

Introduction to Logic

Induction

Michael Genesereth
Computer Science Department
Stanford University

Truth Table for Relational Logic

$$\{p(a) \vee p(b), \forall x.(p(x) \Rightarrow q(x))\} \models \exists x.q(x)?$$

$p(a)$	$p(b)$	$q(a)$	$q(b)$	$p(a) \vee p(b)$	$\forall x.(p(x) \Rightarrow q(x))$	$\exists x.q(x)$
1	1	1	1	1	1	1
1	1	1	0	1	0	1
1	1	0	1	1	0	1
1	1	0	0	1	0	0
1	0	1	1	1	1	1
1	0	1	0	1	1	1
1	0	0	1	1	0	1
1	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	0	1	0	0
0	0	1	1	0	1	1
0	0	1	0	0	1	1
0	0	0	1	0	1	1
0	0	0	0	0	1	0

Relational Logic Sizes

Object constants: n

Binary relation constants: k

Factoids in Herbrand Base: $k*n^2$

Truth Assignments: 2^{k*n^2}

Object constants: 4

Binary relation constants: 4

Factoids in Herbrand Base: 64

Truth Assignments: $2^{64} = 18,446,744,073,709,551,616$

Upshot: Truth tables impractical.

Proofs more practical.

Term Logic Sizes

Object constants: $n \geq 1$

Functions constants: $j \geq 1$

Relation constants: $k \geq 1$

Factoids in Herbrand Base: *countably infinite*

Truth Assignments: *uncountably infinite*

Upshot: Truth tables impossible.

Proofs are our only hope.

Fitch System for Relational Logic

Negation Introduction

Negation Elimination

And Introduction

And Elimination

Or Introduction

Or Elimination

Assumption

Implication Introduction

Implication Elimination

Biconditional Introduction

Biconditional Elimination

Universal Introduction

Universal Elimination

Existential Introduction

Existential Elimination

Domain Closure

Fitch System for Term Logic

Negation Introduction

Negation Elimination

And Introduction

And Elimination

Or Introduction

Or Elimination

Assumption

Implication Introduction

Implication Elimination

Biconditional Introduction

Biconditional Elimination

Universal Introduction

Universal Elimination

Existential Introduction

Existential Elimination

Domain Closure

Induction *New!!*

Overview

Induction

Induction is reasoning from the specific to the general.

If various instances of a schema are true and there are no counterexamples, we are tempted to conclude a universally quantified version of the schema.

$$\begin{array}{l} p(a) \Rightarrow q(a) \\ p(b) \Rightarrow q(b) \\ p(c) \Rightarrow q(c) \end{array} \quad \rightarrow \quad \forall x.(p(x) \Rightarrow q(x))$$

Lucky Guess

Definition

$$f(1) = 1$$

$$f(x + 1) = f(x) + 2*x + 1$$

Data

1	1	=	1	=	1 ²
2	1 + 3	=	4	=	2 ²
3	1 + 3 + 5	=	9	=	3 ²
4	1 + 3 + 5 + 7	=	16	=	4 ²
5	1 + 3 + 5 + 7 + 9	=	25	=	5 ²

Conjecture

$$f(x) = x^2$$

In this case, the answer is correct. Lucky Guess.

Not So Lucky Guess

Data:

$$2^{2^1} + 1 = 2^2 + 1 = 5$$

$$2^{2^2} + 1 = 2^4 + 1 = 17$$

$$2^{2^3} + 1 = 2^8 + 1 = 257$$

$$2^{2^4} + 1 = 2^{16} + 1 = 65537$$

“Theorem” by Fermat (1601-1665):

$$\textit{prime}(2^{2^x} + 1)$$

Fact discovered (mercifully) after his death:

$$2^{2^5} + 1 = 4,294,967,297 = 641 * 6,700,417$$

Oops.

