Induction

Up to this point we have considered proofs of the following type:

> *Given a selected expression* $a \in$ **Aexp** *and some state* $\sigma \in \Sigma$ *compute the semantic value* k,
>
> $$(a, \sigma) \mapsto k.$$

Or we have considered equivalence proofs of the form,

> *Given two selected programs* $c, c' \in$ **Com**, *show that they are semantically equivalent,*
>
> $$c \sim c' \text{ iff } \forall \sigma \in \Sigma, \exists \sigma' \in \Sigma.\ (c, \sigma) \mapsto \sigma' \wedge (c', \sigma) \mapsto \sigma'$$

Question: How would we prove that a certain property holds for all expressions or for all commands? The syntax sets **Aexp**, **Bexp**, and **Com** are infinite and therefore we need a proof technique that can deal with the infinite nature of these sets $\Rightarrow$ Induction.

Induction

Induction is a proof principle that allows us to prove properties of possibly infinite sets.[1]

The power of this technique derives from the fact that we can test a particular property on a few select elements of the set and, if the tests hold, we can conclude that the property holds over the whole set.

In order for this to work the set has to have a certain structure and the tests have to be selected in a manner to provide us with maximum information.

[1] Sidebar: It is interesting to note that there is evidence that inductive proofs were already in use in Aristotle and Plato's time.

Mathematical Induction

We start with mathematical induction over the natural numbers $\mathbb{N}$. Consider the *inductive definition* of the natural numbers,

- $0 \in \mathbb{N}$,
- If $m \in \mathbb{N}$, then $m + 1 \in \mathbb{N}$.

The *inductive structure* of the natural numbers allows us to position the values along the number line without conflict,

$$0\ 1\ 2\ 3\ 4\ 5 \ldots \infty$$

a precise ordering,

$$0 < 1 < 2 < 3 < 4 < 5 < \ldots < \infty$$

Mathematical Induction

Now, if we want to show that a predicate P holds for all $n \in \mathbb{N}$, then we can structure this as an *inductive proof* in two parts:

1. **Base Case** – show that $P(0)$ holds, that is, show that the property of interest holds for the *least* element of the natural numbers.
2. **Inductive Step** – show that if $P(m)$ holds for any $m \in \mathbb{N}$, then $P(m+1)$ also holds.

If both parts can be shown to be true then we can conclude that the statement holds for all $n \in \mathbb{N}$.

This is sometimes also called the *domino effect*; here step 2 above implies $P(0) \Rightarrow P(1)$ and $P(1) \Rightarrow P(2)$ and $P(2) \Rightarrow P(3)$, etc.

Later on we will prove that this domino effect has to hold for inductively defined structures and therefore the proof principle is sound.

Mathematical Induction

Proposition: (Mathematical Induction) Let P be a predicate over the natural numbers $\mathbb{N}$, then

$$\forall n \in \mathbb{N}.P(n) \text{ iff } P(0) \wedge \forall n \in \mathbb{N}.P(n) \Rightarrow P(n+1).$$

Here, $P(0)$ is called the *basis*, $P(n)$ is the *induction hypothesis*, and $P(n) \Rightarrow P(n+1)$ is called the *inductive step*.

Mathematical Induction

Inductive proofs usually follow this pattern:

- Let P be a predicate over the natural numbers $\mathbb{N}$,
- Since the 'only-if' direction of our proposition is trivial, we make use of solely the 'if' direction:

$$\forall n \in \mathbb{N}.P(n) \text{ if } P(0) \wedge \forall n \in \mathbb{N}.P(n) \Rightarrow P(n+1),$$

- Show that the basis holds,
- Show that the inductive step holds *under the induction hypothesis*,
- Conclude that $\forall n \in \mathbb{N}.P(n).\square$

Mathematical Induction

Example: Show that

$$0 + 1 + 2 + \ldots + n = \frac{n(n+1)}{2}$$

for all $n \in \mathbb{N}$.

