Induction

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine

Overview

- Inductive Definitions
- Induction Proofs
- Application: Verifications
- Induction on Sets

Inductive Definitions

Situation

- Recursive definitions on $\mathbb{N}$:

$$
\begin{aligned}
&* : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \\
&x * y := \textbf{if } y = 0 \textbf{ then } 0 \textbf{ else } x + (x * y^{-})
\end{aligned}
$$

- Proposition:

$$
\begin{aligned}
x * 0 &= 0, \\
x * y' &= x + (x * y).
\end{aligned}
$$

Recursive function definition implies pair of equations.

Idea

Also converse is true:

- For each $a * b$, left hand side of only one equation "matches":
  - Either $b = 0$ or $b = y'$ for some y.
  - Consequence of first Peano axiom.
- Equality $b = y'$ determines unique y.
  - $(b = {y_0}' \wedge b = {y_1}') \Rightarrow y_0 = y_1$ for all y_0 and y_1.
  - Consequence of second Peano axiom.

Pair of equations uniquely determines a function.

Alternative Definition Format

Definition by pair of equations ("induction on second argument"):

$$
\begin{aligned}
&* : \mathbb{N} \times \mathbb{N} \to \mathbb{N}\\
&x * 0 := 0,\\
&x * (y+1) := x + (x * y).
\end{aligned}
$$

By syntactic restriction of the equations, the function is well defined.

Inductive Function Definitions

Definition: An inductive definition over $\mathbb{N}$ of an n-ary function f:

$$\begin{aligned} f(x_0, \ldots, 0, \ldots, x_{n-1}) &:= T_\mathrm{b}, \\ f(x_0, \ldots, x_i + 1, \ldots, x_{n-1}) &:= T_\mathrm{r} \end{aligned}$$

- f does not occur in base term T_b.
- Every application of f in recursion term T_r has form

$$f(T_0, \ldots, x_i, \ldots, T_{n-1})$$

- Free variables of terms must occur in definiendum.

Induction runs over x_i.

Inductively Defined Function

Let $A_0, \ldots, A_{i-1}, A_{i+1}, \ldots, A_{n-1}, B$ such that we have

$$T_{\mathrm{b}} \in B \wedge T_{\mathrm{r}} \in B$$

for all $x_0 \in A_0, \ldots, x_{i-1} \in A_{i-1}, x_i \in \mathbb{N}, x_{i+1} \in A_{i+1}, \ldots, x_{n-1} \in A_{n-1}$. Then the definition introduces the unique function

$$f : A_0 \times \ldots \times A_{i-1} \times \mathbb{N} \times A_{i+1} \times \ldots \times A_{n-1} \to B$$

that satisfies

$$\begin{aligned} f(x_0, \ldots, 0, \ldots, x_{n-1}) &= T_{\mathrm{b}} \wedge \\ f(x_0, \ldots, x_i + 1, \ldots, x_{n-1}) &= T_{\mathrm{r}} \end{aligned}$$

for all $x_0 \in A_0, \ldots, x_{i-1} \in A_{i-1}, x_i \in \mathbb{N}, x_{i+1} \in A_{i+1}, \ldots, x_{n-1} \in A_{n-1}$.

Induction with Larger Decrements

Example: Fibonacci Numbers

$$\begin{aligned} \mathrm{fib}(0) &:= 1, \\ \mathrm{fib}(1) &:= 1, \\ \mathrm{fib}(x+2) &:= \mathrm{fib}(x) + \mathrm{fib}(x+1) \end{aligned}$$

$$\mathrm{fib} = [1, 1, 2, 3, 5, 8, 13, 21, \ldots]$$

All base cases must be covered!

Induction over Multiple Arguments

Examples:

$$\begin{aligned} f(0,0) &:= 0, \\ f(x+1,0) &:= 1+f(x,0), \\ f(x,y+1) &:= 1+f(x,y). \end{aligned}$$

$$\begin{aligned} f(0,0) &:= 0, \\ f(x+1,0) &:= 1+f(x,0), \\ f(0,y+1) &:= 1+f(0,y), \\ f(x+1,y+1) &:= 2+f(x,y), \end{aligned}$$

All possible base cases must be covered!

