Teaching Induction

Dave MacQueen

New phase of life: freedom (to live in CA, free time)

What to do?

* maintain and improve SML/NJ

* PL course (online?)
  - useful, thorough tutorial on induction

* survey of module theory

* new projects (theory/design/implementation)
  - learn Coq properly (only dabbled so far)
  - successor ML
  - teaching ML
    [embarrassments of teaching FP with Haskell]
  - new topics? (take advantage of Silicon Valley opportunities?)

Anatomy of an inductive argument

1. an inductive structure (typically terms of some sort)
   e.g. Nat = Z | S Nat

2. a logical statement of the "inductive principle" for the structure
   (a unary 2nd order predicate). E.g.

```
IP(P) =
   P(Z) &                       -- base case
   ∀x.P(x) => P(S x)            -- inductive case, with inductive hypothesis P(x)
   => ∀x.P(x)
```

3. This gives the outline of inductive proofs on the given structure:

```
(1) Lemma: P(Z)    -- the base case
     ....
(2) Lemma: ∀y.P(y) => P(S y)    -- the inductive case
     ....
(3) Theorem: ∀x.P(x)
    by (1), (2), IP(P)
```

This is commonly abbreviated to the following scheme:

```
For any x, show that P(x), by induction on x:
   base case x = Z:
      .... P(Z),
      hence P(x)
   inductive case x = S y:
      assume IH: P(y)
      ... P(S y),   (invoking IH somewhere)
      hence P(x)
   [hence ∀x.P(x) by IP(P)]
```

But often the explicit statement of Induction Hypotheses is omitted.

```
Show that P(x) by induction on x.
case x = Z:
   .... P(Z)
case x = S y:
   ....
   by induction, P(y)
   .... P(S y)
```

Sometimes don't even make the inductive structure explicit,
and we don't have the explicit constructors (like Z, S).

Example: Substitution Lemma

Substitution Lemma from Pierce, Types and Programming Languages.

9.3.8. Lemma [Preservation of types under substitution]:
If Γ, x:S ⊦ t:T and Γ ⊦ s:S, then Γ ⊦ [x → s]t : T.

Proof: By induction on a derivation of the statement Γ, x:S ⊦ t:T. For a given derivation, we proceed by cases on the final typing rule used in the proof. The most interesting cases are the ones for variables and abstractions.

...

Case T-Abs: t = λy:T2.t1
T = T1 → T2
Γ, x:S, y:T2 ⊦ t1:T1

By convention 5.3.4, we may assume x ≠ y and y ∉ FV(s). Using permutation on the given subderivation, we obtain Γ, y:T2, x:S ⊦ t1:T1. Using weakening on the other given derivation (Γ ⊦ s:S), we obtain Γ, y:T2 ⊦ s:S. Now, by the induction hypothesis [?], Γ, y:T2 ⊦ [x → s]t1: T1. By T-Abs, Γ ⊦ λy:T2. [x → s]t1 : T1 → T2. But this is precisely the needed result, since, by the definition of substitution, [x → s]t = λy:T2. [x → s]t1.

Problem: Students can’t “formalize” this proof. They can’t write down the inductive hypothesis that was invoked, and they don’t know what the relevant Induction Principle is.

Abstract syntax of SAEL (Simple Arithmetic Expressions with Let)

```
v ::= x, y, z, ...    (alphanumeric variables)
n ::= 0, 1, 2, ...    (natural numbers)

bop ::= Plus, Times, ...     (primitive binary operators)

e ::= Num(n)                 -- number constants, as before
    | Var(v)                 -- variable expressions
    | Bapp(bop, e, e)
    | Let(v, e, e)
```

Rules for relative closure judgements: $\Gamma \vdash e \text{ ok}$
("e is closed relative to variable set Γ")

Rules:

(1)
$$\frac{}{\Gamma \vdash \mathrm{Num}(n) \text{ ok}}$$

(2)
$$\frac{(x \in \Gamma)}{\Gamma \vdash x \text{ ok}}$$

(3)
$$\frac{\Gamma \vdash e1 \text{ ok} \qquad \Gamma \vdash e2 \text{ ok}}{\Gamma \vdash \mathrm{Bapp}(bop, e1, e2) \text{ ok}}$$

(4)
$$\frac{\Gamma \vdash e1 \text{ ok} \qquad \Gamma \cup \{x\} \vdash e2 \text{ ok}}{\Gamma \vdash \text{let } x = e1 \text{ in } e2 \ \text{ ok}}$$

Lemma 4.4 [Substitution]: $\Gamma \vdash e1\ ok \;\wedge\; \Gamma \cup \{x\} \vdash e2\ ok \;\wedge\; x \notin \Gamma$

$\Rightarrow \;\Gamma \vdash [e1/x]e2\ ok.$

Case: $\Gamma \cup \{x\} \vdash e2$ ok by Rule (4). Then e2 is of the form

```
e2 = let y = e3 in e4
...
```

But how do we state the Induction Hypothesis in this case? If we can't talk about derivations explicitly, we end up trying something like:

(1) $\forall e1. \forall \Gamma 1.\ \Gamma 1 \vdash e1\ ok \Rightarrow$

$$(\forall e2. \forall \Gamma 2. \forall x \in Var.\ x \notin \Gamma 1 \;\wedge\; \Gamma 1 \subseteq \Gamma 2 \;\wedge\; \Gamma 2 \cup \{x\} \vdash e2\ ok \Rightarrow \Gamma 2 \vdash [e1/x]e2\ ok)$$

to deal with the variation in contexts (Γ vs $\Gamma \cup \{x\}$).