Domain Closure

$$\frac{\begin{array}{l} \phi[\sigma_1] \\ \dots \\ \phi[\sigma_n] \end{array} \left. \vphantom{\begin{array}{l} \phi[\sigma_1] \\ \dots \\ \phi[\sigma_n] \end{array}} \right\} \begin{array}{l} \textit{every} \\ \textit{ground} \\ \textit{term} \end{array}}{\forall v. \phi[v]}$$

Finite Example

$$p(a) \Rightarrow q(a)$$

$$p(b) \Rightarrow q(b)$$

$$p(c) \Rightarrow q(c)$$

$$p(d) \Rightarrow q(d)$$

$$\forall x.(p(x) \Rightarrow q(x))$$

Infinite Case

What happens when we have infinitely many terms?

$$\begin{aligned} p(a) &\Rightarrow q(a) \\ p(s(a)) &\Rightarrow q(s(a)) \\ p(s(s(a))) &\Rightarrow q(s(s(a))) \\ p(s(s(s(a)))) &\Rightarrow q(s(s(s(a)))) \\ &\dots \end{aligned}$$

$$\forall x.(p(x) \Rightarrow q(x))$$

Types of Induction

Linear Induction

induction on sequences

Tree Induction

induction on trees

Structural Induction

induction on complex structures

Linear Induction

Linear Worlds

$$a \rightarrow s(a) \rightarrow s(s(a)) \rightarrow s(s(s(a))) \rightarrow \dots$$

Linear Worlds

Linear World:

$$a \rightarrow s(a) \rightarrow s(s(a)) \rightarrow s(s(s(a))) \rightarrow \dots$$

The object constant is called the *base element*, and the function constant is called the *successor function*.

Induction with Linear Worlds

Linear World:

$$a \rightarrow s(a) \rightarrow s(s(a)) \rightarrow s(s(s(a))) \rightarrow \dots$$

Linear Induction:

(1) If we can show our *base element* has some property

and

(2) if we can show that, whenever an arbitrary element has that property, its *successor* has that property,

then we can conclude that *every* element has that property.

Dominoes



Dominoes Example

Object constant: a

Unary function constant: s

Unary relation constant: $falls$

Dominoes Example

Object constant: a

Unary function constant: s

Unary relation constant: $falls$

Axioms:

$$falls(a)$$
$$\forall x.(falls(x) \Rightarrow falls(s(x)))$$

Dominoes Example

Object constant: a

Unary function constant: s

Unary relation constant: $falls$

Axioms:

$$falls(a)$$
$$\forall x.(falls(x) \Rightarrow falls(s(x)))$$

Conclusion:

$$\forall x.falls(x)$$





OFFICIALLY

Linear Induction

$$\begin{array}{l} \phi[a] \\ \forall x.(\phi[x] \Rightarrow \phi[s(x)]) \end{array}$$

$$\forall x.\phi[x]$$

Linear Induction

Base Case

$$\begin{array}{l} \phi[a] \\ \forall x.(\phi[x] \Rightarrow \phi[s(x)]) \\ \hline \forall x.\phi[x] \end{array}$$

Linear Induction

Base Case



$$\phi[a]$$

$$\forall x.(\phi[x] \Rightarrow \phi[s(x)])$$



$$\forall x.\phi[x]$$

Inductive Case

Linear Induction

Base Case



$\phi[a]$

$\forall x.(\phi[x] \Rightarrow \phi[s(x)])$

$\forall x.\phi[x]$

Inductive Case



Overall Conclusion

Linear Induction

Inductive Hypothesis

Base Case



$\phi[a]$



$\forall x.(\phi[x] \Rightarrow \phi[s(x)])$



$\forall x.\phi[x]$

Inductive Case



Overall Conclusion

Linear Induction

Inductive Hypothesis

Base Case

Inductive Conclusion

$\phi[a]$

$\forall x.(\phi[x] \Rightarrow \phi[s(x)])$

$\forall x.\phi[x]$

Inductive Case

Overall Conclusion

Whole Numbers

Object constant: 0

Unary function constant: s

$$0 \rightarrow s(0) \rightarrow s(s(0)) \rightarrow s(s(s(0))) \rightarrow \dots$$

Whole Numbers

Object constant: 0

Unary function constant: s

$$0 \rightarrow s(0) \rightarrow s(s(0)) \rightarrow s(s(s(0))) \rightarrow \dots$$

Unary relation constants: *even, odd*

Whole Numbers

Object constant: 0

Unary function constant: s

$$0 \rightarrow s(0) \rightarrow s(s(0)) \rightarrow s(s(s(0))) \rightarrow \dots$$

Unary relation constants: $even, odd$

Axioms:

$$even(0)$$

$$\forall x.(even(x) \Rightarrow odd(s(x)))$$

$$\forall x.(odd(x) \Rightarrow even(s(x)))$$

Whole Numbers

Object constant: 0

Unary function constant: s

$$0 \rightarrow s(0) \rightarrow s(s(0)) \rightarrow s(s(s(0))) \rightarrow \dots$$

