# Contents

1. Introduction .................................................................................. 1
2. Propositional Logic ........................................................................ 6
3. Gentzen’s Logical Calculi ................................................................. 11
4. Ordered Binary Decision Diagrams .............................................. 16
5. First-Order Logic .......................................................................... 21
6. Formal Reasoning in First-Order Logic ........................................ 26
7. Davis-Putnam & Propositional Resolution ..................................... 31
8. Skolem Functions and Herbrand’s Theorem .................................. 36
9. Unification ..................................................................................... 41
10. Resolution and Prolog .................................................................. 46
11. Modal Logics ................................................................................. 51
12. Tableaux-Based Methods ............................................................. 56
Logic concerns statements in some language

The language can be informal (e.g. English) or formal

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

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

Logical proofs model human reasoning

Statements are declarative assertions:

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

They are not greetings, questions, 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 $X$, $Y$, $Z$, ... 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?

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 $\mathcal{A}$ is valid if all interpretations satisfy $\mathcal{A}$. 
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:

\[
\{X \text{ part of } Y, \ Y \text{ part of } Z, \ X \text{ NOT part of } Z\}
\]

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

satisfiable/unsatisfiable = consistent/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 $S \models A$.

\[
\{X \text{ part of } Y, \ Y \text{ part of } Z\} \models X \text{ part of } Z
\]

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

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

$\models A$ if and only if $A$ is valid
Inference

Want to check $A$ is valid

Checking all interpretations can be effective — but 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

\[
\begin{array}{c}
A_1 \\
\vdots \\
A_n \\
\hline
B
\end{array}
\]

Use inferences to construct finite proofs!

Schematic Inference Rules

\[
\begin{array}{c}
X \text{ part of } Y \\
Y \text{ part of } Z
\end{array}
\Rightarrow
\begin{array}{c}
X \text{ part of } Z
\end{array}
\]

A valid inference:

\[
\begin{array}{c}
\text{spoke part of wheel} \\
\text{wheel part of bike} \\
\hline
\text{spoke part of bike}
\end{array}
\]

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

\[
\begin{array}{c}
\text{cow part of chair} \\
\text{chair part of ant} \\
\hline
\text{cow part of ant}
\end{array}
\]
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. It has been applied to hardware verification.

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

type theories support constructive mathematics.

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.
Syntax of Propositional Logic

- P, Q, R, ... propositional letter
- t true
- f false
- ¬A not A
- A ∧ B A and B
- A ∨ B A or B
- A → B if A then B
- A ↔ B A if and only if B

Semantics of Propositional Logic

¬, ∧, ∨, → and ↔ are truth-functional: functions of their operands

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>¬A</th>
<th>A ∧ B</th>
<th>A ∨ B</th>
<th>A → B</th>
<th>A ↔ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>
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 \( S \)

Implication, Entailment, Equivalence

\( A \rightarrow B \) means simply \( \neg A \lor B \)

\( A \models B \) 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 \( \models A \leftrightarrow B \)
**Equivalences**

\[
\begin{align*}
A \land A & \simeq A \\
A \land B & \simeq B \land A \\
(A \land B) \land C & \simeq A \land (B \land C) \\
A \lor (B \land C) & \simeq (A \lor B) \land (A \lor C) \\
A \land \overline{A} & \simeq \overline{f} \\
A \land \top & \simeq A \\
A \land \neg A & \simeq \overline{f}
\end{align*}
\]

Dual versions: exchange \( \land, \lor \) and \( \top, \overline{f} \) in any equivalence

---

**Negation Normal Form**

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

\[
\begin{align*}
A \leftrightarrow B & \simeq (A \rightarrow B) \land (B \rightarrow A) \\
A \rightarrow B & \simeq \neg A \lor B
\end{align*}
\]

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

\[
\begin{align*}
\neg \neg A & \simeq A \\
\neg (A \land B) & \simeq \neg A \lor \neg B \\
\neg (A \lor B) & \simeq \neg A \land \neg B
\end{align*}
\]
From NNF to Conjunctive Normal Form

3. Push disjunctions in, using distributive laws:

\[ A \lor (B \land C) \simeq (A \lor B) \land (A \lor C) \]
\[ (B \land C) \lor A \simeq (B \lor A) \land (C \lor A) \]

