lec3: Inductive proofs

Jana Dunfield

January 21, 2022

1 Examples

Here are the "big-step" rules from lec2:

$\boxed{e \Downarrow v}$ expression e evaluates to value v (big-step)

$$\frac{}{n \Downarrow n}\ \text{eval-const} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{(+\ e_1\ e_2) \Downarrow n_1 + n_2}\ \text{eval-add}$$

Here are some *alleged* derivations of this judgment. I say "alleged" because some of them cannot be derived according to the two rules above. Before moving to the next page, try to figure out which of them are genuine, complete derivations and which have issues: the derivation might be incomplete, it might not correspond to the given rules, and so forth.

(a) $\dfrac{}{3 \Downarrow 3}\ \text{eval-const}$

(b) $\dfrac{}{3 \Downarrow 2 + 1}\ \text{eval-const}$

(c) $\dfrac{}{3 + 1 \Downarrow 2 + 1}\ \text{eval-const}$

(d) $\dfrac{2 + 1 = 3}{3 \Downarrow 2 + 1}\ \text{eval-const}$

(e) $\dfrac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{(+\ e_1\ e_2) \Downarrow n_1 + n_2}\ \text{eval-add}$

(f) $$\dfrac{\dfrac{}{1 \Downarrow 1}\ \text{eval-const} \qquad \dfrac{}{2 \Downarrow 2}\ \text{eval-const}}{(+\ 1\ 2) \Downarrow 3}\ \text{eval-add}$$

(g) $\dfrac{1 \Downarrow 1 \qquad 2 \Downarrow 2}{(+\ 1\ 2) \Downarrow 3}\ \text{eval-add}$

(h) $\dfrac{}{(+\ 1\ 2) \Downarrow 3}\ \text{eval-add}$

1.1 Impostors revealed

Three of the derivations on the previous page are genuine and complete derivations; one is genuine but incomplete; four are incorrect.

(a) $\dfrac{}{3 \Downarrow 3}$ eval-const

Correct derivation, obtained by replacing n with 3.

(b) $\dfrac{}{3 \Downarrow 2+1}$ eval-const

Correct. Writing 3 as $2+1$ has no evident purpose, but since $2+1=3$, it is correct.

(c) $\dfrac{}{3+1 \Downarrow 2+1}$ eval-const

Incorrect: the first n was replaced with $3+1$, which is 4, but the second n was replaced with $2+1$, which is 3, and $3 \neq 4$.

(d) $\dfrac{2+1=3}{3 \Downarrow 2+1}$ eval-const

Incorrect: eval-const does not have a premise. (This is the "least incorrect" derivation here, because $2+1=3$ is true. Still, it does not belong.)

(e) $\dfrac{e_1 \Downarrow n_1 \qquad e_1 \Downarrow n_2}{(+\ e_1\ e_2) \Downarrow n_1+n_2}$ eval-add

This is just a copy of the rule. It does not say what e_1, n_1, e_2 and n_2 are, and it does not show how the premises were derived, so it is not a derivation.

(f) $\dfrac{\dfrac{}{1 \Downarrow 1}\text{ eval-const} \qquad \dfrac{}{2 \Downarrow 2}\text{ eval-const}}{(+\ 1\ 2) \Downarrow 3}$ eval-add

Correct.

(g) $\dfrac{1 \Downarrow 1 \qquad 2 \Downarrow 2}{(+\ 1\ 2) \Downarrow 3}$ eval-add

Incomplete; otherwise correct. For a derivation to be complete, we need to know how the premises were derived, and this one does not show that.

(h) $\dfrac{}{(+\ 1\ 2) \Downarrow 3}$ eval-add

Incorrect: eval-add has two premises but there are zero here. This is "more incorrect" than (g): with (g) it is clear there is something missing, because the premises are not justified. Here, the premises are omitted entirely, making the derivation seem complete until we compare it to the eval-add rule.

2 Induction ideology

Repeated from lec2:

$\boxed{e \Downarrow v}$ expression e evaluates to value v (big-step)

$$\frac{}{n \Downarrow n}\ \text{eval-const} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{(+\ e_1\ e_2) \Downarrow n_1 + n_2}\ \text{eval-add}$$

Conjecture 1.
For all e, v_1, v_2
such that $e \Downarrow v_1$ *and* $e \Downarrow v_2$, *it is the case that* $v_1 = v_2$.

Proof. By structural induction on e. [I chose e because I know it will work, but some intuition helps: The expression e is the program. The big-step rules decompose e, recursively evaluating smaller expressions. On the other hand, the values v_1 and v_2 are integers, which—being single tokens—are too small to have structure for our purposes. There's no obviously useful induction measure on integers anyway. . .]

Induction hypothesis:
For all e', v_1', v_2'
such that $e' \prec e$
and $e' \Downarrow v_1'$ and $e' \Downarrow v_2'$, it is the case that $v_1' = v_2'$.

