First Order Logic

COMP1600 / COMP6260

Dirk Pattinson Victor Rivera
Australian National University

Semester 2, 2021

First Order Natural Deduction: Example

1	$\forall x(\text{elephant}(x) \to \text{happy}(x))$	
2	$\text{elephant}(\text{Appu})$	
3	$\text{elephant}(\text{Appu}) \to \text{happy}(\text{Appu})$	$\forall$-E, 1
4	$\text{happy}(\text{Appu})$	$\to$-E, 2, 3

Natural deduction in first-order logic

Proof rules for **propositional** natural deduction + **quantifier rules**:

- ∀-E *universal elimination*;
- ∀-I *universal introduction*;
- ∃-E *existential elimination*;
- ∃-I *existential introduction*;

Proof in first-order logic is usually based on these rules, together with the rules for propositional logic.

Elimination

∀-E (universal elimination)

$$\frac{\forall x.\ P(x)}{P(a)}$$

$$\begin{array}{l|ll}
n & \forall x.\ \mathrm{Fish(x)} \rightarrow \mathrm{HasFins(x)} & \\
\vdots & \vdots & \\
m & \mathrm{Fish(a)} \rightarrow \mathrm{HasFins(a)} & \forall\text{-E},\ n
\end{array}$$

If a predicate is true for all members of a domain, then it is also true for a specific one (a must be a member of the domain).

Introduction

$\forall$-I (universal introduction) $\dfrac{P(a) \qquad (a \text{ arbitrary, a variable})}{\forall x.\ P(x)}$

n	a	$\vdots$
$\vdots$		$\vdots$
m		$\mathrm{Cat(a)} \to \mathrm{EatsFish(a)}$
$m+1$	$\forall x.\ \mathrm{Cat(x)} \to \mathrm{EatsFish(x)}$	

- The a on the left of the bar is a *guard* which reminds us that:
  - this variable is local to the inner derivation, and
  - it cannot be *free* in an assumption
- It is like an "assumption" that a is an arbitrary member of the domain.
- That is, the proof from lines n to m must work for *anything* in place of a.

Free and bound variables

Bound: Every occurrence of variable x in $\forall x.\ p(x)$ and in $\exists x.\ p(x)$ is *bound*.

Free: Every occurrence of a variable that is not bound is *free*.

Example

$$(\forall x.\ \ x + y = y + x) \quad \wedge \quad (\forall x.\ \ \exists y.\ \ y > x + z)$$

Q. Which occurrences of variables are free and which are bound?

A. All occurrences of x are bound; none of z are; and just the last 2 occurrences of y are bound.
Hence the instance of z is free, as are the first two occurrences of y.

Breaching the arbitrariness requirement

When we generalise for a variable a, the same proof steps must be possible for all members of the domain.

1	$(\mathrm{Cat}(\mathrm{kitty}) \rightarrow \mathrm{HasFur}(\mathrm{kitty})) \wedge \mathrm{Cat}(\mathrm{kitty})$	
2	$\mathrm{Cat}(\mathrm{kitty}) \rightarrow \mathrm{HasFur}(\mathrm{kitty})$	$\wedge$-E, 1
3	$\mathrm{Cat}(\mathrm{kitty})$	$\wedge$-E, 1
4	$\mathrm{HasFur}(\mathrm{kitty})$	$\rightarrow$-E, 2, 3
5	$\forall x.\ \mathrm{HasFur}(x)$ WRONG	$\forall$-I, 4

WRONG because kitty appears in an assumption (step 1) (and step 4 is still in the scope of that assumption)

Example

$(\forall x \forall y P(x, y)) \vdash (\forall y \forall x P(x, y))$

1	$\forall x \forall y P(x, y)$	
2	b | a | $\forall y P(a, y)$	$\forall$-E, 1
3	$P(a, b)$	$\forall$-E, 2
4	$\forall x P(x, b)$	$\forall$-I, 3
5	$\forall y \forall x P(x, y)$	$\forall$-I, 4

Exercise: also show the reverse to get equivalence, $\dashv\vdash$

Example

$(\forall x \forall y P(x, y)) \leftrightarrow (\forall y \forall x P(x, y))$

1	$\forall x \forall y P(x, y)$	
2	b | a | $\forall y P(a, y)$	∀-E, 1
3	$P(a, b)$	∀-E, 2
4	$\forall x P(x, b)$	∀-I, 3
5	$\forall y \forall x P(x, y)$	∀-I, 4

Exercise: also show the reverse to get equivalence, $\leftrightarrow$

Example

$(\forall x \forall y P(x,y)) \leftrightarrow (\forall y \forall x P(x,y))$

1	$\forall x \forall y P(x,y)$	
2	b | a | $\forall y P(a,y)$	$\forall$-E, 1
3	$P(a,b)$	$\forall$-E, 2
4	$\forall x P(x,b)$	$\forall$-I, 3
5	$\forall y \forall x P(x,y)$	$\forall$-I, 4

Exercise: also show the reverse to get equivalence, $\leftrightarrow$

Example

$(\forall x \forall y P(x,y)) \leftrightarrow (\forall y \forall x P(x,y))$

1				$\forall x \forall y P(x,y)$	
2	b	a		$\forall y P(a,y)$	$\forall$-E, 1
3				$P(a,b)$	$\forall$-E, 2
4			$\forall x P(x,b)$		$\forall$-I, 3
5		$\forall y \forall x P(x,y)$			$\forall$-I, 4

Exercise: also show the reverse to get equivalence, $\leftrightarrow$

Introduction

∃-I (existential introduction)

$$\frac{P(a)}{\exists x P(x)}$$

n	$\mathrm{Dog}(\mathrm{fido})$	
$\vdots$	$\vdots$	
m	$\exists x \mathrm{Dog}(x)$	∃-I, n

An invalid argument

This argument is invalid if the domain is empty.