Unary relation constants: $even, odd$

Axioms:

$$even(0)$$

$$\forall x.(even(x) \Rightarrow odd(s(x)))$$

$$\forall x.(odd(x) \Rightarrow even(s(x)))$$

Goal:

$$\forall x.(even(x) \vee odd(x))$$

Proof

1. $even(0)$ Premise
2. $\forall x.(even(x) \Rightarrow odd(s(x)))$ Premise
3. $\forall x.(odd(x) \Rightarrow even(s(x)))$ Premise

Proof

- | | | |
|----|---|---------|
| 1 | $even(0)$ | Premise |
| 2. | $\forall x.(even(x) \Rightarrow odd(s(x)))$ | Premise |
| 3. | $\forall x.(odd(x) \Rightarrow even(s(x)))$ | Premise |
| 4. | $even(0) \vee odd(0)$ | OI: 1 |

Proof

- | | | |
|----|---|------------|
| 1 | $even(0)$ | Premise |
| 2. | $\forall x.(even(x) \Rightarrow odd(s(x)))$ | Premise |
| 3. | $\forall x.(odd(x) \Rightarrow even(s(x)))$ | Premise |
| 4. | $even(0) \vee odd(0)$ | OI: 1 |
| 5. | $ even(c) \vee odd(c)$ | Assumption |

Proof

1.	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	Assumption
6.	$even(c)$	Assumption

Proof

1.	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$\left even(c) \vee odd(c) \right.$	Assumption
6.	$\left \left even(c) \right. \right.$	Assumption
7.	$\left \left even(c) \Rightarrow odd(s(c)) \right. \right.$	UE: 2
8.	$\left \left odd(s(c)) \right. \right.$	IE: 7, 6

Proof

1.	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	Assumption
6.	$even(c)$	Assumption
7.	$even(c) \Rightarrow odd(s(c))$	UE: 2
8.	$odd(s(c))$	IE: 7, 6
9.	$even(s(c)) \vee odd(s(c))$	OI: 8
10.	$even(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 6, 9

Proof

1.	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$\left \begin{array}{l} even(c) \vee odd(c) \\ \dots \end{array} \right.$	Assumption
10.	$\left even(c) \Rightarrow even(s(c)) \vee odd(s(c)) \right.$	II: 6, 9

Proof

1.	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$\left \begin{array}{l} even(c) \vee odd(c) \\ \dots \end{array} \right.$	Assumption
10.	$\left \begin{array}{l} even(c) \Rightarrow even(s(c)) \vee odd(s(c)) \end{array} \right.$	II: 6, 9
11.	$\left \left odd(c) \right. \right.$	Assumption

Proof

1	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	Assumption
	...	
10.	$even(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 6, 9
11.	$odd(c)$	Assumption
12.	$odd(c) \Rightarrow even(s(c))$	UI: 3
13.	$even(s(c))$	IE: 12, 11
14.	$even(s(c)) \vee odd(s(c))$	OI: 13
15.	$odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 11, 15

Proof

1	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	BE: 4
	...	
10.	$even(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 6,9
	...	
15.	$odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 11,14

Proof

1.	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	BE: 4
	...	
10.	$even(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 6,9
	...	
15.	$odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 11,14
16.	$even(s(c)) \vee odd(s(c))$	OE: 5, 10, 15

Proof

1	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	BE: 4
	...	
10.	$even(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 6,9
	...	
15.	$odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 11,14
16.	$even(s(c)) \vee odd(s(c))$	OE: 5, 10, 15
17.	$even(c) \vee odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 5, 16

Proof

1	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	BE: 4
	...	
10.	$even(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 6,9
	...	
15.	$odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 11,14
16.	$even(s(c)) \vee odd(s(c))$	OE: 5, 10, 15
17.	$even(c) \vee odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 5, 16
18.	$\forall x.(even(x) \vee odd(x) \Rightarrow even(s(x)) \vee odd(s(x)))$	UI: 17

