CIS 301: Lecture Notes on Induction

Torben Amtoft
Department of Computing and Information Sciences
Kansas State University

November 21, 2005

These notes are written as a supplement to [1, Sect. 16.1&16.3], but can be read independently.

1 Loop Invariants: Induction in Disguise

Consider a loop of the form `while` B `do` C `od`, and assume that we know[1]

ψ is established by the preamble of the loop (1)

if with B true, ψ holds prior to C, then ψ also holds after C (2)

Then we can infer that

ψ is an invariant of the loop. (3)
That is, each time control reaches B, ψ holds.

[1] Using the framework of *CIS 301: Lecture Notes on Program Verification*, available at `http://www.cis.ksu.edu/~tamtoft/CIS301/Fall05/verification.pdf`, this amounts to (1) and (2) below being valid assertions.

$\{\psi\}$	(1)
`while` B `do`	
$\{\psi \wedge B\}$	**WhileTrue**
C	
$\{\psi\}$	(2)
`od`	

Now observe that[2]

- (1) amounts to saying that ψ holds after 0 iterations;
- (2) amounts to saying that if ψ holds after k iterations, then after $k+1$ iterations ψ also holds;
- (3) amounts to saying that after any number (≥ 0) of loop iterations, ψ holds.

We thus have the following

Principle 1 (Induction on Iterations). *Assume that for a given loop,*

- *ψ holds after 0 iterations; and*
- *if ψ holds after k iterations then, after $k+1$ iterations, ψ also holds.*

Then, for all $k \geq 0$, after k iterations, ψ will hold.

2 Induction on Natural Numbers

Principle 1 carries over to a general principle:

Principle 2 (Induction on Natural Numbers). *Assume Q is such that*

- *$Q(0)$ holds; and*
- *for all natural numbers k, if $Q(k)$ holds then also $Q(k+1)$ holds.*

Then, for all natural numbers k, $Q(k)$ holds.

This is the rule mentioned in [1, p. 454]; in Fitch format, it can be written

$$
\begin{array}{r|l}
 & Q(0) \\
 & \vdots \\
 & \forall k((\mathsf{Nat}(k) \land Q(k)) \rightarrow Q(k+1)) \\
 & \vdots \\
\triangleright & \forall k(\mathsf{Nat}(k) \rightarrow Q(k))
\end{array}
$$

[2] When we say that ψ holds after k iterations, we mean that *if* the loop iterates at least k times then ψ holds after the k'th iteration. If control exits from the loop earlier, or if one of the first k iterations gives rise to infinite computation (due to a subloop), then it is vacuously true that "ψ holds after k iterations".

Here Nat is a predicate[3] that is true on exactly the numbers $0, 1, 2, 3, 4, \ldots$.

It is instructive to note that a sentence $\forall k(\mathsf{Nat}(k) \to Q(k))$ might also be provable using a "General Conditional Proof":

$$
\begin{array}{r|l}
 & \begin{array}{|l} \boxed{k}\ \mathsf{Nat}(k) \\ \hline \quad \vdots \\ Q(k) \end{array} \\
\triangleright & \forall k(\mathsf{Nat}(k) \to Q(k))
\end{array}
$$

But such an approach is less likely to succeed, since when proving $Q(k)$ for an arbitrary k, we now *cannot* assume $Q(k-1)$. On the other hand, a general conditional proof may be the only way to establish $\forall k(P(k) \to Q(k))$ in the case where the objects satisfying P do not have any "structure".

Example 2.1 ([1, p. 454]). We want to prove that for all natural numbers n we have

$$1 + \cdots + n = \frac{n(n+1)}{2}$$

With LHS and RHS given by

$$
\begin{array}{rcl}
\mathrm{LHS}(n) & = & 1 + \cdots + n \\
\mathrm{RHS}(n) & = & \dfrac{n(n+1)}{2}
\end{array}
$$

the claim is that for all natural numbers n we have $Q(n)$ where $Q(n)$ is given by $\mathrm{LHS}(n) = \mathrm{RHS}(n)$. We prove that by induction:

<u>Basis step.</u> We must establish $Q(0)$, that is $\mathrm{LHS}(0) = \mathrm{RHS}(0)$. Since $\mathrm{LHS}(0) = 0$ (as the sum of zero numbers is 0), this follows since $\mathrm{RHS}(0) = \frac{0 \cdot 1}{2} = 0$.