Proof: by induction,

1. Base case, $0 = 0(0+1)/2$.
2. Inductive step, assume that the formula holds for any n, with this assumption show that it holds for $n+1$,

$$\begin{aligned} (0 + 1 + 2 + \ldots + n) + (n+1) &= \frac{(n+1)\left((n+1)+1\right)}{2} \\ \frac{n(n+1)}{2} + (n+1) &= \frac{(n+1)\left((n+1)+1\right)}{2} \\ \frac{n(n+1) + 2(n+1)}{2} &= \frac{(n+1)\left((n+1)+1\right)}{2} \end{aligned}$$

It clearly does. (I let you finish the algebra.)□

Structural Induction

Let us consider a special kind of induction which is done over *terms* rather than the natural numbers. Because terms tend to be highly structured objects we call this *structural induction*.

Consider the following grammar G for a simple expression language,

$$E ::= (E \mathbin{@} a) \mid a$$

with

$$L(G) = \{a, (a \mathbin{@} a), ((a \mathbin{@} a) \mathbin{@} a), (((a \mathbin{@} a) \mathbin{@} a) \mathbin{@} a), ((((a \mathbin{@} a) \mathbin{@} a) \mathbin{@} a) \mathbin{@} a), \ldots\}$$

The inductive definition of the set $L(G)$ is

- $a \in L(G)$,
- $(e \mathbin{@} a) \in L(G)$ if $e \in L(G)$.

This inductive definition is very similar to the inductive definition of the natural numbers and hints at an ordering where a is the *least element* and

$$e < (e \mathbin{@} a)$$

with $e \in L(G)$.

Structural Induction

The insight that $e < (e \mathbin{@} a)$ for any $e \in L(G)$ allows us to order all the terms in $L(G)$ in an unambiguous way,

$$a < (a \mathbin{@} a) < ((a \mathbin{@} a) \mathbin{@} a) < (((a \mathbin{@} a) \mathbin{@} a) \mathbin{@} a) < ((((a \mathbin{@} a) \mathbin{@} a) \mathbin{@} a) \mathbin{@} a) < \ldots$$

This structural ordering of the terms allows us to formulate our *structural induction principle*: Let P be a predicate over the terms in $L(G)$, then

$$\forall e \in L(G).P(e) \text{ iff } P(a) \wedge \forall e \in L(G).P(e) \Rightarrow P(\ (e \mathbin{@} a)\ ).$$

Structural Induction

Example: Let $V(e)$ be the number of a's and let $O(e)$ be the number of @'s in $e \in L(G)$, show that

$$V(e) = O(e) + 1$$

for all $e \in L(G)$.

Proof: Proof by induction over the elements in $L(G)$,

Base Case: $V(a) = O(a) + 1$ with $V(a) = 1$ and $O(a) = 0$. Clearly, the identity holds.

Inductive Step: Let $e \in L(G)$ and assume that our inductive hypothesis

$$V(e) = O(e) + 1$$

holds. Observe that the following two identities also hold,

$$\begin{aligned} V(\ (e\ @\ a)\ ) &= V(e) + 1 \\ O(\ (e\ @\ a)\ ) &= O(e) + 1 \end{aligned}$$

Then,

$$\begin{aligned} V(\ (e\ @\ a)\ ) &= V(e) + 1 \\ &= O(e) + 1 + 1 \\ &= O(\ (e\ @\ a)\ ) + 1\ \square \end{aligned}$$

Structural Induction

We can apply structural induction to any inductively defined set. Recall the inductive definition of the arithmetic expressions:

- $n \in \mathbf{Aexp}$ if $n \in \mathbf{I}$,
- $x \in \mathbf{Aexp}$ if $x \in \mathbf{Loc}$,
- $a_0 + a_1 \in \mathbf{Aexp}$ if $a_0, a_1 \in \mathbf{Aexp}$,
- $a_0 - a_1 \in \mathbf{Aexp}$ if $a_0, a_1 \in \mathbf{Aexp}$,
- $a_0 * a_1 \in \mathbf{Aexp}$ if $a_0, a_1 \in \mathbf{Aexp}$,
- $(a) \in \mathbf{Aexp}$ if $a \in \mathbf{Aexp}$.