[See Lecture 7 for detailed discussion.]

Make Derivations Explicit!

First let us be precise about the structure of derivations and the definitions of context and subject of a derivation.

Derivations d in Der[ok] are inductively constructed using rule-constructors corresponding to the four rules (1) through (4).

```
d  ::= OK1(Γ, n)
     | OK2(Γ, z)
     | OK3(bop,d1,d2)   -- d1 and d2 are the derivations for e1 and e2
     | OK4(z,d1,d2)     -- d1 is derivation for definiens, d2 for body
```

These rule constructors are not "free" constructors, because a valid construction of a derivation has to satisfy some preconditions, specified as follows:

```
OK2(Γ,z) :         z ∈ Γ
OK3(bop,d1,d2) :   context(d1) = context(d2)
OK4(z,d1,d2) :     context(d2) = context(d1) ∪ {z}
```

Next we define the subject and context functions for derivations as follows:

```
S1: subject(OK1(Γ, n)) = Num n
S2: subject(OK2(Γ, z)) = Var z
S3: subject(OK3(bop,d1,d2)) = Bapp(bop,subject(d1),subject(d2))
S4: subject(OK4(z,d1,d2)) = Let(z,subject(d1),subject(d2))

C1: context(OK1(Γ, n)) = Γ
C2: context(OK2(Γ, z)) = Γ
C3: context(OK3(bop,d1,d2)) = context(d1)
C4: context(OK4(x,d1,d2)) = context(d1)
```

The Induction Principle for Derivations

$IP_{ok}(P)$: (P a predicate over derivations)

```
∀Γ.∀n.P(OK1(Γ,n)) ∧
∀Γ.∀z. x ∈ Γ => P(OK2(Γ,z)) ∧
∀bop.∀d1.∀d2.(P(d1) ∧ P(d2) ^ context(d1) = context(d2)
                => P(OK3(bop,d1,d2)))  ∧
∀z.∀d1.∀d2.(P(d1) ∧ P(d2) ∧ context(d2) = context(d1)∪{z}
                => P(OK4(z,d1,d2)))
=> ∀d.P(d)
```

where it is understood that Γ ranges over variable sets, n over Nat, z over variables, bop over primitive operators, and d,d1,d2 over Der[ok].

Lemma 4.4 [Substitution]: $\Gamma \vdash e1$ ok $\wedge$ $\Gamma\cup\{x\} \vdash e2$ ok $\wedge$ $x \notin \Gamma$ => $\Gamma \vdash [e1/x]e2$ ok. Proof in terms of derivations:

We will assume we are given a derivation d of a judgement

$\Gamma \vdash e1$ ok

(i.e., Γ = context(d) and e1 = subject(d)).

We also assume a variable $x \notin \Gamma$ is given. Then we will prove that $\forall d\in$Der[ok].P(d), where P is the property:

P(d) == $\Gamma \subseteq$ context(d) => context(d)\{x} $\vdash$ [e/x]subject(d) ok

The proof is by induction on the the structure of a derivation d $\in$ Der[ok] as defined above in terms of derivation constructors OK1, OK2, OK3, and OK4.

Ind Case: d = OK4(y,d3,d4).

Let e3 = subject(d3) and e4 = subject(d4) and Γ2 = contect(d3). Then it must be the case that Γ3 = context(d4) = Γ2∪{y} by the OK4 constraint. We then have e2 = Let(y,e3, e4). We can assume that the local let-bound variable y is chosen so that y ≠ x and y ∉ FV(e3) (by α-converting, if necessary, to make it so). We can also assume that Γ1 ⊆ Γ2, since otherwise P(d) is true vacuously. Since Γ1 ⊆ Γ2, we also have Γ1 ⊆ Γ3.

IH1: P(d3) == Γ1 ⊆ Γ2 => Γ2\{x} ⊦ [e1/x]e3 ok
IH2: P(d2) == Γ1 ⊆ Γ3 => Γ3\{x} ⊦ [e1/x]e4 ok

By IH1 and the fact that Γ1 ⊆ Γ2, we have

Γ2\{x} ⊦ [e1/x]e3 ok (1)

and by IH2 and Γ1 ⊆ Γ3 we have

Γ3\{x} ⊦ [e1/x]e4 ok (2)

...

Another Problem: students have trouble writing “prose” proofs. They often confuse the logic.

Possible solution: teach them to write precise proofs in “Lamport” style.

See example in Chapter 7.

Question: Could we support this proof style with a tool for editing and checking proofs?

Why not use Coq?

* If students can’t do conventional proofs using classical logic, they probably won’t find Coq proofs easier.

* Coq is indirect. I want students to write proofs directly and concretely.

* There wasn’t enough time to learn Coq in addition to the primary PL material in a 10 week course.

Can these proofs be automatically checked?

Prospect of creating an online course.

What is the best online format for the text?
- HTML (MathType, blahTeX?)
- PDF (LaTeX)
  PDF -> HTML?
  - iBook

PL Course web site (lecture notes, exercises, etc.)

http://www.classes.cs.uchicago.edu/archive/2011/fall/22100-1/index.html

Draft Tutorial on Induction (in development)

http://www.classes.cs.uchicago.edu/archive/2011/fall/22100-1/handouts/induction.pdf