Logic and Proof

Computer Science Tripos Part IB
Michaelmas Term

Lawrence C Paulson
Computer Laboratory
University of Cambridge

lp15@cam.ac.uk

Contents

Slide 101

Introduction to Logic

Logic concerns *statements* in some *language*.

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).

Slide 102

Statements

Statements are declarative assertions:

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

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!

Slide 103

Schematic Statements

The *meta-variables* X, Y, Z, ... 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 104

Interpretations and Validity

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

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

Black is the colour of Y*.*

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

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

Slide 105

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.*

Examples of inconsistent sets:

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

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

Satisfiable means the same as consistent.

Unsatisfiable means the same as inconsistent.

Slide 106

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 $S \models A$.

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

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

$S \models A$ 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.

Slide 107

Inference

We want to check A is valid.

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

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

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

We can use inference rules to construct finite proofs!

Slide 108

Schematic Inference Rules

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

A valid inference:

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

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

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

Slide 109

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 110

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 201

Syntax of Propositional Logic

$P, Q, R, \ldots$	propositional letter
$\mathbf{t}$	true
$\mathbf{f}$	false
$\neg A$	not A
$A \land B$	A and B
$A \lor B$	A or B
$A \to B$	if A then B
$A \leftrightarrow B$	A if and only if B

Slide 202

Semantics of Propositional Logic

$\neg$, $\land$, $\lor$, $\to$ and $\leftrightarrow$ are *truth-functional*: functions of their operands.

A	B	$\neg A$	$A \land B$	$A \lor B$	$A \to B$	$A \leftrightarrow B$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$
$\mathbf{t}$	$\mathbf{f}$	$\mathbf{f}$	$\mathbf{f}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{f}$
$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$
$\mathbf{f}$	$\mathbf{f}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$

Slide 203

Interpretations of Propositional Logic

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

Interpretation I *satisfies* a formula A if the formula evaluates to $\mathbf{t}$.

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 204

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 205

Equivalences

$$A \land A \simeq A$$
$$A \land B \simeq B \land A$$
$$(A \land B) \land C \simeq A \land (B \land C)$$
$$A \lor (B \land C) \simeq (A \lor B) \land (A \lor C)$$
$$A \land \mathbf{f} \simeq \mathbf{f}$$
$$A \land \mathbf{t} \simeq A$$
$$A \land \neg A \simeq \mathbf{f}$$

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

Slide 206

Negation Normal Form

1. Get rid of $\leftrightarrow$ and $\rightarrow$, leaving just $\land$, $\lor$, $\neg$:

$$A \leftrightarrow B \simeq (A \rightarrow B) \land (B \rightarrow A)$$
$$A \rightarrow B \simeq \neg A \lor B$$

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

$$\neg\neg A \simeq A$$
$$\neg(A \land B) \simeq \neg A \lor \neg B$$
$$\neg(A \lor B) \simeq \neg A \land \neg B$$

Slide 207

From NNF to Conjunctive Normal Form

3. Push disjunctions in, using distributive laws:

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

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

Slide 208

Converting a Non-Tautology to CNF

$$P \vee Q \to Q \vee R$$

1. Elim $\to$:	$\neg(P \vee Q) \vee (Q \vee R)$
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:	$\neg P \vee Q \vee R$

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

Tautology checking using CNF

$$((P \to Q) \to P) \to P$$

1. Elim $\to$:	$\neg[\neg(\neg P \lor Q) \lor P] \lor P$
2. Push $\neg$ in:	$[\neg\neg(\neg P \lor Q) \land \neg P] \lor P$
	$[(\neg P \lor Q) \land \neg P] \lor P$
3. Push $\lor$ in:	$(\neg P \lor Q \lor P) \land (\neg P \lor P)$
4. Simplify:	$\mathbf{t} \land \mathbf{t}$
	$\mathbf{t}$ *It's a tautology!*

Slide 209

Slide 301

A Simple Proof System

Axiom Schemes

K $A \to (B \to A)$

S $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

DN $\neg\neg A \to A$

Inference Rule: Modus Ponens

$$\frac{A \to B \qquad A}{B}$$

Slide 302

A Simple (?) Proof of $A \to A$

$(A \to ((D \to A) \to A)) \to$		(1)
$((A \to (D \to A)) \to (A \to A))$	by S	
$A \to ((D \to A) \to A)$	by K	(2)
$(A \to (D \to A)) \to (A \to A)$	by MP, (1), (2)	(3)
$A \to (D \to A)$	by K	(4)
$A \to A$	by MP, (3), (4)	(5)

Slide 303

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 $S \vdash A \to B$.