1	$\forall x P(x)$	
2	$P(a)$	$\forall$-E, 1
3	$\exists x P(x)$	$\exists$-I, 2

Which step is invalid ??

Elimination

$$\dfrac{\exists x P(x) \qquad \begin{array}{c}[P(a)]\\ \vdots \\ q\end{array} \qquad (a \text{ arbitrary, } a \text{ variable})}{q \quad (a \text{ not free in } q)}$$

Rationale: if $P(x)$ holds for some individual x,

- *let* that individual be called a (so $P(a)$ holds)
- *prove* that q follows
- as q doesn't involve our choice of a,
  q holds regardless of which individual has P true

The proof of q from $P(a)$ must work for *any* individual in place of a

Example

Prove $$\frac{\exists x \mathrm{Elephant}(x) \qquad \forall x \mathrm{Elephant}(x) \to \mathrm{Huge}(x)}{\exists x \mathrm{Huge}(x)}$$

1	$\exists x \mathrm{Elephant}(x)$	
2	$\forall x \mathrm{Elephant}(x) \to \mathrm{Huge}(x)$	
3	a $\mathrm{Elephant}(a)$	
4	$\mathrm{Elephant}(a) \to \mathrm{Huge}(a)$	$\forall$-E, 2
5	$\mathrm{Huge}(a)$	$\to$-E, 3, 4
6	$\exists x \mathrm{Huge}(x)$	$\exists$-I, 5
7	$\exists x \mathrm{Huge}(x)$	$\exists$-E, 1, 3–6

The notation reflects an assumption: since there is some individual x such that $\mathrm{Elephant}(x)$, *assume* that individual is a

Example

Prove $\dfrac{\exists x \mathrm{Elephant}(x) \qquad \forall x \mathrm{Elephant}(x) \to \mathrm{Huge}(x)}{\exists x \mathrm{Huge}(x)}$

1		$\exists x \mathrm{Elephant}(x)$	
2		$\forall x \mathrm{Elephant}(x) \to \mathrm{Huge}(x)$	
3	a	$\mathrm{Elephant}(a)$	
4		$\mathrm{Elephant}(a) \to \mathrm{Huge}(a)$	$\forall$-E, 2
5		$\mathrm{Huge}(a)$	$\to$-E, 3, 4
6		$\exists x \mathrm{Huge}(x)$	$\exists$-I, 5
7		$\exists x \mathrm{Huge}(x)$	$\exists$-E, 1, 3–6

The notation reflects an assumption: since there is some individual x such that $\mathrm{Elephant}(x)$, *assume* that individual is a

Example

Prove $\dfrac{\exists x \mathrm{Elephant}(x) \qquad \forall x \mathrm{Elephant}(x) \rightarrow \mathrm{Huge}(x)}{\exists x \mathrm{Huge}(x)}$

1		$\exists x \mathrm{Elephant}(x)$	
2		$\forall x \mathrm{Elephant}(x) \rightarrow \mathrm{Huge}(x)$	
3	a	$\mathrm{Elephant}(a)$	
4		$\mathrm{Elephant}(a) \rightarrow \mathrm{Huge}(a)$	$\forall$-E, 2
5		$\mathrm{Huge}(a)$	$\rightarrow$-E, 3, 4
6		$\exists x \mathrm{Huge}(x)$	$\exists$-I, 5
7		$\exists x \mathrm{Huge}(x)$	$\exists$-E, 1, 3–6

The notation reflects an assumption: since there is some individual x such that $\mathrm{Elephant}(x)$, *assume* that individual is a

Example

Prove $\dfrac{\exists x \mathrm{Elephant}(x) \qquad \forall x \mathrm{Elephant}(x) \rightarrow \mathrm{Huge}(x)}{\exists x \mathrm{Huge}(x)}$

1	$\exists x \mathrm{Elephant}(x)$	
2	$\forall x \mathrm{Elephant}(x) \rightarrow \mathrm{Huge}(x)$	
3	a $\quad$ $\mathrm{Elephant}(a)$	
4	$\quad$ $\mathrm{Elephant}(a) \rightarrow \mathrm{Huge}(a)$	$\forall$-E, 2
5	$\quad$ $\mathrm{Huge}(a)$	$\rightarrow$-E, 3, 4
6	$\quad$ $\exists x \mathrm{Huge}(x)$	$\exists$-I, 5
7	$\exists x \mathrm{Huge}(x)$	$\exists$-E, 1, 3–6

The notation reflects an assumption: since there is some individual x such that $\mathrm{Elephant}(x)$, *assume* that individual is a

