Expressing Program Properties; Inductive sets and inductive proofs

Lecture 3 Thursday, February 3, 2022

1 Expressing Program Properties

Now that we have defined our small-step operational semantics, we can formally express different properties of programs. For instance:

- **Progress:** For each store σ and expression e that is not an integer, there exists a possible transition for $\langle e, \sigma \rangle$:

$$\forall e \in \textbf{Exp}.\ \forall \sigma \in \textbf{Store}.\ \text{either } e \in \textbf{Int} \text{ or } \exists e', \sigma'.\ \langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle$$

- **Termination:** The evaluation of each expression terminates:

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

- **Deterministic Result:** The evaluation result for any expression is deterministic:

$$\forall e \in \textbf{Exp}.\ \forall \sigma_0, \sigma, \sigma' \in \textbf{Store}.\ \forall n, n' \in \textbf{Int}.$$
$$\text{if } \langle e, \sigma_0 \rangle \longrightarrow^* \langle n, \sigma \rangle \text{ and } \langle e, \sigma_0 \rangle \longrightarrow^* \langle n', \sigma' \rangle \text{ then } n = n' \text{ and } \sigma = \sigma'.$$

How can we prove such kinds of properties? *Inductive proofs* allow us to prove statements such as the properties above. In the next lecture, We will first introduce inductive sets, introduce inductive proofs, and then show how we can prove progress (the first property above) using inductive techniques.

2 Inductive sets

Induction is an important concept in the theory of programming language. We have already seen it used to define language syntax, and to define the small-step operational semantics for the arithmetic language.

An inductively defined set A is a set that is built using a set of axioms and inductive (inference) rules.

Axioms of the form

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

indicate that a is in the set A. Inductive rules

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

indicate that if $a_1, \ldots, a_n$ are all elements of A, then a is also an element of A.

The set A is the set of all elements that can be inferred to belong to A using a (finite) number of applications of these rules, starting only from axioms. In other words, for each element a of A, we must be able to construct a finite proof tree whose final conclusion is $a \in A$.

Example 1. The language of a grammar is an inductive set. For instance, the set of arithmetic expressions can be described with 2 axioms, and 3 inductive rules:

$$\textsc{Var}\ \frac{}{x \in \textbf{Exp}}\ x \in \textbf{Var} \qquad \textsc{Int}\ \frac{}{n \in \textbf{Exp}}\ n \in \textbf{Int}$$

$$\textsc{Add}\ \frac{e_1 \in \textbf{Exp} \quad e_2 \in \textbf{Exp}}{e_1 + e_2 \in \textbf{Exp}} \qquad \textsc{Mul}\ \frac{e_1 \in \textbf{Exp} \quad e_2 \in \textbf{Exp}}{e_1 \times e_2 \in \textbf{Exp}} \qquad \textsc{Asg}\ \frac{e_1 \in \textbf{Exp} \quad e_2 \in \textbf{Exp}}{x := e_1; e_2 \in \textbf{Exp}}\ x \in \textbf{Var}$$

This is equivalent to the grammar $e ::= x \mid n \mid e_1 + e_2 \mid e_1 \times e_2 \mid x := e_1; e_2$.

To show that $(\mathsf{foo} + 3) \times \mathsf{bar}$ is an element of the set **Exp**, it suffices to show that $\mathsf{foo} + 3$ and bar are in the set **Exp**, since the inference rule MUL can be used, with $e_1 \equiv \mathsf{foo} + 3$ and $e_2 \equiv \mathsf{foo}$, and, since if the premises $\mathsf{foo} + 3 \in \textbf{Exp}$ and $\mathsf{bar} \in \textbf{Exp}$ are true, then the conclusion $(\mathsf{foo} + 3) \times \mathsf{bar} \in \textbf{Exp}$ is true.