Slide 304

Gentzen's Natural Deduction Systems

The context of *assumptions* may vary.

Each logical connective is defined *independently*.

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

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

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

$$\frac{A \land B}{A} \qquad \frac{A \land B}{B}$$

Slide 305

The Sequent Calculus

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

if $A_1 \land \ldots \land A_m$ then $B_1 \lor \ldots \lor B_n$

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

Γ and Δ are *sets* in $\Gamma \Rightarrow \Delta$

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

Slide 306

Sequent Calculus Rules

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

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

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

Slide 307

More Sequent Calculus Rules

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

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

Slide 308

Easy Sequent Calculus Proofs

$$\cfrac{\cfrac{\overline{A, B \Rightarrow A}}{A \land B \Rightarrow A} \; (\land l)}{\Rightarrow (A \land B) \to A} \; (\to r)$$

$$\cfrac{\cfrac{\cfrac{\overline{A, B \Rightarrow B, A}}{A \Rightarrow B, B \to A} \; (\to r)}{\Rightarrow A \to B, \; B \to A} \; (\to r)}{\Rightarrow (A \to B) \lor (B \to A)} \; (\lor r)$$

Slide 309

Part of a Distributive Law

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow A, B} \qquad \dfrac{\dfrac{}{B, C \Rightarrow A, B}}{B \land C \Rightarrow A, B}\,(\land l)}{A \lor (B \land C) \Rightarrow A, B}\,(\lor l)}{A \lor (B \land C) \Rightarrow A \lor B}\,(\lor r) \qquad \text{similar}}{A \lor (B \land C) \Rightarrow (A \lor B) \land (A \lor C)}\,(\land r)$$

Second subtree proves $A \lor (B \land C) \Rightarrow A \lor C$ similarly

Slide 310

A Failed Proof

$$\dfrac{\dfrac{\dfrac{A \Rightarrow B, C \qquad \dfrac{}{B \Rightarrow B, C}}{A \lor B \Rightarrow B, C}\,(\lor l)}{A \lor B \Rightarrow B \lor C}\,(\lor r)}{\Rightarrow (A \lor B) \to (B \lor C)}\,(\to r)$$

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

Slide 401

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

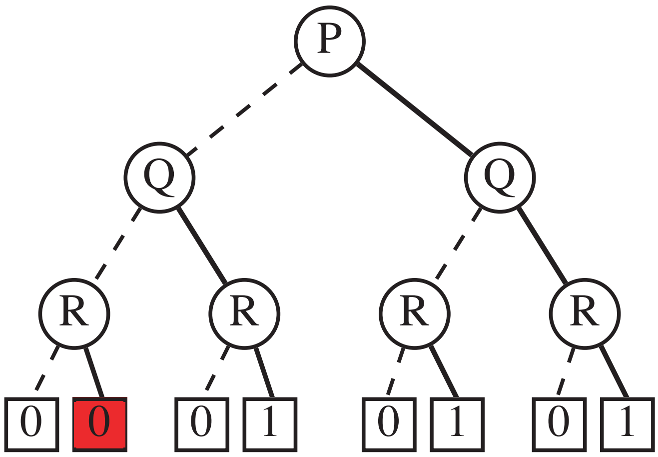
Detects if a formula is tautologous ($\mathbf{t}$) or inconsistent ($\mathbf{f}$).

Exhibits models if the formula is satisfiable.

Excellent for verifying digital circuits, with many other applications.

