

CS 4110

# Programming Languages & Logics

---

## Lecture 3 Inductive Definitions and Proofs



# Arithmetic Expressions

Last time we defined a simple language of arithmetic expressions,

$$e ::= x \mid n \mid e_1 + e_2 \mid e_1 * e_2 \mid x := e_1 ; e_2$$

and a small-step operational semantics,  $\langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle$ .

# Arithmetic Expressions

Last time we defined a simple language of arithmetic expressions,

$$e ::= x \mid n \mid e_1 + e_2 \mid e_1 * e_2 \mid x := e_1 ; e_2$$

and a small-step operational semantics,  $\langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle$ .

Example:

Assuming  $\sigma$  is a store that maps *foo* to 4...

$$\frac{\frac{\frac{\sigma(\text{foo}) = 4}{\langle \sigma, \text{foo} \rangle \rightarrow \langle \sigma, 4 \rangle} \text{VAR}}{\langle \sigma, \text{foo} + 2 \rangle \rightarrow \langle \sigma, 4 + 2 \rangle} \text{LADD}}{\langle \sigma, (\text{foo} + 2) * (\text{bar} + 1) \rangle \rightarrow \langle \sigma, (4 + 2) * (\text{bar} + 1) \rangle} \text{LMUL}$$

# Properties

---

Today we'll prove some useful program properties by induction.

# Properties

Today we'll prove some useful program properties by induction.

- **Determinism:** Every configuration has at most one successor.

$\forall e \in \mathbf{Exp}. \forall \sigma, \sigma', \sigma'' \in \mathbf{Store}. \forall e', e'' \in \mathbf{Exp}.$   
if  $\langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle$  and  $\langle \sigma, e \rangle \rightarrow \langle \sigma'', e'' \rangle$   
then  $e' = e''$  and  $\sigma' = \sigma''$ .

# Properties

Today we'll prove some useful program properties by induction.

- **Determinism:** Every configuration has at most one successor.

$$\begin{aligned} &\forall e \in \mathbf{Exp}. \forall \sigma, \sigma', \sigma'' \in \mathbf{Store}. \forall e', e'' \in \mathbf{Exp}. \\ &\quad \text{if } \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle \text{ and } \langle \sigma, e \rangle \rightarrow \langle \sigma'', e'' \rangle \\ &\quad \text{then } e' = e'' \text{ and } \sigma' = \sigma''. \end{aligned}$$

- **Termination:** Evaluation of every expression terminates.

$$\begin{aligned} &\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}. \exists \sigma' \in \mathbf{Store}. \exists e' \in \mathbf{Exp}. \\ &\quad \langle \sigma, e \rangle \rightarrow^* \langle \sigma', e' \rangle \text{ and } \langle \sigma', e' \rangle \not\rightarrow, \end{aligned}$$

Where  $\langle \sigma', e' \rangle \not\rightarrow$  is shorthand for:

$$\neg (\exists \sigma'' \in \mathbf{Store}. \exists e'' \in \mathbf{Exp}. \langle \sigma', e' \rangle \rightarrow \langle \sigma'', e'' \rangle)$$

# Soundness

- **Soundness:** Evaluation of every expression yields an integer.

$$\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}. \exists \sigma' \in \mathbf{store}. \exists n' \in \mathbf{Int}. \\ \langle \sigma, e \rangle \rightarrow^* \langle \sigma', n' \rangle$$

# Soundness

It is tempting to try to prove this property.

- **Soundness:** Evaluation of every expression yields an integer.

$$\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}. \exists \sigma' \in \mathbf{store}. \exists n' \in \mathbf{Int}. \\ \langle \sigma, e \rangle \rightarrow^* \langle \sigma', n' \rangle$$

But it's not true!



# Soundness

It is tempting to try to prove this property.

- **Soundness:** Evaluation of every expression yields an integer.

$$\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}. \exists \sigma' \in \mathbf{store}. \exists n' \in \mathbf{Int}. \\ \langle \sigma, e \rangle \rightarrow^* \langle \sigma', n' \rangle$$

But it's not true!