Similarly, we can use rule ADD to show that if $\mathsf{foo} \in \textbf{Exp}$ and $3 \in \textbf{Exp}$, then $(\mathsf{foo} + 3) \in \textbf{Exp}$. We can use axiom VAR (twice) to show that $\mathsf{foo} \in \textbf{Exp}$ and $\mathsf{bar} \in \textbf{Exp}$ and rule INT to show that $3 \in \textbf{Exp}$. We can put these all together into a derivation whose conclusion is $(\mathsf{foo} + 3) \times \mathsf{bar} \in \textbf{Exp}$:

$$\textsc{Mul}\ \frac{\textsc{Add}\ \dfrac{\textsc{Var}\ \dfrac{}{\mathsf{foo} \in \textbf{Exp}} \quad \textsc{Int}\ \dfrac{}{3 \in \textbf{Exp}}}{(\mathsf{foo} + 3) \in \textbf{Exp}} \quad \textsc{Var}\ \dfrac{}{\mathsf{bar} \in \textbf{Exp}}}{(\mathsf{foo} + 3) \times \mathsf{bar} \in \textbf{Exp}}$$

Example 2. The natural numbers can be inductively defined:

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

where $succ(n)$ is the successor of n.

Example 3. The small-step evaluation relation $\longrightarrow$ is an inductively defined set. The definition of this set is given by the semantic rules.

Example 4. The transitive, reflexive closure $\longrightarrow^*$ (i.e., the multi-step evaluation relation) can be inductively defined:

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

3 Inductive proofs

We can prove facts about elements of an inductive set using an inductive reasoning that follows the structure of the set definition.

3.1 Mathematical induction

You have probably seen proofs by induction over the natural numbers, called *mathematical induction*. In such proofs, we typically want to prove that some property P holds for all natural numbers, that is, $\forall n \in \mathbb{N}.\ P(n)$. A proof by induction works by first proving that $P(0)$ holds, and then proving for all $m \in \mathbb{N}$, if $P(m)$ then $P(m+1)$. The *inductive reasoning principle* of mathematical induction can be stated as follows:

For any property P,

If

- $P(0)$ holds
- For all natural numbers n, if $P(n)$ holds then $P(n+1)$ holds

then

for all natural numbers k, $P(k)$ holds.

Here, P is the property that we are proving by induction. The assertion that $P(0)$ is the *basis* of the induction (also called the *base case*). Establishing that $P(m) \implies P(m+1)$ is called *inductive step*, or the *inductive case*. While proving the inductive step, the assumption that $P(m)$ holds is called the *inductive hypothesis*.

This inductive reasoning principle gives us a technique to prove that a property holds for all natural numbers, which is an infinite set. Why is the inductive reasoning principle for natural numbers sound? That is, why does it work? One intuition is that for any natural number k you choose, k is either zero, or the result of applying the successor operation a finite number of times to zero. That is, we have a finite proof tree that k is a natural number, using the inference rules given in Example 2 of Lecture 2. Given this proof tree, the leaf of this tree is that $0 \in \mathbb{N}$. We know that $P(0)$ holds. Moreover, since we have for all natural numbers n, if $P(n)$ holds then $P(n+1)$ holds, and we have $P(0)$, we also have $P(1)$. Since we have $P(1)$, we also have $P(2)$, and so on. That is, for each node of the proof tree, we are showing that the property holds of that node. Eventually we will reach the root of the tree, that $k \in N$, and we will have $P(k)$.

3.2 Induction on inductively-defined sets

For every inductively defined set, we have a corresponding inductive reasoning principle. The template for this inductive reasoning principle, for an inductively defined set A, is as follows.

For any property P,

If

- **Base cases:** For each axiom

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

  $P(a)$ holds.

- **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 $\dots$ and $P(a_n)$ then $P(a)$.

then

for all $a \in A$, $P(a)$ holds.

Again, P is the property that we are proving by induction. Each axiom for the inductively defined set (i.e., each inference rule with no premises) is a base case for the induction. Each inductive inference rules (i.e., inference rules with one or more premises) are the inductive cases. When proving an inductive case (i.e., if $P(a_1)$ and $\ldots$ and $P(a_n)$ then $P(a)$), the assumption that $P(a_1)$ and $\ldots$ and $P(a_n)$ are true is the *inductive hypothesis*.