<u>Inductive step.</u> We can assume that $Q(n)$ holds, that is $\mathrm{LHS}(n) = \mathrm{RHS}(n)$,

[3] Alternatively, we could define Nat to hold only on $1, 2, 3, 4, \ldots$ but not on 0. Our choice does not matter much as long as we are *consistent*, unlike the formulation of Proposition 4 in [1, p.454] which—even though 0 just earlier on the page has been declared the first natural number—implicitly assumes that the first n natural numbers are $1, \ldots, n$.

and must prove that $Q(n+1)$ holds. But this follows since

$$\begin{aligned}
\mathrm{LHS}(n+1) &= (1+\cdots+n)+(n+1) = \mathrm{LHS}(n)+(n+1) \\
&= \mathrm{RHS}(n)+(n+1) = \frac{n(n+1)}{2}+(n+1) \\
&= \frac{n(n+1)+2(n+1)}{2} = \frac{(n+1)(n+2)}{2} \\
&= \mathrm{RHS}(n+1)
\end{aligned}$$

where the third equality follows from the induction hypothesis. □

2.1 Alternative formulations

Sometimes, we need to use another starting point than zero, say m_0:

$$\begin{array}{l|l}
 & Q(m_0) \wedge \mathsf{Nat}(m_0) \\
 & \vdots \\
 & \forall k((\mathsf{Nat}(k) \wedge k \geq m_0 \wedge Q(k)) \rightarrow Q(k+1)) \\
 & \vdots \\
\triangleright & \forall k((\mathsf{Nat}(k) \wedge k \geq m_0) \rightarrow Q(k))
\end{array}$$

Below is an application of this principle, with $m_0 = 3$.

Theorem 2.2. *For a k-polygon ($k \geq 3$), the sum of its angles is given by $(k-2) \cdot 180$ degrees.* □

Proof. (Informal.) For the basis step, we must consider $k = 3$; the claim is that the sum of the angles in a triangle is 180 degrees. But this is a fact from elementary geometry.

For the inductive step, consider a $(k+1)$-polygon P. It is not hard to see that P can be split into a triangle and a k-polygon; the sum of the angles in the former is 180 degrees (cf. above), and the sum of the angles in the latter is $(k-2) \cdot 180$ degrees (by the induction hypothesis). Therefore, the sum of the angles in P is $180 + (k-2) \cdot 180 = ((k+1)-2) \cdot 180$, as desired. □

A somewhat different perspective is offered by the following rule[4]:

[4] For simplicity, we implicitly assume that all entities are natural numbers.

Principle 3 (Course-of-values induction). *In order to prove that* $Q(k)$ *holds for all natural numbers* k*, it suffices to show the following property for all* k*: given that* Q *holds for all numbers less than* k*,* Q *also holds for* k*. Expressed in Fitch notation:*

$$
\begin{array}{r|l}
 & \forall k(\forall m(m < k \to Q(m)) \to Q(k)) \\
 & \vdots \\
\rhd & \forall k Q(k)
\end{array}
$$

To justify the validity of course-of-values induction, assume (in order to arrive at a contradiction) that the conclusion does *not* hold. That is, there exists natural numbers not satisfying Q. Let k be the *least* such number. That is, for all $m < k$ we have $Q(m)$. But then our premise tells us that also $Q(k)$, yielding the desired contradiction. (A variation of this proof, where we do a proof by cases depending on whether $k = 0$ or $k > 0$, can be used to establish the validity of the original induction principle.)