Swapping the order of existential quantifiers

$(\exists x \exists y P(x, y)) \leftrightarrow (\exists y \exists x P(x, y))$

1	$\exists x \exists y P(x, y)$	
2	a $\quad \exists y P(a, y)$	
3	b $\quad P(a, b)$	
4	$\exists x P(x, b)$	$\exists$-I, 3
5	$\exists y \exists x P(x, y)$	$\exists$-I, 4
6	$\exists y \exists x P(x, y)$	$\exists$-E, 2, 3–5
7	$\exists y \exists x P(x, y)$	$\exists$-E, 1, 2–6

Exercise: also show the converse to get equivalence, $\leftrightarrow$

Swapping the order of existential quantifiers

$(\exists x \exists y P(x,y)) \leftrightarrow (\exists y \exists x P(x,y))$

1		$\exists x \exists y P(x,y)$	
2	a	$\exists y P(a,y)$	
3	b	$P(a,b)$	
4		$\exists x P(x,b)$	$\exists$-I, 3
5		$\exists y \exists x P(x,y)$	$\exists$-I, 4
6		$\exists y \exists x P(x,y)$	$\exists$-E, 2, 3–5
7		$\exists y \exists x P(x,y)$	$\exists$-E, 1, 2–6

Exercise: also show the converse to get equivalence, $\leftrightarrow$

Swapping the order of existential quantifiers

$$(\exists x \exists y P(x,y)) \leftrightarrow (\exists y \exists x P(x,y))$$

1		$\exists x \exists y P(x,y)$	
2	a	$\exists y P(a,y)$	
3	b	$P(a,b)$	
4		$\exists x P(x,b)$	$\exists$-I, 3
5		$\exists y \exists x P(x,y)$	$\exists$-I, 4
6		$\exists y \exists x P(x,y)$	$\exists$-E, 2, 3–5
7		$\exists y \exists x P(x,y)$	$\exists$-E, 1, 2–6

Exercise: also show the converse to get equivalence, $\leftrightarrow$

Swapping the order of existential quantifiers

$(\exists x \exists y P(x,y)) \leftrightarrow (\exists y \exists x P(x,y))$

1	$\exists x \exists y P(x,y)$	
2	$a \mid \exists y P(a,y)$	
3	$b \mid P(a,b)$	
4	$\exists x P(x,b)$	$\exists$-I, 3
5	$\exists y \exists x P(x,y)$	$\exists$-I, 4
6	$\exists y \exists x P(x,y)$	$\exists$-E, 2, 3–5
7	$\exists y \exists x P(x,y)$	$\exists$-E, 1, 2–6

Exercise: also show the converse to get equivalence, $\leftrightarrow$

Swapping the order of existential and universal quantifiers

Proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

1			$\exists x \forall y P(x,y)$	
2	b	a	$\forall y P(a,y)$	
3			$P(a,b)$	$\forall$-E, 2
4			$\exists x P(x,b)$	$\exists$-I, 3
5		$\exists x P(x,b)$		$\exists$-E, 1, 2–4
6	$\forall y \exists x P(x,y)$			$\forall$-I, 5

Swapping the order of existential and universal quantifiers

Proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

1		$\exists x \forall y P(x,y)$	
2	b	a | $\forall y P(a,y)$	
3		$P(a,b)$	$\forall$-E, 2
4		$\exists x P(x,b)$	$\exists$-I, 3
5		$\exists x P(x,b)$	$\exists$-E, 1, 2–4
6		$\forall y \exists x P(x,y)$	$\forall$-I, 5

Swapping the order of existential and universal quantifiers

Proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

1	$\exists x \forall y P(x,y)$	
2	$b \mid a \mid \forall y P(a,y)$	
3	$P(a,b)$	$\forall$-E, 2
4	$\exists x P(x,b)$	$\exists$-I, 3
5	$\exists x P(x,b)$	$\exists$-E, 1, 2–4
6	$\forall y \exists x P(x,y)$	$\forall$-I, 5

Swapping the order of existential and universal quantifiers

Proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

1			$\exists x \forall y P(x,y)$	
2	b	a	$\forall y P(a,y)$	
3			$P(a,b)$	$\forall$-E, 2
4			$\exists x P(x,b)$	$\exists$-I, 3
5			$\exists x P(x,b)$	$\exists$-E, 1, 2–4
6			$\forall y \exists x P(x,y)$	$\forall$-I, 5

Swapping the order of existential and universal quantifiers

Another proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

We got the previous proof by first looking at the goal, $(\forall y. \ldots)$, so using $\forall$-I. Here we first look at what we have, $(\exists x. \ldots)$, and so use $\exists$-E.

1		$\exists x \forall y P(x,y)$	
2	a	$\forall y P(a,y)$	
3	b	$P(a,b)$	$\forall$-E, 2
4		$\exists x P(x,b)$	$\exists$-I, 3
5		$\forall y \exists x P(x,y)$	$\forall$-I, 4
6		$\forall y \exists x P(x,y)$	$\exists$-E, 1, 2–5

Swapping the order of existential and universal quantifiers

Another proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

We got the previous proof by first looking at the goal, $(\forall y. \ \ldots)$, so using $\forall$-I. Here we first look at what we have, $(\exists x. \ \ldots)$, and so use $\exists$-E.