If the set A is the set of natural numbers (see Example 2 above), then the requirements given above for proving that P holds for all elements of A are equivalent to mathematical induction.

If A describes a syntactic set, then we refer to induction following the requirements above as *structural induction*. If A is an operational semantics relation (such as the small-step operational semantics relation $\longrightarrow$) then such induction is called *induction on derivations*. We will see examples of structural induction and induction on derivations throughout the course.

The intuition for why the inductive reasoning principle works is that same as the intuition for why mathematical induction works, i.e., for why the inductive reasoning principle for natural numbers works.

3.3 Example inductive reasoning principles

Let's consider a specific inductively defined set, and consider the inductive reasoning principle for that set: the set of arithmetic expressions **Exp**, inductively defined by the grammar

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

Here is the inductive reasoning principle for the set **Exp**.

For any property P,

If

- For all variables x, $P(x)$ holds.
- For all integers n, $P(n)$ holds.
- For all $e_1 \in$ **Exp** and $e_2 \in$ **Exp**, if $P(e_1)$ and $P(e_2)$ then $P(e_1 + e_2)$ holds.
- For all $e_1 \in$ **Exp** and $e_2 \in$ **Exp**, if $P(e_1)$ and $P(e_2)$ then $P(e_1 \times e_2)$ holds.
- For all variables x and $e_1 \in$ **Exp** and $e_2 \in$ **Exp**, if $P(e_1)$ and $P(e_2)$ then $P(x := e_1; e_2)$ holds.

then

for all $e \in$ **Exp**, $P(e)$ holds.

Here is the inductive reasoning principle for the small-step relation on arithmetic expressions, i.e., for the set $\longrightarrow$.

For any property P,

If

- VAR: For all variables x, stores σ and integers n such that $\sigma(x) = n$, $P(\langle x, \sigma\rangle \longrightarrow \langle n, \sigma\rangle)$ holds.
- ADD: For all integers n, m, p such that $p = n + m$, and stores σ, $P(\langle n + m, \sigma\rangle \longrightarrow \langle p, \sigma\rangle)$ holds.
- MUL: For all integers n, m, p such that $p = n \times m$, and stores σ, $P(\langle n \times m, \sigma\rangle \longrightarrow \langle p, \sigma\rangle)$ holds.
- ASG: For all variables x, integers n and expressions $e \in$ **Exp**, $P(\langle x := n; e, \sigma\rangle \longrightarrow \langle e, \sigma[x \mapsto n]\rangle)$ holds.
- LADD: For all expressions $e_1, e_2, e_1' \in$ **Exp** and stores σ and σ', if $P(\langle e_1, \sigma\rangle \longrightarrow \langle e_1', \sigma'\rangle)$ holds then $P(\langle e_1 + e_2, \sigma\rangle \longrightarrow \langle e_1' + e_2, \sigma'\rangle)$ holds.
- RADD: For all integers n, expressions $e_2, e_2' \in$ **Exp** and stores σ and σ', if $P(\langle e_2, \sigma\rangle \longrightarrow \langle e_2', \sigma'\rangle)$ holds then $P(\langle n + e_2, \sigma\rangle \longrightarrow \langle n + e_2', \sigma'\rangle)$ holds.
- LMUL: For all expressions $e_1, e_2, e_1' \in$ **Exp** and stores σ and σ', if $P(\langle e_1, \sigma\rangle \longrightarrow \langle e_1', \sigma'\rangle)$ holds then $P(\langle e_1 \times e_2, \sigma\rangle \longrightarrow \langle e_1' \times e_2, \sigma'\rangle)$ holds.
- RMUL: For all integers n, expressions $e_2, e_2' \in$ **Exp** and stores σ and σ', if $P(\langle e_2, \sigma\rangle \longrightarrow \langle e_2', \sigma'\rangle)$ holds then $P(\langle n \times e_2, \sigma\rangle \longrightarrow \langle n \times e_2', \sigma'\rangle)$ holds.
- ASG1: For all variables x, expressions $e_1, e_2, e_1' \in$ **Exp** and stores σ and σ', if $P(\langle e_1, \sigma\rangle \longrightarrow \langle e_1', \sigma'\rangle)$ holds then $P(\langle x := e_1; e_2, \sigma\rangle \longrightarrow \langle x := e_1'; e_2, \sigma'\rangle)$ holds.

