

# *Logic and Proof*

Computer Science Tripos Part IB  
Michaelmas Term

*Lawrence C Paulson*  
Computer Laboratory  
University of Cambridge

lp15@cam.ac.uk

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Propositional Logic</b>	<b>6</b>
<b>3</b>	<b>Gentzen's Logical Calculi</b>	<b>11</b>
<b>4</b>	<b>First-Order Logic</b>	<b>16</b>
<b>5</b>	<b>Formal Reasoning in First-Order Logic</b>	<b>22</b>
<b>6</b>	<b>Davis-Putnam &amp; Propositional Resolution</b>	<b>28</b>
<b>7</b>	<b>Skolem Functions and Herbrand's Theorem</b>	<b>34</b>
<b>8</b>	<b>Unification</b>	<b>40</b>
<b>9</b>	<b>Resolution and Prolog</b>	<b>46</b>
<b>10</b>	<b>Binary Decision Diagrams</b>	<b>51</b>
<b>11</b>	<b>Modal Logics</b>	<b>56</b>
<b>12</b>	<b>Tableaux-Based Methods</b>	<b>61</b>

## Introduction to Logic

Logic concerns *statements* in some *language*.

Slide 101

The language can be informal (say English) or *formal*.

Some statements are *true*, others *false* or *meaningless*.

Logic concerns *relationships* between statements: consistency, entailment, . . .

Logical *proofs* model human reasoning (supposedly).

## Statements

Statements are declarative assertions:

*Black is the colour of my true love's hair.*

Slide 102

They are not greetings, questions or commands:

*What is the colour of my true love's hair?*

*I wish my true love had hair.*

*Get a haircut!*

### Schematic Statements

The *meta-variables*  $X, Y, Z, \dots$  range over 'real' objects

*Black is the colour of  $X$ 's hair.*

*Black is the colour of  $Y$ .*

*$Z$  is the colour of  $Y$ .*

Schematic statements can express general statements, or questions:

*What things are black?*

Slide 103

### Interpretations and Validity

An *interpretation* maps meta-variables to real objects:

The interpretation  $Y \mapsto \text{coal}$  *satisfies* the statement

*Black is the colour of  $Y$ .*

but the interpretation  $Y \mapsto \text{strawberries}$  does not!

A statement  $A$  is *valid* if all interpretations satisfy  $A$ .

Slide 104

### Consistency, or Satisfiability

A set  $S$  of statements is *consistent* if some interpretation satisfies all elements of  $S$  at the same time. Otherwise  $S$  is *inconsistent*.

Slide 105

Examples of inconsistent sets:

$$\{X \text{ part of } Y, Y \text{ part of } Z, X \text{ NOT part of } Z\}$$

$$\{n \text{ is a positive integer, } n \neq 1, n \neq 2, \dots\}$$

*Satisfiable* means the same as consistent.

*Unsatisfiable* means the same as inconsistent.

### Entailment, or Logical Consequence

A set  $S$  of statements *entails*  $A$  if every interpretation that satisfies all elements of  $S$ , also satisfies  $A$ . We write  $S \models A$ .

Slide 106

$$\{X \text{ part of } Y, Y \text{ part of } Z\} \models X \text{ part of } Z$$

$$\{n \neq 1, n \neq 2, \dots\} \models n \text{ is NOT a positive integer}$$

$S \models A$  if and only if  $\{\neg A\} \cup S$  is inconsistent

$\models A$  if and only if  $A$  is valid, if and only if  $\{\neg A\}$  is inconsistent.

### Inference

We want to check  $\bar{A}$  is valid.

Checking all interpretations can be effective — but what if there are infinitely many?

Let  $\{A_1, \dots, A_n\} \models B$ . If  $A_1, \dots, A_n$  are true then  $B$  must be true. Write this as the *inference rule*

$$\frac{A_1 \quad \dots \quad A_n}{B}$$

We can use inference rules to construct finite proofs!

Slide 107

### Schematic Inference Rules

$$\frac{X \text{ part of } Y \quad Y \text{ part of } Z}{X \text{ part of } Z}$$

A valid inference:

$$\frac{\text{spoke part of wheel} \quad \text{wheel part of bike}}{\text{spoke part of bike}}$$

An inference may be valid even if the premises are false!

$$\frac{\text{cow part of chair} \quad \text{chair part of ant}}{\text{cow part of ant}}$$

Slide 108

## Survey of Formal Logics

**propositional logic** is traditional *boolean algebra*.

**first-order logic** can say *for all* and *there exists*.

**higher-order logic** reasons about sets and functions.

**modal/temporal logics** reason about what *must*, or *may*, happen.

**type theories** support *constructive* mathematics.

All have been used to prove correctness of computer systems.

Slide 109

## Why Should the Language be Formal?

Consider this 'definition':

The least integer not definable using eight words

Greater than *The number of atoms in the entire Universe*

Also greater than *The least integer not definable using eight words*

- A formal language prevents AMBIGUITY.

Slide 110

### Syntax of Propositional Logic

$P, Q, R, \dots$  propositional letter

$t$  true

$f$  false

$\neg A$  not  $A$

$A \wedge B$   $A$  and  $B$

$A \vee B$   $A$  or  $B$

$A \rightarrow B$  if  $A$  then  $B$

$A \leftrightarrow B$   $A$  if and only if  $B$

Slide 201

### Semantics of Propositional Logic

$\neg, \wedge, \vee, \rightarrow$  and  $\leftrightarrow$  are *truth-functional*: functions of their operands.

$A$	$B$	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
$t$	$t$	$f$	$t$	$t$	$t$	$t$
$t$	$f$	$f$	$f$	$t$	$f$	$f$
$f$	$t$	$t$	$f$	$t$	$t$	$f$
$f$	$f$	$t$	$f$	$f$	$t$	$t$

Slide 202



### Interpretations of Propositional Logic

An *interpretation* is a function from the propositional letters to  $\{t, f\}$ .

Interpretation  $I$  *satisfies* a formula  $A$  if the formula evaluates to  $t$ .

Write  $\models_I A$

$A$  is *valid* (a *tautology*) if every interpretation satisfies  $A$ .

Write  $\models A$

$S$  is *satisfiable* if some interpretation satisfies every formula in  $S$ .

Slide 203

### Implication, Entailment, Equivalence

$A \rightarrow B$  means simply  $\neg A \vee B$ .

$A \models B$  means if  $\models_I A$  then  $\models_I B$  for every interpretation  $I$ .

$A \models B$  if and only if  $\models A \rightarrow B$ .

#### Equivalence

$A \simeq B$  means  $A \models B$  and  $B \models A$ .

$A \simeq B$  if and only if  $\models A \leftrightarrow B$ .

Slide 204

### Equivalences

$$A \wedge A \simeq A$$

$$A \wedge B \simeq B \wedge A$$

$$(A \wedge B) \wedge C \simeq A \wedge (B \wedge C)$$

$$A \vee (B \wedge C) \simeq (A \vee B) \wedge (A \vee C)$$

$$A \wedge \mathbf{f} \simeq \mathbf{f}$$

$$A \wedge \mathbf{t} \simeq A$$

$$A \wedge \neg A \simeq \mathbf{f}$$

Dual versions: exchange  $\wedge$  with  $\vee$  and  $\mathbf{t}$  with  $\mathbf{f}$  in any equivalence

Slide 205

### Negation Normal Form

1. Get rid of  $\leftrightarrow$  and  $\rightarrow$ , leaving just  $\wedge$ ,  $\vee$ ,  $\neg$ :

$$A \leftrightarrow B \simeq (A \rightarrow B) \wedge (B \rightarrow A)$$

$$A \rightarrow B \simeq \neg A \vee B$$

2. Push negations in, using de Morgan's laws:

$$\neg\neg A \simeq A$$

$$\neg(A \wedge B) \simeq \neg A \vee \neg B$$

$$\neg(A \vee B) \simeq \neg A \wedge \neg B$$

Slide 206

### From NNF to Conjunctive Normal Form

3. Push disjunctions in, using distributive laws:

$$A \vee (B \wedge C) \simeq (A \vee B) \wedge (A \vee C)$$

$$(B \wedge C) \vee A \simeq (B \vee A) \wedge (C \vee A)$$

Slide 207

4. Simplify:

- Delete any disjunction containing  $P$  and  $\neg P$
- Delete any disjunction that includes another: for example, in  $(P \vee Q) \wedge P$ , delete  $P \vee Q$ .
- Replace  $(P \vee A) \wedge (\neg P \vee A)$  by  $A$

### Converting a Non-Tautology to CNF

$$P \vee Q \rightarrow Q \vee R$$

1. Elim  $\rightarrow$ :  $\neg(P \vee Q) \vee (Q \vee R)$
2. Push  $\neg$  in:  $(\neg P \wedge \neg Q) \vee (Q \vee R)$
3. Push  $\vee$  in:  $(\neg P \vee Q \vee R) \wedge (\neg Q \vee Q \vee R)$
4. Simplify:  $\neg P \vee Q \vee R$

Slide 208

Not a tautology: try  $P \mapsto \mathbf{t}$ ,  $Q \mapsto \mathbf{f}$ ,  $R \mapsto \mathbf{f}$

**Tautology checking using CNF**

$$((P \rightarrow Q) \rightarrow P) \rightarrow P$$

1. Elim  $\rightarrow$ :  $\neg[\neg(\neg P \vee Q) \vee P] \vee P$

2. Push  $\neg$  in:  $[\neg\neg(\neg P \vee Q) \wedge \neg P] \vee P$   
 $[(\neg P \vee Q) \wedge \neg P] \vee P$

3. Push  $\vee$  in:  $(\neg P \vee Q \vee P) \wedge (\neg P \vee P)$

4. Simplify:  $\mathbf{t} \wedge \mathbf{t}$

$\mathbf{t}$       *It's a tautology!*

Slide 209

### A Simple Proof System

*Axiom Schemes*

$$\text{K} \quad A \rightarrow (B \rightarrow A)$$

$$\text{S} \quad (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$\text{DN} \quad \neg\neg A \rightarrow A$$

*Inference Rule: Modus Ponens*

$$\frac{A \rightarrow B \quad A}{B}$$

Slide 301

### A Simple (?) Proof of $\bar{A} \rightarrow \bar{A}$

$$(A \rightarrow ((D \rightarrow A) \rightarrow A)) \rightarrow \tag{1}$$

$$((A \rightarrow (D \rightarrow A)) \rightarrow (A \rightarrow A)) \text{ by S}$$

$$A \rightarrow ((D \rightarrow A) \rightarrow A) \text{ by K} \tag{2}$$

$$(A \rightarrow (D \rightarrow A)) \rightarrow (A \rightarrow A) \text{ by MP, (1), (2)} \tag{3}$$

$$A \rightarrow (D \rightarrow A) \text{ by K} \tag{4}$$

$$A \rightarrow A \text{ by MP, (3), (4)} \tag{5}$$

Slide 302

### Some Facts about Deducibility

$A$  is *deducible* from the set  $S$  if there is a finite proof of  $A$  starting from elements of  $S$ . Write  $S \vdash A$ .