1		$\exists x \forall y P(x,y)$	
2	a	$\forall y P(a,y)$	
3	b	$P(a,b)$	$\forall$-E, 2
4		$\exists x P(x,b)$	$\exists$-I, 3
5		$\forall y \exists x P(x,y)$	$\forall$-I, 4
6		$\forall y \exists x P(x,y)$	$\exists$-E, 1, 2–5

Swapping the order of existential and universal quantifiers

Another proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

We got the previous proof by first looking at the goal, $(\forall y. \ldots)$, so using $\forall$-I. Here we first look at what we have, $(\exists x. \ldots)$, and so use $\exists$-E.

1		$\exists x \forall y P(x,y)$	
2	a	$\forall y P(a,y)$	
3	b	$P(a,b)$	$\forall$-E, 2
4		$\exists x P(x,b)$	$\exists$-I, 3
5		$\forall y \exists x P(x,y)$	$\forall$-I, 4
6		$\forall y \exists x P(x,y)$	$\exists$-E, 1, 2–5

Swapping the order of existential and universal quantifiers

Another proof of $\dfrac{\exists x \forall y P(x,y)}{\forall y \exists x P(x,y)}$

We got the previous proof by first looking at the goal, $(\forall y. \ldots)$, so using $\forall$-I. Here we first look at what we have, $(\exists x. \ldots)$, and so use $\exists$-E.

1		$\exists x \forall y P(x,y)$	
2	a	$\forall y P(a,y)$	
3	b	$P(a,b)$	$\forall$-E, 2
4		$\exists x P(x,b)$	$\exists$-I, 3
5		$\forall y \exists x P(x,y)$	$\forall$-I, 4
6		$\forall y \exists x P(x,y)$	$\exists$-E, 1, 2–5

Can quantifiers always be swapped?

$$\exists x \forall y \mathrm{Eats}(x, y) \quad \rightarrow \quad \forall y \exists x \mathrm{Eats}(x, y)$$

There is an animal that can eat all foods. All foods can be eaten by some animal.

$$\forall y \exists x \mathrm{Eats}(x, y) \quad \rightarrow \quad \exists x \forall y \mathrm{Eats}(x, y)$$

All foods can be eaten by some animal. There is an animal that can eat all foods.

Is this second version true? Try to prove it. What happens?

The "quantifier negation" equivalence

$$(\forall x.\ \neg P(x)) \leftrightarrow \neg(\exists x.\ P(x))$$

Prove $\dfrac{\neg(\exists x.\ P(x))}{\forall x.\ \neg P(x)}$

1		$\neg(\exists x.\ P(x))$	
2	a	$P(a)$	
3		$\exists x.\ P(x)$	$\exists$-I, 2
4		F	$\neg$E, 1, 3
5		$\neg P(a)$	$\neg$I, 2–4
6		$\forall x.\ \neg P(x)$	$\forall$-I, 5

The "quantifier negation" equivalence

$$(\forall x.\ \neg P(x)) \leftrightarrow \neg(\exists x.\ P(x))$$

Prove $\dfrac{\neg(\exists x.\ P(x))}{\forall x.\ \neg P(x)}$

1		$\neg(\exists x.\ P(x))$	
2	a	$P(a)$	
3		$\exists x.\ P(x)$	$\exists$-I, 2
4		F	$\neg$E, 1, 3
5		$\neg P(a)$	$\neg$I, 2–4
6		$\forall x.\ \neg P(x)$	$\forall$-I, 5

The "quantifier negation" equivalence

$$(\forall x.\ \neg P(x)) \ \leftrightarrow\ \neg(\exists x.\ P(x))$$

Prove $\dfrac{\neg(\exists x.\ P(x))}{\forall x.\ \neg P(x)}$

1		$\neg(\exists x.\ P(x))$	
2	a	$P(a)$	
3		$\exists x.\ P(x)$	$\exists$-I, 2
4		F	$\neg$E, 1, 3
5		$\neg P(a)$	$\neg$I, 2–4
6		$\forall x.\ \neg P(x)$	$\forall$-I, 5

The "quantifier negation" equivalence

$$(\forall x.\ \neg P(x)) \leftrightarrow \neg(\exists x.\ P(x))$$

Prove $\dfrac{\neg(\exists x.\ P(x))}{\forall x.\ \neg P(x)}$

1	$\neg(\exists x.\ P(x))$	
2	a $P(a)$	
3	$\exists x.\ P(x)$	$\exists$-I, 2
4	F	$\neg$E, 1, 3
5	$\neg P(a)$	$\neg$I, 2–4
6	$\forall x.\ \neg P(x)$	$\forall$-I, 5

The “quantifier negation” equivalence

Proof of the converse:

1	$\forall x.\ \neg P(x)$	
2	$\exists x.\ P(x)$	
3	$a \quad P(a)$	
4	$\neg P(a)$	$\forall$-E, 1
5	F	$\neg$E, 3, 4
6	F	$\exists$-E, 2, 3–6
7	$\neg(\exists x.\ P(x))$	$\neg$I, 2–7

The “quantifier negation” equivalence

Proof of the converse:

1	$\forall x.\ \neg P(x)$	
2	$\quad \exists x.\ P(x)$	
3	$\quad\quad a \quad P(a)$	
4	$\quad\quad\quad \neg P(a)$	$\forall$-E, 1
5	$\quad\quad\quad F$	$\neg$E, 3, 4
6	$\quad\quad F$	$\exists$-E, 2, 3–6
7	$\neg(\exists x.\ P(x))$	$\neg$I, 2–7

The “quantifier negation” equivalence

Proof of the converse:

1	$\forall x.\ \neg P(x)$	
2	$\quad \exists x.\ P(x)$	
3	$\quad\quad a \quad P(a)$	
4	$\quad\quad \neg P(a)$	$\forall$-E, 1
5	$\quad\quad F$	$\neg$E, 3, 4
6	$\quad F$	$\exists$-E, 2, 3–6
7	$\neg(\exists x.\ P(x))$	$\neg$I, 2–7

The “quantifier negation” equivalence

Proof of the converse:

1	$\forall x.\ \neg P(x)$	
2	$\exists x.\ P(x)$	
3	$a \quad P(a)$	
4	$\neg P(a)$	$\forall$-E, 1
5	F	$\neg$E, 3, 4
6	F	$\exists$-E, 2, 3–6
7	$\neg(\exists x.\ P(x))$	$\neg$I, 2–7

Again: Two sides of (the same?) Coin

Validity. A formula is valid (in all structures).

Provability. A formula is provable (via natural deduction).

Recall propositional Logic

- a formula is provable in natural deduction iff it's true for all truth-value assignments

Soundness. All provable formulae are valid.

(As application of proof rules maintains validity of formulae)

Completeness. All valid formulae are provable

(Difficult proof, via so-called "Henkin Models".)

Soundness and completeness is the *glue* between valid and provable.

Metalogic of first order logic

- First order natural deduction is sound and complete
- So you can find a proof of any valid statement
- But the truth-tables aren't finite — you can't actually prove or disprove a statement by truth-tables
- If there is a proof you can find it by mindlessly trying all sequences of rules of length 1, length 2, ...
- But if you don't find a proof, you haven't established anything

Small Gain

- checking for validity in *all* models discounts *infinite* models
- trying all proofs *may* yield a proof.
- First order logic is *semi-decidable* (later in the course)

Structural Induction

So Far.

- the "mechanics" of reasoning
- fully generic, applies to all sets

Now. Induction Principles

- allow us to prove properties about "special" sets
- can be thought of as additional axioms to natural deduction
- but will leave natural deduction implicit from now on.

In more detail

- Induction on the natural numbers: review
- Structural induction over Lists
- Structural induction over Trees
- The principle that: the structural induction rule for a particular data type follows from its definition

Natural Number Induction

This is the induction you already know.

To prove a property P for all natural numbers:

- Prove it for 0
- Prove that, if it is true for n it is true for $n + 1$.

The principle is usually expressed as a rule of inference:

$$\frac{P(0) \qquad \forall n. P(n) \to P(n+1)}{\forall n. P(n)}$$

It is an *additional principle* that allows us to prove facts.

Why does it Work?

The natural numbers are an *inductively defined set*:

1. 0 is a natural number;
2. If n is a natural number, so is $n + 1$;

No object is a natural number unless justified by these clauses.

From the assumptions:

$$P(0) \qquad \forall n.\ P(n) \to P(n+1)$$

we get a sequence of deductions:

$$P(0),\ P(1),\ P(2),\ P(3),\ \ldots$$

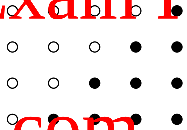
which justifies the conclusion for any n you choose:

- $P(0)$ is given
- obtain $P(0) \to P(1)$ by $(\forall E)$, and then get $P(1)$ using $(\to E)$
- obtain $P(2)$, $P(3)$, ... in the same way.

Example of Mathematical Induction

Let's prove this property of natural numbers:

$$\sum_{i=0}^{n} i = \frac{n \times (n+1)}{2}$$



First the *Base case* $P(0)$

$$\sum_{i=0}^{0} i = \frac{0 \times (0+1)}{2}$$

This is obviously true because both sides equal 0

The Step Case

The *step case.* is of the of form $\forall n. P(n) \rightarrow P(n+1)$.

Q. How do we prove a formula of this form?

A1. How would we do it in natural deduction?

1	a	$P(a)$	
2		horrendous proof	
3		$P(a+1)$	
4		$P(a) \rightarrow P(a+1)$	$\rightarrow$-I, 1–3
5	$\forall n. P(n) \rightarrow P(n+1)$		$\forall$-I, 3

The Step Case

The *step case.* is of the of form $\forall n.P(n) \rightarrow P(n+1)$.

Q. How do we prove a formula of this form?

A1. How would we do it in natural deduction?

1	a	$P(a)$	
2		horrendous proof	
3		$P(a+1)$	
4		$P(a) \rightarrow P(a+1)$	$\rightarrow$-I, 1–3
5		$\forall n.P(n) \rightarrow P(n+1)$	$\forall$-I, 3

The Step Case

The *step case.* is of the of form $\forall n.P(n) \to P(n+1)$.

Q. How do we prove a formula of this form?

A1. How would we do it in natural deduction?