4. Simplify:

- Delete any disjunction containing \( P \) and \( \neg P \)
- Delete any disjunction that includes another
- Replace \((P \lor A) \land (\neg P \lor A)\) by \( A \)

Converting a Non-Tautology to CNF

1. Elim \( \rightarrow \):
\[ \neg(P \lor Q) \lor (Q \lor R) \]

2. Push \( \neg \) in:
\[ (\neg P \land \neg Q) \lor (Q \lor R) \]

3. Push \( \lor \) in:
\[ (\neg P \lor Q \lor R) \land (\neg Q \lor Q \lor R) \]

4. Simplify:
\[ \neg P \lor Q \lor R \]

Not a tautology: try \( P \leftrightarrow t, \ Q \leftrightarrow f, \ R \leftrightarrow f \)
Tautology checking using CNF

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

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

2. Push $\neg$ in: $[\neg\neg(P \lor Q) \land \neg P] \lor P$
   $$[(P \lor Q) \land \neg P] \lor P$$

3. Push $\lor$ in: $(P \lor Q \lor P) \land (\neg P \lor P)$

4. Simplify: $t \land t$
   $$t$$
   It's a tautology!
A Simple Proof System

Axiom Schemes

K  \( A \rightarrow (B \rightarrow A) \)
S  \( (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \)
DN \( \neg \neg A \rightarrow A \)

Inference Rule: Modus Ponens

\[
\begin{array}{c}
A \rightarrow B \\
A
\end{array} \\
\hline
B
\]

A Simple (?) Proof of \( A \rightarrow A \)

\begin{align*}
(A \rightarrow ((D \rightarrow A) \rightarrow A)) & \rightarrow & (1) \\
((A \rightarrow (D \rightarrow A)) \rightarrow (A \rightarrow A)) & \text{ by S} \\
A \rightarrow ((D \rightarrow A) \rightarrow A) & \text{ by K} & (2) \\
(A \rightarrow (D \rightarrow A)) \rightarrow (A \rightarrow A) & \text{ by MP, (1), (2)} & (3) \\
A \rightarrow (D \rightarrow A) & \text{ by K} & (4) \\
A \rightarrow A & \text{ by MP, (3), (4)} & (5)
\end{align*}
Some Facts about Deducibility

A is *deducible from* the set S of 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 $S \vdash A \rightarrow B$.

Gentzen’s Natural Deduction Systems

A varying context of *assumptions*

Each logical connective defined *independently*

*Introduction* rule for $\land$: how to deduce $A \land B$

\[
\frac{A \quad B}{A \land B}
\]

*Elimination* rules for $\land$: what to deduce from $A \land B$

\[
\frac{A \land B}{A} \quad \frac{A \land B}{B}
\]
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

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

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

Sequent Calculus Rules

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

\[
\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta} \quad \text{(-l)}
\]

\[
\frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A} \quad \text{(-r)}
\]

\[
\frac{A, B, \Gamma \Rightarrow \Delta}{A \land B, \Gamma \Rightarrow \Delta} \quad \text{(\lor l)}
\]

\[
\frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, A \land B} \quad \text{(\lor r)}
\]
More Sequent Calculus Rules

\[
\frac{A, \Gamma \Rightarrow \Delta \quad B, \Gamma \Rightarrow \Delta}{A \lor B, \Gamma \Rightarrow \Delta} \quad (\lor L) \\
\frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, A \lor B} \quad (\lor R)
\]

\[
\frac{\Gamma \Rightarrow \Delta, A \quad B, \Gamma \Rightarrow \Delta}{A \rightarrow B, \Gamma \Rightarrow \Delta} \quad (\lor L) \\
\frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \rightarrow B} \quad (\lor R)
\]