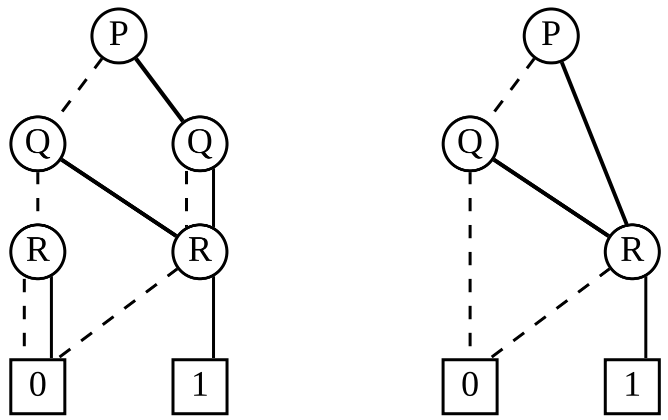
Slide 402

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



Slide 403

Converting a Decision Diagram to a BDD



No duplicates

No redundant tests

Slide 404

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.

Slide 405

Canonical Form Algorithm

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

Trivial if either operand is $\mathbf{t}$ *or* $\mathbf{f}$.

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

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

Slide 406

Canonical Forms of Other Connectives

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

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

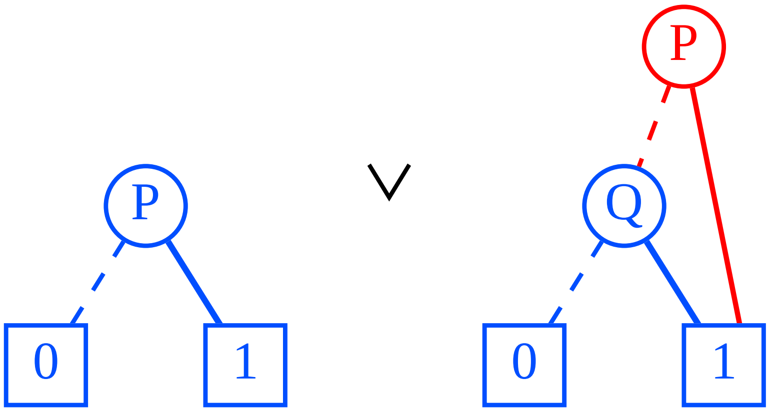
Here is how to convert $\neg Z$, where Z is a BDD:

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

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

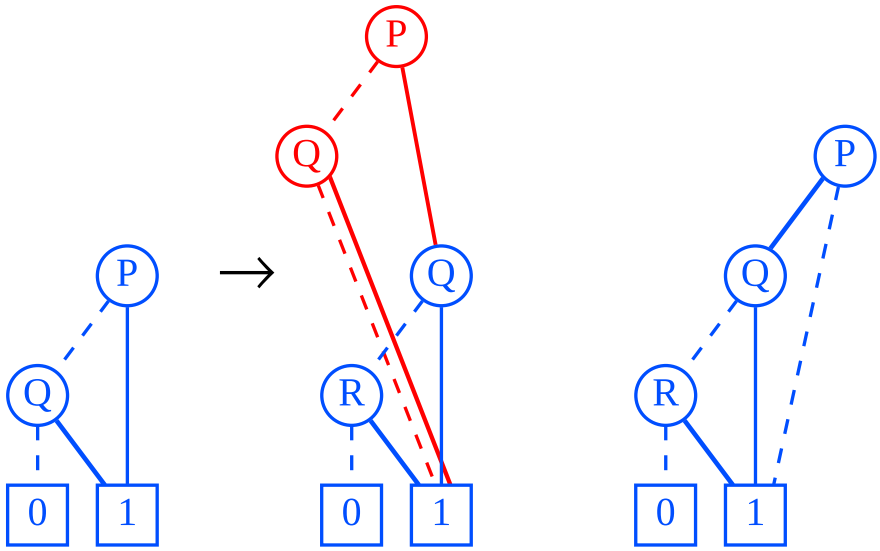
Slide 407

Canonical Form (that is, BDD) of $P \vee Q$



Slide 408

Canonical Form of $P \vee Q \to Q \vee R$



Slide 409

Optimisations Based On Hash Tables

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

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

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

Slide 410

Final Observations

The variable ordering is crucial. Consider this formula:

$$(P_1 \land Q_1) \lor \cdots \lor (P_n \land Q_n)$$

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 501

Outline of First-Order Logic

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

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

Reasons about *all* and *some* individuals:

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

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

Slide 502

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

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

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

We choose the language, adopting any desired function symbols.

Slide 503

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 $R(t_1, \ldots, t_n)$ where R is an n-place relation symbol and t_1, ..., t_n are terms.

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

Later, we can add *quantifiers*.