**Soundness Theorem.** If  $S \vdash A$  then  $S \models A$ .

**Completeness Theorem.** If  $S \models A$  then  $S \vdash A$ .

**Deduction Theorem.** If  $S \cup \{A\} \vdash B$  then  $S \vdash A \rightarrow B$ .

Slide 303

### Gentzen's Natural Deduction Systems

The context of *assumptions* may vary.

Each logical connective is defined *independently*.

The *introduction* rule for  $\wedge$  shows how to deduce  $A \wedge B$ :

$$\frac{A \quad B}{A \wedge B}$$

The *elimination* rules for  $\wedge$  shows what to deduce *from*  $A \wedge B$ :

$$\frac{A \wedge B}{A} \quad \frac{A \wedge B}{B}$$

Slide 304

### The Sequent Calculus

Sequent  $A_1, \dots, A_m \Rightarrow B_1, \dots, B_n$  means,

if  $A_1 \wedge \dots \wedge A_m$  then  $B_1 \vee \dots \vee B_n$

$A_1, \dots, A_m$  are *assumptions*;  $B_1, \dots, B_n$  are *goals*

$\Gamma$  and  $\Delta$  are sets in  $\Gamma \Rightarrow \Delta$

The sequent  $A, \Gamma \Rightarrow A, \Delta$  is trivially true (*basic sequent*).

Slide 305

### Sequent Calculus Rules

$$\frac{\Gamma \Rightarrow \Delta, A \quad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \text{ (cut)}$$

$$\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta} \text{ } (\neg l) \quad \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A} \text{ } (\neg r)$$

$$\frac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta} \text{ } (\wedge l) \quad \frac{\Gamma \Rightarrow \Delta, A \quad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B} \text{ } (\wedge r)$$

Slide 306

### More Sequent Calculus Rules

Slide 307

$$\frac{A, \Gamma \Rightarrow \Delta \quad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta} (\vee l) \qquad \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B} (\vee r)$$

$$\frac{\Gamma \Rightarrow \Delta, A \quad B, \Gamma \Rightarrow \Delta}{A \rightarrow B, \Gamma \Rightarrow \Delta} (\rightarrow l) \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \rightarrow B} (\rightarrow r)$$

### Easy Sequent Calculus Proofs

Slide 308

$$\frac{\overline{A, B \Rightarrow A}}{A \wedge B \Rightarrow A} (\wedge l)$$

$$\frac{A \wedge B \Rightarrow A}{\Rightarrow (A \wedge B) \rightarrow A} (\rightarrow r)$$

$$\frac{\overline{A, B \Rightarrow B, A}}{A \Rightarrow B, B \rightarrow A} (\rightarrow r)$$

$$\frac{A \Rightarrow B, B \rightarrow A}{\Rightarrow A \rightarrow B, B \rightarrow A} (\rightarrow r)$$

$$\frac{\Rightarrow A \rightarrow B, B \rightarrow A}{\Rightarrow (A \rightarrow B) \vee (B \rightarrow A)} (\vee r)$$



Slide 309

**Part of a Distributive Law**

$$\frac{\frac{\overline{A \Rightarrow A, B} \quad \overline{B, C \Rightarrow A, B}}{B \wedge C \Rightarrow A, B} (\wedge l)}{A \vee (B \wedge C) \Rightarrow A, B} (\vee l) \quad \text{similar}$$

$$\frac{A \vee (B \wedge C) \Rightarrow A, B \quad A \vee (B \wedge C) \Rightarrow A \vee B}{A \vee (B \wedge C) \Rightarrow (A \vee B) \wedge (A \vee C)} (\vee r) \quad (\wedge r)$$

Second subtree proves  $A \vee (B \wedge C) \Rightarrow A \vee C$  similarly

Slide 310

**A Failed Proof**

$$\frac{\frac{A \Rightarrow B, C \quad \overline{B \Rightarrow B, C}}{A \vee B \Rightarrow B, C} (\vee l)}{A \vee B \Rightarrow B \vee C} (\vee r)$$

$$\frac{A \vee B \Rightarrow B \vee C}{\Rightarrow (A \vee B) \rightarrow (B \vee C)} (\rightarrow r)$$

$A \mapsto t, B \mapsto f, C \mapsto f$  falsifies unproved sequent!

### Outline of First-Order Logic

Reasons about *functions* and *relations* over a set of *individuals*:

$$\frac{\text{father}(\text{father}(x)) = \text{father}(\text{father}(y))}{\text{cousin}(x, y)}$$

Slide 401

Reasons about *all* and *some* individuals:

$$\frac{\text{All men are mortal} \quad \text{Socrates is a man}}{\text{Socrates is mortal}}$$

Cannot reason about *all functions* or *all relations*, etc.

### Function Symbols; Terms

Each *function symbol* stands for an  $n$ -place function.

A *constant symbol* is a 0-place function symbol.

A *variable* ranges over all individuals.

A *term* is a variable, constant or a function application

$$f(t_1, \dots, t_n)$$

where  $f$  is an  $n$ -place function symbol and  $t_1, \dots, t_n$  are terms.

We choose the language, adopting any desired function symbols.

Slide 402

### Relation Symbols; Formulae

Each *relation symbol* stands for an  $n$ -place relation.

*Equality* is the 2-place relation symbol  $=$

An *atomic formula* has the form  $R(t_1, \dots, t_n)$  where  $R$  is an  $n$ -place relation symbol and  $t_1, \dots, t_n$  are terms.

A *formula* is built up from atomic formulæ using  $\neg, \wedge, \vee$ , and so forth.

Later, we can add *quantifiers*.

Slide 403

### The Power of Quantifier-Free FOL

It is surprisingly expressive, if we include strong induction rules.

It is easy to equivalence of mathematical functions:

$$\begin{array}{ll}
 p(z, 0) = 1 & q(z, 1) = z \\
 p(z, n + 1) = p(z, n) \times z & q(z, 2 \times n) = q(z \times z, n) \\
 & q(z, 2 \times n + 1) = q(z \times z, n) \times z
 \end{array}$$

The prover ACL2 uses this logic and has been used in major hardware proofs.

Slide 404

### Universal and Existential Quantifiers

$\forall x A$  for all  $x$ , the formula  $A$  holds

$\exists x A$  there exists  $x$  such that  $A$  holds

Slide 405

*Syntactic variations:*

$\forall xyz A$  abbreviates  $\forall x \forall y \forall z A$

$\forall z . A \wedge B$  is an alternative to  $\forall z (A \wedge B)$

The variable  $x$  is *bound* in  $\forall x A$ ; compare with  $\int f(x) dx$

### The Expressiveness of Quantifiers

*All men are mortal:*

$$\forall x (\text{man}(x) \rightarrow \text{mortal}(x))$$

Slide 406

*All mothers are female:*

$$\forall x \text{female}(\text{mother}(x))$$

*There exists a unique  $x$  such that  $A$ , sometimes written  $\exists! x A$*

$$\exists x [A(x) \wedge \forall y (A(y) \rightarrow y = x)]$$

### The Point of Semantics

We have to attach meanings to symbols like 1, +, <, etc.

*Why is this necessary? Why can't 1 just mean 1??*

The point is that mathematics derives its flexibility from allowing different interpretations of symbols.

- A *group* has a unit 1, a product  $x \cdot y$  and inverse  $x^{-1}$ .
- In the most important uses of groups, 1 isn't a number but a 'unit permutation', 'unit rotation', etc.

Slide 407

### Constants: Interpreting mortal(Socrates)

An interpretation  $\mathcal{I} = (D, I)$  defines the *semantics* of a first-order language.

$D$  is a non-empty set, called the *domain* or *universe*.

$I$  maps symbols to 'real' elements, functions and relations:

$c$  a *constant* symbol                       $I[c] \in D$

$f$  an  $n$ -place *function* symbol       $I[f] \in D^n \rightarrow D$

$P$  an  $n$ -place *relation* symbol       $I[P] \in D^n \rightarrow \{\mathbf{t}, \mathbf{f}\}$

Slide 408

### Variables: Interpreting $\text{cousin}(\text{Charles}, y)$

A *valuation*  $V : \text{variables} \rightarrow D$  supplies the values of free variables.

An interpretation  $\mathcal{I}$  and valuation function  $V$  jointly specify the value of any term  $t$  by the obvious recursion.

