Worksheet on Induction

Theo Wang

January 31, 2026

The Semantics course has two foci: specifying/designing programming languages using the tools of operational semantics and type systems, and stating/proving properties about them. This sheet focuses on the latter. The aim is to acquaint you with the general notion of *inductively defined data*, as well as the fundamental tool to reason about such data, *induction.* We will see how the syntax, the operational semantics and the type system of a programming language arise as special cases of this general notion. We will also see how the different types of induction (mathematical, structural, rule) arise as special cases of this notion. Finally, we will do a few subtle cases of induction to show that it is not all that trivial.

Adapted from Michael Lee's supervision questions.

1 Inductively Defined Sets

In this section I will deliberately be slightly informal to ease understanding.

A set X is defined inductively if its elements are exactly the ones generated by a set of rules R, where each rule can take two forms:

- Axioms[1]: $\dfrac{}{a \in X}$ (axiom), which says that a must be in X;
- Rules: $\dfrac{x_1 \in X \quad x_2 \in X \quad \ldots \quad x_n \in X}{y \in X}$ (rule), which says that if $x_1 \ldots x_n \in X$, then we can form a new element y, which still is in X.

In other words, X is defined[2] as follows:

$$X \triangleq \{y \mid \text{there is a complete derivation for } y \text{ using rules in } R\}$$

For example, the set of natural numbers $\mathbb{N}$ can be seen as generated by the following two rules:

$$\frac{}{0 \in \mathbb{N}} \qquad\qquad \frac{x \in \mathbb{N}}{s(x) \in \mathbb{N}}$$

Exercise 1. Informally explain why this is the case.

In the future, we will keep the set we are defining with the rules implicit and simply write

$$\frac{}{y}\ \text{(axiom)} \qquad\qquad \frac{x_1 \quad x_2 \quad \ldots \quad x_n}{y}\ \text{(rule)}$$

Exercise 2. This question concerns the List data type.

(i) Give an OCaml data type for lists of natural numbers. You may use the type `nat` for natural numbers.

[1]Note that the definition for axioms is redundant because it is a special case of rules when $n = 0$.

[2]More formally, X is defined as the *smallest set closed under these rules.*

(ii) Define the set of lists of natural numbers inductively.

Exercise 3. Define the set of all (not necessarily well-typed) terms of L1 as an inductively defined set.

Exercise 4. Let X be a set and let R be a binary relation over X, i.e., $R \subseteq X \times X$. Define the reflexive transitive closure $R^* \subseteq X \times X$ of R inductively.

Exercise 5. What set do the typing rules of L1 implicitly define? What set do the transition rules of L1 implicitly define?

2 Induction Principles

Every inductively defined set naturally gives rise to a way of proving properties about them, known as an induction principle. Consider the set of natural numbers $\mathbb{N}$ defined inductively as above. Then, to prove that a property Φ holds for all natural numbers (i.e. that $\forall n \in \mathbb{N}.\ \Phi(n)$ holds), it suffices to prove that

- $\Phi(0)$ holds, and
- if $\Phi(n)$ (the 'inductive hypothesis') holds then $\Phi(s(n))$ holds too.

Notice how each proof obligation corresponds to a rule in the inductive definition of natural numbers. It follows that if an element x is the conclusion of a complete derivation using these two rules, then $\Phi(x)$ must hold. And since $\mathbb{N}$ contains exactly those elements that can be derived from the two rules, it must be that Φ is true for all natural numbers. This proof strategy is the 'inductive principle' for natural numbers, a.k.a. the *principle of mathematical induction.* This shows that mathematical induction is just a special case of rule induction.

Exercise 6. Define the induction principle over lists of natural numbers.

Exercise 7. Define the induction principle over the syntax of L1 terms. Compare it with doing 'structural induction' over L1 terms. What can be said in general about structural induction and rule induction?

Exercise 8. Define the induction principle for (i) the type system of L1 and (ii) the operational semantics of L1.

Exercise 9. Write a general recipe of deriving the induction principle out of the inductive definition (the rules) of a set.