Slide 504

The Power of Quantifier-Free FOL

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

It is easy to equivalence of mathematical functions:

$$p(z, 0) = 1 \qquad q(z, 1) = z$$

$$p(z, n + 1) = p(z, n) \times z \qquad q(z, 2 \times n) = q(z \times z, n)$$

$$q(z, 2 \times n + 1) = q(z \times z, n) \times z$$

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

Slide 505

Universal and Existential Quantifiers

$\forall x\, A$	for all x, the formula A holds
$\exists x\, A$	there exists x such that A holds

Syntactic variations:

$\forall xyz\, A$	abbreviates $\forall x\, \forall y\, \forall z\, A$
$\forall z\,.\, A \land B$	is an alternative to $\forall z\, (A \land B)$

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

Slide 506

The Expressiveness of Quantifiers

All men are mortal:

$$\forall x\, (\mathsf{man}(x) \to \mathsf{mortal}(x))$$

All mothers are female:

$$\forall x\, \mathsf{female}(\mathsf{mother}(x))$$

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

$$\exists x\, [A(x) \land \forall y\, (A(y) \to y = x)]$$

Slide 507

How do we interpret $\mathsf{mortal}(\mathsf{Socrates})$?

Take an interpretation $\mathcal{I} = (D, I)$ of our 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	$I[f] \in D^n \to D$
P an n-place relation symbol	$I[P] \subseteq D^n$

Slide 508

How do we interpret $\mathsf{cousin}(\mathsf{Charles}, y)$?

A *valuation* supplies the values of free variables.

It is a function $V : \text{variables} \to D$.

$\mathcal{I}_V[t]$ extends V to a term t by the obvious recursion:

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

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

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

Slide 509

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 $\mathcal{I}_V[u]$
$\models_{\mathcal{I},V} A \land B$	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.

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

Slide 601

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 602

Substitution for Free Variables

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

$$\begin{array}{rcll} (B \land C)[t/x] & \textit{is} & B[t/x] \land C[t/x] & \\ (\forall x\, B)[t/x] & \textit{is} & \forall x\, B & \\ (\forall y\, B)[t/x] & \textit{is} & \forall y\, B[t/x] & \quad (x \neq y) \\ (P(u))[t/x] & \textit{is} & P(u[t/x]) & \end{array}$$

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 603

Some Equivalences for Quantifiers

$$\neg(\forall x\, A) \simeq \exists x\, \neg A$$
$$\forall x\, A \simeq \forall x\, A \land A[t/x]$$
$$(\forall x\, A) \land (\forall x\, B) \simeq \forall x\, (A \land B)$$

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

Dual versions: exchange $\forall$ with $\exists$ and $\land$ with $\lor$

Slide 604

Further Quantifier Equivalences

These hold only if x is not free in B.

$$(\forall x\, A) \land B \simeq \forall x\, (A \land B)$$
$$(\forall x\, A) \lor B \simeq \forall x\, (A \lor B)$$
$$(\forall x\, A) \to B \simeq \exists x\, (A \to B)$$

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

Slide 605

Reasoning by Equivalences

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

$$\begin{aligned}&\exists z\,(P(z) \to P(a) \land P(b))\\ &\qquad \simeq \forall z\,P(z) \to P(a) \land P(b)\\ &\qquad \simeq \forall z\,P(z) \land P(a) \land P(b) \to P(a) \land P(b)\\ &\qquad \simeq \mathbf{t}\end{aligned}$$

Slide 606

Sequent Calculus Rules for $\forall$

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

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

$$\frac{\overline{P(y) \Rightarrow P(y)}}{P(y) \Rightarrow \forall y\,P(y)}\ (\forall r)$$

THIS IS NONSENSE!