This value is written  $\mathcal{I}_V[t]$ , and here are the recursion rules:

$$\mathcal{I}_V[x] \stackrel{\text{def}}{=} V(x) \quad \text{if } x \text{ is a variable}$$

$$\mathcal{I}_V[c] \stackrel{\text{def}}{=} I[c]$$

$$\mathcal{I}_V[f(t_1, \dots, t_n)] \stackrel{\text{def}}{=} I[f](\mathcal{I}_V[t_1], \dots, \mathcal{I}_V[t_n])$$

Slide 409

### Tarski's Truth-Definition

An interpretation  $\mathcal{I}$  and valuation function  $V$  similarly specify the truth value (**t** or **f**) of any formula  $A$ .

*Quantifiers* are the only problem, as they bind variables.

$V\{\alpha/x\}$  is the valuation that maps  $x$  to  $\alpha$  and is otherwise like  $V$ .

With the help of  $V\{\alpha/x\}$ , we now formally define  $\models_{\mathcal{I}, V} A$ , the truth value of  $A$ .

Slide 410

### The Meaning of Truth—In FOL!

For interpretation  $\mathcal{I}$  and valuation  $V$ , define  $\models_{\mathcal{I}, V}$  by recursion.

$\models_{\mathcal{I}, V} P(t)$	if $\mathcal{I}_V[t] \in I[P]$ holds
$\models_{\mathcal{I}, V} t = u$	if $\mathcal{I}_V[t]$ equals $\mathcal{I}_V[u]$
$\models_{\mathcal{I}, V} A \wedge B$	if $\models_{\mathcal{I}, V} A$ and $\models_{\mathcal{I}, V} B$
$\models_{\mathcal{I}, V} \exists x A$	if $\models_{\mathcal{I}, V\{m/x\}} A$ holds for some $m \in D$

Finally, we define

$\models_{\mathcal{I}} A$                     if  $\models_{\mathcal{I}, V} A$  holds for all  $V$ .

A closed formula  $A$  is *satisfiable* if  $\models_{\mathcal{I}} A$  for some  $\mathcal{I}$ .

Slide 411

### Free vs Bound Variables

All occurrences of  $x$  in  $\forall x A$  and  $\exists x A$  are *bound*

An occurrence of  $x$  is *free* if it is not bound:

$$\forall y \exists z R(y, z, f(y, x))$$

In this formula,  $y$  and  $z$  are bound while  $x$  is free.

May *rename* bound variables:

$$\forall w \exists z' R(w, z', f(w, x))$$

Slide 501

### Substitution for Free Variables

$A[t/x]$  means *substitute*  $t$  for  $x$  in  $A$ :

$$(B \wedge C)[t/x] \text{ is } B[t/x] \wedge C[t/x]$$

$$(\forall x B)[t/x] \text{ is } \forall x B$$

$$(\forall y B)[t/x] \text{ is } \forall y B[t/x] \quad (x \neq y)$$

$$(P(u))[t/x] \text{ is } P(u[t/x])$$

With  $A[t/x]$ , no variable of  $t$  may be bound in  $A$ !

$(\forall y (x = y)) [y/x]$  IS NOT EQUIVALENT TO  $\forall y (y = y)$

Slide 502



### Some Equivalences for Quantifiers

$$\neg(\forall x A) \simeq \exists x \neg A$$

$$\forall x A \simeq \forall x A \wedge A[t/x]$$

$$(\forall x A) \wedge (\forall x B) \simeq \forall x (A \wedge B)$$

BUT WE DO NOT HAVE  $(\forall x A) \vee (\forall x B) \simeq \forall x (A \vee B)$ .

*Dual versions: exchange  $\forall$  with  $\exists$  and  $\wedge$  with  $\vee$*

Slide 503

### Further Quantifier Equivalences

These hold only if  $x$  is not free in  $B$ .

$$(\forall x A) \wedge B \simeq \forall x (A \wedge B)$$

$$(\forall x A) \vee B \simeq \forall x (A \vee B)$$

$$(\forall x A) \rightarrow B \simeq \exists x (A \rightarrow B)$$

These let us expand or contract a quantifier's scope.

Slide 504

### Reasoning by Equivalences

$$\begin{aligned}\exists x (x = a \wedge P(x)) &\simeq \exists x (x = a \wedge P(a)) \\ &\simeq \exists x (x = a) \wedge P(a) \\ &\simeq P(a)\end{aligned}$$

Slide 505

$$\begin{aligned}\exists z (P(z) \rightarrow P(a) \wedge P(b)) \\ &\simeq \forall z P(z) \rightarrow P(a) \wedge P(b) \\ &\simeq \forall z P(z) \wedge P(a) \wedge P(b) \rightarrow P(a) \wedge P(b) \\ &\simeq \mathbf{t}\end{aligned}$$

### Sequent Calculus Rules for $\forall$

$$\frac{A[t/x], \Gamma \Rightarrow \Delta}{\forall x A, \Gamma \Rightarrow \Delta} (\forall l) \quad \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \forall x A} (\forall r)$$

Slide 506

Rule  $(\forall l)$  can create many instances of  $\forall x A$

Rule  $(\forall r)$  holds *provided*  $x$  is not free in the conclusion!

NOT allowed to prove

$$\frac{\overline{P(y) \Rightarrow P(y)}}{P(y) \Rightarrow \forall y P(y)} (\forall r) \quad \text{THIS IS NONSENSE!}$$

### A Simple Example of the $\forall$ Rules

Slide 507

$$\frac{\overline{P(f(y)) \Rightarrow P(f(y))}}{\forall x P(x) \Rightarrow P(f(y))} \quad (\forall l)$$

$$\frac{\forall x P(x) \Rightarrow P(f(y))}{\forall x P(x) \Rightarrow \forall y P(f(y))} \quad (\forall r)$$

### A Not-So-Simple Example of the $\forall$ Rules

Slide 508

$$\frac{\overline{P \Rightarrow Q(y)}, P \quad \overline{P, Q(y) \Rightarrow Q(y)}}{P, P \rightarrow Q(y) \Rightarrow Q(y)} \quad (\rightarrow l)$$

$$\frac{P, P \rightarrow Q(y) \Rightarrow Q(y)}{P, \forall x (P \rightarrow Q(x)) \Rightarrow Q(y)} \quad (\forall l)$$

$$\frac{P, \forall x (P \rightarrow Q(x)) \Rightarrow Q(y)}{P, \forall x (P \rightarrow Q(x)) \Rightarrow \forall y Q(y)} \quad (\forall r)$$

$$\frac{P, \forall x (P \rightarrow Q(x)) \Rightarrow \forall y Q(y)}{\forall x (P \rightarrow Q(x)) \Rightarrow P \rightarrow \forall y Q(y)} \quad (\rightarrow r)$$

In  $(\forall l)$ , we must replace  $x$  by  $y$ .

### Sequent Calculus Rules for $\exists$

$$\frac{A, \Gamma \Rightarrow \Delta}{\exists x A, \Gamma \Rightarrow \Delta} (\exists l) \quad \frac{\Gamma \Rightarrow \Delta, A[t/x]}{\Gamma \Rightarrow \Delta, \exists x A} (\exists r)$$

Slide 509

Rule  $(\exists l)$  holds *provided*  $x$  is not free in the conclusion!

Rule  $(\exists r)$  can create many instances of  $\exists x A$

For example, to prove this counter-intuitive formula:

$$\exists z (P(z) \rightarrow P(a) \wedge P(b))$$

### Part of the $\exists$ Distributive Law

$$\frac{\frac{\frac{P(x) \Rightarrow P(x), Q(x)}{P(x) \Rightarrow P(x) \vee Q(x)} (\vee r)}{P(x) \Rightarrow \exists y (P(y) \vee Q(y))} (\exists r)}{\exists x P(x) \Rightarrow \exists y (P(y) \vee Q(y))} (\exists l) \quad \frac{\text{similar}}{\exists x Q(x) \Rightarrow \exists y \dots} (\exists l)}{\exists x P(x) \vee \exists x Q(x) \Rightarrow \exists y (P(y) \vee Q(y))} (\vee l)$$

Slide 510

Second subtree proves  $\exists x Q(x) \Rightarrow \exists y (P(y) \vee Q(y))$  similarly

In  $(\exists r)$ , we must replace  $y$  by  $x$ .

### A Failed Proof

$$\begin{array}{r}
 \frac{P(x), Q(y) \Rightarrow P(x) \wedge Q(x)}{P(x), Q(y) \Rightarrow \exists z (P(z) \wedge Q(z))} \quad (\exists r) \\
 \frac{P(x), Q(y) \Rightarrow \exists z (P(z) \wedge Q(z))}{P(x), \exists x Q(x) \Rightarrow \exists z (P(z) \wedge Q(z))} \quad (\exists l) \\
 \frac{P(x), \exists x Q(x) \Rightarrow \exists z (P(z) \wedge Q(z))}{\exists x P(x), \exists x Q(x) \Rightarrow \exists z (P(z) \wedge Q(z))} \quad (\exists l) \\
 \frac{\exists x P(x), \exists x Q(x) \Rightarrow \exists z (P(z) \wedge Q(z))}{\exists x P(x) \wedge \exists x Q(x) \Rightarrow \exists z (P(z) \wedge Q(z))} \quad (\wedge l)
 \end{array}$$

Slide 511

We cannot use  $(\exists l)$  twice with the same variable

This attempt renames the  $x$  in  $\exists x Q(x)$ , to get  $\exists y Q(y)$

### Clause Form

*Clause*: a disjunction of *literals*

$$\neg K_1 \vee \dots \vee \neg K_m \vee L_1 \vee \dots \vee L_n$$

Slide 601

Set notation:  $\{\neg K_1, \dots, \neg K_m, L_1, \dots, L_n\}$

Kowalski notation:  $K_1, \dots, K_m \rightarrow L_1, \dots, L_n$

$L_1, \dots, L_n \leftarrow K_1, \dots, K_m$

Empty clause:  $\square$

Empty clause is equivalent to  $\mathbf{f}$ , meaning CONTRADICTION!