1	a	$P(a)$	
2		horrendous proof	
3		$P(a+1)$	
4		$P(a) \to P(a+1)$	$\to$-I, 1–3
5		$\forall n.P(n) \to P(n+1)$	$\forall$-I, 3

The Step Case

The *step case.* is of the of form $\forall n.P(n) \to P(n+1)$.

Q. How do we prove a formula of this form?

A1. How would we do it in natural deduction?

1	a	$P(a)$	
2		horrendous proof	
3		$P(a+1)$	
4		$P(a) \to P(a+1)$	$\to$-I, 1–3
5		$\forall n.P(n) \to P(n+1)$	$\forall$-I, 3

The Step Case, Again

The *step case* is of the form $\forall n. P(n) \to P(n+1)$.

Q. How do we prove a formula of this form?

A2. What the natural deduction proof really means

- pick an arbitrary variable a
- assume that $P(a)$ and prove $P(a+1)$
- this amounts to $P(a) \to P(a+1)$
- as a was arbitrary, this amounts to $\forall n. P(n) \to P(n+1)$

How the Step Case Plays Out

Recall. Want to prove $\forall n. \underbrace{\sum_{i=0}^{n} i = \frac{n \times (n+1)}{2}}_{P(n)}$

Step case.

$$\forall n. \left(\sum_{i=0}^{n} i = \frac{n \times (n+1)}{2}\right) \to \left(\sum_{i=0}^{n+1} i = \frac{(n+1) \times (n+1+1)}{2}\right)$$

Let a be arbitrary and assume $P(a)$, i.e.

$$\text{(IH)} \qquad \underbrace{\sum_{i=0}^{a} i = \frac{a \times (a+1)}{2}}_{P(a)}$$

The assumption (IH) is called the *induction hypothesis*. Need to use it to prove $P(a+1)$.

Step Case - Detailed Proof

Assume $P(a)$, that is $\sum_{i=0}^{a} i = \dfrac{a \times (a+1)}{2}$.

Prove $P(a+1)$, that is, $\sum_{i=0}^{a+1} i = \dfrac{(a+1) \times ((a+1)+1)}{2}$

$$
\begin{aligned}
\sum_{i=0}^{a+1} i &= \sum_{i=0}^{a} i + (a+1) \\
&= \frac{a \times (a+1)}{2} + (a+1) && \text{(by IH)} \\
&= \frac{a \times (a+1)}{2} + \frac{2 \times (a+1)}{2} \\
&= \frac{(a+2) \times (a+1)}{2} \\
&= \frac{(a+1) \times (a+2)}{2}
\end{aligned}
$$

Wrapping up the proof

Recall. Proof rule for induction over natural numbers:

$$\frac{P(0) \quad \forall n. P(n) \to P(n+1)}{\forall n. P(n)}$$

We have proved *both* premises of the induction rule

- $P(0)$ and $\forall n. P(n) \to P(n+1)$

so that *applying* the rule gives $\forall n. P(n)$.

We have *demonstrated* this for a particular $P(n)$, i.e.

$$P(n) = \sum_{i=0}^{n} i = \frac{n \times (n+1)}{2}$$

in both the base case and the induction step.

Back to Programs

Q. How would we *implement* summation, e.g. in Haskell?

A. For example, like so:

```
sfz :: Int -> Int
sfz 0 = 0
sfz n = n + sfz (n-1)
```


Similarity to induction proofs

- prove that $\forall n.P(n)$
- prove that $P(0)$
- prove that $P(a) \to P(a+1)$

Slogan.

$\boxed{\textit{Recursive} \text{ definitions} \approx \textit{inductive} \text{ proofs}}$

Example: Proofs about a Program

Given. The definition of the program, in our case:

```
sfz :: Int -> Int
sfz 0 = 0                  -- SFZ0
sfz n = n + sfz (n-1)      -- SFZ1
```


Goal. To prove that $\forall n.\mathtt{sfz}(n) = \frac{1}{2}(n \times (n+1))$.

Strategy. Induction on n.

Base Case.

$$\mathtt{sfz}(0) = 0 = \frac{1}{2}(0 \times (0+1)) \qquad \text{(by SFZ0)}$$

Example: Proofs about a Program

Given. The definition of the program, in our case:

```
sfz :: Int -> Int
sfz 0 = 0                  -- SFZ0
sfz n = n + sfz (n-1)      -- SFZ1
```


Step Case. Pick an arbitrary a and assume

$$(\mathrm{IH}) \qquad \mathtt{sfz}\ a = \frac{1}{2}(a \times (a+1))$$

Goal. Show that $\mathtt{sfz}(a+1) = \frac{1}{2}((a+1)(a+1+1))$.

$$\begin{aligned}
\mathtt{sfz}(a+1) &= (a+1) + \mathtt{sfz}(a+1-1) && \text{(by SFZ1)} \\
&= (a+1) + \mathtt{sfz}(a) && \text{(by arithmetic)} \\
&= (a+1) + \frac{1}{2}(a \times (a+1)) && \text{(by IH)} \\
&= \frac{1}{2}((a+1) \times (a+1+1)) && \text{(arithmetic, see before)}
\end{aligned}$$

Basic Anatomy of an Induction Proof

Base Case $(n = 0)$.

