

HW I

On the course web page:

Finger exercises in Racket

Assignment is due **Friday**