Easy Sequent Calculus Proofs

\[
\frac{A, B \Rightarrow A}{A, B \Rightarrow A} \quad (\land L) \\
\frac{A \land B \Rightarrow A}{\Rightarrow A \land B \rightarrow A} \quad (\rightarrow R)
\]

\[
\frac{A, B \Rightarrow B, A}{A \Rightarrow B, B \rightarrow A} \quad (\rightarrow R) \\
\frac{A \Rightarrow B, B \rightarrow A}{A \Rightarrow B \rightarrow B, B \rightarrow A} \quad (\lor R)
\]

\[
\Rightarrow (A \rightarrow B) \lor (B \rightarrow A)
\]
Part of a Distributive Law

\[ \begin{align*}
A \implies A, B & \quad B, C \implies A, B \quad (\land L) \\
A \land (B \land C) \implies A, B & \quad B \land C \implies A, B \quad (\lor L) \\
A \lor (B \land C) \implies A, B & \quad (\lor R) \\
A \lor (B \land C) \implies A \lor B & \quad \text{similar} \quad \land R \\
A \lor (B \land C) \implies (A \lor B) \land (A \lor C) & \quad \text{similarly}
\end{align*} \]

Second subtree proves \( A \lor (B \land C) \implies A \lor C \)

A Failed Proof

\[ \begin{align*}
A \implies B, C & \quad B \implies B, C \quad (\lor L) \\
A \lor B \implies B, C & \quad (\lor R) \\
A \lor B \implies B \lor C & \quad (\rightarrow R) \\
\implies A \lor B \rightarrow B \lor C & \quad \text{similarly}
\end{align*} \]

\( A \mapsto t, \ B \mapsto f, \ C \mapsto f \) falsifies unproved sequent!
Ordered Binary Decision Diagrams

Canonical form: essentially 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)

A **FAST** way of verifying digital circuits, . . .

---

Decision Diagram for \((P \lor Q) \land R\)

```
         P
        /\  
       Q  Q
      / \ / \  
     R  R R  R
    / \ / \ / \
   0  0 0 1 0 1 0 1
```
Converting a Decision Diagram to an OBDD

No duplicates
No redundant tests

Building OBDDs Efficiently

Do not construct full tree! (see Bryant, §3.1)
Do not expand $\rightarrow$, $\leftrightarrow$, $\oplus$ (exclusive OR) to other connectives
Treat $\neg Z$ as $Z \rightarrow \text{f}$ or $Z \oplus \text{t}$
Recursively convert operands
Combine operand OBDDs — respecting ordering and sharing
Delete test if it proves to be redundant
**Canonical Form Algorithm**

To do $Z \land Z'$, where $Z$ and $Z'$ are already canonical:

*Trivial if either is $t$ or $f$. Treat $\lor$, $\rightarrow$, $\leftrightarrow$ similarly!*

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

If $P = P'$ then recursively do $\text{if}(P, X \land X', Y \land Y')$

If $P < P'$ then recursively do $\text{if}(P, X \land Z', Y \land Z')$

If $P > P'$ then recursively do $\text{if}(P', Z \land X', Z \land Y')$

---

**Canonical Form of $P \lor Q$**

![Diagram showing the canonical form of $P \lor Q$]
Canonical Form of $P \lor Q \rightarrow Q \lor R$

Optimisations Based On Hash Tables

Never build the same OBDD twice: share pointers

- Pointer identity: $X = Y$ whenever $X \leftrightarrow Y$
- Fast removal of redundant tests by $\text{if}(P, X, X) \simeq X$
- Fast processing of $X \land X, X \lor X, X \rightarrow X, \ldots$

Never process $X \land Y$ twice; keep table of canonical forms
The variable ordering is crucial. Consider

\[(P_1 \land Q_1) \lor \cdots \lor (P_n \land Q_n)\]

A good ordering is \(P_1 < Q_1 < \cdots < P_n < Q_n\)

A dreadful ordering is \(P_1 < \cdots < P_n < Q_1 < \cdots < Q_n\)

Many digital circuits have small OBDDs \((not\ multiplication!)\)

OBDDs can solve problems in hundreds of variables

General case remains intractable!
Outline of First-Order Logic

Reasons about functions and relations over a set of individuals:

\[ \text{father} (\text{father} (x)) = \text{father} (\text{father} (y)) \]

\[ \text{cousin} (x, y) \]

Reasons about all and some individuals:

- All men are mortal
- Socrates is a man
- Socrates is mortal

Does not reason about all functions or all relations, ...
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 \( t_1, \ldots, t_n \) are terms

A formula is built up from atomic formulæ using \( \neg, \land, \lor, \ldots \)

(Later we add quantifiers)