3 Doing Induction

For supervisions, when dealing with proof questions, I ask that you make sure you follow the following three steps:

1. Explicitly identify all inductively defined sets and thus the inductive principles available to you.
2. Choose the inductive principle to apply, and what to apply it on, by explicitly writing out the *inductive hypothesis*, Φ, which corresponds to the property we'd like to prove. **Remember that you are only allowed to use induction principles available to you; anything else has to be proven.**
3. Prove each of the cases following the chosen inductive principle.

For exams, step 1 should be done implicitly and step 2 should distinguish between mathematical, structural and rule induction, even if we now know that they are the same thing.

Exercise 10. This question was taken from the 12-13 DiscMath II exercises. The set of well-bracketed strings is inductively defined as follows:

$$\frac{}{\epsilon} \qquad\qquad \frac{x}{(x)} \qquad\qquad \frac{x \quad y}{xy}$$

(i) Define the induction principle for this set.

(ii) Use it to prove that any well-bracketed string has an equal number of left and right brackets.

Most of the times when proving semantic properties like type preservation or progress, there would be multiple inductively defined structures in the assumption. This means that one has to *choose* a structure to induct on, and use *inversion* to reason about the other one.

Exercise 11. This question concerns exactly the task of doing induction when multiple inductive structures are in the assumption.

(i) Without looking at your notes, state the inversion lemma for L1's typing rules.

(ii) Let e be an arbitrary expression such that $\Gamma \vdash e + 1 : A$; what can we say about A and about e?

(iii) Consider a version of L1 with its type system extended with a new type $\mathsf{bool2}$ and a new rule

$$\frac{}{\Gamma \vdash \mathsf{true} : \mathsf{bool2}}$$

What is a valid answer to the question: "what is the type of true"? What is the answer to the question: "Let A be an arbitrary type such that $\cdot \vdash \mathsf{true} : A$, what must A be"? Use your answer to explain why inversion is different from direct reasoning.

(iv) Without looking at your notes, prove type preservation for L1. Explicitly indicate where you use the inversion lemma.

4 When Induction Goes Wrong

This section aims to show you that induction is actually quite a subtle business and should be treated with care and rigour. Consider the following pure fragment of L2.

$$\begin{aligned} e &:= x \mid \mathbf{fn}\ x \Rightarrow e \mid e_1\ e_2 \mid \mathbf{skip} \\ A &:= 1 \mid A \to B \\ v &:= \mathbf{fn}\ x \Rightarrow e \mid \mathbf{skip} \end{aligned}$$

Its type system is as follows:

$$\frac{\Gamma(x) = A}{\Gamma \vdash x : A}\ (\text{var}) \qquad \frac{}{\Gamma \vdash \mathbf{skip} : 1}\ (\text{skip}) \qquad \frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \mathbf{fn}\ x \Rightarrow e : A \to B}\ (\text{fn}) \qquad \frac{\Gamma \vdash e : A \to B \quad \Gamma \vdash e' : A}{\Gamma \vdash e\ e' : B}\ (\text{app})$$

where $\Gamma : \mathsf{Var} \rightharpoonup \mathsf{Types}$ is a usual typing context.

And we also give a left-to-right CBV operational semantics:

$$\frac{e_1 \to e_1'}{e_1\ e_2 \to e_1'\ e_2} \qquad \frac{e_2 \to e_2'}{v\ e_2 \to v\ e_2'} \qquad \frac{}{(\mathbf{fn}\ x \Rightarrow e)\ v \to e[v/x]}$$

This fragment is also called the *simply typed lambda calculus*, or STLC. For the purpose of this section, we shall only consider this fragment.

Exercise 12. What are the three inductively defined sets in this definition, i.e., what inductive principles are available?

Now, I claim that the following lemmata and propositions hold.

Lemma 4.1 (Weakening)**.** If $\Gamma \vdash e : A$ and $x \notin \mathsf{dom}(\Gamma)$ then $\Gamma, x : B \vdash e : A$.