### Outline of Clause Form Methods

To prove  $A$ , obtain a contradiction from  $\neg A$ :

1. Translate  $\neg A$  into CNF as  $A_1 \wedge \dots \wedge A_m$
2. This is the set of clauses  $A_1, \dots, A_m$
3. Transform the clause set, *preserving consistency*

Deducing the empty *clause* refutes  $\neg A$ .

An empty *clause set* (all clauses deleted) means  $\neg A$  is satisfiable.

The basis for SAT SOLVERS and RESOLUTION PROVERS.

Slide 602

### The Davis-Putnam-Logeman-Loveland Method

1. Delete tautological clauses:  $\{P, \neg P, \dots\}$
2. For each unit clause  $\{L\}$ ,
  - delete all clauses containing  $L$
  - delete  $\neg L$  from all clauses
3. Delete all clauses containing *pure literals*
4. Perform a *case split* on some literal

DPLL is a **decision procedure**: it finds a contradiction or a model.

Slide 603

### Davis-Putnam on a Non-Tautology

Consider  $P \vee Q \rightarrow Q \vee R$

Clauses are  $\{P, Q\}$   $\{\neg Q\}$   $\{\neg R\}$

$\{P, Q\}$	$\{\neg Q\}$	$\{\neg R\}$	initial clauses
$\{P\}$	$\{\neg R\}$		unit $\neg Q$
	$\{\neg R\}$		unit $P$ (also pure)
			unit $\neg R$ (also pure)

Clauses satisfiable by  $P \mapsto \mathbf{t}$ ,  $Q \mapsto \mathbf{f}$ ,  $R \mapsto \mathbf{f}$

Slide 604

### Example of a Case Split on P

$$\{\neg Q, R\} \quad \{\neg R, P\} \quad \{\neg R, Q\} \quad \{\neg P, Q, R\} \quad \{P, Q\} \quad \{\neg P, \neg Q\}$$

$\{\neg Q, R\}$	$\{\neg R, Q\}$	$\{Q, R\}$	$\{\neg Q\}$	if P is true
	$\{\neg R\}$	$\{R\}$		unit $\neg Q$
	$\square$			unit R

---

$\{\neg Q, R\}$	$\{\neg R\}$	$\{\neg R, Q\}$	$\{Q\}$	if P is false
$\{\neg Q\}$			$\{Q\}$	unit $\neg R$
			$\square$	unit $\neg Q$

Slide 605

### SAT solvers in the Real World

- Progressed from joke to killer technology in 10 years.
- Princeton's zChaff has solved problems with more than one million variables and 10 million clauses.
- Applications include finding bugs in device drivers (Microsoft's SLAM project).
- Typical approach: approximate the problem with a finite model; encode it using Boolean logic; supply to a SAT solver.

Slide 606



### The Resolution Rule

From  $B \vee A$  and  $\neg B \vee C$  infer  $A \vee C$

In set notation,

$$\frac{\{B, A_1, \dots, A_m\} \quad \{\neg B, C_1, \dots, C_n\}}{\{A_1, \dots, A_m, C_1, \dots, C_n\}}$$

Some special cases:

$$\frac{\{B\} \quad \{\neg B, C_1, \dots, C_n\}}{\{C_1, \dots, C_n\}} \quad \frac{\{B\} \quad \{\neg B\}}{\square}$$

Slide 607

### Simple Example: Proving $P \wedge Q \rightarrow Q \wedge P$

*Hint:* use  $\neg(A \rightarrow B) \simeq A \wedge \neg B$

1. Negate!  $\neg[P \wedge Q \rightarrow Q \wedge P]$

2. Push  $\neg$  in:  $(P \wedge Q) \wedge \neg(Q \wedge P)$

$$(P \wedge Q) \wedge (\neg Q \vee \neg P)$$

Clauses:  $\{P\} \quad \{Q\} \quad \{\neg Q, \neg P\}$

Resolve  $\{P\}$  and  $\{\neg Q, \neg P\}$  getting  $\{\neg Q\}$ .

Resolve  $\{Q\}$  and  $\{\neg Q\}$  getting  $\square$ : we have refuted the negation.

Slide 608

### Another Example

Refute  $\neg[(P \vee Q) \wedge (P \vee R) \rightarrow P \vee (Q \wedge R)]$

From  $(P \vee Q) \wedge (P \vee R)$ , get clauses  $\{P, Q\}$  and  $\{P, R\}$ .

From  $\neg[P \vee (Q \wedge R)]$  get clauses  $\{\neg P\}$  and  $\{\neg Q, \neg R\}$ .

Resolve  $\{\neg P\}$  and  $\{P, Q\}$  getting  $\{Q\}$ .

Resolve  $\{\neg P\}$  and  $\{P, R\}$  getting  $\{R\}$ .

Resolve  $\{Q\}$  and  $\{\neg Q, \neg R\}$  getting  $\{\neg R\}$ .

Resolve  $\{R\}$  and  $\{\neg R\}$  getting  $\square$ , contradiction.

Slide 609

### The Saturation Algorithm

At start, all clauses are *passive*. None are *active*.

1. Transfer a clause (*current*) from *passive* to *active*.
2. Form all resolvents between *current* and an *active* clause.
3. Use new clauses to simplify both *passive* and *active*.
4. Put the new clauses into *passive*.

Repeat until CONTRADICTION found or *passive* becomes empty.

Slide 610

## Heuristics and Hacks for Resolution

*Orderings* to focus the search on specific literals

*Subsumption*, or deleting redundant clauses

*Indexing*: elaborate data structures for speed

*Preprocessing*: removing tautologies, symmetries . . .

*Weighting*: giving priority to “good” clauses over those containing unwanted constants

Slide 611

## Reducing FOL to Propositional Logic

*Prenex:* Move quantifiers to the front

*Skolemize:* Remove quantifiers, preserving consistency

*Herbrand models:* Reduce the class of interpretations

*Herbrand's Thm:* Contradictions have *finite, ground* proofs

*Unification:* Automatically find the right instantiations

Finally, combine unification with *resolution*

Slide 701

## Prenex Normal Form

Convert to Negation Normal Form using additionally

$$\neg(\forall x A) \simeq \exists x \neg A$$

$$\neg(\exists x A) \simeq \forall x \neg A$$

Move quantifiers to the front using (provided  $x$  is not free in  $B$ )

$$(\forall x A) \wedge B \simeq \forall x (A \wedge B)$$

$$(\forall x A) \vee B \simeq \forall x (A \vee B)$$

and the similar rules for  $\exists$

Slide 702

### Skolemization, or Getting Rid of $\exists$

Start with a formula of the form (Can have  $k = 0$ ).

$$\forall x_1 \forall x_2 \cdots \forall x_k \exists y A$$

Slide 703

Choose a fresh  $k$ -place function symbol, say  $f$

Delete  $\exists y$  and replace  $y$  by  $f(x_1, x_2, \dots, x_k)$ . We get

$$\forall x_1 \forall x_2 \cdots \forall x_k A[f(x_1, x_2, \dots, x_k)/y]$$

Repeat until no  $\exists$  quantifiers remain

### Example of Conversion to Clauses

For proving  $\exists x [P(x) \rightarrow \forall y P(y)]$

$\neg [\exists x [P(x) \rightarrow \forall y P(y)]]$	negated goal
$\forall x [P(x) \wedge \exists y \neg P(y)]$	conversion to NNF
$\forall x \exists y [P(x) \wedge \neg P(y)]$	pulling $\exists$ out
$\forall x [P(x) \wedge \neg P(f(x))]$	Skolem term $f(x)$
$\{P(x)\} \quad \{\neg P(f(x))\}$	Final clauses

Slide 704

### Correctness of Skolemization

The formula  $\forall x \exists y A$  is consistent

$\iff$  it holds in some interpretation  $\mathcal{I} = (D, I)$

$\iff$  for all  $x \in D$  there is some  $y \in D$  such that  $A$  holds

$\iff$  some function  $\hat{f}$  in  $D \rightarrow D$  yields suitable values of  $y$

$\iff A[f(x)/y]$  holds in some  $\mathcal{I}'$  extending  $\mathcal{I}$  so that  $f$  denotes  $\hat{f}$

$\iff$  the formula  $\forall x A[f(x)/y]$  is consistent.

*Don't panic if you can't follow this reasoning!*

Slide 705

### Simplifying the Search for Models

$S$  is satisfiable if even *one* model makes all of its clauses true.

There are *infinitely many* models to consider!

Also many *duplicates*: “states of the USA” and “the integers 1 to 50”

Fortunately, nice models exist.

- They have a *uniform structure* based on the language's *syntax*.
- They satisfy the clauses if any model does.

Slide 706

### The Herbrand Universe for a Set of Clauses $S$

$H_0 \stackrel{\text{def}}{=} \text{the set of constants in } S \text{ (must be non-empty)}$

$H_{i+1} \stackrel{\text{def}}{=} H_i \cup \{f(t_1, \dots, t_n) \mid t_1, \dots, t_n \in H_i$

and  $f$  is an  $n$ -place function symbol in  $S\}$

$H \stackrel{\text{def}}{=} \bigcup_{i \geq 0} H_i \quad \textit{Herbrand Universe}$

$H_i$  contains just the terms with at most  $i$  nested function applications.

$H$  consists of the terms in  $S$  that contain no variables (*ground terms*).

Slide 707

### The Herbrand Semantics of Terms

In a Herbrand model, every constant stands for itself.

Every function symbol stands for a term-forming operation:

$f$  denotes the function that puts ' $f$ ' in front of the given arguments.

In a Herbrand model,  $X + 0$  can never equal  $X$ .

Every ground term denotes itself.

This is the promised uniform structure!