## Counterexample

If  $\sigma = \emptyset$ , then  $\langle \sigma, x \rangle \not\rightarrow$ .

In general, evaluation of an expression can *get stuck*...

# Well-Formedness

---

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

# Well-Formedness

---

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

**Free Variables:** Let's define a function  $fvs(e)$  *inductively*.

# Well-Formedness

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

**Free Variables:** Let's define a function  $fvs(e)$  *inductively*.

$$fvs(x) \triangleq \{x\}$$

# Well-Formedness

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

**Free Variables:** Let's define a function  $fvs(e)$  *inductively*.

$$\begin{aligned} fvs(x) &\triangleq \{x\} \\ fvs(n) &\triangleq \{\} \end{aligned}$$

# Well-Formedness

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

**Free Variables:** Let's define a function  $fvs(e)$  *inductively*.

$$\begin{aligned} fvs(x) &\triangleq \{x\} \\ fvs(n) &\triangleq \{\} \\ fvs(e_1 + e_2) &\triangleq fvs(e_1) \cup fvs(e_2) \end{aligned}$$

# Well-Formedness

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

**Free Variables:** Let's define a function  $fvs(e)$  *inductively*.

$$\begin{aligned} fvs(x) &\triangleq \{x\} \\ fvs(n) &\triangleq \{\} \\ fvs(e_1 + e_2) &\triangleq fvs(e_1) \cup fvs(e_2) \\ fvs(e_1 * e_2) &\triangleq fvs(e_1) \cup fvs(e_2) \end{aligned}$$

# Well-Formedness

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

**Free Variables:** Let's define a function  $fvs(e)$  *inductively*.

$$\begin{aligned} fvs(x) &\triangleq \{x\} \\ fvs(n) &\triangleq \{\} \\ fvs(e_1 + e_2) &\triangleq fvs(e_1) \cup fvs(e_2) \\ fvs(e_1 * e_2) &\triangleq fvs(e_1) \cup fvs(e_2) \\ fvs(x := e_1 ; e_2) &\triangleq fvs(e_1) \cup (fvs(e_2) \setminus \{x\}) \end{aligned}$$



# Well-Formedness

**Idea:** Restrict our attention to **well-formed** configurations  $\langle \sigma, e \rangle$ , where all the variables that  $e$  refers to have values in  $\sigma$ .

**Free Variables:** Let's define a function  $fvs(e)$  *inductively*.

$$\begin{aligned} fvs(x) &\triangleq \{x\} \\ fvs(n) &\triangleq \{\} \\ fvs(e_1 + e_2) &\triangleq fvs(e_1) \cup fvs(e_2) \\ fvs(e_1 * e_2) &\triangleq fvs(e_1) \cup fvs(e_2) \\ fvs(x := e_1 ; e_2) &\triangleq fvs(e_1) \cup (fvs(e_2) \setminus \{x\}) \end{aligned}$$

**Well-Formedness:**

A configuration  $\langle \sigma, e \rangle$  is *well-formed* if and only if  $fvs(e) \subseteq dom(\sigma)$ .

# Progress and Preservation

Now we can formulate two properties that imply soundness:

- Progress:

$\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}.$

$\langle \sigma, e \rangle \text{ well-formed} \implies$

$e \in \mathbf{Int} \text{ or } (\exists e' \in \mathbf{Exp}. \exists \sigma' \in \mathbf{Store}. \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle)$

# Progress and Preservation

Now we can formulate two properties that imply soundness:

- Progress:

$\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}.$

$\langle \sigma, e \rangle \text{ well-formed} \implies$

$e \in \mathbf{Int} \text{ or } (\exists e' \in \mathbf{Exp}. \exists \sigma' \in \mathbf{Store}. \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle)$

- Preservation:

$\forall e, e' \in \mathbf{Exp}. \forall \sigma, \sigma' \in \mathbf{Store}.$

$\langle \sigma, e \rangle \text{ well-formed and } \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle \implies$

$\langle \sigma', e' \rangle \text{ well-formed.}$

# Progress and Preservation

Now we can formulate two properties that imply soundness:

- Progress:

$$\begin{aligned} &\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}. \\ &\quad \langle \sigma, e \rangle \text{ well-formed} \implies \\ &\quad e \in \mathbf{Int} \text{ or } (\exists e' \in \mathbf{Exp}. \exists \sigma' \in \mathbf{Store}. \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle) \end{aligned}$$

- Preservation:

$$\begin{aligned} &\forall e, e' \in \mathbf{Exp}. \forall \sigma, \sigma' \in \mathbf{Store}. \\ &\quad \langle \sigma, e \rangle \text{ well-formed and } \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle \implies \\ &\quad \langle \sigma', e' \rangle \text{ well-formed.} \end{aligned}$$

How are we going to prove these properties? Induction!

# Inductive Sets

# Inductive Sets

An *inductively-defined set*  $A$  is one that can be described using a finite collection of inference rules:

$$\frac{a_1 \in A \quad \dots \quad a_n \in A}{a \in A}$$

This rule states that if  $a_1$  through  $a_n$  are elements of  $A$ , then  $a$  is also an element of  $A$ .

An inference rule with no premises is often called an *axiom*.

The set  $A$  is the smallest set “closed” under these axioms and rules.

# Inductive Set Examples

The natural numbers are an inductive set.

$$\frac{}{0 \in \mathbb{N}} \qquad \frac{n \in \mathbb{N}}{\text{succ}(n) \in \mathbb{N}}$$

# Inductive Set Examples

Every BNF grammar defines an inductive set.

$$e ::= x \mid n \mid e_1 + e_2 \mid e_1 * e_2 \mid x := e_1 ; e_2$$

can be equivalently defined as:

$$\begin{array}{c} \frac{}{x \in \mathbf{Exp}} \qquad \frac{}{n \in \mathbf{Exp}} \\[10pt] \frac{e_1 \in \mathbf{Exp} \quad e_2 \in \mathbf{Exp}}{e_1 + e_2 \in \mathbf{Exp}} \qquad \frac{e_1 \in \mathbf{Exp} \quad e_2 \in \mathbf{Exp}}{e_1 * e_2 \in \mathbf{Exp}} \\[10pt] \frac{e_1 \in \mathbf{Exp} \quad e_2 \in \mathbf{Exp}}{x := e_1 ; e_2 \in \mathbf{Exp}} \end{array}$$



# Inductive Set Examples

The small-step evaluation relation  $\rightarrow$  is an inductive set.

$$\frac{n = \sigma(x)}{\langle \sigma, x \rangle \rightarrow \langle \sigma, n \rangle} \text{VAR}$$

$$\frac{\langle \sigma, e_1 \rangle \rightarrow \langle \sigma', e'_1 \rangle}{\langle \sigma, e_1 + e_2 \rangle \rightarrow \langle \sigma', e'_1 + e_2 \rangle} \text{LADD}$$

$$\frac{\langle \sigma, e_2 \rangle \rightarrow \langle \sigma', e'_2 \rangle}{\langle \sigma, n + e_2 \rangle \rightarrow \langle \sigma', n + e'_2 \rangle} \text{RADD}$$

$$\frac{p = m + n}{\langle \sigma, n + m \rangle \rightarrow \langle \sigma, p \rangle} \text{ADD}$$

$$\frac{\langle \sigma, e_1 \rangle \rightarrow \langle \sigma', e'_1 \rangle}{\langle \sigma, e_1 * e_2 \rangle \rightarrow \langle \sigma', e'_1 * e_2 \rangle} \text{LMUL}$$

$$\frac{\langle \sigma, e_2 \rangle \rightarrow \langle \sigma', e'_2 \rangle}{\langle \sigma, n * e_2 \rangle \rightarrow \langle \sigma', n * e'_2 \rangle} \text{RMUL}$$