Slide 607

A Simple Example of the $\forall$ Rules

$$\cfrac{\cfrac{\cfrac{}{P(f(y)) \Rightarrow P(f(y))}}{\forall x\, P(x) \Rightarrow P(f(y))}\,(\forall l)}{\forall x\, P(x) \Rightarrow \forall y\, P(f(y))}\,(\forall r)$$

Slide 608

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

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{P \Rightarrow Q(y), P} \qquad \cfrac{}{P, Q(y) \Rightarrow Q(y)}}{P,\ P \to Q(y) \Rightarrow Q(y)}\,(\to l)}{P,\ \forall x\, (P \to Q(x)) \Rightarrow Q(y)}\,(\forall l)}{P,\ \forall x\, (P \to Q(x)) \Rightarrow \forall y\, Q(y)}\,(\forall r)}{\forall x\, (P \to Q(x)) \Rightarrow P \to \forall y\, Q(y)}\,(\to r)$$

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

Slide 609

Sequent Calculus Rules for $\exists$

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

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:

$$\exists z\, (P(z) \to P(a) \land P(b))$$

Slide 610

Part of the $\exists$ Distributive Law

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(x) \Rightarrow P(x), Q(x)}}{P(x) \Rightarrow P(x) \lor Q(x)}\ (\lor r)}{P(x) \Rightarrow \exists y\, (P(y) \lor Q(y))}\ (\exists r)}{\exists x\, P(x) \Rightarrow \exists y\, (P(y) \lor Q(y))}\ (\exists l) \qquad \dfrac{\text{similar}}{\exists x\, Q(x) \Rightarrow \exists y\ \ldots}\ (\exists l)}{\exists x\, P(x) \lor \exists x\, Q(x) \Rightarrow \exists y\, (P(y) \lor Q(y))}\ (\lor l)$$

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

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

Slide 611

A Failed Proof

$$\dfrac{\dfrac{\dfrac{\dfrac{P(x), Q(y) \Rightarrow P(x) \land Q(x)}{P(x), Q(y) \Rightarrow \exists z\,(P(z) \land Q(z))}\,(\exists r)}{P(x), \exists x\, Q(x) \Rightarrow \exists z\,(P(z) \land Q(z))}\,(\exists l)}{\exists x\, P(x), \exists x\, Q(x) \Rightarrow \exists z\,(P(z) \land Q(z))}\,(\exists l)}{\exists x\, P(x) \land \exists x\, Q(x) \Rightarrow \exists z\,(P(z) \land Q(z))}\,(\land l)$$

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

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

Slide 701

Clause Form

Clause: a disjunction of *literals*

$$\neg K_1 \lor \cdots \lor \neg K_m \lor L_1 \lor \cdots \lor L_n$$

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

Kowalski notation: $K_1, \cdots, K_m \to L_1, \cdots, L_n$

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

Empty clause: $\Box$

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

Slide 702

Outline of Clause Form Methods

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

1. Translate $\neg A$ into CNF as $A_1 \land \cdots \land A_m$
2. This is the set of clauses $A_1, \ldots, A_m$
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 703

The Davis-Putnam-Logeman-Loveland Method

1. Delete tautological clauses: $\{P, \neg P, \ldots\}$
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 704

Davis-Putnam on a Non-Tautology

Consider $P \vee Q \to Q \vee R$

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 705

Example of a Case Split on P

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

$\{\neg Q, R\}$	$\{\neg R, Q\}$	$\{Q, R\}$	$\{\neg Q\}$	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 706

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 707

The Resolution Rule

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

In set notation,

$$\frac{\{B, A_1, \ldots, A_m\} \qquad \{\neg B, C_1, \ldots, C_n\}}{\{A_1, \ldots, A_m, C_1, \ldots, C_n\}}$$

Some special cases:

$$\frac{\{B\} \qquad \{\neg B, C_1, \ldots, C_n\}}{\{C_1, \ldots, C_n\}} \qquad\qquad \frac{\{B\} \qquad \{\neg B\}}{\Box}$$

Slide 708

Simple Example: Proving $P \wedge Q \to Q \wedge P$

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

1. Negate!	$\neg[P \wedge Q \to Q \wedge P]$
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 $\Box$: we have refuted the negation.

Slide 709

Another Example

Refute $\neg[(P \lor Q) \land (P \lor R) \to P \lor (Q \land R)]$

From $(P \lor Q) \land (P \lor R)$, get clauses $\{P, Q\}$ and $\{P, R\}$.

From $\neg\, [P \lor (Q \land R)]$ 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 $\Box$, contradiction.