Observe that we have two base or minimal elements of **Aexp** on the left and we have the inductively defined elements of **Aexp** on the right. More precisely, we can formulate a structural ordering based on the inductive definition,

- n and x are the least elements and form the basis,
- for the remaining terms we have the following ordering
  - $a_0 < a_0 + a_1$ and $a_1 < a_0 + a_1$
  - $a_0 < a_0 - a_1$ and $a_1 < a_0 - a_1$
  - $a_0 < a_0 * a_1$ and $a_1 < a_0 * a_1$
  - $a < (\, a \,)$

Now, it is not possible to create a linear order for the elements of **Aexp**, but we can sort them into a *partially ordered lattice structure*. It turns out that that is enough to be able to perform structural induction as long as there are no ambiguities (which there are none). The only consequence of this lattice structure is that our induction principle will be more complicated ...

Structural Induction

Given the ordering of the terms we can now state our *structural induction principle* to show that some predicate P holds for all arithmetic expressions:

$$
\begin{array}{lll}
\forall a \in \mathbf{Aexp}.P(a) & \text{iff} & (\forall n \in \mathbf{I}.P(n)) \wedge \\
 & & (\forall x \in \mathbf{Loc}.P(x)) \wedge \\
 & & (\forall a_0, a_1 \in \mathbf{Aexp}.P(a_0) \wedge\ P(a_1) \Rightarrow P(a_0 + a_1)) \wedge \\
 & & (\forall a_0, a_1 \in \mathbf{Aexp}.P(a_0) \wedge\ P(a_1) \Rightarrow P(a_0 - a_1)) \wedge \\
 & & (\forall a_0, a_1 \in \mathbf{Aexp}.P(a_0) \wedge\ P(a_1) \Rightarrow P(a_0 * a_1)) \wedge \\
 & & (\forall a \in \mathbf{Aexp}.P(a) \Rightarrow P(\ (a)\ ))
\end{array}
$$

As expected, here we also take advantage of the precise ordering of terms and their sub terms and therefore the domino effect also works here.

Structural Induction

Observation: As many cases as there are expression forms

- Atomic expressions (with no subexpressions) are all base cases
- Composite expressions are the inductive cases

Exercise: State the structural induction principle for the remaining syntax sets.

Structural Induction

Example: Let $a \in \mathbf{Aexp}$, let $L(a)$ be the number of literals and variable occurrences in a, let $O(a)$ be the number of operators in a, and define the predicate P as

$$P(a) \equiv L(a) \leq O(a) + 1.$$

Show that $\forall a \in \mathbf{Aexp}.P(a)$.

Proof: By induction over the structure of **Aexp**. We make use of the structural induction principle for **Aexp**,

$$\begin{aligned}
\forall a \in \mathbf{Aexp}.P(a) \quad \text{if} \quad & (\forall n \in \mathbf{I}.P(n)) \wedge \\
& (\forall x \in \mathbf{Loc}.P(x)) \wedge \\
& (\forall a_0, a_1 \in \mathbf{Aexp}.P(a_0) \wedge P(a_1) \Rightarrow P(a_0 + a_1)) \wedge \\
& (\forall a_0, a_1 \in \mathbf{Aexp}.P(a_0) \wedge P(a_1) \Rightarrow P(a_0 - a_1)) \wedge \\
& (\forall a_0, a_1 \in \mathbf{Aexp}.P(a_0) \wedge P(a_1) \Rightarrow P(a_0 * a_1)) \wedge \\
& (\forall a \in \mathbf{Aexp}.P(a) \Rightarrow P(\ (a)\ ))
\end{aligned}$$

Structural Induction

Case n.

$L(n) = 1$ and $O(n) = 0$, clearly $P(n)$ holds.

Case x.

$L(x) = 1$ and $O(x) = 0$, clearly $P(x)$ holds.

Structural Induction

Case (a).

