Proofs by Induction

Proposition: *If* $f(0) = 0$ *and* $f(n+1) = f(n) + n + 1$ *then, for all* $n \in \mathbb{N}$, *we have* $f(n) = n(n+1)/2$

Let $S(n)$ be $f(n) = n(n+1)/2$

We prove $S(0)$ holds

We prove that $S(n)$ implies $S(n+1)$

We deduce that $S(1),\ S(2),\ S(3), \ldots$ hold and more generally $S(n)$ holds for *all* n

Proofs by Induction

Proposition: *If $A \subseteq \mathbb{N}$ and A does not have a least element then $A = \emptyset$*

Assume that A has no least element

Let $S(n)$ be that, forall $a \in A$ we have $n < a$

We prove $S(0)$ holds: if $0 \in A$ then 0 is the least element of A

We prove that $S(n)$ implies $S(n+1)$. We *assume* $S(n)$. If $n+1 \in A$ then $n+1$ is the least element of A

We deduce that $S(1),\ S(2),\ S(3), \ldots$ hold and more generally $S(n)$ holds for *all* n. This implies $A = \emptyset$

Any nonempty subset of $\mathbb{N}$ has a least element

Proofs by Induction

Proposition: *If $n \geq 8$ then n can be written as a sum of 3's and 5's*

Let $S(n)$ be "n can be written as a sum of 3's and 5's".

$S(7)$ does not hold. But $S(8), S(9), S(10)$ hold.

Let $T(n)$ be "$S(k)$ hold for $k = 8, 9, \ldots, n$"

We prove $T(n) \Rightarrow T(n+1)$ for $n \geq 10$

If $T(n)$ holds then $S(n-2)$ holds and so does $S(n+1)$.

Proofs by Induction

All horses have the same color

$P(n)$: for any set of n horses they are all of the same color

$P(1)$ is clearly true

We claim that $P(n)$ implies $P(n+1)$

Take $h_1, \ldots, h_n$ they are all of the same color

Also $h_2, \ldots, h_{n+1}$. Hence $h_1, \ldots, h_{n+1}$ all have the same color!

Proof by Mutual Induction

One can represent a *circuit* as a set of functions from natural numbers to $\{0,1\}$ defined recursively

For instance

$f(0) = 0,\ g(0) = 1,\ h(0) = 0$

$f(n+1) = g(n),\ g(n+1) = f(n),\ h(n+1) = 1 - h(n)$

Proposition: *We have* $h(n) = f(n)$ *for all* n

If $S(n)$ is $h(n) = f(n)$ it does not seem possible to prove $S(n) \Rightarrow S(n+1)$ directly

Proof by Mutual Induction

We prove, by induction on n the statement $T(n)$

$h(n) = f(n) \ \wedge \ h(n) = 1 - g(n)$

BASIS: $h(0) = f(0) \ \wedge \ h(0) = 1 - g(0)$

STEP: $T(n) \Rightarrow T(n+1)$

One needs to *strengthen* the statement $S(n)$ to the statement $T(n)$

Proof by Mutual Induction

This can be represented as a state machine

The states are the possible values of $s(n) = (f(n), g(n), h(n))$

The transitions are from the states $s(n)$ to the state $s(n+1)$

One can check the invariant $f(n) = h(n)$ on all the states *accessible* from the initial state $(0, 1, 0)$.

Proofs by Induction

In *mathematics*, this is almost the only form of induction that is used

In *computer science*, proofs by induction play a more important rôle

Other *data types* than natural numbers: lists, trees, . . .

Notion of *inductively defined sets* (that we shall see later in the course)

Other data types

Finitely branching trees

Basis: the empty tree $()$ is a tree

Inductive step: if we have a finite list of trees $t_1, \ldots, t_k$ we can form a new tree $(t_1, \ldots, t_k)$

We can then *define* functions on the set of trees by induction, and *prove* properties of these functions by induction