Inductive Predicate Definitions

Definition: An inductive definition over $\mathbb{N}$ of an n-ary predicate p:

$$\begin{aligned} p(x_0, \ldots, 0, \ldots, x_{n-1}) &:\Leftrightarrow F_{\mathrm{b}}, \\ p(x_0, \ldots, x_i + 1, \ldots, x_{n-1}) &:\Leftrightarrow F_{\mathrm{r}} \end{aligned}$$

- p does not occur in base formula F_{b}.
- Every application of p in recursion formula F_{r} has form

$$p(T_0, \ldots, x_i, \ldots, T_{n-1})$$

- Free variables of terms must occur in definiendum.

Induction runs over x_i.

Inductively Defined Predicate

Take sets $A_0, \ldots, A_{i-1}, A_{i+1}, \ldots, A_{n-1}$.
The definition introduces the predicate

$$p \subseteq A_0 \times \ldots \times A_{i-1} \times \mathbb{N} \times A_{i+1} \times \ldots \times A_{n-1}$$

that satisfies

$$\begin{aligned} p(x_0, \ldots, 0, \ldots, x_{n-1}) &\Leftrightarrow T_{\mathrm{b}} \wedge \\ p(x_0, \ldots, x_i + 1, \ldots, x_{n-1}) &\Leftrightarrow T_{\mathrm{r}} \end{aligned}$$

for all $x_0 \in A_0, \ldots, x_{i-1} \in A_{i-1}, x_i \in \mathbb{N}, x_{i+1} \in A_{i+1}, \ldots, x_{n-1} \in A_{n-1}$.

Example

We can introduce the predicate $\mathrm{iseven}(x) :\Leftrightarrow 2|x$ also as

$$\begin{aligned}
&\mathrm{iseven}(0) :\Leftrightarrow \mathrm{T},\\
&\mathrm{iseven}(x+1) :\Leftrightarrow \neg\mathrm{iseven}(x).
\end{aligned}$$

or as

$$\begin{aligned}
&\mathrm{iseven}(0) :\Leftrightarrow \mathrm{T},\\
&\mathrm{iseven}(1) :\Leftrightarrow \mathrm{F},\\
&\mathrm{iseven}(x+2) :\Leftrightarrow \mathrm{iseven}(x).
\end{aligned}$$

$$\mathrm{iseven} = [\mathrm{T}, \mathrm{F}, \mathrm{T}, \mathrm{F}, \mathrm{T}, \ldots]$$

Induction Proofs

Mathematical Induction

Third Peano Axiom:

$$(F[x \leftarrow 0] \wedge (\forall x \in \mathbb{N} : F \Rightarrow F[x \leftarrow x+1])) \Rightarrow \forall x \in \mathbb{N} : F.$$

Proposition: In order to prove

$$\forall x \in \mathbb{N} : F,$$

it suffices to prove

1. $F[x \leftarrow 0]$,
2. $(\forall x \in \mathbb{N} : F \Rightarrow F[x \leftarrow x+1])$.

Typical Format

We want to prove

$$\forall x \in \mathbb{N} : F.$$

1. Induction Base: We show $F[x \leftarrow 0]$.
2. Induction Hypothesis: We take arbitrary $x \in \mathbb{N}$ and assume F.
3. Induction Step: We show $F[x \leftarrow x + 1]$.

Proof strategy for formulas that are universally quantified over $\mathbb{N}$.