[Notice that, once we decide what we're inducting on and how we're inducting on it, the IH is obtained *systematically* from the statement of the conjecture.]

[To be continued. . .] □

You may have been exposed to explanations of induction that talk about a "base case" and an "induction step". I don't like such explanations because they combine two techniques that should be distinguished:

- reasoning by cases (case analysis), and
- induction.

Case analysis is a common technique that's often used in non-inductive proofs. Using $|\cdots|$ to mean absolute value, suppose we want to prove

For all real numbers x, the equation $|-x| = |x|$ holds.

We consider two possible cases: $x < 0$, and $x \geq 0$:

- First case: $x < 0$, so $-x > 0$; by the definition of absolute value, $|-x| = -x$.
  Since $x < 0$, we have $|x| = -x$.
  Therefore $|-x| = |x|$.
- Second case: $x \geq 0$, so by the definition of absolute value, $|x| = x$.
  Since x is positive, $-x$ is negative, so $|-x| = -(-x) = x$, which is equal to $|x|$.

Since we proved $|{-x}| = |x|$ for each possible case, we have proved $|{-x}| = |x|$. (CISC 204 note: This is like using the $\vee$e rule.)

We can use case analysis on mathematical objects that are defined by induction *without* using induction in the proof. For example, to prove the statement

For all expressions e, the last character of the expression is either a decimal digit, or a right parenthesis.

we don't need to use induction, only case analysis: Either e was constructed (1) using the n alternative in the grammar, or (2) using the `(+ `e_1 e_2`)` alternative. In case (1), $e = n$ where n is an integer, so its last character must be a decimal digit. In case (2), $e =$ `(+ `e_1 e_2`)`, and the last character of `(+ `e_1 e_2`)` is a right parenthesis.

Explanations of inductive proof that insist on a base case and an induction step are treating these two steps as if they are *necessarily* connected. This leads to wrong proofs of the variety "all horses have the same colour"—such proofs *feel* valid because they seem to follow the "[ ] base case; [ ] induction step" checklist, hiding a case analysis that is pretty clearly wrong if you can inspect it in isolation.[1]

(Syllogisms in ancient logic have a somewhat similar issue: "If something is smoky, it is on fire. The hillside is smoky. Therefore, the hill is on fire." From a modern perspective of mathematical logic, this syllogism combines the universal quantification "for all x…" with the implication "if x is smoky, then it is on fire".[2])

Instead of forcing the format of base case + induction step, it's better to think of induction as a proof technique that gives you an induction hypothesis (IH), which you can use wherever you want as long as you do it correctly, e.g. by making sure that $e' \prec e$. You will find that many proofs look like "By induction on $\cdots$. Consider the following possible cases: ...." where some of the cases don't use the IH (and can be called base cases) and some cases do (and can be called inductive cases). Strictly speaking, the IH *is* available in the "base cases", it just isn't useful in those cases. When we prove our conjecture about big-step determinacy, there will be a case for $e = n$ that doesn't use the IH. Since $e = n$, the IH

"for all e', ... such that $e' \prec e$ ..."

is

"for all e', ... such that $e' \prec n$ ..."

[1] One version of that proof: Consider a herd of horses. We proceed by mathematical induction on the number of horses in the herd:

- **Base case,** $n = 1$**:** If there is one horse, then it has the same colour as itself.
- **Inductive step,** $n > 1$**:** Assume the result holds for n, we show it for $n + 1$. We have $n + 1$ horses, which we can number $1, \ldots, n, n + 1$.
  The set S of horses obtained by omitting the last ($n + 1$th) horse has n horses, so by induction, all horses in that set have the same colour.
  The set T of horses obtained by omitting the *first* horse also has n horses, so by induction, all horses in that set have the same colour.

1	2	...	n	$n+1$
in S only	in both S and T	in both S and T	in both S and T	in T only

  Therefore, all $n + 1$ horses have the same colour.

[2] This is a dubious rendition of a traditional syllogism in India: I didn't include an example, as "If something is smoky, it is on fire, like a kitchen." Whether an example was required in such syllogisms was debated among logicians.

But there are no subexpressions of n, because n is a token. Thus, the IH is equivalent to "for all e', ... such that FALSE!!...", or "for all $e' \in \{\}$". A statement about all members of the empty set is useless. The IH in a base case is talking about animals (e') that are smaller than the animal you have (e), but if $e = n$ it's already the smallest possible animal.

3 Actual proofs

Conjecture 2.
For all e, v_1, v_2
such that $e \Downarrow v_1$ *and* $e \Downarrow v_2$, *it is the case that* $v_1 = v_2$.

Proof. By structural induction on e.