then

for all $\langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$, $P(\langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle)$ holds.

Note that there is one case for each inference rule: 4 axioms (VAR, ADD, MUL and ASG) and 5 inductive rules (LADD, RADD, LMUL, RMUL, ASG1).

The inductive reasoning principles give us a technique for showing that a property holds of every element in an inductively defined set. Let's consider some examples. Make sure you understand how the appropriate inductive reasoning principle is being used in each of these examples.

3.4 Example: Proving progress

Let's consider the progress property defined above, and repeated here:

Progress: For each store σ and expression e that is not an integer, there exists a possible transition for $\langle e, \sigma \rangle$:

$$\forall e \in \textbf{Exp}.\ \forall \sigma \in \textbf{Store}.\ \text{either } e \in \textbf{Int} \text{ or } \exists e', \sigma'.\ \langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle$$

Let's rephrase this property as: for all expressions e, $P(e)$ holds, where:

$$P(e) = \forall \sigma.\ (e \in \textbf{Int}) \vee (\exists e', \sigma'.\ \langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle)$$

The idea is to build a proof that follows the inductive structure in the grammar of expressions:

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

This is called "structural induction on the expressions e". We must examine each case in the grammar and show that $P(e)$ holds for that case. Since the grammar productions $e = e_1 + e_2$ and $e = e_1 \times e_2$ and $e = x := e_1; e_2$ are inductive definitions of expressions, they are inductive steps in the proof; the other two cases $e = x$ and $e = n$ are the basis of induction. The proof goes as follows:

We will prove by structural induction on expressions **Exp** that for all expressions $e \in$ **Exp** we have

$$P(e) = \forall \sigma.\ (e \in \textbf{Int}) \vee (\exists e', \sigma'.\ \langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle).$$

Consider the possible cases for e.

- Case $e = x$. By the VAR axiom, we can evaluate $\langle x, \sigma \rangle$ in any state: $\langle x, \sigma \rangle \longrightarrow \langle n, \sigma \rangle$, where $n = \sigma(x)$. So $e' = n$ is a witness that there exists e' such that $\langle x, \sigma \rangle \longrightarrow \langle e', \sigma \rangle$, and $P(x)$ holds.
- Case $e = n$. Then $e \in$ **Int**, so $P(n)$ trivially holds.
- Case $e = e_1 + e_2$. This is an inductive step. The inductive hypothesis is that P holds for subexpressions e_1 and e_2. We need to show that P holds for e. In other words, we want to show that $P(e_1)$ and $P(e_2)$ implies $P(e)$. Let's expand these properties. We know that the following hold:

$$P(e_1) = \forall\sigma.\ (e_1 \in \textbf{Int}) \vee (\exists e', \sigma'.\ \langle e_1, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle)$$
$$P(e_2) = \forall\sigma.\ (e_2 \in \textbf{Int}) \vee (\exists e', \sigma'.\ \langle e_2, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle)$$

and we want to show:

$$P(e) = \forall\sigma.\ (e \in \textbf{Int}) \vee (\exists e', \sigma'.\ \langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle)$$

We must inspect several subcases.

First, if both e_1 and e_2 are integer constants, say $e_1 = n_1$ and $e_2 = n_2$, then by rule ADD we know that the transition $\langle n_1 + n_2, \sigma\rangle \longrightarrow \langle n, \sigma\rangle$ is valid, where n is the sum of n_1 and n_2. Hence, $P(e) = P(n_1 + n_2)$ holds (with witness $e' = n$).

