CS 4110

Programming Languages & Logics

Lecture 3
Inductive Definitions and Proofs

`29 August 2016`

~~3 September 2014~~

Announcements

Office Hours

- ~~Fran: Thursday at 2-3pm~~
- ~~Nate: Friday at 11-12pm~~
- ~~Nitesh: TBA~~

Temporary course web page:
https://www.cs.cornell.edu/Courses/cs4110/2016fa/

See the web page for course staff contact info and office hours

Piazza and CMS

- ~~Sign up!~~

Piazza: self-signup
CMS: email Dexter with your netId
see the web page for links

Homework #1

- Out: ~~today~~ Wed Aug 31
- Due: ~~in a week~~ Wed Sept 7

Arithmetic Expressions

Last time we defined a simple language of arithmetic expressions,

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

as well as a small-step operational semantics

Arithmetic Expressions

Last time we defined a simple language of arithmetic expressions,

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

as well as a small-step operational semantics

Example:

Assuming σ is a store that maps *foo* to 4...

$$\dfrac{\dfrac{\dfrac{\sigma(\mathit{foo}) = 4}{\langle \sigma, \mathit{foo} \rangle \rightarrow \langle \sigma, 4 \rangle}\text{ Var}}{\langle \sigma, \mathit{foo} + 2 \rangle \rightarrow \langle \sigma, 4 + 2 \rangle}\text{ LAdd}}{\langle \sigma, (\mathit{foo} + 2) * (\mathit{bar} + 1) \rangle \rightarrow \langle \sigma, (4 + 2) * (\mathit{bar} + 1) \rangle}\text{ LMul}$$

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

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

Properties

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

- Determinism: every configuration has at most one successor

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

- Termination: evaluation of every expression terminates,

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

where $\langle \sigma', e'\rangle \not\longrightarrow$ is shorthand for
$\neg\,(\exists \sigma'' \in \mathbf{Store}.\ \exists e'' \in \mathbf{Exp}.\ \langle \sigma', e'\rangle \longrightarrow \langle \sigma'', e''\rangle)$.

Soundness

It is tempting to try to prove the following property.

- Soundness: evaluation of every expression yields an integer,

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

But unfortunately it is not true!

Soundness

It is tempting to try to prove the following property.

- Soundness: evaluation of every expression yields an integer,

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

But unfortunately it is not true!

Counterexample

If σ is the undefined function, then $\langle \sigma, x \rangle \not\longrightarrow$.

Soundness

It is tempting to try to prove the following property.

- Soundness: evaluation of every expression yields an integer,

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

But unfortunately it is not true!

Counterexample

If σ is the undefined function, then $\langle \sigma, x \rangle \not\longrightarrow$.

In generally, evaluation of an expression can “get stuck”...

Well-Formedness

Idea: restrict our attention to *well-formed* configurations $\langle \sigma, e \rangle$, where σ is defined on (at least) the *free variables* in e.

Well-Formedness

Idea: restrict our attention to *well-formed* configurations $\langle \sigma, e \rangle$, where σ is defined on (at least) the *free variables* in e.

Free Variables:

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

Well-Formedness

Idea: restrict our attention to *well-formed* configurations $\langle \sigma, e \rangle$, where σ is defined on (at least) the *free variables* in *e*.

Free Variables:

$$\begin{array}{lcl} \mathit{fvs}(x) & \triangleq & \{x\} \\ \mathit{fvs}(n) & \triangleq & \{\} \end{array}$$

Well-Formedness

Idea: restrict our attention to *well-formed* configurations $\langle \sigma, e \rangle$, where σ is defined on (at least) the *free variables* in *e*.

Free Variables:

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

Well-Formedness

Idea: restrict our attention to *well-formed* configurations $\langle \sigma, e \rangle$, where σ is defined on (at least) the *free variables* in e.

Free Variables:

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

Well-Formedness

Idea: restrict our attention to *well-formed* configurations $\langle \sigma, e \rangle$, where σ is defined on (at least) the *free variables* in *e*.

Free Variables:

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

Well-Formedness

Idea: restrict our attention to *well-formed* configurations $\langle \sigma, e \rangle$, where σ is defined on (at least) the *free variables* in e.

Free Variables:

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

Well-Formedness:

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

Progress and Preservation

Now we can formulate two properties that imply soundness:

- Progress:

$$
\begin{array}{l}
\forall e \in \mathbf{Exp}.\ \forall \sigma \in \mathbf{Store}. \\
\quad \langle \sigma, e \rangle \text{ well-formed } \implies \\
\quad e \in \mathbf{Int} \text{ or } \left(\exists e' \in \mathbf{Exp}.\ \exists \sigma' \in \mathbf{Store}.\ \langle \sigma, e \rangle \longrightarrow \langle \sigma', e' \rangle\right)
\end{array}
$$

Progress and Preservation

Now we can formulate two properties that imply soundness:

- Progress:

$$\begin{array}{l} \forall e \in \textbf{Exp}.\ \forall \sigma \in \textbf{Store}. \\ \quad \langle \sigma, e \rangle \text{ well-formed } \implies \\ \quad e \in \textbf{Int} \text{ or } (\exists e' \in \textbf{Exp}.\ \exists \sigma' \in \textbf{Store}.\ \langle \sigma, e \rangle \longrightarrow \langle \sigma', e' \rangle) \end{array}$$

- Preservation:

$$\begin{array}{l} \forall e, e' \in \textbf{Exp}.\ \forall \sigma, \sigma' \in \textbf{Store}. \\ \quad \langle \sigma, e \rangle \text{ well-formed and } \langle \sigma, e \rangle \longrightarrow \langle \sigma', e' \rangle \implies \\ \quad \langle \sigma', e' \rangle \text{ well-formed.} \end{array}$$

Progress and Preservation

Now we can formulate two properties that imply soundness:

- Progress:

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

- Preservation:

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

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:

$$\frac{a_1 \in A \qquad \ldots \qquad a_n \in A}{a \in A}$$

This rules 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.