Power of Quantifier-Free FOL

Very expressive, given strong induction rules

Prove equivalence of mathematical functions:

\[
\begin{align*}
p(z, 0) &= 1 \\
p(z, n + 1) &= p(z, n) \times z \\
q(z, 1) &= z \\
q(z, 2 \times n) &= q(z \times z, n) \\
q(z, 2 \times n + 1) &= q(z \times z, n) \times z
\end{align*}
\]

Boyer/Moore Theorem Prover: checked Gödel's Theorem, \ldots

Many systems based on equational reasoning
Universal and Existential Quantifiers

\( \forall x \ A \) for all \( x \), \( A \) 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 \( \int f(x) \, dx \)

Expressiveness of Quantifiers

All men are mortal:

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

All mothers are female:

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

There exists a unique \( x \) such that \( A \), written \( \exists! x \ A \)

\( \exists x [A(x) \land \forall y (A(y) \rightarrow y = x)] \)
How do we interpret mortal(Socrates)?

Interpretation $\mathcal{I} = (D, I)$ of our first-order language

$D$ is a non-empty universe

$I$ maps symbols to 'real' functions, 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 \[ I[P] \subseteq D^n \]

How do we interpret cousin(Charles, y)?

A valuation supplies the values of free variables

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

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

- $\mathcal{I}_V[x] \overset{\text{def}}{=} V(x)$ if $x$ is a variable
- $\mathcal{I}_V[c] \overset{\text{def}}{=} I[c]$
- $\mathcal{I}_V[f(t_1, \ldots, t_n)] \overset{\text{def}}{=} I[f](\mathcal{I}_V[t_1], \ldots, \mathcal{I}_V[t_n])$
The Meaning of Truth — in FOL

For interpretation $\mathcal{I}$ and valuation $V$

$\models_{\mathcal{I}, V} P(t)$ if $I[P](\mathcal{I}[t])$ holds

$\models_{\mathcal{I}, V} t = u$ if $\mathcal{I}[t]$ equals $\mathcal{I}[u]$

$\models_{\mathcal{I}, V} A \land B$ 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 $m \in D$

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

$A$ is satisfiable if $\models_{\mathcal{I}} A$ for some $\mathcal{I}$
Free v 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:

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

May rename bound variables:

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

Substitution for Free Variables

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

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

No variable in $t$ may be bound in $A$!

$$\forall y \ x = y)[y/x] \text{ is not } \forall y \ y = y!$$
Some Equivalences for Quantifiers

\[ \neg (\forall x \ A) \simeq \exists x \neg A \]

\[ (\forall x \ A) \land B \simeq \forall x (A \land B) \]

\[ (\forall x \ A) \lor B \simeq \forall x (A \lor B) \]

\[ (\forall x \ A) \land (\forall x \ B) \simeq \forall x (A \land B) \]

\[ (\forall x \ A) \rightarrow B \simeq \exists x (A \rightarrow B) \]

\[ \forall x \ A \simeq \forall x A \land A[t/x] \]

*Dual versions:* exchange \( \forall, \exists \) and \( \land, \lor \)

Reasoning by Equivalences

\[ \exists x (x = a \land P(x)) \simeq \exists x (x = a \land P(a)) \]

\[ \simeq \exists x (x = a) \land P(a) \]

\[ \simeq P(a) \]

\[ \exists z (P(z) \rightarrow P(a) \land P(b)) \]

\[ \simeq \forall z P(z) \rightarrow P(a) \land P(b) \]

\[ \simeq \forall z P(z) \land P(a) \land P(b) \rightarrow P(a) \land P(b) \]

\[ \simeq t \]
Sequent Calculus Rules for $\forall$

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

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

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

Not allowed to prove

\[ \frac{P(y) \Rightarrow P(y)}{P(y) \Rightarrow \forall y P(y)} \quad (\forall r) \]

Simple Example of the $\forall$ Rules

\[ \frac{P(f(y)) \Rightarrow P(f(y))}{\forall x P(x) \Rightarrow P(f(y))} \quad (\forall l) \]
\[ \frac{\forall x P(x) \Rightarrow P(f(y))}{\forall x P(x) \Rightarrow \forall y P(f(y))} \quad (\forall r) \]
Not-So-Simple Example of the $\forall$ Rules

\[
\begin{align*}
P \Rightarrow Q(y), P & \quad P, Q(y) \Rightarrow Q(y) \\
& \quad P, P \rightarrow Q(y) \Rightarrow Q(y) \\
& \quad P, \forall x (P \rightarrow Q(x)) \Rightarrow Q(y) \\
& \quad P, \forall x (P \rightarrow Q(x)) \Rightarrow \forall y Q(y) \\
\end{align*}
\]

