First Order Logic

COMP1600 / COMP6260

Semester 2, 202

First Order Natural Deduction: Example

3	elephant(Appu) $\rightarrow$ happy(Appu)	-E, 1
4	happy(Appu)	

Natural deduction in first-order logic

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

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

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

Elimination

∀-E (universal elimination)

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

$$\begin{array}{r|l} \vdots & \vdots \\ m & \mathrm{Fish}(a) \to \mathrm{HasFins}(a \end{array}$$

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

Introduction

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

```
n   | a |  ⋮
⋮   |   |  ⋮
    |
    |  ∀        →
```

- The *a* on the left of the bar is a *guar*
  - 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*.

Exampl

)

Q. Which occurrences of variables are free and which

A. All occurrences of x are bound; none of 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	(Cat(kitty) HasFur(kitty)) Cat(kitty)	
2		∧-E, 1
3		∧-E, 1
4	HasFur(kitty)	, 2, 3
5	∀*x*. HasFur(x) WRONG	

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

Example

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

1 $x \quad y P(x, y)$

4 $\forall x P(x, b)$

5 $\forall y \forall x P(x, y)$

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 $x \quad y P(x, y)$

4 $\forall x P(x, b)$

5 $\forall y \forall x P(x, y)$

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 $\quad x \;\; yP(x,y)$

4 $\quad \forall x P(x,b)$

5 $\quad \forall y \forall x P(x,y)$

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 $x \;\; y P(x,y)$

4 $\forall x P(x,b)$

5 $\forall y \forall x P(x,y)$

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

Introduction

∃-I (existential introduction)

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

An invalid argument

This argument is invalid if the domain is empty.

3 | $\exists x P(x)$

Which step is invalid ??

Elimination

$$\begin{array}{c}[P(a)] \\ \vdots \\ \hline \end{array}$$

Rational

- *let* that individual be called a (so $P(a$
- *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 $\dfrac{\exists x \mathrm{Elephant}(x) \qquad \forall x \mathrm{Elephant}(x) \to \mathrm{Huge}(x)}{\exists x \mathrm{Huge}(x)}$

1 $\exists x \mathrm{Elephant}(x)$

2 $x \mathrm{Elephant}(x) \quad \mathrm{Huge}(x)$

5 $\mathrm{Huge}(a)$

6 $\exists x \mathrm{Huge}(x)$

7 $\exists x \mathrm{Huge}(x)$

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 $x \mathrm{Elephant}(x) \qquad \mathrm{Huge}(x)$

5 $\mathrm{Huge}(a)$

6 $\exists x \mathrm{Huge}(x)$

7 $\exists x \mathrm{Huge}(x)$

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

Example

Prove $$\frac{\exists x \text{Elephant}(x) \qquad \forall x \text{Elephant}(x) \rightarrow \text{Huge}(x)}{\exists x \text{Huge}(x)}$$

1 $\exists x \text{Elephant}(x)$

2 $x \text{Elephant}(x) \quad \text{Huge}(x)$

[illegible]

5 $\text{Huge}(a)$

6 $\exists x \text{Huge}(x)$

7 $\exists x \text{Huge}(x)$

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

5 $\mathrm{Huge}(a)$

6 $\exists x \mathrm{Huge}(x)$

7 $\exists x \mathrm{Huge}(x)$

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

$\exists$,

5 $\exists y \exists x P(x,y)$

6 $\exists y \exists x P(x,y)$

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	$\exists x \exists y P(x,y)$	
	$\exists$,	
5	$\exists y \exists x P(x,y)$	
6	$\exists y \exists x P(x,y)$	
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 | ∃x∃yP(x,y)
  |----------
  | |
  | |----------
  | | |
  | | |----------
  | | | ∃        ,
5 | | | ∃y∃xP(x,y)
6 | | ∃y∃xP(x,y)
7 | ∃y∃xP(x,y)                  ∃-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 $\exists x \exists y P(x,y)$

$\exists$,

5 $\exists y \exists x P(x,y)$

6 $\exists y \exists x P(x,y)$

7 $\exists y \exists x P(x,y)$ ∃-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	$x \quad y P(x, y)$	
4	$\exists x P(x, b)$	
5	$\exists x P(x, b)$	
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	$x \quad y P(x,y)$	
4	$\exists x P(x,b)$	
5	$\exists x P(x,b)$	
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 $x \quad y P(x,y)$

4 $\exists x P(x,b)$

5 $\exists x P(x,b)$

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 $x \quad y P(x, y)$

4 $\exists x P(x, b)$

5 $\exists x P(x, b)$

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.

3	b	$P(a,b)$	
4		$\exists x P(x,b)$	
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.

3	b	$P(a,b)$	
4		$\exists x P(x,b)$	
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.

3	b $P(a,b)$	
4	$\exists x P(x,b)$	
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.

3	$b \mid P(a,b)$	
4	$\exists x P(x,b)$	
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.

$\forall y \exists$

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

3 $\exists x.\ P(x)$

4 F

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

3 $\exists x.\ P(x)$

4 F

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

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

The "quantifier negation" equivalence

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

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

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

The “quantifier negation” equivalence

Proof of the converse:

1 $\forall x.\ \neg P(x)$

5 F

6 F

7 $\neg(\exists x.\ P(x))$ $\neg$I, 2–7

The “quantifier negation” equivalence

Proof of the converse:

1 $\forall x.\ \neg P(x)$

5 F

6 F

7 $\neg(\exists x.\ P(x))$ $\neg$I, 2–7

The "quantifier negation" equivalence

Proof of the converse:

1 $\forall x.\ \neg P(x)$

5 F

6 F

7 $\neg(\exists x.\ P(x))$ $\neg$I, 2–7

The “quantifier negation” equivalence

Proof of the converse:

1 $\forall x.\ \neg P(x)$

5 F

6 F

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 for
  trut

Soundness. All provable formulae are valid.