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

Inductive Set Examples

Every BNF grammar defines an inductive set.

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

can be equivalently defined as:

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

$$\frac{e_1 \in \mathbf{Exp} \qquad e_2 \in \mathbf{Exp}}{e_1 \mathbf{+} e_2 \in \mathbf{Exp}} \qquad \frac{e_1 \in \mathbf{Exp} \qquad e_2 \in \mathbf{Exp}}{e_1 \mathbf{*} e_2 \in \mathbf{Exp}}$$

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

Inductive Set Examples

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

$$\frac{n = \sigma(x)}{\langle \sigma, x \rangle \to \langle \sigma, n \rangle}\ \text{Var}$$

$$\frac{\langle \sigma, e_1 \rangle \to \langle \sigma', e_1' \rangle}{\langle \sigma, e_1 + e_2 \rangle \to \langle \sigma', e_1' + e_2 \rangle}\ \text{LAdd} \qquad \frac{\langle \sigma, e_2 \rangle \to \langle \sigma', e_2' \rangle}{\langle \sigma, n + e_2 \rangle \to \langle \sigma', n + e_2' \rangle}\ \text{RAdd}$$

$$\frac{p = m + n}{\langle \sigma, n + m \rangle \to \langle \sigma, p \rangle}\ \text{Add} \qquad \frac{\langle \sigma, e_1 \rangle \to \langle \sigma', e_1' \rangle}{\langle \sigma, e_1 * e_2 \rangle \to \langle \sigma', e_1' * e_2 \rangle}\ \text{LMul}$$

$$\frac{\langle \sigma, e_2 \rangle \to \langle \sigma', e_2' \rangle}{\langle \sigma, n * e_2 \rangle \to \langle \sigma', n * e_2' \rangle}\ \text{RMul} \qquad \frac{p = m \times n}{\langle \sigma, m * n \rangle \to \langle \sigma, p \rangle}\ \text{Mul}$$

$$\frac{\langle \sigma, e_1 \rangle \to \langle \sigma', e_1' \rangle}{\langle \sigma, x := e_1 \ ;\ e_2 \rangle \to \langle \sigma', x := e_1' \ ;\ e_2 \rangle}\ \text{Assgn1} \qquad \frac{\sigma' = \sigma[x \mapsto n]}{\langle \sigma, x := n\ ;\ e_2 \rangle \to \langle \sigma', e_2 \rangle}\ \text{Assgn}$$

Inductive Set Examples

The multi-step evaluation relation is an inductive set.

$$\frac{}{\langle \sigma, e \rangle \longrightarrow^{*} \langle \sigma, e \rangle}\ \textsf{Refl}$$

$$\frac{\langle \sigma, e \rangle \longrightarrow \langle \sigma', e' \rangle}{\langle \sigma, e \rangle \longrightarrow^{*} \langle \sigma', e' \rangle}\ \textsf{Step}$$

$$\frac{\langle \sigma, e \rangle \longrightarrow^{*} \langle \sigma', e' \rangle \qquad \langle \sigma', e' \rangle \longrightarrow^{*} \langle \sigma'', e'' \rangle}{\langle \sigma, e \rangle \longrightarrow^{*} \langle \sigma'', e'' \rangle}\ \textsf{Trans}$$

Inductive Set Examples

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

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

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

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

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

$$\frac{}{a \in A}$$

- Inductive cases: For each inference rule

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

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

Example: Progress

Recall the progress property.

$$\begin{array}{l} \forall e \in \textbf{Exp}.\ \forall \sigma \in \textbf{Store}. \\ \quad \langle \sigma, e \rangle \text{ well-formed } \implies \\ \quad e \in \textbf{Int} \text{ or } \left(\exists e' \in \textbf{Exp}.\ \exists \sigma' \in \textbf{Store}.\ \langle \sigma, e \rangle \longrightarrow \langle \sigma', e' \rangle\right) \end{array}$$

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

$$\frac{}{x \in \textbf{Exp}} \qquad \frac{}{n \in \textbf{Exp}}$$

$$\frac{e_1 \in \textbf{Exp} \qquad e_2 \in \textbf{Exp}}{e_1 + e_2 \in \textbf{Exp}} \qquad \frac{e_1 \in \textbf{Exp} \qquad e_2 \in \textbf{Exp}}{e_1 * e_2 \in \textbf{Exp}}$$

$$\frac{e_1 \in \textbf{Exp} \qquad e_2 \in \textbf{Exp}}{x := e_1\ ;\ e_2 \in \textbf{Exp}}$$