Example

We prove by induction on n

$$\forall n \in \mathbb{N} : n < 2^n.$$

The induction base holds because $0 < 1 = 2^0$.
Now we take arbitrary $n \in \mathbb{N}$ and assume (induction hypothesis)

$$(1)\ n < 2^n.$$

We have to show (induction step)

$$(2)\ n + 1 < 2^{n+1}.$$

By (1) we have

$$(3)\ n + 1 < 2^n + 1$$

and therefore

$$(4)\ n + 1 < 2^n + 1 \leq 2^n + 2^n = 2 * 2^n = 2^{n+1}$$

which implies (2).

Example

We prove by induction on n

$$\forall n \in \mathbb{N} : 3 | n^3 + 2n$$

The induction base holds because $3|0$ and $0 = 0^3 + 2 * 0$.
We take arbitrary $n \in \mathbb{N}$ and assume

$$(1)\ 3|n^3 + 2n.$$

We have to show

$$(2)\ 3|(n+1)^3 + 2(n+1).$$

Example (Continued)

By (1) and definition of $|$ we have some $a \in \mathbb{N}$ such that

$$(3)\ 3a = n^3 + 2n.$$

We therefore have

$$\begin{aligned}
(n+1)^3 + 2(n+1) &= \\
(n^3 + 3n^2 + 3n + 1) + (2n + 2) &= \\
(n^3 + 2n) + (3n^2 + 3n + 3) &= (3) \\
3a + 3(n^2 + n + 1) &= \\
3(a + n^2 + n + 1) &
\end{aligned}$$

which implies (2) by definition of $|$.

Example

We prove by induction on n

$$\forall n \in \mathbb{N} : \sum_{1 \leq i \leq n} i = \frac{(n+1)n}{2}$$

The induction base holds because

$$\sum_{1 \leq i \leq 0} i = 0 = \frac{(0+1) * 0}{2}.$$

We take arbitrary $n \in \mathbb{N}$ and assume

$$(1) \quad \sum_{1 \leq i \leq n} i = \frac{(n+1)n}{2}.$$

We have to show

$$(2) \quad \sum_{1 \leq i \leq n+1} i = \frac{((n+1)+1)(n+1)}{2}.$$

Example (Continued)

We have

$$
\begin{array}{rcl}
\sum_{1\leq i\leq n+1} i & = & (\text{definition } \sum) \\
\sum_{1\leq i\leq n} i + (n+1) & = & (1) \\
\frac{(n+1)n}{2} + (n+1) & = & \\
\frac{(n+1)n+2(n+1)}{2} & = & \\
\frac{(n+1)(n+2)}{2} & = & \\
\frac{(n+1)((n+1)+1)}{2}. & &
\end{array}
$$

which implies (2).

Example

We can prove by induction the "computing laws" in $\mathbb{N}$:

We prove

$$\forall x \in \mathbb{N}, y \in \mathbb{N}, z \in \mathbb{N} : x + (y + z) = (x + y) + z.$$

We take arbitrary $x \in \mathbb{N}$ and $y \in \mathbb{N}$ and prove by induction on z.

$$\forall z \in \mathbb{N} : x + (y + z) = (x + y) + z$$

We have to show

$$x + (y + 0) = (x + y) + 0.$$

...

See lecture notes.

Complete Induction

Generalization of the induction principle:

Proposition: In order to prove

$$\forall x \in \mathbb{N} : F$$

it suffices to prove

$$(\forall x \in \mathbb{N} : (\forall n < x : F[x \leftarrow n]) \Rightarrow F).$$

1. Induction Hypothesis. We take arbitrary $x \in \mathbb{N}$ and assume

$$\forall n < x : F[x \leftarrow n].$$

2. Induction Step: We show F.

Example

We prove that every natural number greater than 1 can be factorized into a sequence of prime numbers, i.e.,

$$\forall n \in \mathbb{N} : n > 1 \Rightarrow$$
$$(\exists k \in \mathbb{N}, f : \mathbb{N}_k \to \mathbb{N} : n = \prod_{0 \leq i < k} f(i) \land \forall i \in \mathbb{N}_k : f(i) \text{ is prime}).$$