- usually trivial
- uses base case of recursive definition

Step Case $a \to a + 1$

- assume the property for an arbitrary a – the *induction hypothesis* (IH)
- massage the goal (the property for $a + 1$) so that (IH) applies
- this usually uses the recursive step in the definition
- apply (IH) to prove the step case – the property for $a + 1$

Justification.

- simple facts (e.g. arithmetic) can be justified by saying just that
- applied equations need to be justified explicitly.

Why do we care?

Program Correctness.

- have formal *proof* that a function computes what it should
- function is *operational* whereas property is *descriptive*
- two different angles on the same function.

Optimisation.

- given: slow implementation of a function – say `slow`
- hypothesis: faster implementation – say `fast` does the same job
- proof of $\forall n.\mathtt{slow}(n) = \mathtt{fast}(n)$ allows us to swap `slow` for `fast`

Another Example

Given.


```
sumodd :: Int -> Int
sumodd 0 = 0
sumodd n = (2 * n - 1) + sumodd (n-1)
```


Q. What does this function do?

Answer. It computes the square of n, for $n \geq 0$.

Inductive Proof of sumodd

Given.


```
sumodd :: Int -> Int
sumodd 0 = 0                                    -- SO1
sumodd n = (2 * n - 1) + sumodd (n-1)           -- SO2
```


Goal. $\forall n.\texttt{sumodd}\ n = n^2$.

Base Case. Show that sumodd $0 = 0^2$.

$$\texttt{sumodd}\ 0 = 0 = 0^2 \text{ (by SO1 and arithmetic)}$$

Inductive Proof of `sumodd`

Given.

```
sumodd :: Int -> Int
sumodd 0 = 0                              -- S01
sumodd n = (2 * n - 1) + sumodd (n-1)     -- S02
```


Step Case.

- assume (IH) : $\texttt{sumodd}\ a = a^2$
- prove that $\texttt{sumodd}\ (a+1) = (a+1)^2$.

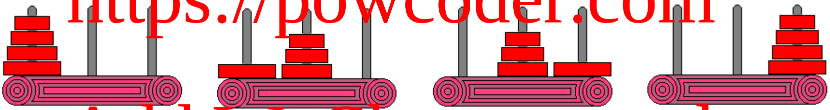
$$
\begin{aligned}
\texttt{sumodd}\ (a+1) &= 2*(a+1) - 1 + \texttt{sumodd}\ (a+1-1) && \text{(by S02)}\\
&= 2a + 1 + \texttt{sumodd}\ (a) && \text{(arithmetic)}\\
&= 2a + 1 + a^2 && \text{(by IH)}\\
&= (a+1)^2 && \text{(arithmetic)}
\end{aligned}
$$

Optimisation Example: Towers of Hanoi

Rules.

- three poles with disks of varying sizes
- larger disks may *never* be on top of smaller ones
- may only move one disk at a time.

Q. How many moves to get n discs from pole one to pole 3?



A. Here's a program!

```
t :: Int -> Int
t 0 = 0
t n = t (n-1) + 1 + t (n-1)
```

Critique 1: This is super inefficient

Compare the two programs:

```
t :: Int -> Int
t 0 = 0
t n = t (n-1) + 1 + t (n-1)
```

```
tb :: Int -> Int
tb 0 = 0
tb n = 2*tb (n-1) + 1
```

Clearly the left one is bogged down by *two* identical recursive calls!

Show that $\forall a.\texttt{t}\ (a) = \texttt{tb}\ (a)$ by induction:

Base Case. $\texttt{t}\ (0) = 0 = \texttt{tb}\ (0)$

Step Case. If $\texttt{t}\ (a) = \texttt{tb}\ (a)$, then $\texttt{t}\ (a+1) = \texttt{tb}\ (a+1)$.

$$
\begin{aligned}
\texttt{t}\ (a+1) &= \texttt{t}\ (a) + 1 + \texttt{t}\ (a) && \text{(def'n of t)} \\
&= 2 * \texttt{t}\ (a) + 1 && \text{(arith)} \\
&= 2 * \texttt{tb}\ (a) + 1 && \text{(IH)} \\
&= \texttt{tb}\ (a+1) && \text{(def'n of tb)}
\end{aligned}
$$

Critique 2: Even `tb` is not tail recursive

Compare the two programs:


```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb (n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```



```
tt :: Int -> Int
tt n = ta n 0
```


Observation. `tt` is even better (faster) than `tb`. Let's see if we can prove it . . .

Goal. $\forall n. \texttt{tb}\ (n) = \texttt{tt}\ (n)$.

Health Warning

The following slides contain lots of attempts of failed proofs.

- don't be discouraged: the example is difficult by design
- it's intended to demonstrate how things can be fixed

Proof Take 1: Let's just do it!