$$\frac{p = m \times n}{\langle \sigma, m * n \rangle \rightarrow \langle \sigma, p \rangle} \text{MUL}$$

$$\frac{\langle \sigma, e_1 \rangle \rightarrow \langle \sigma', e'_1 \rangle}{\langle \sigma, x := e_1 ; e_2 \rangle \rightarrow \langle \sigma', x := e'_1 ; e_2 \rangle} \text{ASSGN1}$$

$$\frac{\sigma' = \sigma[x \mapsto n]}{\langle \sigma, x := n ; e_2 \rangle \rightarrow \langle \sigma', e_2 \rangle} \text{ASSGN}$$

# Inductive Set Examples

The multi-step evaluation relation is an inductive set.

$$\frac{}{\langle \sigma, e \rangle \rightarrow^* \langle \sigma, e \rangle} \text{ REFL}$$

$$\frac{\langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle}{\langle \sigma, e \rangle \rightarrow^* \langle \sigma', e' \rangle} \text{ STEP}$$

$$\frac{\langle \sigma, e \rangle \rightarrow^* \langle \sigma', e' \rangle \quad \langle \sigma', e' \rangle \rightarrow^* \langle \sigma'', e'' \rangle}{\langle \sigma, e \rangle \rightarrow^* \langle \sigma'', e'' \rangle} \text{ TRANS}$$

# Inductive Set Examples

The set of free variables of an expression is an inductive set.

$$\frac{}{y \in fvs(y)}$$

$$\frac{y \in fvs(e_1)}{y \in fvs(e_1 + e_2)}$$

$$\frac{y \in fvs(e_2)}{y \in fvs(e_1 + e_2)}$$

$$\frac{y \in fvs(e_1)}{y \in fvs(e_1 * e_2)}$$

$$\frac{y \in fvs(e_2)}{y \in fvs(e_1 * e_2)}$$

$$\frac{y \in fvs(e_1)}{y \in fvs(x := e_1 ; e_2)}$$

$$\frac{y \neq x \quad y \in fvs(e_2)}{y \in fvs(x := e_1 ; e_2)}$$

# Induction Principle

---

Recall the principle of mathematical induction.

To prove  $\forall n. P(n)$ , we must establish several cases.

- Base case:  $P(0)$
- Inductive case:  $P(m) \Rightarrow P(m + 1)$

# Induction Principle

Every inductive set has an analogous principle.

To prove  $\forall a. P(a)$  we must establish several cases.

- **Base cases:**  $P(a)$  holds for each axiom

$$\overline{a \in A}$$

- **Inductive cases:** For each inference rule

$$\frac{a_1 \in A \quad \dots \quad a_n \in A}{a \in A}$$

if  $P(a_1)$  and ... and  $P(a_n)$  then  $P(a)$

# Example: Progress

Recall the progress property.

$\forall e \in \mathbf{Exp}. \forall \sigma \in \mathbf{Store}.$

$\langle \sigma, e \rangle \text{ well-formed} \implies$

$e \in \mathbf{Int} \text{ or } (\exists e' \in \mathbf{Exp}. \exists \sigma' \in \mathbf{Store}. \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle)$

We'll prove this by structural induction on  $e$ .

$$\frac{}{x \in \mathbf{Exp}}$$

$$\frac{}{n \in \mathbf{Exp}}$$

$$\frac{e_1 \in \mathbf{Exp} \quad e_2 \in \mathbf{Exp}}{e_1 + e_2 \in \mathbf{Exp}}$$

$$\frac{e_1 \in \mathbf{Exp} \quad e_2 \in \mathbf{Exp}}{e_1 * e_2 \in \mathbf{Exp}}$$

$$\frac{e_1 \in \mathbf{Exp} \quad e_2 \in \mathbf{Exp}}{x := e_1 ; e_2 \in \mathbf{Exp}}$$