We proceed by complete induction over n.
We take arbitrary $n \in \mathbb{N}$ and assume

$$(1)\ \forall m < n : m > 1 \Rightarrow$$
$$(\exists k \in \mathbb{N}, f : \mathbb{N}_k \to \mathbb{N} : m = \prod_{0 \leq i < k} f(i) \land \forall i \in \mathbb{N}_k : f(i) \text{ is prime}).$$

We have to show

$$n > 1 \Rightarrow$$
$$(\exists k \in \mathbb{N}, f : \mathbb{N}_k \to \mathbb{N} : n = \prod_{0 \leq i < k} f(i) \land \forall i \in \mathbb{N}_k : f(i) \text{ is prime}).$$

See lecture notes.

Induction over Term Values

Proposition: In order to prove F, it suffices to prove

$$(\forall y \in \mathbb{N} : y = T \Rightarrow F)$$

where y does not occur freely in T or F.

Consequence: in order to prove

$$\forall x_0, \ldots, x_{n-1} : F$$

we may prove

$$(\forall x_0, \ldots, x_{n-1}, y \in \mathbb{N} : y = T \Rightarrow F)$$

where T is a term with free variables $x_0, \ldots, x_{n-1}$.

We introduce a variable over $\mathbb{N}$ to proceed by induction.

Application: Verification

Specifications

Definition: For every function $f : A \to B$, a relation $I \subseteq A$ and a relation $O \subseteq A \times B$, we call the formula

$$\forall x : I(x) \Rightarrow O(x, f(x))$$

a specification of f with input condition I and output condition O.

If the formula is true, then f implements the specification.

We want to verify whether a function implements a specification.

Example

Exponentiation function:

$$
\begin{aligned}
x^0 &:= 1, \\
x^{n+1} &:= x * x^n.
\end{aligned}
$$

We want to verify that the function implements the specification

$$\forall x, n \in \mathbb{N} : x^n = \prod_{1 \leq i \leq n} x.$$

Example (Continued)

Take arbitrary x; we proceed by induction over n.
We have $x^0 = 1 = \prod_{1 \leq i \leq 0} x$ and thus the induction base holds.
We take arbitrary $n \in \mathbb{N}$ and assume

$$(1)\ x^n = \prod_{1 \leq i \leq n} x.$$

We have to prove

$$(2)\ x^{n+1} = \prod_{1 \leq i \leq n+1} x.$$

We know

$$\begin{array}{rcl} x^{n+1} & = & \text{(definition exponentiation)} \\ x * x^n & = & (1) \\ x * \prod_{1 \leq i \leq n} x & = & \text{(definition } \prod \text{)} \\ \prod_{1 \leq i \leq n+1} x & & \end{array}$$

which implies (2).

Purpose of Verification

- Given: input condition I and output condition O.
  - Abstract definition of a function ("what is to be done").
  - May be inconstructive (does not immediately yield an algorithm).
  - Even if constructive, the corresponding algorithm may be too inefficient.
- Given: definition of a function f.
  - Concrete definition of a function ("how is it done").
  - Intended to yield (efficient) algorithm.
- Verification: show that f implements corresponding specification.

Definition of such a function and its verification needs more knowledge; more knowledge gives better algorithms.

Example: Greatest Common Divisor

$$\gcd(x, y) := \textbf{such } z \in \mathbb{N} : z|x \wedge z|y \wedge (\forall w : (w|x \wedge w|y) \Rightarrow w \leq z).$$

```
fun gcd(x, y) =
  let(m = if(=(x, N0), y, x):
    such(z in nat(N0, m):
      and(divides(z, x), divides(z, y),
        forall(w in nat(+N(z, N1), m):
          or(not(divides(w, x)), not(divides(w, y))))),
      z));
```

Extremely inefficient way to compute the greatest common divisor.