$(\forall 1)$

In $(\forall 1)$ we have replaced $x$ by $y$

Sequent Calculus Rules for $\exists$

\[
\begin{align*}
\Lambda, \Gamma \Rightarrow \Delta & \quad (\exists l) \\
\exists x \Lambda, \Gamma \Rightarrow \Delta & \quad (\exists r)
\end{align*}
\]

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

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

Say, to prove

\[
\exists z (P(z) \rightarrow P(a) \land P(b))
\]
Part of the ∃ Distributive Law

\[
\begin{align*}
P(x) & \Rightarrow P(x), Q(x) \quad (\forall r) \\
P(x) & \Rightarrow P(x) \lor Q(x) \quad (\forall r) \\
P(x) & \Rightarrow \exists y \left( P(y) \lor Q(y) \right) \quad (\exists r) \\
\exists x P(x) & \Rightarrow \exists y \left( P(y) \lor Q(y) \right) \quad (\exists 1) \\
\exists x P(x) \lor \exists x Q(x) & \Rightarrow \exists y \left( P(y) \lor Q(y) \right) \quad (\lor 1)
\end{align*}
\]

Second subtree proves \( \exists x Q(x) \Rightarrow \exists y \left( P(y) \lor Q(y) \right) \) similarly

In (\( \exists r \)) we have replaced \( y \) by \( x \)

A Failed Proof

\[
\begin{align*}
P(x), Q(y) & \Rightarrow P(x) \land Q(x) \quad (\exists r) \\
P(x), Q(y) & \Rightarrow \exists z \left( P(z) \land Q(z) \right) \quad (\exists 1) \\
P(x), \exists x Q(x) & \Rightarrow \exists z \left( P(z) \land Q(z) \right) \quad (\exists 1) \\
\exists x P(x), \exists x Q(x) & \Rightarrow \exists z \left( P(z) \land Q(z) \right) \quad (\land 1)
\end{align*}
\]

We cannot use (\( \exists 1 \)) twice with the same variable

We rename the bound variable in \( \exists x Q(x) \) and get \( \exists y Q(y) \)
**Clause Form**

Clause: a disjunction of literals

\[-K_1 \lor \cdots \lor -K_m \lor L_1 \lor \cdots \lor L_n\]

Set notation: \[\{-K_1, \ldots, -K_m, L_1, \ldots, L_n\}\]

Kowalski notation: \[K_1, \ldots, K_m \rightarrow L_1, \ldots, L_n\]

\[L_1, \ldots, L_n \leftarrow K_1, \ldots, K_m\]

Empty clause: \[\square\]

**EMPTY CLAUSE MEANS CONTRADICTION!**

---

**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

Empty clause refutes \(\neg A\)

Empty clause set means \(\neg A\) is satisfiable
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

It's a decision procedure: it finds either a contradiction or a model.

Davis-Putnam on a Non-Tautology

Consider P \lor Q \rightarrow Q \lor 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 t, Q \mapsto f, R \mapsto f
Example of a Case Split on $P$

\[
\{\neg Q, R\} \quad \{\neg R, P\} \quad \{\neg R, Q\} \quad \{\neg P, Q, R\} \quad \{P, Q\} \quad \{\neg P, \neg Q\}
\]

\[
\{\neg Q, R\} \quad \{\neg R, Q\} \quad \{Q, R\} \quad \{\neg Q\} \quad \text{if } P \text{ is true}
\]

\[
\{\neg R\} \quad \{R\} \quad \text{unit } \neg Q
\]

\[
\square \quad \text{unit } R
\]

\[
\{\neg Q, R\} \quad \{\neg R\} \quad \{\neg R, Q\} \quad \{Q\} \quad \text{if } P \text{ is false}
\]

\[
\{\neg Q\} \quad \{Q\} \quad \text{unit } \neg R
\]

\[
\square \quad \text{unit } \neg Q
\]