Second, if e_1 is not an integer constant, then by the inductive hypothesis $P(e_1)$ we know that $\langle e_1, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$ for some e' and σ'. We can then use rule LADD to conclude $\langle e_1 + e_2, \sigma\rangle \longrightarrow \langle e' + e_2, \sigma'\rangle$, so $P(e) = P(e_1 + e_2)$ holds.

Third, if e_1 is an integer constant, say $e_1 = n_1$, but e_2 is not, then by the inductive hypothesis $P(e_2)$ we know that $\langle e_2, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$ for some e' and σ'. We can then use rule RADD to conclude $\langle n_1 + e_2, \sigma\rangle \longrightarrow \langle n_1 + e', \sigma'\rangle$, so $P(e) = P(n_1 + e_2)$ holds.

- Case $e = e_1 \times e_2$ and case $e = x := e_1; e_2$. These are also inductive cases, and their proofs are similar to the previous case. [Note that if you were writing this proof out for a homework, you should write these cases out in full.]

3.5 A recipe for inductive proofs

In this class, you will be asked to write inductive proofs. Until you are used to doing them, inductive proofs can be difficult. Here is a recipe that you should follow when writing inductive proofs. Note that this recipe was followed above.

1. State what you are inducting over. In the example above, we are doing structural induction on the expressions e.

2. State the property P that you are proving by induction. (Sometimes, as in the proof above the property P will be essentially identical to the theorem/lemma/property that you are proving; other times the

property we prove by induction will need to be stronger than theorem/lemma/property you are proving in order to get the different cases to go through.)

3. Make sure you know the inductive reasoning principle for the set you are inducting on.

4. Go through each case. For each case, don't be afraid to be verbose, spelling out explicitly how the meta-variables in an inference rule are instantiated in this case.

3.6 Example: the store change is incremental

Let's see another example of an inductive proof, this time doing an induction on the derivation of the small-step operational semantics relation. The property we will prove is that for all expressions e and stores σ, if $\langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle$ then either $\sigma = \sigma'$ or there is some variable x and integer n such that $\sigma' = \sigma[x \mapsto n]$. That is, in one small step, either the new store is identical to the old store, or is the result of updating a single program variable.

Theorem 1. *For all expressions e and stores σ, if $\langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle$ then either $\sigma = \sigma'$ or there is some variable x and integer n such that $\sigma' = \sigma[x \mapsto n]$.*

Proof of Theorem 1. We proceed by induction on the derivation of $\langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle$. Suppose we have e, σ, e' and σ' such that $\langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle$. The property P that we will prove of e, σ, e' and σ', which we will write as $P(\langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle)$, is that either $\sigma = \sigma'$ or there is some variable x and integer n such that $\sigma' = \sigma[x \mapsto n]$:

$$P(\langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle) \triangleq \sigma = \sigma' \vee (\exists x \in \textbf{Var}, n \in \textbf{Int}.\ \sigma' = \sigma[x \mapsto n]).$$

Consider the cases for the derivation of $\langle e, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle$.

- Case ADD. This is an axiom. Here, $e \equiv n + m$ and $e' = p$ where p is the sum of m and n, and $\sigma' = \sigma$. The result holds immediately.

- Case LADD. This is an inductive case. Here, $e \equiv e_1 + e_2$ and $e' \equiv e_1' + e_2$ and $\langle e_1, \sigma \rangle \longrightarrow \langle e_1', \sigma' \rangle$. By the inductive hypothesis, applied to $\langle e_1, \sigma \rangle \longrightarrow \langle e_1', \sigma' \rangle$, we have that either $\sigma = \sigma'$ or there is some variable x and integer n such that $\sigma' = \sigma[x \mapsto n]$, as required.

- Case ASG. This is an axiom. Here $e \equiv x := n; e_2$ and $e' \equiv e_2$ and $\sigma' = \sigma[x \mapsto n]$. The result holds immediately.