Specification

We can show that

$$\forall z \in \mathbb{N} : \exists w : w | 0 \land w > z$$

i.e., $\mathrm{gcd}(0,0)$ is undefined, but that, if $x \neq 0 \lor y \neq 0$,

$$\exists z \in \mathbb{N} : z | x \land z | y \land (\forall w : (w | x \land w | y) \Rightarrow w \leq z).$$

i.e., $\mathrm{gcd}(x,y)$ is well defined.

Thus our problem is to find some f that implements the specification

$$\forall m \in \mathbb{N}, n \in \mathbb{N} : (m \neq 0 \lor n \neq 0) \Rightarrow \mathrm{f}(m,n) = \mathrm{gcd}(m,n).$$

in a more efficient way than gcd does.

Euclid's Algorithm

New knowledge:

$$(0)\ \forall m \in \mathbb{N}, n \leq m : \gcd(m,n) = \gcd(m-n,n).$$

Idea for recursive function definition (termination term $m+n$):

$$\begin{array}{l} \mathrm{Euclid}(m,n) := \\ \quad \textbf{if } m = 0 \textbf{ then } n \\ \quad \textbf{else if } n = 0 \textbf{ then } m \\ \quad \textbf{else if } n \leq m \textbf{ then } \mathrm{Euclid}(m-n,n) \\ \quad \textbf{else } \mathrm{Euclid}(m,n-m). \end{array}$$

Verification

$$\forall m \in \mathbb{N}, n \in \mathbb{N} : (m \neq 0 \lor n \neq 0) \Rightarrow \mathrm{Euclid}(m, n) = \mathrm{gcd}(m, n).$$

Proof by complete induction on term $m + n$.

We take arbitrary $m \in \mathbb{N}$ and $n \in \mathbb{N}$ and assume

$$\begin{array}{l}(1)\ \forall x \in \mathbb{N}, y \in \mathbb{N} : x + y < m + n \Rightarrow \\ \qquad (x \neq 0 \lor y \neq 0) \Rightarrow \mathrm{Euclid}(x, y) = \mathrm{gcd}(x, y).\end{array}$$

We have to prove

$$(2)\ (m \neq 0 \lor n \neq 0) \Rightarrow \mathrm{Euclid}(m, n) = \mathrm{gcd}(m, n).$$

We assume (3) $(m \neq 0 \lor n \neq 0)$ and prove (4) $\mathrm{Euclid}(m, n) = \mathrm{gcd}(m, n)$.

Verification (Continued)

By function definition, we have four cases:

- $m = 0$.

  By (3), we have $n \neq 0$ and, by definition of gcd and Euclid,

  $$\mathrm{gcd}(m,n) = n = \mathrm{Euclid}(m,n)$$

  which implies (4).

- $m \neq 0 \wedge n = 0$.

  We have, by definition of gcd and Euclid,

  $$\mathrm{gcd}(m,n) = m = \mathrm{Euclid}(m,n)$$

  which implies (4).

Verification (Continued)

- $m \neq 0 \land n \neq 0 \land n \leq m$.

  We know

$$\begin{aligned}
\gcd(m, n) &= (0) \\
\gcd(m - n, n) &= (1) \\
\text{Euclid}(m - n, n) &= (\text{definition Euclid}) \\
\text{Euclid}(m, n)
\end{aligned}$$

  which implies (4).

- $m \neq 0 \land n \neq 0 \land n \not\leq m$.

  The proof is analogous to the previous case.

Improvements

More knowledge:

$$(0')\ \forall m \in \mathbb{N}, n \neq 0 : \mathrm{gcd}(m,n) = \mathrm{gcd}(m, m \bmod n)$$

Function definition (with recursion term $m+n$):

$$
\begin{array}{l}
\mathrm{Euclid}'(m,n) := \\
\quad \textbf{if } m = 0 \textbf{ then } n \\
\quad \textbf{else if } n = 0 \textbf{ then } m \\
\quad \textbf{else if } n \leq m \textbf{ then } \mathrm{Euclid}'(m \bmod n, n) \\
\quad \textbf{else } \mathrm{Euclid}'(m, n \bmod m)
\end{array}
$$

Logic Evaluator

```
fun Euclid(m: N, n: N) recursive +(m, n) =
  if(=(m, 0), n,
  if(=(n, 0), m,
  if(<=(m, n), Euclid(m, -(n, m)),
    Euclid(-(m, n), n))));
fun Euclid'(m: N, n: N) recursive +(m, n) =
  if(=(m, 0), n,
  if(=(n, 0), m,
  if(<=(m, n), Euclid'(m, modN(n, m)),
    Euclid'(modN(m, n), n))));
```

Much faster than gcd!