The Resolution Rule

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

In set notation,

\[
\{B, A_1, \ldots, A_m\} \quad \{\neg B, C_1, \ldots, C_n\}
\]

\[
\{A_1, \ldots, A_m, C_1, \ldots, C_n\}
\]

Some special cases:

\[
\{B\} \quad \{\neg B, C_1, \ldots, C_n\}
\]

\[
\{C_1, \ldots, C_n\}
\]

\[
\square
\]

\[
\{B\} \quad \{\neg B\}
\]

\[
\square
\]
**Simple Example: Proving** \( P \land Q \rightarrow Q \land P \)

Hint: use \( \neg(A \rightarrow B) \simeq A \land \neg B \)

1. Negate! \( \neg[P \land Q \rightarrow Q \land P] \)
2. Push \( \neg \) in: 
   \( (P \land Q) \land \neg(Q \land P) \)
   
   \( (P \land Q) \land (\neg Q \lor \neg P) \)

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 \( \square \)

**Another Example**

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

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 \( \square \)
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.

Refinements of Resolution

Preprocessing: removing tautologies, symmetries . . .
Set of Support: working from the goal
Weighting: priority to the smallest clauses
Subsumption: deleting redundant clauses
Hyper-resolution: avoiding intermediate clauses
Indexing: data structures for speed
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

Prenex Normal Form

Convert to Negation Normal Form using additionally

\neg (\forall x \ A) \sim \exists x \neg A

\neg (\exists x \ A) \sim \forall x \neg A

Then move quantifiers to the front using

(\forall x \ A) \land B \sim \forall x (A \land B)

(\forall x \ A) \lor B \sim \forall x (A \lor B)

and the similar rules for \exists
Skolemization

Take a formula of the form

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

Choose a new 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

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

$$\forall x \ [P(x) \land \exists y \neg P(y)]$$ 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)

$$\{P(x)\} \quad \{\neg P(f(x))\}$$ Final clauses
**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.

**Herbrand Interpretations for a set of clauses $S$**

$H_0 \overset{\text{def}}{=} \text{the set of constants in } S$

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

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

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

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

and $P$ is an $n$-place predicate symbol in $S$
Example of an Herbrand Model

\[ \begin{align*}
\neg \text{even}(1) \\
\text{even}(2) \\
\text{even}(X \cdot Y) \leftarrow \text{even}(X), \text{even}(Y)
\end{align*} \]

\[ H = \{1, 2, 1 \cdot 1, 1 \cdot 2, 2 \cdot 1, 2 \cdot 2, 1 \cdot (1 \cdot 1), \ldots \} \]
\[ HB = \{\text{even}(1), \text{even}(2), \text{even}(1 \cdot 1), \text{even}(1 \cdot 2), \ldots \} \]
\[ I[\text{even}] = \{\text{even}(2), \text{even}(1 \cdot 2), \text{even}(2 \cdot 1), \text{even}(2 \cdot 2), \ldots \} \]

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

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
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**: and no variables remain: it's propositional!
Unification

Finding a common instance of two terms

- Logic programming (Prolog)
- Polymorphic type-checking (ML)
- Constraint satisfaction problems
- Resolution theorem proving for FOL
- Many other theorem proving methods

Substitutions

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 $t_i \neq x_i$

- $f(t, u)\theta = f(t\theta, u\theta)$  \hspace{1cm} (terms)
- $P(t, u)\theta = P(t\theta, u\theta)$  \hspace{1cm} (literals)
- $\{L_1, \ldots, L_m\}\theta = \{L_1\theta, \ldots, L_m\theta\}$  \hspace{1cm} (clauses)
Composing Substitutions

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

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

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

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

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

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

Most General Unifiers

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

$\theta$ is more general than $\phi$ if $\phi = \theta \circ \sigma$

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

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 $f(a, g(z))$
**Algorithm for Unifying Two Terms**

Represent terms by *binary trees*

Each term is a *Variable* x, y . . ., *Constant* a, b . . ., or *Pair* (t, t’)

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!