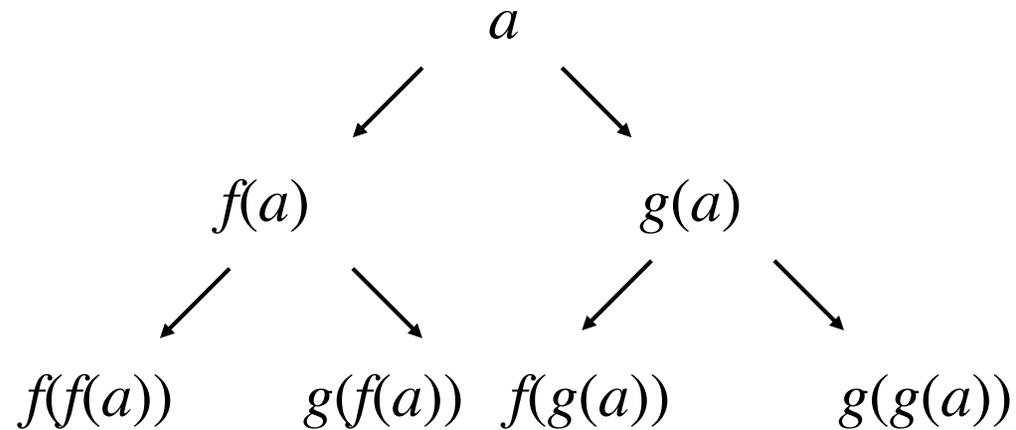
Proof

1	$even(0)$	Premise
2.	$\forall x.(even(x) \Rightarrow odd(s(x)))$	Premise
3.	$\forall x.(odd(x) \Rightarrow even(s(x)))$	Premise
4.	$even(0) \vee odd(0)$	OI: 1
5.	$even(c) \vee odd(c)$	BE: 4
	...	
10.	$even(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 6,9
	...	
15.	$odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 11,14
16.	$even(s(c)) \vee odd(s(c))$	OE: 5, 10, 15
17.	$even(c) \vee odd(c) \Rightarrow even(s(c)) \vee odd(s(c))$	II: 5, 16
18.	$\forall x.(even(x) \vee odd(x) \Rightarrow even(s(x)) \vee odd(s(x)))$	UI: 17
19.	$\forall x.(even(x) \vee odd(x))$	Ind: 4, 18

Tree Induction

Tree Languages

Tree-Like World:



Languages like this are called *tree languages*.

Tree Induction

$$\phi[a]$$
$$\forall \mu. (\phi[\mu] \Rightarrow \phi[f(\mu)])$$
$$\forall \mu. (\phi[\mu] \Rightarrow \phi[g(\mu)])$$

$$\forall v. \phi[v]$$

Canine Ancestry Example

Object constant: *rex*

Unary function constants: *f*, *g*

Unary relation constant: *p*

Canine Ancestry Example

Object constant: rex

Unary function constants: f, g

Unary relation constant: p

Axioms:

$$p(rex)$$
$$\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$$

Canine Ancestry Example

Object constant: rex

Unary function constants: f, g

Unary relation constant: p

Axioms:

$$p(rex) \\ \forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$$

Goal:

$$\forall x.p(x)$$

Canine Ancestry Proof

1. $p(\text{rex})$ Premise
2. $\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$ Premise

Canine Ancestry Proof

1. $p(\textit{rex})$ Premise
2. $\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$ Premise
3. $p(c) \Leftrightarrow p(f(c)) \wedge p(g(c))$ UE: 2
4. $p(c) \Rightarrow p(f(c)) \wedge p(g(c))$ BE: 3

Canine Ancestry Proof

- | | | |
|----|---|------------|
| 1. | $p(\text{rex})$ | Premise |
| 2. | $\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$ | Premise |
| 3. | $p(c) \Leftrightarrow p(f(c)) \wedge p(g(c))$ | UE: 2 |
| 4. | $p(c) \Rightarrow p(f(c)) \wedge p(g(c))$ | BE: 3 |
| 5. | $\left p(c) \right.$ | Assumption |

Canine Ancestry Proof

- | | | |
|----|---|------------|
| 1. | $p(\text{rex})$ | Premise |
| 2. | $\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$ | Premise |
| 3. | $p(c) \Leftrightarrow p(f(c)) \wedge p(g(c))$ | UE: 2 |
| 4. | $p(c) \Rightarrow p(f(c)) \wedge p(g(c))$ | BE: 3 |
| 5. | $\left p(c) \right.$ | Assumption |
| 6. | $\left p(f(c)) \wedge p(g(c)) \right.$ | IE: 4, 5 |