Induction on Sets

Inductive Set Definition

Definition: An inductive definition of a set S is a collection of formulas

$$
\begin{array}{l}
(\forall x_1, \ldots, x_{m_1}, y_1 \in S, \ldots, y_{n_1} \in S : \\
\quad f_1(x_1, \ldots, x_{m_1}, y_1, \ldots, y_{n_1}) \in S) \\
, \; \ldots \; , \\
(\forall x_1, \ldots, x_{m_c}, y_1 \in S, \ldots, y_{n_c} \in S : \\
\quad f_c(x_1, \ldots, x_{m_c}, y_1, \ldots, y_{n_c}) \in S)
\end{array}
$$

where we call the function constants $f_1, \ldots, f_c$ the constructors of S.

Defined Set

S is the smallest set on which the conjunction of these formulas holds, i.e., every element of S is described by a constructor term

$$f_i(T_1, \ldots, T_{m_i}, S_1, \ldots, S_{n_i})$$

for some terms $T_1, \ldots, T_{m_i}, S_1, \ldots, S_{n_i}$ where the $S_1, \ldots, S_{n_i}$ are also such constructor terms.

Example

The set $\mathbb{N}$ is inductively defined by

$$\begin{aligned}
&0 \in \mathbb{N}, \\
&\forall x \in \mathbb{N} : x' \in \mathbb{N}
\end{aligned}$$

with constructors 0 and '.

Every element of $\mathbb{N}$ is of the form

$$0^{\prime \cdots \prime},$$

e.g. the number 4 in $\mathbb{N}$ is denoted by $0''''$.

Example

For every set T, the set $\mathrm{List}(T)$ is defined by

$$\begin{aligned}
&\mathrm{nil} \in \mathrm{List}(T),\\
&\forall e \in T, l \in \mathrm{List}(T) : \mathrm{cons}(e, l) \in \mathrm{List}(T).
\end{aligned}$$

with constructors nil and cons.

Every element of $\mathrm{List}(T)$ is of the form

$$\mathrm{cons}(e_0, \ldots, \mathrm{cons}(e_{n-1}, \mathrm{nil})),$$

e.g. the list $[2, 3]$ in $\mathrm{List}(\mathbb{N})$ is denoted by $\mathrm{cons}(2, \mathrm{cons}(3, \mathrm{nil}))$.

Example

For every set T, the set $\mathrm{Tree}(T)$ is defined by

$$\begin{aligned}&\mathrm{empty} \in \mathrm{Tree}(T),\\&\forall e \in T, l \in \mathrm{Tree}(T), r \in \mathrm{List}(T) : \mathrm{node}(e, l, r) \in \mathrm{Tree}(T).\end{aligned}$$

with constructors empty and node.
Every element of $\mathrm{Tree}(T)$ is of the form

$$\mathrm{node}(n_0, \mathrm{node}(n_{11}, \ldots), \mathrm{node}(n_{21}, \ldots)),$$

1
2 5
3 4

$\mathrm{node}(1, \mathrm{node}(2, \mathrm{node}(3, \mathrm{empty}, \mathrm{empty}), \mathrm{node}(4, \mathrm{empty}, \mathrm{empty})), \mathrm{node}(5, \mathrm{empty}, \mathrm{empty}))$

Term

The set Term is defined by

$$\begin{aligned}
&0 \in \mathrm{Term},\\
&1 \in \mathrm{Term},\\
&\forall x \in \mathrm{Term} : -x \in \mathrm{Term},\\
&\forall x \in \mathrm{Term}, y \in \mathrm{Term} : x + y \in \mathrm{Term},\\
&\forall x \in \mathrm{Term}, y \in \mathrm{Term} : x * y \in \mathrm{Term}
\end{aligned}$$

with constructors $0, 1, -, +, *$.