Slide 708

### The Herbrand Semantics of Predicates

$$\text{HB} \stackrel{\text{def}}{=} \{P(t_1, \dots, t_n) \mid t_1, \dots, t_n \in H\}$$

and  $P$  is an  $n$ -place predicate symbol in  $S$

HB contains all *ground atoms*: predicates applied to ground terms.

We view a ground atom as stating that the formula is true.

In a Herbrand model, each predicate  $P$  stands for

*the set of ground atoms  $P(x)$  that we want to be true.*

We can make whatever we want true!

Slide 709

### Example of an Herbrand Model

$$\left. \begin{array}{l} \neg \text{even}(1) \\ \text{even}(2) \\ \text{even}(X \cdot Y) \leftarrow \text{even}(X), \text{even}(Y) \end{array} \right\} \text{clauses}$$

$$H = \{1, 2, 1 \cdot 1, 1 \cdot 2, 2 \cdot 1, 2 \cdot 2, 1 \cdot (1 \cdot 1), \dots\}$$

$$\text{HB} = \{\text{even}(1), \text{even}(2), \text{even}(1 \cdot 1), \text{even}(1 \cdot 2), \dots\}$$

$$I[\text{even}] = \{\text{even}(2), \text{even}(1 \cdot 2), \text{even}(2 \cdot 1), \text{even}(2 \cdot 2), \dots\}$$

*(for model where  $\cdot$  means product; could instead use sum!)*

Slide 710



### A Key Fact about Herbrand Interpretations

Let  $S$  be a set of clauses.

$S$  is unsatisfiable  $\iff$  no Herbrand interpretation satisfies  $S$

- Holds because some Herbrand model mimicks every 'real' model
- We must consider only a small class of models
- Herbrand models are syntactic, easily processed by computer

Slide 711

### Herbrand's Theorem

Let  $S$  be a set of clauses.

$S$  is unsatisfiable  $\iff$  there is a finite unsatisfiable set  $S'$  of ground instances of clauses of  $S$ .

- **Finite:** we can compute it
- **Instance:** result of substituting for variables
- **Ground:** no variables remain—it's propositional!

Example:  $S$  could be  $\{P(x)\} \quad \{\neg P(f(y))\}$ ,  
and  $S'$  could be  $\{P(f(a))\} \quad \{\neg P(f(a))\}$ .

Slide 712

## Unification

Finding a *common instance* of two terms. Lots of applications:

- **Prolog** and other logic programming languages
- **Theorem proving**: resolution and other procedures
- Tools for reasoning with **equations**
- Tools for satisfying multiple **constraints**
- Polymorphic type-checking (**ML** and other functional languages)

It's an intuitive generalization of pattern-matching.

Slide 801

## Substitutions: A Mathematical Treatment

A substitution is a finite set of *replacements*

$$\theta = [t_1/x_1, \dots, t_k/x_k]$$

where  $x_1, \dots, x_k$  are distinct variables and  $t_i \neq x_i$ .

$$f(t, u)\theta = f(t\theta, u\theta) \quad (\text{substitution in } \textit{terms})$$

$$P(t, u)\theta = P(t\theta, u\theta) \quad (\text{in } \textit{literals})$$

$$\{L_1, \dots, L_m\}\theta = \{L_1\theta, \dots, L_m\theta\} \quad (\text{in } \textit{clauses})$$

Slide 802

### Composing Substitutions

Composition of  $\phi$  and  $\theta$ , written  $\phi \circ \theta$ , satisfies for all terms  $t$

$$t(\phi \circ \theta) = (t\phi)\theta$$

Slide 803

It is defined by (for all relevant  $x$ )

$$\phi \circ \theta \stackrel{\text{def}}{=} [(x\phi)\theta / x, \dots]$$

Consequences include  $\theta \circ [] = \theta$ , and *associativity*:

$$(\phi \circ \theta) \circ \sigma = \phi \circ (\theta \circ \sigma)$$

### Most General Unifiers

$\theta$  is a *unifier* of terms  $t$  and  $u$  if  $t\theta = u\theta$ .

$\theta$  is *more general* than  $\phi$  if  $\phi = \theta \circ \sigma$  for some substitution  $\sigma$ .

$\theta$  is *most general* if it is more general than every other unifier.

Slide 804

If  $\theta$  unifies  $t$  and  $u$  then so does  $\theta \circ \sigma$ :

$$t(\theta \circ \sigma) = t\theta\sigma = u\theta\sigma = u(\theta \circ \sigma)$$

A most general unifier of  $f(a, x)$  and  $f(y, g(z))$  is  $[a/y, g(z)/x]$ .

The common instance is  $f(a, g(z))$ .

### The Unification Algorithm

Represent terms by *binary trees*.

Each term is a *Variable*  $x, y \dots$ , *Constant*  $a, b \dots$ , or *Pair*  $(t, t')$

SKETCH OF THE ALGORITHM.

Constants do not unify with different Constants.

Constants do not unify with Pairs.

Variable  $x$  and term  $t$ : unifier is  $[t/x]$ , **unless**  $x$  occurs in  $t$

**Cannot unify  $f(x)$  with  $x$ !**

Slide 805

### The Unification Algorithm: The Case of Two Pairs

$\theta \circ \theta'$  unifies  $(t, t')$  with  $(u, u')$

if  $\theta$  unifies  $t$  with  $u$  and  $\theta'$  unifies  $t'\theta$  with  $u'\theta$ .

*We unify the left sides, then the right sides.*

In an implementation, substitutions are formed by updating pointers.

Composition happens automatically as more pointers are updated.

Slide 806

### Mathematical justification

It's easy to check that  $\theta \circ \theta'$  unifies  $(t, t')$  with  $(u, u')$ :

$$\begin{aligned}
 (t, t')(\theta \circ \theta') &= (t, t')\theta\theta' \\
 &= (t\theta\theta', t'\theta\theta') \\
 &= (u\theta\theta', u'\theta\theta') \\
 &= (u, u')\theta\theta' \\
 &= (u, u')(\theta \circ \theta')
 \end{aligned}$$

$\theta \circ \theta'$  is even a most general unifier, if  $\theta$  and  $\theta'$  are!

Slide 807

### Four Unification Examples

$f(x, b)$	$f(x, x)$	$f(x, x)$	$j(x, x, z)$
$f(a, y)$	$f(a, b)$	$f(y, g(y))$	$j(w, a, h(w))$
$f(a, b)$	<b>None</b>	<b>None</b>	$j(a, a, h(a))$
$[a/x, b/y]$	<b>Fail</b>	<b>Fail</b>	$[a/w, a/x, h(a)/z]$

Remember, the output is a *substitution*.

The algorithm yields a *most general* unifier.

Slide 808

**Theorem-Proving Example 1**

$$(\exists y \forall x R(x, y)) \rightarrow (\forall x \exists y R(x, y))$$

**Slide 809**

After negation, the clauses are  $\{R(x, a)\}$  and  $\{\neg R(b, y)\}$ .

The literals  $R(x, a)$  and  $R(b, y)$  have unifier  $[b/x, a/y]$ .

We have the contradiction  $R(b, a)$  and  $\neg R(b, a)$ .

THE THEOREM IS PROVED BY CONTRADICTION!

**Theorem-Proving Example 2**

$$(\forall x \exists y R(x, y)) \rightarrow (\exists y \forall x R(x, y))$$

**Slide 810**

After negation, the clauses are  $\{R(x, f(x))\}$  and  $\{\neg R(g(y), y)\}$ .

The literals  $R(x, f(x))$  and  $R(g(y), y)$  are not unifiable.

(They fail the *occurs check*.)

We can't get a contradiction. FORMULA IS NOT A THEOREM!

## Variations on Unification

*Efficient unification algorithms*: near-linear time

*Indexing & Discrimination networks*: fast retrieval of a unifiable term

*Associative/commutative unification*

- *Example*: unify  $a + (y + c)$  with  $(c + x) + b$ , get  $[a/x, b/y]$
- Algorithm is very complicated
- The number of unifiers can be exponential

Unification in many other theories (often undecidable!)

Slide 811

### The Binary Resolution Rule

$$\frac{\{B, A_1, \dots, A_m\} \quad \{\neg D, C_1, \dots, C_n\}}{\{A_1, \dots, A_m, C_1, \dots, C_n\}\sigma} \quad \text{provided } B\sigma = D\sigma$$

Slide 901

First, *rename variables apart* in the clauses! For example, given

$$\{P(x)\} \quad \text{and} \quad \{\neg P(g(x))\}$$

rename  $x$  in one of the clauses before attempting unification.

Always use a *most general* unifier (MGU).

### The Factoring Rule

This inference collapses unifiable literals *in one clause*:

$$\frac{\{B_1, \dots, B_k, A_1, \dots, A_m\}}{\{B_1, A_1, \dots, A_m\}\sigma} \quad \text{provided } B_1\sigma = \dots = B_k\sigma$$

Slide 902

*Example:* Prove  $\forall x \exists y \neg(P(y, x) \leftrightarrow \neg P(y, y))$

The clauses are  $\{\neg P(y, a), \neg P(y, y)\} \quad \{P(y, y), P(y, a)\}$

Factoring yields  $\{\neg P(a, a)\} \quad \{P(a, a)\}$

Resolution yields the empty clause!