Canine Ancestry Proof

1.	$p(\text{rex})$	Premise
2.	$\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$	Premise
3.	$p(c) \Leftrightarrow p(f(c)) \wedge p(g(c))$	UE: 2
4.	$p(c) \Rightarrow p(f(c)) \wedge p(g(c))$	BE: 3
5.	$p(c)$	Assumption
6.	$p(f(c)) \wedge p(g(c))$	IE: 4, 5
7.	$p(f(c))$	AE: 6

Canine Ancestry Proof

1.	$p(\text{rex})$	Premise
2.	$\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$	Premise
3.	$p(c) \Leftrightarrow p(f(c)) \wedge p(g(c))$	UE: 2
4.	$p(c) \Rightarrow p(f(c)) \wedge p(g(c))$	BE: 3
5.	$\left \begin{array}{l} p(c) \end{array} \right.$	Assumption
6.	$\left \begin{array}{l} p(f(c)) \wedge p(g(c)) \end{array} \right.$	IE: 4, 5
7.	$\left \begin{array}{l} p(f(c)) \end{array} \right.$	AE: 6
8.	$p(c) \Rightarrow p(f(c))$	II: 5, 7
9.	$\forall x.(p(x) \Rightarrow p(f(x)))$	UI: 8

Canine Ancestry Proof

- | | | |
|-----|---|---------|
| 1. | $p(\text{rex})$ | Premise |
| 2. | $\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$ | Premise |
| 3. | $p(c) \Leftrightarrow p(f(c)) \wedge p(g(c))$ | UE: 2 |
| 4. | $p(c) \Rightarrow p(f(c)) \wedge p(g(c))$ | BE: 3 |
| | ... | |
| 9. | $\forall x.(p(x) \Rightarrow p(f(x)))$ | UI: 8 |
| | ... | |
| 14. | $\forall x.(p(x) \Rightarrow p(g(x)))$ | UI: 13 |

Canine Ancestry Proof

- | | | |
|-----|---|---------------|
| 1. | $p(\text{rex})$ | Premise |
| 2. | $\forall x.(p(x) \Leftrightarrow p(f(x)) \wedge p(g(x)))$ | Premise |
| 3. | $p(c) \Leftrightarrow p(f(c)) \wedge p(g(c))$ | UE: 2 |
| 4. | $p(c) \Rightarrow p(f(c)) \wedge p(g(c))$ | BE: 3 |
| | ... | |
| 9. | $\forall x.(p(x) \Rightarrow p(f(x)))$ | UI: 8 |
| | ... | |
| 14. | $\forall x.(p(x) \Rightarrow p(g(x)))$ | UI: 13 |
| 15. | $\forall x.p(x)$ | Ind: 1, 9, 14 |

Linear Induction

Linear Language:

$$a \rightarrow s(a) \rightarrow s(s(a)) \rightarrow s(s(s(a))) \rightarrow \dots$$

Induction:

$$\frac{\begin{array}{l} \phi[a] \\ \forall \mu. (\phi[\mu] \Rightarrow \phi[s(\mu)]) \end{array}}{\forall v. \phi[v]}$$

Multiple Object Constants

Linear Language:

$$a \rightarrow s(a) \rightarrow s(s(a)) \rightarrow s(s(s(a))) \rightarrow \dots$$

$$b \rightarrow s(b) \rightarrow s(s(b)) \rightarrow s(s(s(b))) \rightarrow \dots$$

Induction:

$$\phi[a]$$

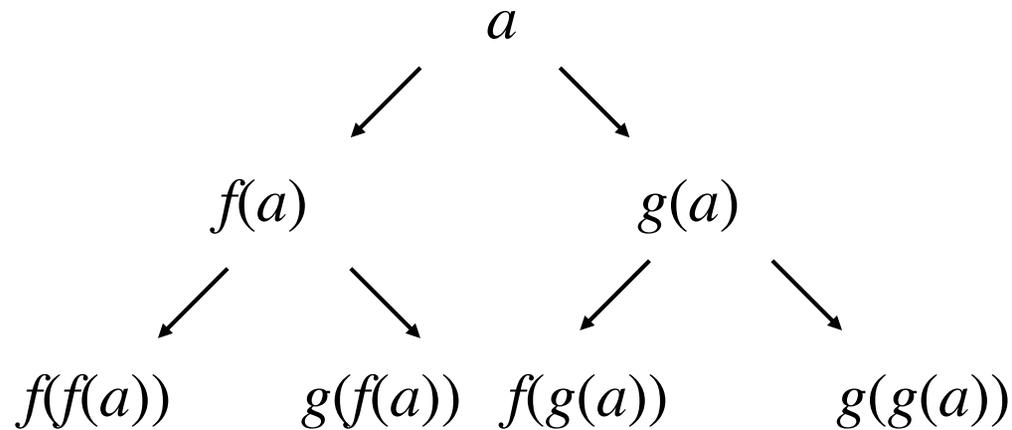
$$\phi[b]$$

$$\forall \mu. (\phi[\mu] \Rightarrow \phi[s(\mu)])$$

$$\forall v. \phi[v]$$

Tree Induction

Tree Language:

