15–212: Fundamental Structures of Computer Science II

Some Notes on Induction

Frank Pfenning

Draft of September 8, 1997

These notes provide a brief introduction to induction for proving properties of ML programs. We assume that the reader is already familiar with ML and the notes on evaluation for pure ML programs.

Recall that we write $\overline{n}$ for the representation of the integer n in ML (and, in particular, $\overline{0} = \texttt{0}$, $\overline{1} = \texttt{1}$, etc.). We write $e \stackrel{k}{\Longrightarrow} e'$ for a computation of k steps, $e \Longrightarrow e'$ for a computation of any number of steps (including 0), $e \hookrightarrow v$ for a complete computation of e to a value v, and $n = m$ or $e = e'$ for mathematical equality.

1 Lemmas

When proving the correctness of ML programs, we usually establish the correctness of the functions we define in sequence. The correctness property of a function corresponds to a lemma we can use in proving the correctness of later functions.

There is another form of lemma, which is a mathematical property which is necessary for a particular step in a program correctness proof. Usually, we take mathematical properties for granted and concentrate on the program property.

As a simple example, consider the function

```
fun square (n:int) = n * n;
```

It is trivial to see that this function correctly implements the squaring function $f(n) = n^2$. In assignments we would take such properties for granted, but if we wanted to establish the correctness formally, we would proceed as follows.

Lemma 1 *For every integer n, `square` $(\overline{n}) \hookrightarrow \overline{n^2}$.*

Proof: We prove this directly, relying on the operational semantics of ML.

	$\texttt{square}\ (\overline{n})$	
$\stackrel{1}{\Longrightarrow}$	$\overline{n}\ \texttt{*}\ \overline{n}$	by evaluation rule for function application
$\stackrel{1}{\Longrightarrow}$	$\overline{n \times n}$	by evaluation rule for `*`
$=$	$\overline{n^2}$	by mathematical property

□

2 Mathematical Induction

The simplest form of induction over natural numbers $0, 1, \ldots$ is *mathematical induction.* Assume we have to prove a property for every natural number n. We first prove that the property holds for 0 (induction basis). Then we assume the property holds for n and establish it for $n+1$ (induction step). Basis and step together guarantee that the property holds for all natural numbers.

There are small variations of this scheme which can be justified easily and which we also call mathmatical induction. For example, the induction might start 1 if we want to prove a property of all positive integers, or there might be two base cases for 0 and 1. A distinguishing characteristic of mathematical induction is the step from n to $n+1$.

As an example, we consider a straightforward, but inefficient function to compute n^k for n an arbitrary integer, and k a natural number. We take $0^0 = 1$.

```
(* power (n:int, k:int) for k >= 0 *)
fun power (n:int, k:int) =
    if k = 0 then 1 else n * power (n, k-1);
```

Theorem 2 *power* $(\overline{n}, \overline{k}) \hookrightarrow \overline{n^k}$ *for* $k \geq 0$.

Proof: By mathematical induction on k.

Induction Basis $k = 0$.

$$\begin{array}{ll}
& \texttt{power }(\overline{n}\texttt{, 0)} \\
\stackrel{1}{\Longrightarrow} & \texttt{if 0 = 0 then 1 else } \overline{n} \texttt{ * power (}\overline{n}\texttt{, 0-1)} \\
\stackrel{1}{\Longrightarrow} & \texttt{if true then 1 else } \overline{n} \texttt{ * power (}\overline{n}\texttt{, 0-1)} \\
\stackrel{1}{\Longrightarrow} & \overline{1} \\
= & \overline{n^0}
\end{array}$$

Induction Step Assume that power $(\overline{n}, \overline{k}) \hookrightarrow \overline{n^k}$. We have to show that power $(\overline{n}, \overline{k+1}) \hookrightarrow \overline{n^{k+1}}$.

$$\begin{array}{lll}
& \texttt{power (}\overline{n}\texttt{, }\overline{k+1}\texttt{)} & \\
\stackrel{1}{\Longrightarrow} & \texttt{if } \overline{k+1} \texttt{ = 0 then 1 else } \overline{n} \texttt{ * power (}\overline{n}\texttt{, }\overline{k+1}\texttt{-1)} & \\
\stackrel{1}{\Longrightarrow} & \texttt{if false then 1 else } \overline{n} \texttt{ * power (}\overline{n}\texttt{, }\overline{k+1}\texttt{-1)} & \text{since } k \geq 0 \\
\stackrel{1}{\Longrightarrow} & \overline{n} \texttt{ * power (}\overline{n}\texttt{, }\overline{k+1}\texttt{-1)} & \\
\stackrel{1}{\Longrightarrow} & \overline{n} \texttt{ * power (}\overline{n}\texttt{, }\overline{k}\texttt{)} & \\
\Longrightarrow & \overline{n} \texttt{ * } \overline{n^k} & \text{by induction hypothesis} \\
\stackrel{1}{\Longrightarrow} & \overline{n \times n^k} & \\
= & \overline{n^{k+1}} &
\end{array}$$

□

The level of detail in a proof generally depends on the context in which the proof is carried out and the mathematical sophistication of the expected reader. In homework assignments you should feel free to omit the number of computation steps (unless we are investigating computational complexity) and combine obvious steps. Appeals to the induction hypothesis or other non-obvious steps should be justified as in the example above.