(As application of proof rules maintains validit

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 disp
- If the rule
- But if you don't find a proof, you haven't established anything

Small Gain

- checking for validity in *all* models disco
- 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

- allo
- can b
- but

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:

- Pro
- Pro

The principle is usually expressed as a rule of inference:

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

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 a

we get a sequence of deductions:

$$P(0), P(1), P(2)$$

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:

$$\sum^{n} i = \frac{n\,(n+1)}{}$$

First the

$$\sum_{i=0}^{0} i = \frac{0 \times (}{}$$

This is obviously true because both sides equal 0

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?

2 horrendous proof

3 $P(a+1)$

4 $P(a) \to P(a+1)$

5 $\forall n.P(n) \to P(n+1)$ $\forall$-I, 3

The Step Case

The *step case.* is of the 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	[illegible]	
2	horrendous proof	
3	$P(a+1)$	
4	$P(a) \rightarrow P(a+1)$	
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?

2 horrendous proof

3 $P(a+1)$

4 $P(a) \to P(a+1)$

5 $\forall n.P(n) \to P(n+1)$ $\forall$-I, 3

The Step Case

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

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

A1. How would we do it in natural deduction?

2 horrendous proof

3 $P(a+1)$

4 $P(a) \rightarrow P(a+1)$

5 $\forall n.P(n) \rightarrow P(n+1)$ $\forall$-I, 3

The Step Case, Again

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

Q. How d

A2. Wh

- pick
- 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 $\forall$

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.

$\forall n.$ [illegible] $1))$

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

$$\text{(IH)} \qquad \underbrace{\sum_{i=0}^{a} i = \frac{a \text{ [illegible]}}{\text{[illegible]}}}_{P(a)}$$

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

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

$$= \frac{a \times (a+1)}{2} \quad (\qquad \text{(by IH)}$$

$$= \frac{a \times (a+1)}{2} \quad \frac{\qquad}{2}$$

$$= \frac{(a+2) \times (a+1)}{2}$$

$$= \frac{(a+1) \times (a+2)}{2}$$

Wrapping up the proof

Recall. Proof rule for induction over natural numbers:

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

We have pr

- $P(0$

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

We have *demonstrated* this for a particular

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

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:

s

s

s

Similarity to induction proofs

-
-
- $P(a+1)$

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 + s
```

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

Strategy. Induction on n.

Base Case.

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

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 Cas

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

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

Basic Anatomy of an Induction Proof

Base Case ($n = 0$).

- usually trivial
- uses base case of recursive definition

Step Cas

- ass *hesis* (IH)
- mas
- this usually uses the recursive step in the definitio
- apply (IH) to prove the step case – the property for

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
- fun
- two

Optimisation.

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

Another Example

Given.

s

s

s

Q. What does this function do?

Answer. It computes the square of n, for

Inductive Proof of `sumodd`

Given.

```
sumodd :: Int -> Int
s                                                   -- SO1
s
```

Goal. $\forall n.$`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                              -- SO1
sumodd n = (2 * n - 1) + sumodd (n-1)   -- SO2
```

Step Ca

- ass
- prove that sumodd $(a+1) = (a+1)^2$

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

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

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

Clearly th s!

Show that

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

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

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

Critique 2: Even tb is not tail recursive

Compare the two programs:

```
tb :: Int -> Int                    ta :: Int -> Int -> Int
tb 0 = 0
tb n = 2 * tb (
```

```
                                    tt n = ta n 0
```

Observation. tt is even better (faster) than
it . . .

Goal. $\forall n.$tb $(n) =$ tt (n).

Health Warning

- [illegible]
- it's intended to demonstrate how things can be fixed

Proof Take 1: Let's just do it!

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

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

Base Cas tt 0

(def'n o

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

$$
\begin{aligned}
\texttt{tb}\ (n+1) &= 2 * \texttt{tb}\ (n) + 1 && \text{(def'n of} \\
&= 2 * \texttt{tt}\ (n) + 1 && \text{(IH)} \\
&= 2 * \texttt{ta}\ n\ 0 && \text{(def'n of tt)} \\
&= ??? && \text{(we're stuck!)} \\
&= \texttt{ta}\ (n+1)\ 0 \\
&= tt\ (n+1) && \text{(def'n of tt)}
\end{aligned}
$$

Analysis of Failure

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

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

Step Cas tt $(n+1)$

Failure. We *couldn't* go

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

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.

Base Cas

Step Case. Assume tb n = ta n 0, prove

$$
\begin{aligned}
\text{tb}(n+1) &= 2 * \text{tb}\ (n) + 1 && \text{(def'n of}\\
&= 2 * \text{ta}\ n\ 0 + 1 && \text{(IH)}\\
&= \quad ??? && \text{(stuck again!)}\\
&= \text{ta}\ n\ (2 * 0 + 1) && \\
&= \text{ta}\ (n+1)\ 0 && \text{(def'n of ta)}
\end{aligned}
$$

Analysis of Failure, Again ...

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

```
ta :: Int -> Int -> Int
ta 0 a = a
```

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

Problem. The second argument of ta ch

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 $a = (\mathrm{tb}\ n)$

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
```

Show. [illegible]

Base Case.

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

so base case still works.

Proof Take 3: Stronger Property

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

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

Step Cas

- Ass
- Show that ta $(n+1)$ $a =$ tb $(n+1) +$

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

so step case also works!

Finally: Wrapping Up!

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

Show.

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

Conceptual Digression

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

Changing Arguments.

- ta i
- rec
- *but*

Solution.

- find a *stronger* property that involves these
- 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!