Induction hypothesis:
For all e', v_1', v_2'
such that $e' \prec e$
and $e' \Downarrow v_1'$ and $e' \Downarrow v_2'$, it is the case that $v_1' = v_2'$.

Consider cases of e.

- Case: $e = n$.

  [Inspecting our big-step rules, we notice that if $e = n$ then the only rule that can derive $e \Downarrow v_1$ is eval-const. So the concluding rule in the derivation that derives $e \Downarrow v_1$ must be eval-const. Therefore, v_1 must be n. We condense this reasoning as follows.]

$e \Downarrow v_1$	Given [assumption]
$e = n$	Given [assumption within this case]
$n \Downarrow v_1$	By above equation
$v_1 = n$	By inversion on rule eval-const

  [We proceed similarly for $e \Downarrow v_2$.]

$e \Downarrow v_2$	Given [assumption]
$e = n$	Given [assumption within this case]
$n \Downarrow v_2$	By above equation
$v_2 = n$	By inversion on rule eval-const

  [We're basically done with this case; just need to say explicitly that $v_1 = v_2$.]

☞	$v_1 = v_2$	By above equations

- Case: $e =$ `(+` e_1 e_2`)`.

  [Inversion helps us here, too. I use "ditto marks", 〃, when the justification for a step is the same as the justification on the line above: Inverting eval-add tells us about v_1 *and* gives us two premises.]

$e \Downarrow v_1$	Given [assumption]
$e =$ `(+` e_1 e_2`)`	Given [assumption within this case]
`(+` e_1 e_2`)` $\Downarrow v_1$	By above equation
$v_1 = n_{11} + n_{12}$	By inversion on rule eval-add
$e_1 \Downarrow n_{11}$	〃
$e_2 \Downarrow n_{12}$	〃

  [I originally wrote n_1 and n_2 instead of n_{11} and n_{12}, but saw a naming collision with the next part of the proof.]

[We proceed similarly for $e \Downarrow v_2$. If I had named n_{21} and n_{22} the same as n_{11} and n_{12}, I would have used the same names for things that actually *are* the same, but I haven't proved it yet! Inversion only tells me that those integers exist and their sum is related to v_2; inversion doesn't prove that $n_{11} = n_{21}$ and $n_{12} = n_{22}$.]

$e \Downarrow v_2$	Given [assumption]
$e = (+\ e_1\ e_2)$	Given [assumption within this case]
$(+\ e_1\ e_2) \Downarrow v_2$	By above equation
$v_2 = n_{21} + n_{22}$	By inversion on rule eval-add
$e_1 \Downarrow n_{21}$	〃
$e_2 \Downarrow n_{22}$	〃

[Inversion is needed so frequently that I usually do it even if I'm not quite sure that it will help. This is a pretty effective strategy. The next part of the strategy is to use the IH everywhere we can!]

[Where can we use the IH? What results do we already know that are "shaped like" the IH?]

$e_1 \Downarrow n_{11}$	Above
$e_1 \Downarrow n_{21}$	Above

[We need to check that e_1 is a smaller expression than e. This step is often implicit. But you should *think* it.]

$e = (+\ e_1\ e_2)$	Given [assumption within this case]
$e_1 \prec e$	By above line
$e_1 \Downarrow n_{11}$	Above
$e_1 \Downarrow n_{21}$	Above
$n_{11} = n_{21}$	By IH [with e_1 as e' and n_{11} as v_1' and n_{21} as v_2']

[That gave us something interesting. Let's do the same thing with e_2. When two things feel symmetric, apply the same techniques to each of them. I won't be quite as detailed this time.]

$e_2 \prec e$	By $e = (+\ e_1\ e_2)$
$e_2 \Downarrow n_{12}$	Above
$e_2 \Downarrow n_{22}$	Above
$n_{12} = n_{22}$	By IH [with e_2 as e' and n_{12} as v_1' and n_{22} as v_2']

[Wait, what are we trying to prove? That $v_1 = v_2$. Don't we know something about those?]

☞ $v_1 = v_2$???

[now:]

$n_{11} = n_{21}$	Above
$n_{12} = n_{22}$	Above
$n_{11} + n_{12} = n_{11} + n_{12}$	Identical things are equal
$n_{11} + n_{12} = n_{21} + n_{22}$	By above equations
$v_1 = n_{11} + n_{12}$	Above
$v_2 = n_{21} + n_{22}$	Above
☞ $v_1 = v_2$	By above equations

□

3.1 Induction on derivations

Besides structural induction on expressions, we can do structural induction on *derivations*.

The statement below is identical to Conjecture 2, except that I have named the derivations of $e \Downarrow v_1$ and $e \Downarrow v_2$. Naming the derivations is useful, because I want to say what I am inducting on, and (when I use the IH) why I am allowed to do so.