Note that to establish the premise required for course-of-values induction, a proof of the following form is probably needed:

$$
\begin{array}{|l}
\begin{array}{|l}
\boxed{k}\ \forall m(m < k \to Q(m)) \\
\hline
\quad \vdots \\
Q(k)
\end{array} \\
\forall k(\forall m(m < k \to Q(m)) \to Q(k))
\end{array}
$$

We now give an example that illustrates the usefulness of course-of-values induction. We shall consider the Fibonacci numbers[5] given by

$$
\begin{array}{lll}
\mathrm{fib}(n) & = \texttt{case}\ n\ \texttt{of} & \\
 & \quad 0 & \Rightarrow 1 \\
 & \quad 1 & \Rightarrow 1 \\
 & \quad m+2 & \Rightarrow \mathrm{fib}(m+1) + \mathrm{fib}(m)
\end{array}
$$

Theorem 2.3. *If* $n + 1$ *is divisible by 3, then* $\mathrm{fib}(n)$ *is an even number, otherwise* $\mathrm{fib}(n)$ *is an odd number.* □

[5] See `http://en.wikipedia.org/wiki/Fibonacci_number` (which uses a slightly different definition) for a survey of some properties of these.

Proof. We shall employ course-of-values induction; there is thus no base step but "only" the inductive step where we have to establish that for an arbitrary n, if (with Q the property mentioned in the theorem) $Q(m)$ holds for all $m < n$ then also $Q(n)$ holds. We have to do a case analysis.

<u>Case 1: $n = 0$ or $n = 1$.</u> Then $n + 1$ is not divisible by 3, and accordingly $\text{fib}(n) = 1$ which is odd.

<u>Case 2: $n + 1$ is divisible by 3.</u> Then neither $(n - 1) + 1$ nor $(n - 2) + 1$ is divisible by 3. Our induction hypothesis thus tells us that $\text{fib}(n - 1)$ and $\text{fib}(n-2)$ are both odd. As $\text{fib}(n) = \text{fib}(n-1)+\text{fib}(n-2)$, this implies that $\text{fib}(n)$ is even, as desired.

<u>Case 3: $n > 2$ and $n + 1$ is not divisible by 3.</u> Then exactly one of $(n-1)+1$ and $(n-2)+1$ is divisible by 3. Our induction hypothesis then tells us that exactly one of $\text{fib}(n - 1)$ and $\text{fib}(n - 2)$ is even. As $\text{fib}(n) = \text{fib}(n - 1) + \text{fib}(n - 2)$, this implies that $\text{fib}(n)$ is odd, as desired. $\square$

Note that the last two steps could not have been carried out using the original principle of induction (Principle 2), where we in order to establish $Q(n)$ can assume only $Q(n - 1)$ but not $Q(n - 2)$.

We can actually be much more specific about the value of $\text{fib}(n)$:

Theorem 2.4. *For all n we have*

$$\text{fib}(n) = \frac{\phi^{n+1} - (\frac{-1}{\phi})^{n+1}}{\sqrt{5}}$$

where ϕ, also called the golden ratio[6]*, is the positive solution to the equation*

$$\phi^2 - \phi - 1 = 0 \tag{1}$$

Thus we have $\square$

$$\phi = \frac{1 + \sqrt{5}}{2} \approx 1.618 \tag{2}$$

Proof. Again, we do a course-of-values induction. First, however, observe (by successively dividing by ϕ on both sides of (1)) that

$$\phi - 1 - \frac{1}{\phi} \quad = \quad 0 \tag{3}$$

$$1 - \frac{1}{\phi} - \frac{1}{\phi^2} \quad = \quad 0 \tag{4}$$

[6]See, e.g., `http://mathworld.wolfram.com/GoldenRatio.html` for background information on ϕ.

There are three cases; in the first two, it is convenient to work "backwards".

<u>Case 1: $n = 0$.</u> By (3) and then (2), we infer the desired equality:

$$\frac{\phi^{0+1} - (\frac{-1}{\phi})^{0+1}}{\sqrt{5}} = \frac{\phi + \frac{1}{\phi}}{\sqrt{5}} = \frac{\phi + \phi - 1}{\sqrt{5}} = \frac{\sqrt{5}}{\sqrt{5}} = 1 = \mathrm{fib}(0)$$

<u>Case 2: $n = 1$.</u> Using that $\phi + \frac{1}{\phi} = \sqrt{5}$ (established in previous case), and then (3), we infer the desired equality:

$$\frac{\phi^{1+1} - (\frac{-1}{\phi})^{1+1}}{\sqrt{5}} = \frac{\phi^2 - \frac{1}{\phi^2}}{\sqrt{5}} = \frac{(\phi + \frac{1}{\phi})(\phi - \frac{1}{\phi})}{\sqrt{5}} = \phi - \frac{1}{\phi} = 1 = \mathrm{fib}(1)$$