Slide 710

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 711

Refinements of Resolution

Subsumption:	deleting redundant clauses
Preprocessing:	removing tautologies, symmetries . . .
Indexing:	elaborate data structures for speed
Ordered resolution:	restrictions to focus the search
Weighting:	giving priority to the smallest clauses
Set of Support:	working on the goal, not the axioms

Slide 801

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 802

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)

$$(\forall x\, A) \land B \simeq \forall x\, (A \land B)$$

$$(\forall x\, A) \lor B \simeq \forall x\, (A \lor B)$$

and the similar rules for $\exists$

Slide 803

Skolemization, or Getting Rid of $\exists$

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

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

Choose a fresh k-place function symbol, say f

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

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

Repeat until no $\exists$ quantifiers remain

Slide 804

Example of Conversion to Clauses

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

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

Slide 805

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 $D \to D$ 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 806

The Herbrand Universe for a set of clauses S

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

$$H_{i+1} \stackrel{\text{def}}{=} H_i \cup \{f(t_1, \ldots, t_n) \mid t_1, \ldots, t_n \in H_i$$
$$\text{and } f \text{ is an } n\text{-place function symbol in } S\}$$

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

H contains the terms expressible using the function symbols of S.

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

Slide 807

Herbrand Interpretations for a set of clauses S

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

$$\text{and } P \text{ is an } n\text{-place predicate symbol in } S\}$$

HB contains all applications of predicates to elements of H.

Each subset of HB defines the cases where the predicates are true.

A Herbrand model will interpret the predicates by some subset of HB.

It will interpret function symbols by term-forming operations:

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

Slide 808

Example of an Herbrand Model

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

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

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

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

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

Slide 809

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 810

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

Slide 901

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 902

Substitutions: A Mathematical Treatment

A substitution is a finite set of *replacements*

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

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

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

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

$$\{L_1, \ldots, L_m\}\theta = \{L_1\theta, \ldots, L_m\theta\} \qquad \text{(in \textit{clauses})}$$

Composing Substitutions

Slide 903

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

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

It is defined by (for all relevant x)

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

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

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

Most General Unifiers

Slide 904

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

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

θ is *most general* if it is more general than every other unifier.

If θ 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 $f(a, g(z))$.

Slide 905

The Unification Algorithm

Represent terms by *binary trees*.

Each term is a *Variable* x, y . . ., *Constant* a, b . . ., 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 906

The Unification Algorithm: The Case of Two Pairs

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

if θ unifies t with u and θ' 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 907

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 θ and θ' are!

Slide 908

Four Unification Examples

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

Remember, the output is a *substitution*.

The algorithm yields a *most general* unifier.

Slide 909

Theorem-Proving Example 1

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

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!

Slide 910

Theorem-Proving Example 2

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

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!

Slide 911

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 1001

The Binary Resolution Rule

$$\frac{\{B, A_1, \ldots, A_m\} \qquad \{\neg D, C_1, \ldots, C_n\}}{\{A_1, \ldots, A_m, C_1, \ldots, C_n\}\sigma} \qquad \textit{provided } B\sigma = D\sigma$$

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).

Slide 1002

The Factoring Rule

This inference collapses unifiable literals *in one clause*:

$$\frac{\{B_1, \ldots, B_k, A_1, \ldots, A_m\}}{\{B_1, A_1, \ldots, A_m\}\sigma} \qquad \textit{provided } B_1\sigma = \cdots = B_k\sigma$$

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

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

Resolution yields the empty clause!

Slide 1003

A Non-Trivial Proof

$$\exists x\,[P \to Q(x)] \land \exists x\,[Q(x) \to P] \to \exists x\,[P \leftrightarrow Q(x)]$$

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 1004

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*

$$\frac{\{B[t'], A_1, \ldots, A_m\} \qquad \{t = u, C_1, \ldots, C_n\}}{\{B[u], A_1, \ldots, A_m, C_1, \ldots, C_n\}\sigma} \qquad \textit{(if } t\sigma = t'\sigma\textit{)}$$

Slide 1005

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" $A_1, \ldots, A_m$ is

$$B \leftarrow A_1, \ldots, A_m$$

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

$$\leftarrow A_1, \ldots, A_m$$

Slide 1006

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*. (UNSOUND, but needed for speed)

