Inductive proofs and Large-step semantics

Lecture 3 Tuesday, February 2, 2016

1 Inductive proofs, continued

Last lecture we considered inductively defined sets, and saw how the principle of mathematical induction (i.e., induction on the natural numbers) could be generalized to induction on other inductively-defined sets.

1.1 Inductive reasoning principle

The inductive reasoning principle for natural numbers 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.

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 $P(k)$.

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
>
> $$\frac{a_1 \in A \quad \ldots \quad a_n \in A}{a \in A}\ ,$$
>
> if $P(a_1)$ and $\ldots$ and $P(a_n)$ then $P(a)$.
>
> **then**
>
> for all $a \in A$, $P(a)$ holds.

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.

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

For any property P,
If

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

then

for all $e \in$ **AExp**, $P(e)$ 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 $e \in$ **AExp**, $P(\langle x := n; e, \sigma\rangle \longrightarrow \langle e, \sigma[x \mapsto n]\rangle)$ holds.
- LADD: For all expressions $e_1, e_2, e_1' \in$ **AExp** 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 $e_2, e_2' \in$ **AExp** 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 $e_1, e_2, e_1' \in$ **AExp** 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 $e_2, e_2' \in$ **AExp** 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 $e_1, e_2, e_1' \in$ **AExp** 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.

1.2 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.]

1.3 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.

1.4 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 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.

□

2 Large-step semantics

So far we have defined the small step evaluation relation $\longrightarrow \subseteq \textbf{Config} \times \textbf{Config}$ for our simple language of arithmetic expressions, and used its transitive and reflexive closure $\longrightarrow^*$ to describe the execution of multiple steps of evaluation. In particular, if $\langle e, \sigma\rangle$ is some start configuration, and $\langle n, \sigma'\rangle$ is a final configuration, the evaluation $\langle e, \sigma\rangle \longrightarrow^* \langle n, \sigma'\rangle$ shows that by executing expression e starting with the store σ, we get the result n, and the final store σ'.

Large-step semantics is an alternative way to specify the operational semantics of a language. Large-step semantics directly give the final result.

We'll use the same configurations as before, but define a large step evaluation relation:

$$\Downarrow \subseteq \textbf{Config} \times \textbf{FinalConfig}$$

where

$$\textbf{Config} = \textbf{Exp} \times \textbf{Store}$$
$$\text{and } \textbf{Final Config} = \textbf{Int} \times \textbf{Store} \subseteq \textbf{Config}.$$

We write $\langle e, \sigma\rangle \Downarrow \langle n, \sigma'\rangle$ to mean that $(\langle e, \sigma\rangle, \langle n, \sigma'\rangle) \in \Downarrow$. In other words, configuration $\langle e, \sigma\rangle$ evaluates in one big step directly to final configuration $\langle n, \sigma'\rangle$. In general, the big step semantics takes a configuration to an "answer". For our language of arithmetic expressions, "answers" are a subset of configurations, but this is not always true in general.

The large step semantics boils down to defining the relation $\Downarrow$. We use inference rules to inductively define the relation $\Downarrow$, similar to how we specified the small-step operational semantics $\longrightarrow$.

$$\textsc{Int}_{\textsc{Lrg}}\ \frac{}{\langle n, \sigma\rangle \Downarrow \langle n, \sigma\rangle} \qquad \textsc{Var}_{\textsc{Lrg}}\ \frac{}{\langle x, \sigma\rangle \Downarrow \langle n, \sigma\rangle} \text{ where } \sigma(x) = n$$

$$\textsc{Add}_{\textsc{Lrg}}\ \frac{\langle e_1, \sigma\rangle \Downarrow \langle n_1, \sigma''\rangle \qquad \langle e_2, \sigma''\rangle \Downarrow \langle n_2, \sigma'\rangle}{\langle e_1 + e_2, \sigma\rangle \Downarrow \langle n, \sigma'\rangle} \text{ where } n \text{ is the sum of } n_1 \text{ and } n_2$$

$$\textsc{Mul}_{\textsc{Lrg}}\ \frac{\langle e_1, \sigma\rangle \Downarrow \langle n_1, \sigma''\rangle \qquad \langle e_2, \sigma''\rangle \Downarrow \langle n_2, \sigma'\rangle}{\langle e_1 \times e_2, \sigma\rangle \Downarrow \langle n, \sigma'\rangle} \text{ where } n \text{ is the product of } n_1 \text{ and } n_2$$

$$\textsc{Asg}_{\textsc{Lrg}}\ \frac{\langle e_1, \sigma\rangle \Downarrow \langle n_1, \sigma''\rangle \qquad \langle e_2, \sigma''[x \mapsto n_1]\rangle \Downarrow \langle n_2, \sigma'\rangle}{\langle x := e_1; e_2, \sigma\rangle \Downarrow \langle n_2, \sigma'\rangle}$$