<u>Case 3: $n = m + 2$.</u> Here we infer the desired equality as follows:

$$\begin{array}{rcl}
\mathrm{fib}(n) & = & \text{(definition of fib}(n)) \\
\mathrm{fib}(m) + \mathrm{fib}(m+1) & = & \text{(induction hypothesis)} \\
\frac{\phi^{m+1} - (\frac{-1}{\phi})^{m+1}}{\sqrt{5}} + \frac{\phi^{m+2} - (\frac{-1}{\phi})^{m+2}}{\sqrt{5}} & = & \text{(rearrangement)} \\
\frac{\phi^{m+1} + \phi^{m+2} - (\frac{-1}{\phi})^{m+1} - (\frac{-1}{\phi})^{m+2}}{\sqrt{5}} & = & \text{(common factor)} \\
\frac{\phi^{m+1}(1+\phi) - (\frac{-1}{\phi})^{m+1}(1 - \frac{1}{\phi})}{\sqrt{5}} & = & \text{(using (1) and (4))} \\
\frac{\phi^{m+1}\phi^2 - (\frac{-1}{\phi})^{m+1}(\frac{1}{\phi^2})}{\sqrt{5}} & = & \text{(since } n = m+2) \\
\frac{\phi^{n+1} - (\frac{-1}{\phi})^{n+1}}{\sqrt{5}} & &
\end{array}$$

□

3 Induction on Lists

Lists, a very common data structure, are inductively defined as follows:

base clause: $\mathsf{List}(\mathsf{nil})$ holds;

inductive clause: if $\mathsf{List}(x)$ and v is a value then also $\mathsf{List}(v \copyright x)$.

That is, a list is either empty (nil), or a value v in front of a list; here values could be natural numbers but also characters etc. (and they could even be lists themselves!)

EXAMPLE 3.1. Consider a list with the elements 5,7,4 (note that the order matters). This list is in our syntax written as

$$5 \copyright (7 \copyright (4 \copyright \mathsf{nil}))$$

which we may abbreviate as $[5, 7, 4]$. A graphic representation is

```
   C
  / \
 5   C
    / \
   7   C
      / \
     4  nil
```

□

Lists, as defined above, are very similar to the "linked lists" seen in pointer languages like C. In particular, unlike what is the case for arrays, one does not have direct access to each element, but must instead follow a chain of pointers. A key difference, however, is that lists are *immutable*: in the above example, we *cannot* replace say 7 by 8; if we do want a list $[5, 8, 4]$, we must construct it from scratch! This might seem inconvenient, but in fact makes reasoning about programs much simpler: if two pointers p_1 and p_2 may *alias*, that is, point to the same location, then modifying the object denoted by p_1 would (perhaps inadvertently) change what p_2 denotes.

Principle 4 (List induction). *In order to show that a property holds for all lists, it suffices to show that it holds for the empty list, and that the property holds for a non-empty list provided it holds for its "tail". Expressed in Fitch notation:*

$$\begin{array}{r|l} & Q(\mathsf{nil}) \\ & \vdots \\ & \forall x \forall v((\mathsf{List}(x) \wedge Q(x)) \rightarrow Q(v \copyright x)) \\ & \vdots \\ \rhd & \forall x(\mathsf{List}(x) \rightarrow Q(x)) \end{array}$$

We also call this induction principle structural induction.

To justify the validity of list induction, assume (in order to arrive at a contradiction) that the conclusion does *not* hold. That is, there exists lists not satisfying Q. Let x be among the "shortest" such lists (there might be several choices). That is, for all y such that y is shorter than x we have $Q(y)$.

We shall do a case analysis depending on whether x is nil or not, in both cases arriving at a contradiction. If $x = \mathsf{nil}$ the contradiction comes since our first premise then tells us that $Q(x)$ does hold. If x is of the form $v \copyright y$, then y is shorter than x so $Q(y)$ holds, which by our second premise implies that also $Q(x)$ holds, yielding the desired contradiction.