### A Non-Trivial Proof

$$\exists x [P \rightarrow Q(x)] \wedge \exists x [Q(x) \rightarrow P] \rightarrow \exists x [P \leftrightarrow Q(x)]$$

Clauses are  $\{P, \neg Q(b)\}$   $\{P, Q(x)\}$   $\{\neg P, \neg Q(x)\}$   $\{\neg P, Q(a)\}$

Resolve  $\{P, \underline{\neg Q(b)}\}$  with  $\{P, \underline{Q(x)}\}$  getting  $\{P, P\}$

Factor  $\{P, P\}$  getting  $\{P\}$

Resolve  $\{\neg P, \underline{\neg Q(x)}\}$  with  $\{\neg P, \underline{Q(a)}\}$  getting  $\{\neg P, \neg P\}$

Factor  $\{\neg P, \neg P\}$  getting  $\{\neg P\}$

Resolve  $\{P\}$  with  $\{\neg P\}$  getting  $\square$

Slide 903

### What About Equality?

In theory, it's enough to add the *equality axioms*:

- The *reflexive, symmetric and transitive* laws.
- *Substitution* laws like  $\{x \neq y, f(x) = f(y)\}$  for each  $f$ .
- *Substitution* laws like  $\{x \neq y, \neg P(x), P(y)\}$  for each  $P$ .

In practice, we need something special: the *paramodulation rule*

$$\frac{\{B[t'], A_1, \dots, A_m\} \quad \{t = u, C_1, \dots, C_n\}}{\{B[u], A_1, \dots, A_m, C_1, \dots, C_n\}\sigma} \quad (\text{if } t\sigma = t'\sigma)$$

Slide 904

### Prolog Clauses

Prolog clauses have a restricted form, with *at most one* positive literal.

The *definite clauses* form the program. Procedure  $B$  with body “commands”  $A_1, \dots, A_m$  is

$$B \leftarrow A_1, \dots, A_m$$

The single *goal clause* is like the “execution stack”, with say  $m$  tasks left to be done.

$$\leftarrow A_1, \dots, A_m$$

Slide 905

### Prolog Execution

*Linear* resolution:

- Always resolve some program clause with the goal clause.
- The result becomes the new goal clause.

Try the program clauses in *left-to-right* order.

Solve the goal clause’s literals in *left-to-right* order.

Use *depth-first search*. (Performs *backtracking*, using little space.)

Do unification without *occurs check*. (UN SOUND, but needed for speed)

Slide 906

### A (Pure) Prolog Program

```
parent(elizabeth,charles).
parent(elizabeth,andrew).

parent(charles,william).
parent(charles,henry).

parent(andrew,beatrice).
parent(andrew,eugenia).

grand(X,Z) :- parent(X,Y), parent(Y,Z).
cousin(X,Y) :- grand(Z,X), grand(Z,Y).
```

Slide 907

### Prolog Execution

```

                                     :- cousin(X,Y).
                                :- grand(Z1,X), grand(Z1,Y).
        :- parent(Z1,Y2), parent(Y2,X), grand(Z1,Y).
*           :- parent(charles,X), grand(elizabeth,Y).
X=william           :- grand(elizabeth,Y).
                   :- parent(elizabeth,Y5), parent(Y5,Y).
*
Y=beatrice           :- parent(andrew,Y).
                                     :- □.
```

Slide 908

\* = backtracking choice point

**16 solutions** including `cousin(william,william)`  
and `cousin(william,henry)`

### Another FOL Proof Procedure: Model Elimination

A Prolog-like method to run on fast Prolog architectures.

*Contrapositives*: treat clause  $\{A_1, \dots, A_m\}$  like the  $m$  clauses

$$A_1 \leftarrow \neg A_2, \dots, \neg A_m$$

$$A_2 \leftarrow \neg A_3, \dots, \neg A_m, \neg A_1$$

$$\vdots$$

$$A_m \leftarrow \neg A_1, \dots, \neg A_{m-1}$$

*Extension rule*: when proving goal  $P$ , assume  $\neg P$ .

Slide 909

### A Survey of Automatic Theorem Provers

**Saturation** (that is, resolution): E, Gandalf, SPASS, Vampire, ...

**Higher-Order Logic**: TPS, LEO

**Model Elimination**: Prolog Technology Theorem Prover, SETHEO

**Parallel ME**: PARTHENON, PARTHEO

**Tableau (sequent) based**: LeanTAP, 3TAP, ...

Slide 910

## BDDs: Binary Decision Diagrams

A *canonical form* for boolean expressions: decision trees with sharing.

- *ordered* propositional symbols ('variables')
- *sharing* of identical subtrees
- *hashing* and other optimisations

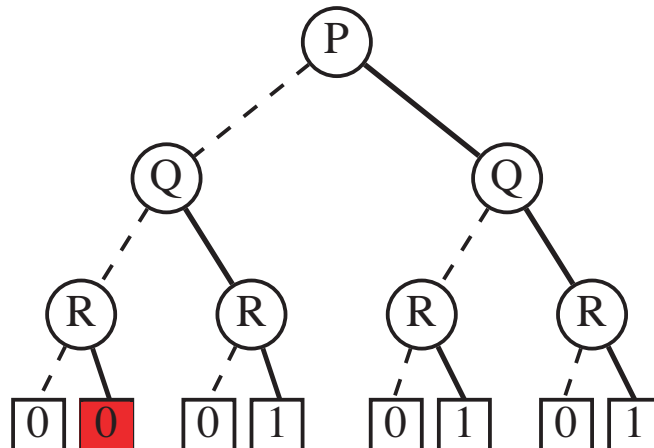
Detects if a formula is tautologous (**t**) or inconsistent (**f**).

Exhibits models if the formula is satisfiable.

Excellent for verifying digital circuits, with many other applications.

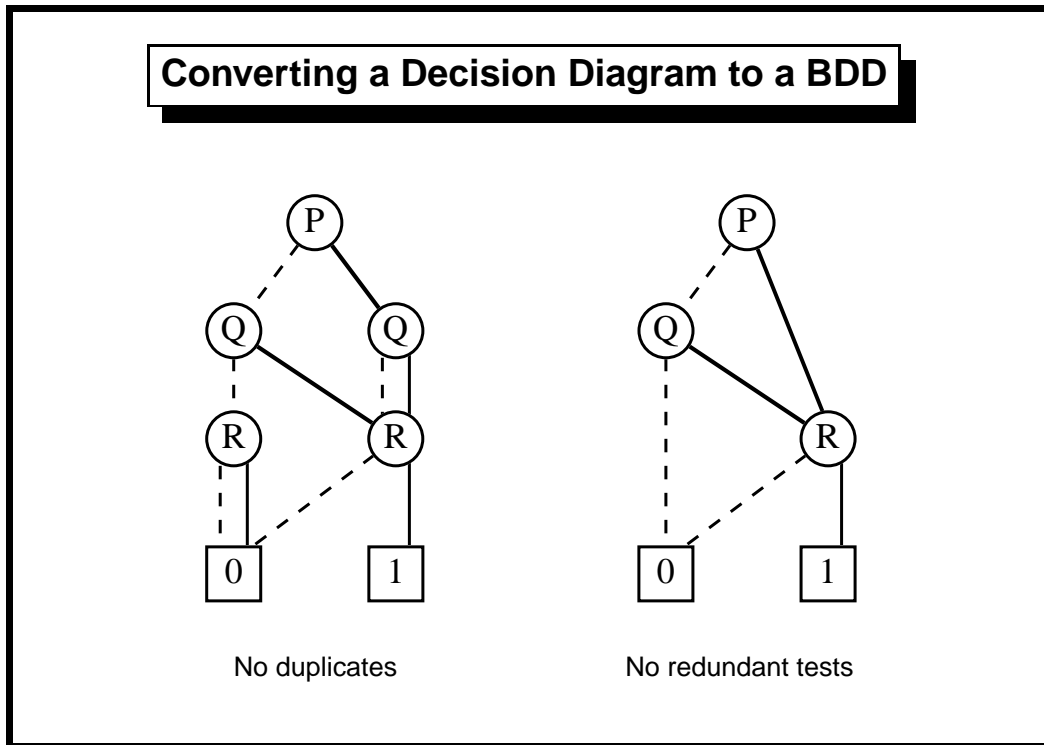
Slide 1001

## Decision Diagram for $(P \vee Q) \wedge R$



Slide 1002

Slide 1003



Slide 1004

### Building BDDs Efficiently

Do not construct full tree!

Do not expand  $\rightarrow$ ,  $\leftrightarrow$ ,  $\oplus$  (exclusive OR) to other connectives.

Treat  $\neg Z$  as  $Z \rightarrow \mathbf{f}$  or  $Z \oplus \mathbf{t}$ .

Recursively convert operands to BDDs.

Combine operand BDDs, respecting the ordering and sharing.

Delete redundant variable tests.

### Canonical Form Algorithm

To convert  $Z \wedge Z'$ , where  $Z$  and  $Z'$  are already BDDs:

*Trivial if either operand is t or f.*

Let  $Z = \text{if}(P, X, Y)$  and  $Z' = \text{if}(P', X', Y')$

- If  $P = P'$  then recursively convert  $\text{if}(P, X \wedge X', Y \wedge Y')$ .
- If  $P < P'$  then recursively convert  $\text{if}(P, X \wedge Z', Y \wedge Z')$ .
- If  $P > P'$  then recursively convert  $\text{if}(P', Z \wedge X', Z \wedge Y')$ .

Slide 1005

### Canonical Forms of Other Connectives

$Z \vee Z'$ ,  $Z \rightarrow Z'$  and  $Z \leftrightarrow Z'$  are converted to BDDs similarly.

Some cases, like  $Z \rightarrow \text{t}$ , reduce to negation.