Your primary concern should be the appropriateness of the induction principle you use and the correctness of the individual steps (even if not all details are given).

3 Complete Induction

The principle of *complete induction* (also called *course-of-values induction*) formalizes a frequent pattern of reasoning. It can be justified entirely from the principle of mathematical induction, but it is useful enough to be stated as another admissible induction principle.

To prove a property by complete induction on a variable n, we first establish the induction basis for $n = 0$. Then we prove the induction step for $n \geq 0$ by assuming the property for all $n' < n$ and establishing it for n. One can think of it like mathematical induction, except that we are allowed to appeal to the induction hypothesis for any $n' < n$ and not just the immediate predecessor.

As an example we write a more efficient implementation of the `power` function which requires fewer recursive calls. It uses the `square` function above, and one new auxiliary function to test if a number is even.

```
fun even (n:int) = (n mod 2 = 0);
```

We assume without proof that `even` $(\overline{n}) \hookrightarrow$ `true` if n is even, and `even` $(\overline{n}) \hookrightarrow$ `false` if n is odd. We also depart from the first definition of `power` in that we use pattern matching to define the function.

```
(* power (n:int, k:int) where k >= 0 *)
fun power (n, 0) = 1
  | power (n, k) = (* k > 0 *)
    if even k then square (power (n, k div 2))
              else n * power (n, k-1);
```

When we compute the application of a function defined by pattern matching to an argument, we do not consider the sequential matching of the argument value against the patterns as separate steps. However, to understand a proof it may be important to note why a particular case matched or did not match the arguments.

Theorem 3 *power* $(\overline{n},\ \overline{k}) \hookrightarrow \overline{n^k}$ *for* $k \geq 0$.

Proof: By complete induction on k.

Induction Basis $k = 0$.

$$\begin{array}{lll} & \texttt{power}\ (\overline{n},\ \texttt{0}) & \\ \stackrel{1}{\Longrightarrow} & 1 & \text{by evaluation, including pattern matching} \end{array}$$

Induction Step $k > 0$.

$$\begin{array}{lll} & \texttt{power}\ (\overline{n},\ \overline{k}) & \\ \stackrel{1}{\Longrightarrow} & \texttt{if even}\ \overline{k}\ \texttt{then square (power}\ (\overline{n},\ \overline{k}\ \texttt{div 2))} & \\ & \texttt{else}\ \overline{n}\ \texttt{* power}\ (\overline{n},\ \overline{k}\texttt{-1}) & \text{since } k > 0 \end{array}$$

Now we distinguish two subcases, depending on whether k is even or odd.

Case $k = 2k'$ for some $k' < k$. Then continuing the computation above (assuming the correctness of `even`) yields

$$\begin{array}{lll}
& \texttt{power (}\overline{n}\texttt{, }\overline{k}\texttt{)} & \\
\Longrightarrow & \texttt{square (power (}\overline{n}\texttt{, }\overline{2k'}\texttt{ div 2))} & \\
\stackrel{1}{\Longrightarrow} & \texttt{square (power (}\overline{n}\texttt{, }\overline{k'}\texttt{))} & \\
\Longrightarrow & \texttt{square (}\overline{n^{k'}}\texttt{)} & \text{by induction hypothesis on } k' \\
\Longrightarrow & \overline{(n^{k'})^2} & \text{by Lemma 1} \\
= & \overline{n^{2k'}} = \overline{n^k} &
\end{array}$$

Note that $k' < k$, so we the induction hypothesis applies.

Case $k = 2k' + 1$ for some $k' < k$. then continuing the computation above (assuming the correctness of `even`) yields

$$\begin{array}{lll}
& \texttt{power (}\overline{n}\texttt{, }\overline{k}\texttt{)} & \\
\Longrightarrow & \overline{n}\texttt{ * power (}\overline{n}\texttt{, }\overline{k}\texttt{-1)} & \\
\stackrel{1}{\Longrightarrow} & \overline{n}\texttt{ * power (}\overline{n}\texttt{, }\overline{k-1}\texttt{)} & \\
\stackrel{1}{\Longrightarrow} & \overline{n}\texttt{ * }\overline{n^{k-1}} & \text{by induction hypothesis on } k-1 \\
\stackrel{1}{\Longrightarrow} & \overline{n \times n^{k-1}} = \overline{n^k} &
\end{array}$$

Here we can apply the induction hypothesis since $k - 1 < k$ and $k > 0$.