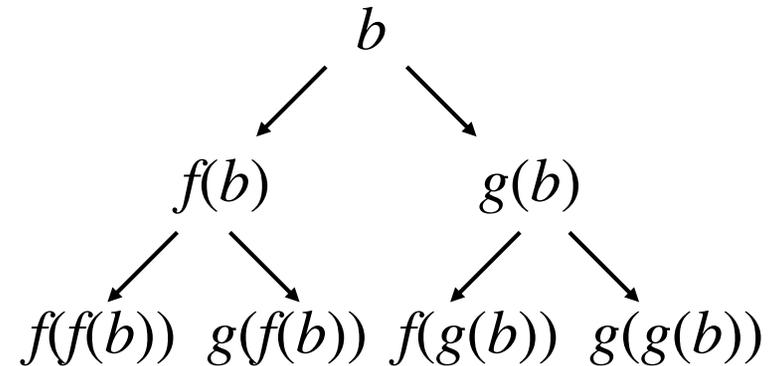
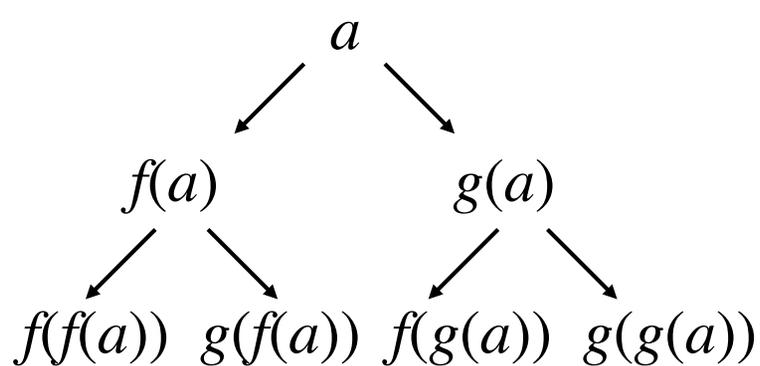


Induction:

$$\begin{array}{l} \phi[a] \\ \forall \mu. (\phi[\mu] \Rightarrow \phi[f(\mu)]) \\ \forall \mu. (\phi[\mu] \Rightarrow \phi[g(\mu)]) \\ \hline \forall v. \phi[v] \end{array}$$

Multiple Object and Function Constants

Tree Language:



Induction:

$$\phi[a]$$

$$\phi[b]$$

$$\forall \mu. (\phi[\mu] \Rightarrow \phi[f(\mu)])$$

$$\forall \mu. (\phi[\mu] \Rightarrow \phi[g(\mu)])$$

$$\forall v. \phi[v]$$

Structural Induction

n -ary Function Constant

Object constant: a

Binary function constant: f

Sample Terms

$a,$

$f(a,a),$

$f(a, f(a,a)),$

$f(f(a,a),a),$

$f(f(a,a), f(a,a)),$

$f(a, f(a, f(a,a))),$

$f(a, f(f(a,a),a)),$

...

Structural Induction

Object constant: a

Binary function constant: f

$$\frac{\begin{array}{l} \phi[a] \\ \forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[f(\lambda, \mu)]) \end{array}}{\forall v. \phi[v]}$$

Structural Induction

Object constants: a, b

Binary function constant: f

$$\frac{\begin{array}{l} \phi[a] \\ \phi[b] \\ \forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[f(\lambda, \mu)]) \end{array}}{\forall v. \phi[v]}$$