An element of Term is $1 + (1 + 0) * 1$.

Formula

The set Formula is defined by

$$\begin{aligned}
&\text{T} \in \text{Formula} \\
&\forall x \in \text{Formula} : \text{not}(x) \in \text{Formula}, \\
&\forall x \in \text{Formula}, y \in \text{Formula} : \text{and}(x, y) \in \text{Formula}, \\
&\forall x \in \text{Variable}, y \in \text{Formula} : \text{forall}(x, y) \in \text{Formula}
\end{aligned}$$

with constructors "T", "not", "and", "forall".

An element of Formula is $\text{forall}(\text{X}, \text{and}(\text{T}, \text{or}(\text{T}, \text{F})))$ (assuming $\text{X} \in \text{Variable}$).

Term Algebra

An inductively defined set is a term algebra if we have for every constructor f of this set

$$\forall x, y : f(x) = f(y) \Rightarrow x = y$$

i.e., different arguments are mapped to different results.

Furthermore, for all constructors f and g

$$\forall x, y : f(x) \neq g(y)$$

i.e., different constructors yield different results.

Consequence

- Every element of a term algebra is denoted by one and only one constructor term

$$f_i(T_1, \ldots, T_{m_i}, S_1, \ldots, S_{n_i})$$

  for some terms $T_1, \ldots, T_{m_i}, S_1, \ldots, S_{n_i}$ where the $S_1, \ldots, S_{n_i}$ are also constructor terms.
- One to one correspondence between terms and set elements.

We may define functions and predicates in term algebras inductively .

Example

Take the set $\mathrm{List}(T)$ defined in the previous example and assume that it is a term algebra. We define the length of a list as

$$\begin{aligned}
&\mathrm{length} : \mathrm{List}(T) \to \mathbb{N} \\
&\mathrm{length}(\mathrm{nil}) := 0 \\
&\mathrm{length}(\mathrm{cons}(e, l)) := 1 + \mathrm{length}(l).
\end{aligned}$$

Then we have $\mathrm{length}(\mathrm{cons}(1, \mathrm{cons}(2, \mathrm{nil}))) = 2.$

Example

Take the set Term defined in the previous example and assume that it is a term algebra. We define the value of a term as

$$
\begin{aligned}
&\mathrm{value} : \mathrm{Term} \to \mathbb{N} \\
&\mathrm{value}(0) := 0_{\mathbb{N}} \\
&\mathrm{value}(1) := 1_{\mathbb{N}} \\
&\mathrm{value}(-x) := -_{\mathbb{N}}\mathrm{value}(x) \\
&\mathrm{value}(x + y) := \mathrm{value}(x) +_{\mathbb{N}} \mathrm{value}(y) \\
&\mathrm{value}(x * y) := \mathrm{value}(x) *_{\mathbb{N}} \mathrm{value}(y)
\end{aligned}
$$

Then we have $\mathrm{value}(1 + (1 + 0) * 1) = 2$.

Generalized Induction Principle

We want to prove

$$\forall x \in S : F.$$

Idea: exery element x in S is denoted by some term

$$f_i(x_1, \ldots, x_{m_i}, y_1, \ldots, y_{n_i}).$$

Let the induction run over the structure of every such term:

- assume that F holds for every "S-component" y_j of x, and
- show that F is propagated to x itself.