Definition 3.2. The *append* function, taking two lists x and y as arguments and returning their concatenation $x +\!\!+\, y$ (also a list), is given by the following recursive definition

$$\begin{array}{lll} x +\!\!+\, y & = \texttt{case}\ x\ \texttt{of} & \\ & \quad \mathsf{nil} & \Rightarrow y \\ & \quad (v \copyright x') & \Rightarrow v \copyright (x' +\!\!+\, y) \end{array}$$

□

For example, we have $[5,7] +\!\!+ [8,4] = [5,7,8,4]$ since

$$\begin{array}{rl} & (5 \copyright (7 \copyright \mathsf{nil})) +\!\!+ (8 \copyright (4 \copyright \mathsf{nil})) \\ = & 5 \copyright ((7 \copyright \mathsf{nil}) +\!\!+ (8 \copyright (4 \copyright \mathsf{nil}))) \\ = & 5 \copyright (7 \copyright (\mathsf{nil} +\!\!+ (8 \copyright (4 \copyright \mathsf{nil})))) \\ = & 5 \copyright (7 \copyright (8 \copyright (4 \copyright \mathsf{nil}))) \end{array}$$

By definition, nil is a "left neutral element" for the append function. We shall now show that it is also a right neutral element.

Theorem 3.3. *For all lists x, we have $x +\!\!+ \mathsf{nil} = x$.* □

Proof. With $Q(x)$ given by $x +\!\!+ \mathsf{nil} = x$, we shall prove by list induction that $Q(x)$ holds for all lists x. For the basis step, we must establish $Q(\mathsf{nil})$, that is

$$\mathsf{nil} +\!\!+ \mathsf{nil} = \mathsf{nil}$$

which is trivial from Definition 3.2.

For the inductive step, we can assume $Q(x)$ and must show $Q(v \copyright x)$, which follows from the calculation

$$(v \copyright x) +\!\!+ \mathsf{nil} = v \copyright (x +\!\!+ \mathsf{nil}) = v \copyright x$$

Here we used Definition 3.2 for the first equality, and the induction hypothesis for the second equality. □

We can also prove that the append function is *associative*:

Theorem 3.4. *For all x,y,z, we have* $(x +\!\!+ y) +\!\!+ z = x +\!\!+ (y +\!\!+ z)$. □

Proof. We do structural induction on x: for given y and z, we define $Q(x)$ as the predicate $(x +\!\!+ y) +\!\!+ z = x +\!\!+ (y +\!\!+ z)$.

For the basis step, we must establish Q(nil), that is

$$(\mathsf{nil} +\!\!+ y) +\!\!+ z = \mathsf{nil} +\!\!+ (y +\!\!+ z)$$

which follows as both left hand side and right hand side reduces to $y +\!\!+ z$.

For the inductive step, we can assume $Q(x)$, and must establish $Q(v \copyright x)$, which follows from the calculation

$$\begin{array}{rcl}
((v \copyright x) +\!\!+ y) +\!\!+ z & = & \text{(Definition 3.2)} \\
(v \copyright (x +\!\!+ y)) +\!\!+ z & = & \text{(Definition 3.2)} \\
v \copyright ((x +\!\!+ y) +\!\!+ z) & = & \text{(Induction hypothesis)} \\
v \copyright (x +\!\!+ (y +\!\!+ z)) & = & \text{(Definition 3.2, backwards)} \\
(v \copyright x) +\!\!+ (y +\!\!+ z) & &
\end{array}$$

□

Note that induction in y or in z would not have worked.

3.1 Other kinds of structural induction

In Sect. 2.1 we saw that for natural numbers, "course-of-values induction" is often more applicable than the standard induction principle. Similarly, for other inductively defined data structures (like binary trees), it is often more convenient to apply the following induction principle

$$\begin{array}{l|l}
 & \forall x(\forall y(y \text{ "smaller than" } x \rightarrow Q(y)) \rightarrow Q(x)) \\
 & \vdots \\
\rhd & \forall x Q(x)
\end{array}$$

The unspecific “smaller than” can be defined in a numerous ways. For binary trees, one could for instance say that “y is smaller than x” iff y has fewer nodes than x.

References

[1] Jon Barwise and John Etchemendy. *Language, Proof and Logic*. CSLI Publications, 1999.