Structural Induction

Object constant: a

Binary function constants: f, g

$$\phi[a]$$

$$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[f(\lambda, \mu)])$$

$$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[g(\lambda, \mu)])$$

$$\forall v. \phi[v]$$

Structural Induction

Object constants: a, b

Binary function constants: f, g

$$\phi[a]$$
$$\phi[b]$$
$$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[f(\lambda, \mu)])$$
$$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[g(\lambda, \mu)])$$

$$\forall v. \phi[v]$$

Objects and Base Cases

Object constants: a, b

Binary function constants: f, g

$\phi[a]$

$\phi[b]$

$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[f(\lambda, \mu)])$

$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[g(\lambda, \mu)])$

$\forall v. \phi[v]$

Functions and Inductive Cases

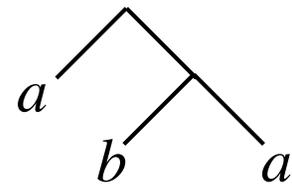
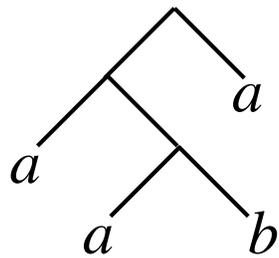
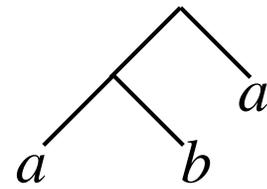
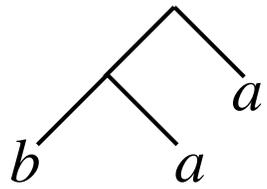
Object constants: a, b

Binary function constants: f, g

$$\phi[a]$$
$$\phi[b]$$
$$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[f(\lambda, \mu)])$$
$$\forall \lambda. \forall \mu. (\phi[\lambda] \wedge \phi[\mu] \Rightarrow \phi[g(\lambda, \mu)])$$

$$\forall v. \phi[v]$$

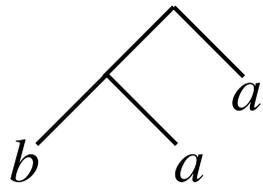
Trees



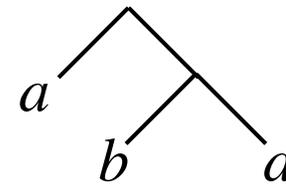
Tree Vocabulary

Object constants: a, b

Unary function constants: $cons$



$cons(cons(b,a),a)$



$cons(a,cons(b,a))$

Unary relation constants: $symmetric, uniform$

Binary relation constant: $subtree, congruent$

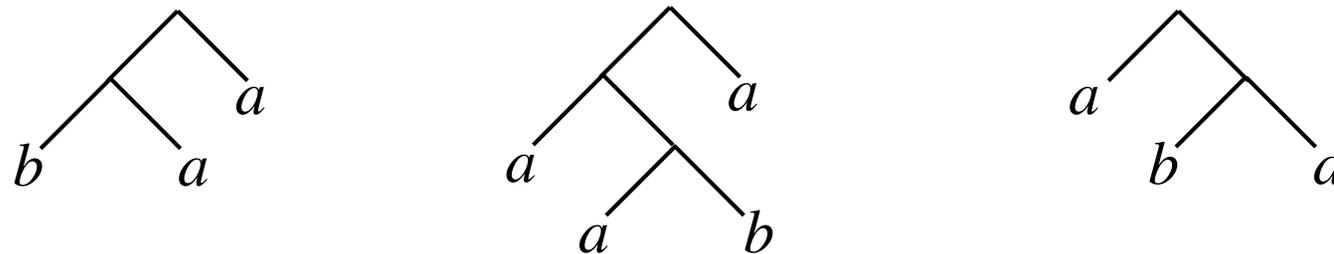
Congruence

Two trees are *congruent* if and only if they have the *same shape*. (Labels on leaf nodes irrelevant.)

Examples:



Non-Examples:



Definition

Congruence of "atomic" trees

congruent(a, a)

congruent(a, b)

congruent(b, a)

congruent(b, b)

Congruence of "compound" trees:

$$\forall u. \forall v. \forall x. \forall y. (\text{congruent}(\text{cons}(u, v), \text{cons}(x, y)) \Leftrightarrow \text{congruent}(u, x) \wedge \text{congruent}(v, y))$$

Non-Congruence of mixed trees:

$$\forall x. \forall y. (\neg \text{congruent}(a, \text{cons}(x, y)) \wedge \neg \text{congruent}(\text{cons}(x, y), a))$$

$$\forall x. \forall y. (\neg \text{congruent}(b, \text{cons}(x, y)) \wedge \neg \text{congruent}(\text{cons}(x, y), b))$$

Properties

Congruence is an equivalence relation.