Conjecture 3.
For all e, v_1, v_2
such that $\mathcal{D}_1$ *derives* $e \Downarrow v_1$ *and* $\mathcal{D}_2$ *derives* $e \Downarrow v_2$, *it is the case that* $v_1 = v_2$.

Proof. By structural induction on $\mathcal{D}_1$ (the derivation of $e \Downarrow v_1$).

Consider cases of the rule concluding $\mathcal{D}_1$.

- Case: The rule concluding $\mathcal{D}_1$ is eval-const.

  $e = n$ By inversion on rule eval-const
  $v_1 = n$ ″

  Since $e = n$, the concluding rule of the *other* derivation $\mathcal{D}_2$ must be eval-const. By inversion on eval-const, $v_2 = n$.

  Therefore $v_1 = v_2$.

- Case: The rule concluding $\mathcal{D}_1$ is eval-add.

  $e = (+\ e_1\ e_2)$ By inversion on rule eval-add
  $e_1 \Downarrow n_{11}$ ″
  $e_2 \Downarrow n_{12}$ ″

  Since $e = (+\ e_1\ e_2)$, the concluding rule of the *other* derivation $\mathcal{D}_2$ must be eval-add. By inversion on eval-add:

  $e_1 \Downarrow n_{21}$
  $e_2 \Downarrow n_{22}$

  [The rest of the proof is similar to the previous proof—but our justification for being able to use the IH is different. Instead of showing that e_1 is a subexpression of e, we show that the derivation of $e_1 \Downarrow n_{11}$ is a subderivation of the derivation of $e \Downarrow v_1$.]

  ...

□

Exercise 1. Try to complete the proof above.

This proof and the earlier one (Conjecture 2) should seem fairly similar. However, some of the uses of inversion here are subtly different: rather than exhaustively going through all rules and concluding that only eval-const (for example) could have been used, we know that eval-const was used because we chose to do case analysis on the concluding rule. The high-level idea behind these two kinds of inversion is the same:

- We have a derivation. We may know something about it, but not everything.

- We look at rules and get more information.

In our earlier use of inversion, we had $e \Downarrow v_1$ and knew that $e = n$, because we were doing case analysis on e and were inside the $e = n$ case. We observed that only one rule, eval-const, can derive $n \Downarrow v_1$—and the conclusion of eval-const says $n \Downarrow n$, so $v_1 = n$. So we started from "the left-hand side of the conclusion of this derivation is some integer n", but we had to go through both rules to determine that eval-const was used, and then to use that fact.

In this new use of inversion, we have $e \Downarrow v_1$ and know that the *concluding rule* is eval-const, because we are doing case analysis on the concluding rule, and are inside the eval-const case. Since we *know* (within the case) that the concluding rule is eval-const, we notice that the conclusion of eval-const says $n \Downarrow n$... From this point, the use of inversion is the same as the earlier one. Here we started from "the concluding rule is eval-const", so we did *not* have to inspect all the rules to make sure that only eval-const was possible.

Hopefully, that explains how the uses of inversion are different. On a more general level, though, why do the proofs "feel" similar? I think the reason is that the big-step evaluation rules are "in sync" with the inductive definition of expressions: the first part of the inductive definition corresponds to the first rule, eval-const, while the second part of the inductive definition corresponds to the second rule, eval-add.

Rules of this kind are often called *syntax-directed*: each syntactic form (each part of the inductive definition of syntax) has exactly one rule, and each rule applies to exactly one syntactic form. Thus, when we prove things about such rules, the "eval-add case" (when case-analyzing the concluding rule) will look similar to the "`(+ ···)` case". However, not all of the rules we'll see in the course are syntax-directed. I have learned to reflexively case-analyze on the concluding rule: for syntax-directed rules it seems to work about as well as case-analyzing syntax, and for rules that are not syntax-directed, it works better.

$\boxed{e \mapsto^* e'}$ expression e steps zero or more times to e'

$$\frac{}{e \mapsto^* e}\ \text{steps-zero} \qquad \frac{e \mapsto e_1 \qquad e_1 \mapsto^* e_2}{e \mapsto^* e_2}\ \text{steps-multi}$$

$\boxed{e \mapsto e'}$ expression e steps to e'

$$\frac{}{(+\ n_1\ n_2) \ \mapsto\ n_1{+}n_2}\ \text{step-add}$$

$$\frac{e_1 \ \mapsto\ e_1'}{(+\ e_1\ e_2) \ \mapsto\ (+\ e_1'\ e_2)}\ \text{step-add-left} \qquad \frac{e_2 \ \mapsto\ e_2'}{(+\ e_1\ e_2) \ \mapsto\ (+\ e_1\ e_2')}\ \text{step-add-right}$$

Conjecture 4.
If $e \Downarrow v_B$ *and* $e \mapsto^* v_S$
then $v_B = v_S$.

I'm a little worried about this one.