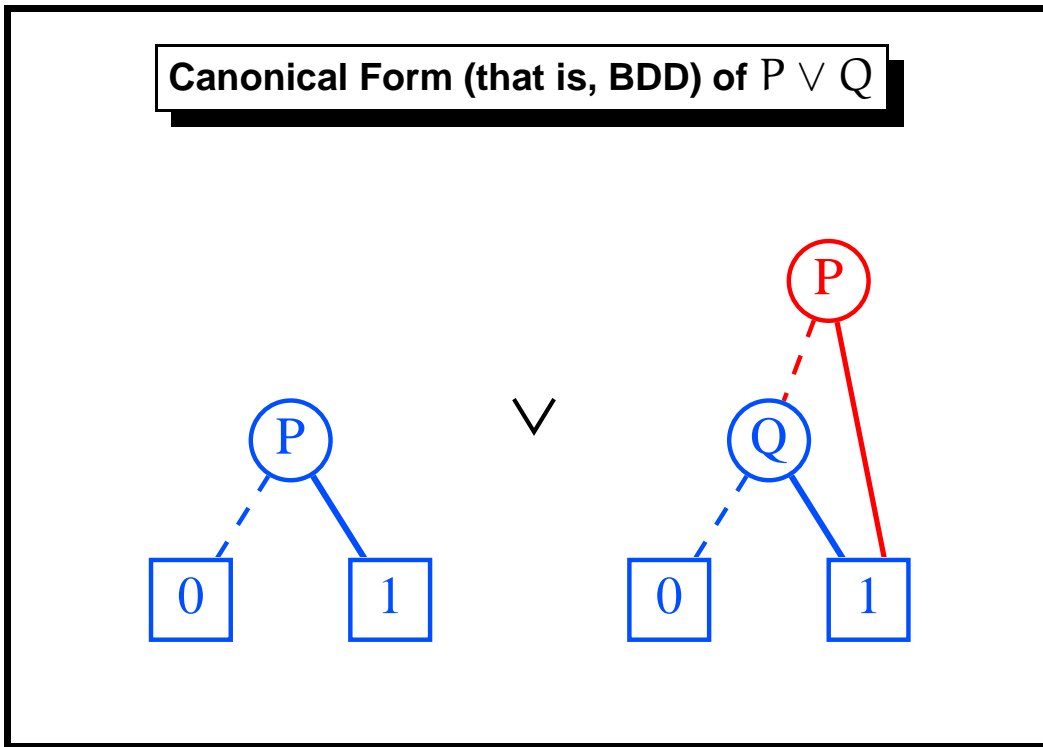
Here is how to convert  $\neg Z$ , where  $Z$  is a BDD:

- If  $Z = \text{if}(P, X, Y)$  then recursively convert  $\text{if}(P, \neg X, \neg Y)$ .
- if  $Z = \text{t}$  then return  $\text{f}$ , and if  $Z = \text{f}$  then return  $\text{t}$ .

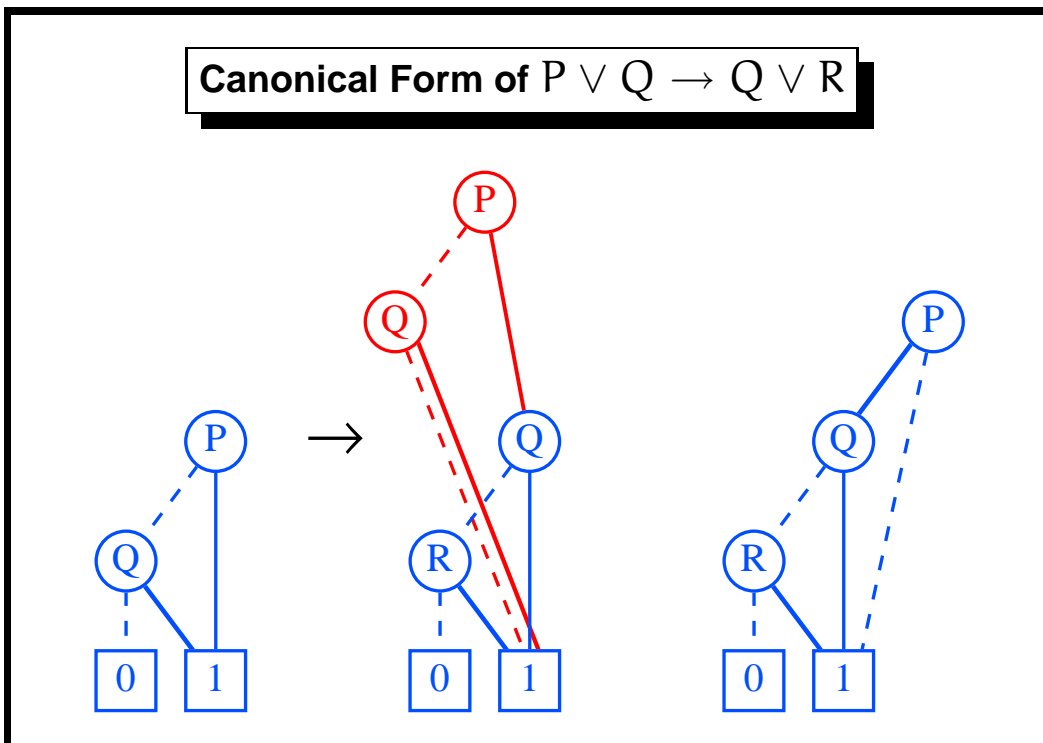
In effect we copy the BDD but swap  $\text{t}$  and  $\text{f}$  at the leaves.

Slide 1006

Slide 1007



Slide 1008





### Optimisations Based On Hash Tables

*Never build the same BDD twice, but share pointers. Advantages:*

- If  $X \simeq Y$ , then the addresses of  $X$  and  $Y$  are equal.
- Can see if  $\text{if}(P, X, Y)$  is redundant by checking if  $X = Y$ .
- Can quickly simplify special cases like  $X \wedge X$ .

*Never convert  $X \wedge Y$  twice, but keep a table of known canonical forms.*

Slide 1009

### Final Observations

The variable ordering is crucial. Consider this formula:

$$(P_1 \wedge Q_1) \vee \cdots \vee (P_n \wedge Q_n)$$

A **good ordering** is  $P_1 < Q_1 < \cdots < P_n < Q_n$ : the BDD is linear.

With  $P_1 < \cdots < P_n < Q_1 < \cdots < Q_n$ , the BDD is EXPONENTIAL.

Many digital circuits have small BDDs: adders, but not multipliers.

BDDs can solve problems in hundreds of variables.

The general case remains hard (it is NP-complete).

Slide 1010

### Modal Operators

$W$ : set of *possible worlds* (machine states, future times, . . .)

$R$ : *accessibility relation* between worlds

$(W, R)$  is called a *modal frame*

$\Box A$  means  $A$  is *necessarily true* } — in all **accessible** worlds  
 $\Diamond A$  means  $A$  is *possibly true* }

$$\neg \Diamond A \simeq \Box \neg A$$

$$A \text{ cannot be true} \iff A \text{ must be false}$$

Slide 1101

### Semantics of Propositional Modal Logic

For a particular frame  $(W, R)$

An *interpretation*  $I$  maps the propositional letters to *subsets* of  $W$

$w \Vdash A$  means  $A$  is true in world  $w$

$$w \Vdash P \iff w \in I(P)$$

$$w \Vdash A \wedge B \iff w \Vdash A \text{ and } w \Vdash B$$

$$w \Vdash \Box A \iff v \Vdash A \text{ for all } v \text{ such that } R(w, v)$$

$$w \Vdash \Diamond A \iff v \Vdash A \text{ for some } v \text{ such that } R(w, v)$$

Slide 1102

### Truth and Validity in Modal Logic

For a particular frame  $(W, R)$ , and interpretation  $I$

$w \Vdash A$  means  $A$  is true in world  $w$

$\models_{W,R,I} A$  means  $w \Vdash A$  for all  $w$  in  $W$

$\models_{W,R} A$  means  $w \Vdash A$  for all  $w$  and all  $I$

$\models A$  means  $\models_{W,R} A$  for all frames;  $A$  is *universally valid*

. . . but typically we constrain  $R$  to be, say, **transitive**

*All tautologies are universally valid*

Slide 1103

### A Hilbert-Style Proof System for $\mathbb{K}$

Extend your favourite propositional proof system with

Dist  $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$

Inference Rule: *Necessitation*

$$\frac{A}{\Box A}$$

Treat  $\Diamond$  as a *definition*

$$\Diamond A \stackrel{\text{def}}{=} \neg \Box \neg A$$

Slide 1104

### Variant Modal Logics

Start with pure modal logic, which is called K

Add *axioms* to constrain the accessibility relation:

- T  $\Box A \rightarrow A$  (*reflexive*) logic T
- 4  $\Box A \rightarrow \Box \Box A$  (*transitive*) logic S4
- B  $A \rightarrow \Box \Diamond A$  (*symmetric*) logic S5

And countless others!