Reflexivity

$$\forall x. \text{congruent}(x, x)$$

Symmetry

$$\forall x. \forall y. (\text{congruent}(x, y) \Rightarrow \text{congruent}(y, x))$$

Transitivity

$$\forall x. \forall y. \forall z. (\text{congruent}(x, y) \wedge \text{congruent}(y, z) \Rightarrow \text{congruent}(x, z))$$

Problem

Given:

congruent(a, a)

congruent(a, b)

congruent(b, a)

congruent(b, b)

$\forall u. \forall v. \forall x. \forall y. (\text{congruent}(\text{cons}(u, v), \text{cons}(x, y)) \Leftrightarrow$
 $\text{congruent}(u, x) \wedge \text{congruent}(v, y))$

Prove:

$\forall x. \text{congruent}(x, x)$

Problem Restated using \cong

Given:

$$a \cong a$$

$$a \cong b$$

$$b \cong a$$

$$b \cong b$$

$$\forall u. \forall v. \forall x. \forall y. (\text{cons}(u, v) \cong \text{cons}(x, y) \Leftrightarrow u \cong x \wedge v \cong y)$$

Prove:

$$\forall x. x \cong x$$

Induction Strategy

Our goal is to prove that congruence is reflexive.

$$\forall x. x \cong x$$

Base Cases:

$$a \cong a$$

$$b \cong b$$

Inductive Case:

$$\forall x. \forall y. (x \cong x \wedge y \cong y \Rightarrow \text{cons}(x, y) \cong \text{cons}(x, y))$$

Proof of Reflexivity

1. $a \cong a$ Premise
2. $a \cong b$ Premise
3. $b \cong a$ Premise
4. $b \cong b$ Premise
5. $\forall u. \forall v. \forall x. \forall y. (\text{cons}(u, v) \cong \text{cons}(x, y) \Leftrightarrow u \cong x \wedge v \cong y)$ Premise

Proof of Reflexivity

1. $a \cong a$ Premise
2. $a \cong b$ Premise
3. $b \cong a$ Premise
4. $b \cong b$ Premise
5. $\forall u. \forall v. \forall x. \forall y. (\text{cons}(u, v) \cong \text{cons}(x, y) \Leftrightarrow u \cong x \wedge v \cong y)$ Premise
6. $\text{cons}(c, d) \cong \text{cons}(c, d) \Leftrightarrow c \cong c \wedge d \cong d$ 4 x UE: 5
7. $c \cong c \wedge d \cong d \Rightarrow \text{cons}(c, d) \cong \text{cons}(c, d)$ BE: 6

Proof of Reflexivity

1. $a \cong a$ Premise
2. $a \cong b$ Premise
3. $b \cong a$ Premise
4. $b \cong b$ Premise
5. $\forall u. \forall v. \forall x. \forall y. (\text{cons}(u, v) \cong \text{cons}(x, y) \Leftrightarrow u \cong x \wedge v \cong y)$ Premise
6. $\text{cons}(c, d) \cong \text{cons}(c, d) \Leftrightarrow c \cong c \wedge d \cong d$ 4 x UE: 5
7. $c \cong c \wedge d \cong d \Rightarrow \text{cons}(c, d) \cong \text{cons}(c, d)$ BE: 6
8. $\forall x. \forall y. (x \cong x \wedge y \cong y \Rightarrow \text{cons}(x, y) \cong \text{cons}(x, y))$ 2 x UI: 7

Proof of Reflexivity

1. $a \cong a$ Premise
2. $a \cong b$ Premise
3. $b \cong a$ Premise
4. $b \cong b$ Premise
5. $\forall u. \forall v. \forall x. \forall y. (\text{cons}(u, v) \cong \text{cons}(x, y) \Leftrightarrow u \cong x \wedge v \cong y)$ Premise
6. $\text{cons}(c, d) \cong \text{cons}(c, d) \Leftrightarrow c \cong c \wedge d \cong d$ 4 x UE: 5
7. $c \cong c \wedge d \cong d \Rightarrow \text{cons}(c, d) \cong \text{cons}(c, d)$ BE: 6
8. $\forall x. \forall y. (x \cong x \wedge y \cong y \Rightarrow \text{cons}(x, y) \cong \text{cons}(x, y))$ 2 x UI: 7
9. $\forall x. (x \cong x)$ Ind: 1, 4, 8