□

Before starting a proof, it is generally useful to examine the pattern of recursion of the function one wishes to prove correct. If it calls itself on the immediate predecessor only, this signals a use of mathematical induction. If it calls itself on other, small terms, a use of complete induction is more likely.

4 Generalizing the Induction Hypothesis

From the examples so far it may seem that induction is always completely straightforward. While many induction proofs that arise in program correctness are indeed simple, there is the occasional function whose correctness proof turns out to be difficult. This is often because we have to prove something more general than the final result we are aiming at. This is referred to a *generalizing the induction hypothesis* and it can be shown that there exists no general recipe for generalization which will always work. However, one can isolate certain common cases.

As an example, consider the following implementation of the factorial function.

```
(* fact' (n, a) where n >= 0 *)
fun fact' (0, a) = a
  | fact' (n, a) = fact' (n-1, n*a);

(* fact (n) where n >= 0 *)
fun fact (n) = fact' (n, 1);
```

We would like to prove that `fact` ($\overline{n}$) $\hookrightarrow \overline{n!}$ for every $n \geq 0$. This requires a lemma about `fact'`, which takes one more argument. So we have to determine which specification `fact'` satisfies. Simply checking if

`fact'` ($\overline{n}$, `1`) $\hookrightarrow \overline{n!}$

will not work, since a proof by induction fails.

Here is the problematic case. We assume the induction hypothesis:

$$\texttt{fact' }(\overline{n}\texttt{, 1}) \hookrightarrow \overline{n!}$$

and try to prove

$$\texttt{fact' }(\overline{n+1}\texttt{, 1}) \hookrightarrow \overline{(n+1)!}.$$

We start as as before:

$$\begin{array}{ll} & \texttt{fact' }(\overline{n+1}\texttt{, 1}) \\ \stackrel{1}{\Longrightarrow} & \texttt{fact' }(\overline{n+1}\texttt{-1, }\overline{n}\texttt{*1}) \\ \stackrel{1}{\Longrightarrow} & \texttt{fact' }(\overline{n}\texttt{, }\overline{n}\texttt{*1}) \\ \stackrel{1}{\Longrightarrow} & \texttt{fact' }(\overline{n}\texttt{, }\overline{n}) \end{array}$$

but now we cannot apply the induction hypothesis since the second argument in the call to `fact'` is not `1` but $\overline{n}$.

The solution is to generalize the induction property to allow any a in such a way that the desired result above follows easily. The following theorem generalizes appropriately.

Lemma 4 *`fact'` $(\overline{n}, \overline{a}) \hookrightarrow \overline{(n!) \times a}$ for any $n \geq 0$ and a.*

Proof: By mathematical induction on n.

Induction Basis $n = 0$.

$$\begin{array}{ll} & \texttt{fact' }(\texttt{0, }\overline{a}) \\ \stackrel{1}{\Longrightarrow} & \overline{a} = \overline{1 \times a} = \overline{0! \times a} \end{array}$$

Induction Step Assume the induction hypothesis for n.

$$\begin{array}{lll} & \texttt{fact' }(\overline{n+1}\texttt{, }\overline{a}) & \\ \stackrel{1}{\Longrightarrow} & \texttt{fact' }(\overline{n+1}\texttt{-1, }\overline{n+1}\texttt{*}\overline{a}) & \\ \stackrel{2}{\Longrightarrow} & \texttt{fact' }(\overline{n}\texttt{, }\overline{(n+1) \times a}) & \\ \Longrightarrow & \overline{n! \times ((n+1) \times a)} & \text{by induction hypothesis on } n \\ = & \overline{(n+1)! \times a} & \end{array}$$

Note that the lemma (and therefore the induction hypothesis) is stated *for every* a, we can apply the induction hypothesis when the second argument to `fact'` is $\overline{(n+1) \times a}$

□

The main theorem now follows directly.

Theorem 5 *`fact` $(\overline{n}) \hookrightarrow \overline{n!}$ for $n \geq 0$.*

Proof: By direct computation and Lemma 4.

$$\begin{array}{lll} & \texttt{fact }(\overline{n}) & \\ \stackrel{1}{\Longrightarrow} & \texttt{fact' }(\overline{n}\texttt{, 1}) & \\ \Longrightarrow & \overline{n \times 1!} & \text{by Lemma 4} \\ = & \overline{n} & \end{array}$$

□