Slide 1007

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 1008

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).
*                                  :- parent(andrew,Y).
Y=beatrice                                         :- □.
```

* = backtracking choice point

16 solutions including `cousin(william,william)`

and `cousin(william,henry)`

Slide 1009

Another FOL Proof Procedure: Model Elimination

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

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

$$A_1 \leftarrow \neg A_2, \ldots, \neg A_m$$
$$A_2 \leftarrow \neg A_3, \ldots, \neg A_m, \neg A_1$$
$$\vdots$$
$$A_m \leftarrow \neg A_1, \ldots, \neg A_{m-1}$$

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

Slide 1010

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 1101

Modal Operators

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

R: *accessibility relation* between worlds

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

$$\left.\begin{array}{l}\Box A \text{ means } A \text{ is } \textit{necessarily true} \\ \Diamond A \text{ means } A \text{ is } \textit{possibly true}\end{array}\right\} \text{— in all } \textbf{accessible} \text{ worlds}$$

$\neg\Diamond A \simeq \Box\neg A$ $\qquad$ A *cannot be true* $\iff$ A *must be false*

Slide 1102

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

$$\begin{array}{lcl}
w \Vdash P & \iff & w \in I(P) \\
w \Vdash A \land B & \iff & w \Vdash A \text{ and } w \Vdash B \\
w \Vdash \Box A & \iff & v \Vdash A \text{ for all } v \text{ such that } R(w, v) \\
w \Vdash \Diamond A & \iff & v \Vdash A \text{ for some } v \text{ such that } R(w, v)
\end{array}$$

Slide 1103

Truth and Validity in Modal Logic

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

$$
\begin{array}{ll}
w \Vdash A & \text{means } A \text{ is true in world } w \\
\models_{W,R,I} A & \text{means } w \Vdash A \text{ for all } w \text{ in } W \\
\models_{W,R} A & \text{means } w \Vdash A \text{ for all } w \text{ and all } I
\end{array}
$$

$\models A$ 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 1104

A Hilbert-Style Proof System for K

Extend your favourite propositional proof system with

$$\text{Dist} \quad \Box(A \to B) \to (\Box A \to \Box B)$$

Inference Rule: *Necessitation*

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

Treat $\Diamond$ as a *definition*

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

Slide 1105

Variant Modal Logics

Start with pure modal logic, which is called K

Add *axioms* to constrain the accessibility relation:

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

And countless others!

We shall mainly look at $S4$

Slide 1106

Extra Sequent Calculus Rules for $S4$

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

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

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

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

Slide 1107

A Proof of the Distribution Axiom

$$\cfrac{\cfrac{\cfrac{\cfrac{\overline{A \Rightarrow B, A} \qquad \overline{B, A \Rightarrow B}}{A \to B, A \Rightarrow B}\,(\to l)}{A \to B, \Box A \Rightarrow B}\,(\Box l)}{\Box(A \to B), \Box A \Rightarrow B}\,(\Box l)}{\Box(A \to B), \Box A \Rightarrow \Box B}\,(\Box r)$$

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

Must apply $(\Box r)$ first!

Slide 1108

Part of an Operator String Equivalence

$$\cfrac{\cfrac{\cfrac{\cfrac{\overline{\Diamond A \Rightarrow \Diamond A}}{\Box\Diamond A \Rightarrow \Diamond A}\,(\Box l)}{\Diamond\Box\Diamond A \Rightarrow \Diamond A}\,(\Diamond l)}{\Box\Diamond\Box\Diamond A \Rightarrow \Diamond A}\,(\Box l)}{\Box\Diamond\Box\Diamond A \Rightarrow \Box\Diamond A}\,(\Box r)$$

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

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

Slide 1109

Two Failed Proofs

$$\dfrac{\dfrac{\Rightarrow A}{\Rightarrow \Diamond A}\,(\Diamond r)}{A \Rightarrow \Box\Diamond A}\,(\Box r)$$