Structural Induction

Proposition: In order to prove a property

$$\forall x \in S : F$$

for an inductively defined set S, it suffices to prove

$$\begin{array}{l} \forall x_1, \ldots, x_{m_i}, y_1 \in S, \ldots, y_{n_i} \in S : \\ \quad (F[x := y_1] \wedge \ldots \wedge F[x := y_{n_i}]) \Rightarrow \\ \qquad F[x := f_i(x_1, \ldots, x_{m_i}, y_1, \ldots, y_{n_i})] \end{array}$$

for every constructor f_i of S.

Example

Take the set $\mathrm{List}(T)$ defined inductively as

$$\begin{aligned} &\mathrm{nil} \in \mathrm{List}(T), \\ &\forall e \in T, l \in \mathrm{List}(T) : \mathrm{cons}(e, l) \in \mathrm{List}(T). \end{aligned}$$

We define

$$\begin{aligned} &\mathrm{append} : \mathrm{List}(T) \times \mathrm{List}(T) \to \mathrm{List}(T) \\ &\mathrm{append}(\mathrm{nil}, y) := y \\ &\mathrm{append}(\mathrm{cons}(e, x), y) := \mathrm{cons}(e, \mathrm{append}(x, y)) \end{aligned}$$

and claim that the following holds:

$$\begin{aligned} &\forall x \in \mathrm{List}(T), y \in \mathrm{List}(T) : \\ &\quad \mathrm{length}(\mathrm{append}(x, y)) = \mathrm{length}(x) + \mathrm{length}(y). \end{aligned}$$

Example (Continued)

We proceed by structural induction on x:

Case $x = \mathrm{nil}$**:** We have to show

$$\forall y \in \mathrm{List}(T) : \\ \mathrm{length}(\mathrm{append}(\mathrm{nil}, y)) = \mathrm{length}(\mathrm{nil}) + \mathrm{length}(y).$$

Take arbitrary $y \in \mathrm{List}(T)$. We have

$$\begin{aligned} \mathrm{length}(\mathrm{append}(\mathrm{nil}, y)) &= \text{(definition append)} \\ \mathrm{length}(y) &= \\ 0 + \mathrm{length}(y) &= \text{(definition length)} \\ \mathrm{length}(\mathrm{nil}) + \mathrm{length}(y). \end{aligned}$$

Example (Continued)

Case $x = \text{cons}(e, l)$**:** Take arbitrary $e \in T$ and $l \in \text{List}(T)$.

We assume (induction hypothesis)

$$\begin{array}{l} \forall y \in \text{List}(T) : \\ \quad \text{length}(\text{append}(l, y)) = \text{length}(l) + \text{length}(y) \end{array}$$

and have to show

$$\begin{array}{l} \forall y \in \text{List}(T) : \\ \quad \text{length}(\text{append}(\text{cons}(e, l), y)) = \text{length}(\text{cons}(e, l)) + \text{length}(y). \end{array}$$

Example (Continued)

Take arbitrary $y \in \mathrm{List}(T)$. We have

$$\begin{aligned}
\mathrm{length}(\mathrm{append}(\mathrm{cons}(e,l),y)) &= \text{(definition append)}\\
\mathrm{length}(\mathrm{cons}(e,\mathrm{append}(l,y))) &= \text{(definition length)}\\
1+\mathrm{length}(\mathrm{append}(l,y)) &= \text{(induction hypothesis)}\\
1+(\mathrm{length}(l)+\mathrm{length}(y)) &=\\
(1+\mathrm{length}(l))+\mathrm{length}(y) &= \text{(definition length)}\\
\mathrm{length}(\mathrm{cons}(e,l))+\mathrm{length}(y).
\end{aligned}$$

Summary

- Inductive definitions on $\mathbb{N}$.
  - Single induction parameter.
  - Multiple base cases.
  - Multiple induction parameters.
- Induction proofs on $\mathbb{N}$.
  - Mathematical induction.
  - Complete induction.
  - Induction over term values.
- Induction on sets.
  - Inductive set definitions.
  - Inductive function/predicate definitions on term algebras.
  - Induction proofs on inductively defined sets.