As the induction hypothesis we assume that $P(a) \equiv L(a) \leq O(a) + 1$ holds. We need to show that

$$\forall a \in \mathbf{Aexp}.P(a) \Rightarrow P(\ (a)\ ).$$

We now show that $P(\ (a)\ ) \equiv L(\ (a)\ ) \leq O(\ (a)\ ) + 1$ holds. It is easy to see that

$$L(\ (a)\ ) = L(a)$$

and

$$O(\ (a)\ ) = O(a),$$

then

$$\begin{aligned} L(\ (a)\ ) &\leq L(a) \\ &\leq O(a) + 1 \\ &\leq O(\ (a)\ ) + 1. \end{aligned}$$

Structural Induction

Case $a_0 + a_1$.

As the induction hypothesis we assume that $P(a_0) \equiv L(a_0) \leq O(a_0) + 1$ and $P(a_1) \equiv L(a_1) \leq O(a_1) + 1$ are true. We need to show that

$$\forall a_0, a_1 \in \mathbf{Aexp}. P(a_0) \wedge P(a_1) \Rightarrow P(a_0 + a_1).$$

We now show that $P(a_0 + a_1) \equiv L(a_0 + a_1) \leq O(a_0 + a_1) + 1$ holds. It is easy to see that

$$L(a_0 + a_1) = L(a_0) + L(a_1)$$

and

$$O(a_0 + a_1) = O(a_0) + O(a_1) + 1,$$

then

$$\begin{aligned} L(a_0 + a_1) &\leq L(a_0) + L(a_1) \\ &\leq (O(a_0) + 1) + (O(a_1) + 1) \\ &\leq (O(a_0) + O(a_1) + 1) + 1 \\ &\leq O(a_0 + a_1) + 1. \end{aligned}$$

The remaining cases can be shown to hold in a similar fashion. This concludes the proof.□

Structural Induction

Example: Prove that every computation with arithmetic expressions terminates. Formally,

$$\forall a \in \textbf{Aexp}, \forall \sigma \in \Sigma, \exists k \in \mathbb{I}.(a, \sigma) \mapsto k.$$

Proof: By structural induction over **Aexp**, let

$$P(a) \equiv \forall \sigma \in \Sigma, \exists k \in \mathbb{I}.(a, \sigma) \mapsto k.$$

Case n.

$\forall \sigma \in \Sigma$, we have $(n, \sigma) \mapsto \text{eval}(n)$. Clearly, $P(n)$ holds.

Case x.

$\forall \sigma \in \Sigma$, we have $(x, \sigma) \mapsto \sigma(x)$. Clearly, $P(x)$ holds.

Structural Induction

Case (a).

As the inductive hypothesis assume that $P(a)$ holds, then $\forall \sigma \in \Sigma$ we have,

$$\frac{(a, \sigma) \mapsto k}{(\ (a)\ , \sigma) \mapsto k}$$

The premise holds by assumption. Therefore, $P(\ (a)\ )$ holds.

Structural Induction

Case $a_0 + a_1$.

As the inductive hypothesis assume that $P(a_0)$ and $P(a_1)$ hold, then $\forall \sigma \in \Sigma$ we have,

$$\frac{(a_0, \sigma) \mapsto k_0 \qquad (a_1, \sigma) \mapsto k_1}{(a_0 + a_1, \sigma) \mapsto k}$$, where $k = k_0 + k_1$.

The premises hold by assumption. Therefore, $P(a_0 + a_1)$ holds.

The remaining cases can be shown to hold in a similar fashion. This proves that all arithmetic expressions terminate. $\square$

Limits of Structural Induction

Structural induction can only be applied where syntax directs the computation such that only subexpressions appear in the premisses of inductive proof steps. Consider the **while** loop; its evaluation does not depend only on the evaluation of its strict subexpressions:

$$\frac{(b, \sigma) \mapsto \textbf{true} \qquad (c, \sigma) \mapsto \sigma'' \qquad (\textbf{while } b \textbf{ do } c, \sigma'') \mapsto \sigma'}{(\textbf{while } b \textbf{ do } c, \sigma) \mapsto \sigma'}$$

This means we *cannot* use structural induction on while loops.

To get around this problem we can resort to well-founded induction on states, mathematical induction on loop invariants (in the case of **while** loops), or induction on derivations.