```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb(n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```



```
tt :: Int -> Int
tt n = ta n 0
```

Base Case. $tb\ (0) = 0$ (def'n of `tb`) $=$ `ta 0 0` (def'n of `ta`) $=$ `tt 0` (def'n of `tt`)

Step Case. Assume that `tb` $(n) =$ `tt` (n), prove `tb` $(n+1) =$ `tt` $(n+1)$

`tb` $(n+1)$	$= 2 *$ `tb` $(n) + 1$	(def'n of `tb`)
	$= 2 *$ `tt` $(n) + 1$	(IH)
	$= 2 *$ `ta` $n\ 0$	(def'n of `tt`)
	$= ???$	(we're stuck!)
	$=$ `ta` $(n+1)\ 0$	
	$= tt\ (n+1)$	(def'n of `tt`)

Analysis of Failure

```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb(n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```



```
tt :: Int -> Int
tt n = ta n 0
```


Step Case. Assume that tb (n) = tt (n), prove tb $(n+1)$ = tt $(n+1)$

Failure. We *couldn't* go

- from $2 *$ ta n 0 (which we have obtained by applying IH)
- to ta $(n+1)$ (which is equal to tt $(n+1)$)

Analysis.

- the recursion *really* happens in ta
- so maybe need a statement that relates tb and ta?

Proof Take 2: Relate ta and tb

```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb(n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```



```
tt :: Int -> Int
```

tt n = ta n 0

Show. $\forall n.$tb $(n) =$ ta (n) (0).

Base Case. Left as exercise ☺

Step Case. Assume tb $n =$ ta n 0, prove tb $(n+1) =$ ta $(n+1)$ 0

tb $(n+1)$	$= 2 *$ tb $(n) + 1$	(def'n of tb)
	$= 2 *$ ta n $0 + 1$	(IH)
	$=$???	(stuck again!)
	$=$ ta n $(2 * 0 + 1)$	
	$=$ ta $(n+1)$ 0	(def'n of ta)

Analysis of Failure, Again . . .


```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb(n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```



```
tt :: Int -> Int
tt n = ta n 0
```

We wanted. $2 * \texttt{ta}\ n\ 0 + 1 = \texttt{ta}\ n\ (2 * 0 + 1)$.

Problem. The second argument of ta changes!

Solution. Find a property that involves the second argument of ta.

Experiments

```
tb 3   =  7
ta 3 0 =  7   ta 3 1 = 15   ta 3 2 = 23   ta 3 3 = 31

tb 4   = 15
ta 4 0 = 15   ta 4 1 = 31   ta 4 2 = 47   ta 4 3 = 63
```


Wild Guess. How about ta n $a = (\mathrm{tb}\ n) + a * (\mathrm{tb}\ n+1)$??

This would give

$$\mathrm{tb}\ n = (\mathrm{tb}\ n) + 0 * (\mathrm{tb}\ n+1) = \mathrm{ta}\ n\ 0 = \mathrm{tt}\ 0$$

so would solve our problem.

Proof Take 3: Stronger Property

```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb (n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```

```
tt :: Int -> Int
tt n = ta n 0
```


Show. $\forall n.\forall a.\texttt{ta}\ n\ a = \texttt{tb}\ n + a * (\texttt{tb}\ n + 1)$.

Base Case.

$$\begin{aligned} \texttt{ta}\ 0\ a &= a && \text{(def'n ta)} \\ &= 0 + a * (0 + 1) && \text{(arith)} \\ &= (\texttt{tb}\ 0) + a * ((\texttt{tb}\ 0) + 1) && \text{(def'n of tb)} \end{aligned}$$

so base case still works.

Proof Take 3: Stronger Property

```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb (n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```



```
tt :: Int -> Int
tt n = ta n 0
```

Step Case

- Assume `ta` $n\ a =$ `tb` $n + a * ($`tb` $n + 1)$
- Show that `ta` $(n+1)\ a =$ `tb` $(n+1) + a * ($`tb` $(n+1) + 1)$

$$
\begin{aligned}
\texttt{ta}\ (n+1)\ a &= \texttt{ta}\ n\ (2*a+1) && \text{(def'n of ta)}\\
&= \texttt{tb}\ n + (2*a+1)(\texttt{tb}\ n+1) && \text{(IH)}\\
&= 2*\texttt{tb}\ n+1+2*a*(mathtttb\ n+1) && \text{(lots of arith)}\\
&= \texttt{tb}\ (n+1)+a*(\texttt{tb}\ (n+1)+1) && \text{(def'n of tb)}
\end{aligned}
$$

so step case also works!

Finally: Wrapping Up!

```
tb :: Int -> Int
tb 0 = 0
tb n = 2 * tb(n-1) + 1
```

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```

```
tt ::  Int -> Int
tt n = ta n 0
```


Show. $\forall n.\texttt{tb}\ n = \texttt{tt}\ n$.

$\texttt{tt}\ n$	$= \texttt{ta}\ n\ 0$	(def'n of tt)
	$= (\texttt{tb}\ n) + 0 * (\texttt{tb}\ n + 1)$	(we have now!)
	$= \texttt{tb}\ n$	(arith)

Conceptual Digression

```
ta :: Int -> Int -> Int
ta 0 a = a
ta n a = ta (n-1) (2 * a + 1)
```


Changing Arguments.

- ta is a *two-place* (binary) function
- recursion is on *first* argument (n)
- *but* second argument (a) is *not* constant!

Solution.

- find a *stronger* property that involves the second argument
- usually: universally quantified

Example.

$$P(n) = \forall a.\texttt{ta}\ n\ a = \texttt{tb}\ n + a * (\texttt{tb}\ n + 1)$$

- as a is universally quantified, property holds for *all* a
- even if a changes in recursive call!