$$\dfrac{\dfrac{B \Rightarrow A \land B}{B \Rightarrow \Diamond(A \land B)}\,(\Diamond r)}{\Diamond A, \Diamond B \Rightarrow \Diamond(A \land B)}\,(\Diamond l)$$

Can extract a countermodel from the proof attempt

Slide 1201

Simplifying the Sequent Calculus

7 connectives *(or 9 for modal logic)*:

$$\neg \quad \land \quad \lor \quad \to \quad \leftrightarrow \quad \forall \quad \exists \qquad (\Box \quad \Diamond)$$

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

Idea! Work in **Negation Normal Form**

Fewer connectives: $\land \quad \lor \quad \forall \quad \exists \qquad (\Box \quad \Diamond)$

Sequents need *one side only!*

Slide 1202

Simplified Calculus: Left-Only

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

$$\frac{A, B, \Gamma \Rightarrow}{A \land B, \Gamma \Rightarrow} (\land l) \qquad \frac{A, \Gamma \Rightarrow \qquad B, \Gamma \Rightarrow}{A \lor B, \Gamma \Rightarrow} (\lor l)$$

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

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

Slide 1203

Left-Only Sequent Rules for $S4$

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

$\Gamma^* \stackrel{\text{def}}{=} \{\Box B \mid \Box B \in \Gamma\}$ Erase non-$\Box$ 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 1204

Proving $\forall x\,(P \to Q(x)) \Rightarrow P \to \forall y\, Q(y)$

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

$P \land \exists y\, \neg Q(y),\ \forall x\,(\neg P \lor Q(x)) \Rightarrow$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{P,\ \neg Q(y),\ \neg P \Rightarrow} \qquad \overline{P,\ \neg Q(y),\ Q(y) \Rightarrow}}{P,\ \neg Q(y),\ \neg P \lor Q(y) \Rightarrow}\ (\lor l)}{P,\ \neg Q(y),\ \forall x\,(\neg P \lor Q(x)) \Rightarrow}\ (\forall l)}{P,\ \exists y\,\neg Q(y),\ \forall x\,(\neg P \lor Q(x)) \Rightarrow}\ (\exists l)}{P \land \exists y\,\neg Q(y),\ \forall x\,(\neg P \lor Q(x)) \Rightarrow}\ (\land l)$$

Slide 1205

Adding Unification

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

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

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 $(\exists l)$? *Skolemize!*

Slide 1206

Skolemization from NNF

Don't pull quantifiers out! Skolemize

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

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

Example: proving $\exists x\, \forall y\, [P(x) \to P(y)]$:

Negate; convert to NNF:	$\forall x\, \exists y\, [P(x) \land \neg P(y)]$
Push in the $\exists y$:	$\forall x\, [P(x) \land \exists y\, \neg P(y)]$
Push in the $\forall x$:	$(\forall x\, P(x)) \land (\exists y\, \neg P(y))$
Skolemize:	$\forall x\, P(x) \land \neg P(a)$

Slide 1207

A Proof of $\exists x\, \forall y\, [P(x) \to P(y)]$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{y \mapsto f(z)}{P(y),\ \neg P(f(y)),\ P(z),\ \neg P(f(z)) \Rightarrow}\ \text{(basic)}}{P(y),\ \neg P(f(y)),\ P(z) \land \neg P(f(z)) \Rightarrow}\ (\land l)}{P(y),\ \neg P(f(y)),\ \forall x\, [P(x) \land \neg P(f(x))] \Rightarrow}\ (\forall l)}{P(y) \land \neg P(f(y)),\ \forall x\, [P(x) \land \neg P(f(x))] \Rightarrow}\ (\land l)}{\forall x\, [P(x) \land \neg P(f(x))] \Rightarrow}\ (\forall l)
$$

Unification chooses the term for $(\forall l)$

Slide 1208

A Failed Proof

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

NNF: $\exists x\, \neg P(x) \land \exists x\, \neg Q(x),\ \forall x\, [P(x) \lor Q(x)] \Rightarrow$

Skolemize: $\neg P(a) \land \neg Q(b),\ \forall x\, [P(x) \lor Q(x)] \Rightarrow$

$$
\dfrac{\dfrac{\dfrac{\dfrac{y \mapsto a}{\neg P(a),\ \neg Q(b),\ P(y) \Rightarrow} \qquad \dfrac{y \mapsto b???}{\neg P(a),\ \neg Q(b),\ Q(y) \Rightarrow}}{\neg P(a),\ \neg Q(b),\ P(y) \lor Q(y) \Rightarrow}\ (\lor l)}{\neg P(a),\ \neg Q(b),\ \forall x\, [P(x) \lor Q(x)] \Rightarrow}\ (\forall l)}{\neg P(a) \land \neg Q(b),\ \forall x\, [P(x) \lor Q(x)] \Rightarrow}\ (\land l)
$$

Slide 1209

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).
```