Other data types

We can represent graphically the trees like in 1.4.3 and define the functions $ne(t)$ (number of *edges*) and $nn(t)$ (number of *nodes*)

$$ne() = 0, \quad ne(t_1, \ldots, t_k) = k + ne(t_1) + \cdots + ne(t_k)$$

$$nn() = 1, \quad nn(t_1, \ldots, t_k) = 1 + nn(t_1) + \cdots + nn(t_k)$$

Proposition: *for all tree* t *we have* $nn(t) = 1 + ne(t)$

Proof by *induction* with *Basis* case and *Inductive step* case

Other example

We define the function

$rev() = (),\ rev(t_1, \ldots, t_k) = (rev(t_k), \ldots, rev(t_1))$

Proposition: *for all tree* t *we have* $rev(rev(t)) = t$

We prove

Basis: $P()$

Inductive step: $P(t_1, \ldots, t_k)$ follow from $P(t_1), \ldots, P(t_k)$

Other data types

Abstract syntax of a language

Arithmetical expression E

Basis: if n natural number then $n \in E$

Inductive step: if $e_1, e_2 \in E$ then $minus(e_1),\ plus(e_1, e_2),\ times(e_1, e_2) \in E$

We can then define the *semantics* of an arithmetical expression by induction

$s(n) = n,\quad s(minus(e)) = -s(e),\quad s(plus(e_1, e_2)) = s(e_1) + s(e_2),\ s(times(e_1, e_2)) = s(e_1) \times s(e_2)$

Central concepts: alphabet and words

Σ given finite set

Alphabet finite set of symbols (events) Σ

String (or *word*, or *trace*: finite sequence of symbols (behaviour)

type convention: $a, b, c, \ldots$ for symbols (events) and $x, y, z, \ldots$ for strings (words)

Words

Σ^* is the set of all words for a given alphabet Σ

This can be described inductively in at least two different ways

Basis: the empty word ϵ is in Σ^*

Inductive step: if $a \in \Sigma$ and $x \in \Sigma^*$ then $ax \in \Sigma^*$

Words

The other description is

Basis: the empty word ϵ is in Σ^*

Inductive step: if $a \in \Sigma$ and $x \in \Sigma^*$ then $xa \in \Sigma^*$

We can *define* functions and *prove* properties of these functions by induction

Length

The length function is defined by

Basis: $|\epsilon| = 0$

Inductive step $|ax| = 1 + |x|$

$|p_0 p_1 p_0 p_0 p_1| = 5$

Concatenation

The *concatenation* function xy is defined by

Basis: $\epsilon y = y$

Inductive step: $(ax)y = a(xy)$

Proposition: *for all* x, y *we have* $|xy| = |x| + |y|$

Example: if $x = p_0 p_1$ and $y = p_0 p_0 p_1$ then

$xy = p_0 p_1 p_0 p_0 p_1$ and $yx = p_0 p_0 p_1 p_0 p_1$

In general $xy \neq yx$: concatenation is not commutative

Concatenation

Proposition: *for all* x *we have* $x\epsilon = \epsilon x = x$

Proposition: *for all* x, y, z *we have* $x(yz) = (xy)z$

We write it simply xyz

Power

We define x^n by

$x^0 = \epsilon$ and $x^{n+1} = x^n x$

We define it by induction on n

For instance $(p_0 p_1)^3 = p_0 p_1 p_0 p_1 p_0 p_1$

Languages

Given an alphabet Σ

A *language* is simply a *subset* of Σ^*

Common languages, programming languages, can be seen as sets of words

A language can be finite or infinite

Reverse functions

Intuitively $rev(a_1 \ldots a_n) = a_n \ldots a_1$

We can define $rev(x)$ by induction

$rev(\epsilon) = \epsilon$

$rev(ax) = rev(x)a$

Lemma: $rev(xy) = rev(y)rev(x)$

Some terminology

x is a *prefix* of y iff there exists z such that $y = xz$

x is a *suffix* of y iff there exists z such that $y = zx$

x is a *palindrome* iff $x = rev(x)$

A proof by induction

Proposition: *If* $x = z^k$ *and* $y = z^l$ *then* $xy = yx = z^{k+l}$

Theorem: *We have* $xy = yx$ *iff there exists* z, k, l *such that* $x = z^k$ *and* $y = z^l$

Exercice: What are the words x such that there exists y such that $x^3 = y^2$

Function bewteen languages

We consider functions $f : \Sigma^* \rightarrow \Theta^*$ such that

$f(\epsilon) = \epsilon$

$f(xy) = f(x)f(y)$

If $x = a_1 \ldots a_k$ we have $f(x) = f(a_1) \ldots f(a_k)$

Such a function f is a *coding* iff f is *injective*

Example: file compression