**We shall mainly look at S4**

Slide 1105

### Extra Sequent Calculus Rules for S4

$$\frac{A, \Gamma \Rightarrow \Delta}{\Box A, \Gamma \Rightarrow \Delta} (\Box l) \qquad \frac{\Gamma^* \Rightarrow \Delta^*, A}{\Gamma \Rightarrow \Delta, \Box A} (\Box r)$$

$$\frac{A, \Gamma^* \Rightarrow \Delta^*}{\Diamond A, \Gamma \Rightarrow \Delta} (\Diamond l) \qquad \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \Diamond A} (\Diamond r)$$

$$\Gamma^* \stackrel{\text{def}}{=} \{\Box B \mid \Box B \in \Gamma\} \quad \text{Erase } non\text{-}\Box \text{ assumptions}$$

$$\Delta^* \stackrel{\text{def}}{=} \{\Diamond B \mid \Diamond B \in \Delta\} \quad \text{Erase } non\text{-}\Diamond \text{ goals!}$$

Slide 1106

**A Proof of the Distribution Axiom**

$$\frac{\overline{A \Rightarrow B, A} \quad \overline{B, A \Rightarrow B}}{A \rightarrow B, A \Rightarrow B} (\rightarrow l)$$

$$\frac{A \rightarrow B, A \Rightarrow B}{A \rightarrow B, \Box A \Rightarrow B} (\Box l)$$

$$\frac{A \rightarrow B, \Box A \Rightarrow B}{\Box(A \rightarrow B), \Box A \Rightarrow B} (\Box l)$$

$$\frac{\Box(A \rightarrow B), \Box A \Rightarrow B}{\Box(A \rightarrow B), \Box A \Rightarrow \Box B} (\Box r)$$

Slide 1107

And thus  $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$

**Must** apply ( $\Box r$ ) first!

**Part of an Operator String Equivalence**

$$\frac{\overline{\Diamond A \Rightarrow \Diamond A}}{\Box \Diamond A \Rightarrow \Diamond A} (\Box l)$$

$$\frac{\Box \Diamond A \Rightarrow \Diamond A}{\Diamond \Box \Diamond A \Rightarrow \Diamond A} (\Diamond l)$$

$$\frac{\Diamond \Box \Diamond A \Rightarrow \Diamond A}{\Box \Diamond \Box \Diamond A \Rightarrow \Diamond A} (\Box l)$$

$$\frac{\Box \Diamond \Box \Diamond A \Rightarrow \Diamond A}{\Box \Diamond \Box \Diamond A \Rightarrow \Box \Diamond A} (\Box r)$$

Slide 1108

In fact,  $\Box \Diamond \Box \Diamond A \simeq \Box \Diamond A$  also  $\Box \Box A \simeq \Box A$

The S4 operator strings are  $\Box \Diamond \Box \Diamond \Box \Diamond \Box \Diamond$

### Two Failed Proofs

$$\frac{\Rightarrow A}{\Rightarrow \Diamond A} (\Diamond r)$$

$$\frac{\Rightarrow \Diamond A}{A \Rightarrow \Box \Diamond A} (\Box r)$$

$$\frac{B \Rightarrow A \wedge B}{B \Rightarrow \Diamond(A \wedge B)} (\Diamond r)$$

$$\frac{B \Rightarrow \Diamond(A \wedge B)}{\Diamond A, \Diamond B \Rightarrow \Diamond(A \wedge B)} (\Diamond l)$$

Can extract a countermodel from the proof attempt

Slide 1109

### Simplifying the Sequent Calculus

7 connectives (or 9 for modal logic):

$$\neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftrightarrow \quad \forall \quad \exists \quad (\Box \quad \Diamond)$$

Slide 1201

Left and right: so 14 rules (or 18) plus basic sequent, cut

Idea! Work in **Negation Normal Form**

Fewer connectives:  $\wedge \quad \vee \quad \forall \quad \exists \quad (\Box \quad \Diamond)$

Sequents need *one side only!*

### Simplified Calculus: Left-Only

$$\frac{}{\neg A, A, \Gamma \Rightarrow} \text{ (basic)} \quad \frac{\neg A, \Gamma \Rightarrow \quad A, \Gamma \Rightarrow}{\Gamma \Rightarrow} \text{ (cut)}$$

$$\frac{A, B, \Gamma \Rightarrow}{A \wedge B, \Gamma \Rightarrow} \text{ } (\wedge l) \quad \frac{A, \Gamma \Rightarrow \quad B, \Gamma \Rightarrow}{A \vee B, \Gamma \Rightarrow} \text{ } (\vee l)$$

$$\frac{A[t/x], \Gamma \Rightarrow}{\forall x A, \Gamma \Rightarrow} \text{ } (\forall l) \quad \frac{A, \Gamma \Rightarrow}{\exists x A, \Gamma \Rightarrow} \text{ } (\exists l)$$

Rule  $(\exists l)$  holds *provided*  $x$  is not free in the conclusion!

Slide 1202

### Left-Only Sequent Rules for S4

$$\frac{A, \Gamma \Rightarrow}{\Box A, \Gamma \Rightarrow} (\Box\text{I}) \quad \frac{A, \Gamma^* \Rightarrow}{\Diamond A, \Gamma \Rightarrow} (\Diamond\text{I})$$

$$\Gamma^* \stackrel{\text{def}}{=} \{\Box B \mid \Box B \in \Gamma\} \quad \text{Erase non-}\Box \text{ assumptions}$$

From 14 (or 18) rules to 4 (or 6)

Left-side only system uses **proof by contradiction**

Right-side only system is an exact *dual*

Slide 1203

### Proving $\forall x (P \rightarrow Q(x)) \Rightarrow P \rightarrow \forall y Q(y)$

Move the right-side formula to the left and convert to NNF:

$$P \wedge \exists y \neg Q(y), \forall x (\neg P \vee Q(x)) \Rightarrow$$

$$\frac{\frac{\frac{\frac{\overline{P, \neg Q(y), \neg P \Rightarrow}}{\overline{P, \neg Q(y), \neg P \vee Q(y) \Rightarrow}} (\vee\text{I})}{\overline{P, \neg Q(y), \forall x (\neg P \vee Q(x)) \Rightarrow}} (\forall\text{I})}{\overline{P, \exists y \neg Q(y), \forall x (\neg P \vee Q(x)) \Rightarrow}} (\exists\text{I})}{\overline{P \wedge \exists y \neg Q(y), \forall x (\neg P \vee Q(x)) \Rightarrow}} (\wedge\text{I})$$

Slide 1204



### Adding Unification

Rule  $(\forall\iota)$  now inserts a **new** free variable:

$$\frac{A[z/x], \Gamma \Rightarrow}{\forall x A, \Gamma \Rightarrow} \quad (\forall\iota)$$

Slide 1205

Let unification instantiate *any free variable*

In  $\neg A, B, \Gamma \Rightarrow$  try unifying  $A$  with  $B$  to make a basic sequent

**Updating a variable affects *entire proof tree***

What about rule  $(\exists\iota)$ ? *Skolemize!*

### Skolemization from NNF

*Don't* pull quantifiers out! Skolemize

$$[\forall y \exists z Q(y, z)] \wedge \exists x P(x) \quad \text{to} \quad [\forall y Q(y, f(y))] \wedge P(a)$$

Slide 1206

It's better to push quantifiers in (called **miniscoping**)

*Example:* proving  $\exists x \forall y [P(x) \rightarrow P(y)]$ :

$$\text{Negate; convert to NNF:} \quad \forall x \exists y [P(x) \wedge \neg P(y)]$$

$$\text{Push in the } \exists y : \quad \forall x [P(x) \wedge \exists y \neg P(y)]$$

$$\text{Push in the } \forall x : \quad (\forall x P(x)) \wedge (\exists y \neg P(y))$$

$$\text{Skolemize:} \quad \forall x P(x) \wedge \neg P(a)$$

### A Proof of $\exists x \forall y [P(x) \rightarrow P(y)]$

Slide 1207

$$\begin{array}{r}
 \frac{y \mapsto f(z)}{P(y), \neg P(f(y)), P(z), \neg P(f(z)) \Rightarrow} \text{(basic)} \\
 \frac{}{P(y), \neg P(f(y)), P(z) \wedge \neg P(f(z)) \Rightarrow} (\wedge I) \\
 \frac{}{P(y), \neg P(f(y)), \forall x [P(x) \wedge \neg P(f(x))] \Rightarrow} (\forall I) \\
 \frac{}{P(y) \wedge \neg P(f(y)), \forall x [P(x) \wedge \neg P(f(x))] \Rightarrow} (\wedge I) \\
 \frac{}{\forall x [P(x) \wedge \neg P(f(x))] \Rightarrow} (\forall I)
 \end{array}$$

*Unification* chooses the term for  $(\forall I)$

### A Failed Proof

Try to prove  $\forall x [P(x) \vee Q(x)] \Rightarrow \forall x P(x) \vee \forall x Q(x)$

*NNF*:  $\exists x \neg P(x) \wedge \exists x \neg Q(x), \forall x [P(x) \vee Q(x)] \Rightarrow$

*Skolemize*:  $\neg P(a) \wedge \neg Q(b), \forall x [P(x) \vee Q(x)] \Rightarrow$

Slide 1208

$$\begin{array}{r}
 \frac{y \mapsto a}{\neg P(a), \neg Q(b), P(y) \Rightarrow} \quad \frac{y \mapsto b???}{\neg P(a), \neg Q(b), Q(y) \Rightarrow} \\
 \frac{}{\neg P(a), \neg Q(b), P(y) \vee Q(y) \Rightarrow} (\vee I) \\
 \frac{}{\neg P(a), \neg Q(b), \forall x [P(x) \vee Q(x)] \Rightarrow} (\forall I) \\
 \frac{}{\neg P(a) \wedge \neg Q(b), \forall x [P(x) \vee Q(x)] \Rightarrow} (\wedge I)
 \end{array}$$

### The World's Smallest Theorem Prover?

```

prove((A,B),UnExp,Lits,FreeV,VarLim) :- !,           and
    prove(A,[B|UnExp],Lits,FreeV,VarLim).
prove((A;B),UnExp,Lits,FreeV,VarLim) :- !,         or
    prove(A,UnExp,Lits,FreeV,VarLim),
    prove(B,UnExp,Lits,FreeV,VarLim).
prove(all(X,Fml),UnExp,Lits,FreeV,VarLim) :- !,     forall
    \+ length(FreeV,VarLim),
    copy_term((X,Fml,FreeV),(X1,Fml1,FreeV)),
    append(UnExp,[all(X,Fml)],UnExp1),
    prove(Fml1,UnExp1,Lits,[X1|FreeV],VarLim).
prove(Lit,_,[L|Lits],_,_) :-                       literals; negation
    (Lit = -Neg; -Lit = Neg) ->
    (unify(Neg,L); prove(Lit,[],Lits,_,_)).
prove(Lit,[Next|UnExp],Lits,FreeV,VarLim) :-       next formula
    prove(Next,UnExp,[Lit|Lits],FreeV,VarLim).

```

Slide 1209