- We leave the other cases (VAR, RADD, LMUL, RMUL, MUL, and ASG1) as exercises for the reader. Seriously, try them. Make sure you can do them. Go on, you're reading these notes, you may as well try the exercise.

□

3.7 Heuristic: structural induction vs. rule induction

In the previous section, we prove that the store change is incremental by induction on the small-step derivation (Theorem 1).

Can we prove the same theorem by structural induction on e instead? Yes, we can because the rules we consider operate on subexpressions.

First, our property $P(e)$ now includes the step $\langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$ as a condition:

$$P(e) \triangleq \sigma = \forall \sigma, e', \sigma'.\ \langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle \Longrightarrow \sigma = \sigma' \vee (\exists x \in \mathbf{Var}, n \in \mathbf{Int}.\ \sigma' = \sigma[x \mapsto n]).$$

We need to show $P(e)$ for each syntax case of e. To show $P(e)$, we *assume* $\langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$ and show $\sigma = \sigma' \vee (\exists x \in \mathbf{Var}, n \in \mathbf{Int}.\ \sigma' = \sigma[x \mapsto n])$. To tailor the step assumption to a syntax case, we *invert* it, keeping only the rules that could possibly match the syntax case. Then for each rule, we can assume the premise and side-condition to show our overall conclusion. When making use of an indutive hypothesis $P(e_0)$ for some sub-expression, we must provide the step assumption which we could obtain from the premise of an inverted rule.

We will walk through some cases to clarify the process. Because we're doing structural induction, we have to consider all 5 syntax cases to complete the proof.

- Case $e = x$. We assume the step $\langle x, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$, and need to show $\sigma = \sigma' \vee (\exists x \in \mathbf{Var}, n \in \mathbf{Int}.\ \sigma' = \sigma[x \mapsto n])$. By inversion of the step assumption only VAR applies, and we know $e' = n$, $n = \sigma(x)$, $\sigma = \sigma'$, which shows the left-hand side of the conclusion.

- Case $e = n$. We assume the step $\langle n, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$, and need to show $\sigma = \sigma' \vee (\exists x \in \mathbf{Var}, n \in \mathbf{Int}.\ \sigma' = \sigma[x \mapsto n])$. By inversion of the step assumption, no rule applies and we are done.

- Case $e = e_1 + e_2$. We assume the step $\langle e_1 + e_2, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$, and need to show $\sigma = \sigma' \vee (\exists x \in \mathbf{Var}, n \in \mathbf{Int}.\ \sigma' = \sigma[x \mapsto n])$. By inversion, we have three cases to consider: LADD, RADD, ADD.

– Case LADD. We know $\langle e_1, \sigma \rangle \longrightarrow \langle e_1', \sigma' \rangle$. By in the induction hypothesis on e_1:

$$P(e_1) = \sigma = \forall \sigma, e', \sigma'.\ \langle e_1, \sigma \rangle \longrightarrow \langle e', \sigma' \rangle \Longrightarrow \sigma = \sigma' \vee (\exists x \in \textbf{Var}, n \in \textbf{Int}.\ \sigma' = \sigma[x \mapsto n]).$$

By instantiating $P(e_1)$ with σ, e_1', σ' and the step premise of the inverted rule, we get $\sigma = \sigma' \vee (\exists x \in \textbf{Var}, n \in \textbf{Int}.\ \sigma' = \sigma[x \mapsto n])$, which is what we need to show.

– Case RADD. Left as an exercise.

– Case ADD. Left as an exercise.

- Case $e = e_1 \times e_2$. Left as an exercise.

- Case $x := e_1; e_2$. Left as an exercise.

3.8 Heuristic: Picking an inductive measure

Pick an inductive measure that follows your argument (e.g. structural induction e vs mathematical induction on the number of steps) and be sure that you have a tangible member of an inductive set (e.g. you can induct on a premise but not on a conclusion!).