Lemma 4.2 (Substitution)**.** If $\Gamma \vdash e : A$ and $\Gamma, x : A \vdash e' : B$ (with $x \notin \mathsf{dom}(\Gamma)$) then $\Gamma \vdash e'[e/x] : B$.

Proposition 4.3 (Progress)**.** If $\cdot \vdash e : A$ then e is a value or $\exists e'.\ e \to e'$.

Proposition 4.4 (Type Preservation)**.** If $\cdot \vdash e : A$ and $e \to e'$ then $\cdot \vdash e' : A$.

Let's try to prove some of these. As usual, we work with all definitions up to alpha equivalence. If I ask you to prove a particular lemma or proposition, you should assume that all previous propositions hold. You may also assume inversion. Finally, make sure you follow the recipe given in the previous section.

Exercise 13. Prove Type Preservation (assuming weakening, exchange and substitution).

Exercise 14. This exercise asks you to prove Progress by inducting over the typing derivations of e.

(i) It is tempting to write the inductive hypothesis Φ as a proposition taking a closed expression and its type: $\Phi(e, A) = ...$ Why is this incorrect? *Hint: What induction principles are available to us? (exercise 12)*

(ii) Explicitly write out the right Φ.

(iii) Finish the proof.

Exercise 15. This exercise concerns the proof of the substitution lemma.

(i) Say we attempt to prove substitution using rule induction over the typing derivation of e'. State Φ and do the (fn) case. What goes wrong?

(ii) Now prove substitution instead using structural induction over the syntax of e'. What made this proof work compared to the previous attempt?

The previous exercises show you why induction can be very subtle. The next one shows you the limitations of naive induction. It is perhaps not too hard to see that intuitively, any well-typed closed term in our language necessarily terminates. Let us attempt to prove it.

Proposition 4.5 (Termination Attempt 1)**.** If $\cdot \vdash e : A$ then e terminates.

Exercise 16. Say we attempt to prove it by induction over the typing derivation. Write out the inductive hypothesis explicitly and try the (fn) case. What goes wrong?

Clearly, our inductive hypothesis needs to be stronger: it cannot only apply on cases where the expression is closed. Instead, let's define simultaneous substitution as follows:

$$\frac{}{\cdot \vDash \cdot} \qquad\qquad \frac{\rho \vDash \Gamma \qquad \vdash v : A \text{ is a value}}{\rho, v/x \vDash \Gamma, x : A}$$

Proposition 4.6 (Termination Attempt 2)**.** If $\Gamma \vdash e : A$ then for any simultaneous substitution $\rho \vDash \Gamma$, $e[\rho]$ terminates.

Exercise 17. Attempt to prove it by induction over typing again: state Φ, and do the cases for (fn) and (app). What goes wrong?

It turns out that to prove this statement we actually need an even more powerful tool called a 'logical relation', which, informally, can be seen as having a different inductive hypothesis for different types. Here, it means to have a different (stronger) notion of 'termination' for certain types, so that overall we get the weaker property of termination for all types.

Define

$$\Theta(1) \triangleq \{\cdot \vdash e : 1 \mid e \text{ terminates}\}$$
$$\Theta(A \to B) \triangleq \{\cdot \vdash e : A \to B \mid e \text{ terminates and } \forall e' \in \Theta(A).\ e\ e' \in \Theta(B)\}.$$

Exercise 18 (Optional)**.** This question concerns the logical relations based proof of termination for our language.

(i) Show that Θ is closed under reduction, that is, for any type A and $\cdot \vdash e : A$, if $e \to e'$ then $[e' \in \Theta(A) \iff e \in \Theta(A)]$. You may assume determinacy.

(ii) Naturally extend the definition of Θ to contexts Γ, such that $\Theta(\Gamma) \subseteq \{\rho \mid \rho \vDash \Gamma\}$.

(iii) Show that for all $\Gamma \vdash e : A$ and all simultaneous closing substitution $\rho \in \Theta(\Gamma)$, it holds that $e[\rho] \in \Theta(A)$.