Inductive definitions and proofs

Lecture 2 Thursday, January 30, 2014

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:
$$\forall e \in \textbf{Exp}.\ \forall \sigma_0 \in \textbf{Store}.\ \exists \sigma \in \textbf{Store}.\ \exists n \in \textbf{Int}.\ \langle e, \sigma_0 \rangle \longrightarrow^* \langle n, \sigma \rangle$$
- **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. We 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 (without assignment) can be described with 2 axioms, and 2 inductive rules:

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

$$\textsc{Add}\ \frac{e_1 \in \textbf{Exp} \quad e_2 \in \textbf{Exp}}{e_1 + e_2 \in \textbf{Exp}} \qquad \textsc{Mul}\ \frac{e_1 \in \textbf{Exp} \quad e_2 \in \textbf{Exp}}{e_1 \times e_2 \in \textbf{Exp}}$$

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

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 $\mathsf{foo} + 3 \in$ **Exp** and $\mathsf{bar} \in$ **Exp** are true, then the conclusion $(\mathsf{foo} + 3) \times \mathsf{bar} \in$ **Exp** is true.

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

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

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 principle of mathematical induction can be stated succinctly as

$$P(0) \text{ and } (\forall m \in N.\ P(m) \implies P(m+1)) \implies \forall n \in \mathbb{N}.\ P(n).$$

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*.

3.2 Structural induction

Given an inductively defined set A, to prove that property P holds for all elements of A, we need to show:

1. **Base cases:** For each axiom

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

   $P(a)$ holds.

2. **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 $\ldots$ and $P(a_n)$ then $P(a)$.

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 is 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.

3.3 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 show by structural induction that for all expressions e 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 $e \in \textbf{Int}$, 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.]

3.4 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 *inductive hypothesis* P that you are proving by induction. (Sometimes, as in the proof above the inductive hypothesis P will be essentially identical to the theorem/lemma/property that you are proving; other times the inductive hypothesis will need to be stronger than theorem/lemma/property you are proving in order to get the different cases to go through.)
3. 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.

3.5 Example: the store changes 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. We proceed by induction on the derivation of $\langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$. The inductive hypothesis is that 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 $\sigma' = \sigma[x \mapsto n]$.

Suppose we have a derivation of $\langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$ for some e, σ, e', and σ'. Assume that the inductive hypothesis holds for any subderivation $\langle e_0, \sigma_0\rangle \longrightarrow \langle e_0', \sigma_0'\rangle$ used in the derivation of $\langle e, \sigma\rangle \longrightarrow \langle e', \sigma'\rangle$. Consider the last rule used in 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.

□