---

**Unifying Two Pairs**

θ o θ’ unifies (t, t’) with (u, u’)

if θ unifies t with u and θ’ unifies t’ θ with u’ θ

\[(t, t')(θ o θ’) = (t, t’)(θθ’)
\]

\[= (tθθ’, t’(θθ’))\]

\[= (uθθ’, u’θθ’))\]

\[= (u, u’)(θθ’)\]

\[= (u, u’)(θ o θ’)]\]
### Examples of Unification

<table>
<thead>
<tr>
<th></th>
<th>f(x, b)</th>
<th>f(x, x)</th>
<th>f(x, x)</th>
<th>j(x, x, z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f(a, y)</td>
<td>f(a, b)</td>
<td>f(y, g(y))</td>
<td>j(w, a, h(w))</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>f(a, b)</th>
<th>?</th>
<th>?</th>
<th>j(a, a, h(a))</th>
</tr>
</thead>
</table>

\[ [a/x, b/y] \quad \text{FAIL} \quad \text{FAIL} \quad [a/w, a/x, h(a)/z] \]

We always get a **most general** unifier.

---

### Theorem-Proving Examples

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

Clauses after negation are \( \{R(x, a)\} \) and \( \{\neg R(b, y)\} \)

\( R(x, a) \) and \( R(b, y) \) have unifier \( [b/x, a/y] \): **contradiction!**

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

Clauses after negation are \( \{R(x, f(x))\} \) and \( \{\neg R(g(y), y)\} \)

\( R(x, f(x)) \) and \( R(g(y), y) \) are not unifiable: **occurs check**

Formula is not a theorem!
Variations on Unification

Efficient unification algorithms: near-linear time

Indexing & Discrimination networks: fast retrieval of a unifiable term

Order-sorted unification: type-checking in Haskell

Associative/commutative operators: problems in group theory

Higher-order unification: support $\lambda$-calculus

Boolean unification: reasoning about sets
**Binary Resolution**

\[
\frac{\{B, A_1, \ldots, A_m\} \quad \{-D, C_1, \ldots, C_n\}}{\{A_1, \ldots, A_m, C_1, \ldots, C_n\}_\sigma}
\]

provided \(B_\sigma = D_\sigma\)

First *rename variables apart* in the clauses! — say, to resolve

\[
\{P(x)\} \quad \text{and} \quad \{-P(g(x))\}
\]

Always use a *most general* unifier (MGU)

Soundness? Same argument as for the propositional version

---

**Factorisation**

Collapsing similar literals *in one clause*:

\[
\frac{\{B_1, \ldots, B_k, A_1, \ldots, A_m\}}{\{B_1, A_1, \ldots, A_m\}_\sigma}
\]

provided \(B_1_\sigma = \cdots = B_k_\sigma\)

Normally combined with resolution

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

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

Factoring yields \(\{-P(a, a)\} \quad \{P(a, a)\}\)

Resolution yields the empty clause!
A Non-Trivial Example

\[ \exists x [P \rightarrow Q(x)] \land \exists x [Q(x) \rightarrow P] \rightarrow \exists x [P \leftrightarrow Q(x)] \]

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

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

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

Resolve \{P\} with \{\neg P\} getting \Box

*Implicit factoring:* \{P, P\} \mapsto \{P\}  Many other proofs!

Prolog Clauses and Their Execution

*At most one* positive literal per clause!

*Definite clause* \{\neg A_1, \ldots, \neg A_m, B\} or \(B \leftarrow A_1, \ldots, A_m\).

*Goal clause* \{\neg A_1, \ldots, \neg A_m\} or \(\leftarrow A_1, \ldots, A_m\).

*Linear resolution:* a program clause with last goal clause

*Left-to-right* through program clauses

*Left-to-right* through goal clause’s literals

*Depth-first search:* backtracks, but still incomplete

*Unification without occurs check:* fast, but unsound!
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).

Prolog Execution

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

* = backtracking choice point

16 solutions including cousin(william, william)
and cousin(william, henry)
The Method of Model Elimination

A Prolog-like method; complete for First-Order Logic
Contrapositives: treat clause \{A_1, \ldots, A_m\} as m clauses

\[
\begin{align*}
A_1 & \leftarrow \neg A_2, \ldots, \neg A_m \\
A_2 & \leftarrow \neg A_3, \ldots, \neg A_m, \neg A_1 \\
& \vdots
\end{align*}
\]

Extension rule: when proving goal P, may assume \neg P

A brute force method: efficient but no refinements such as subsumption

A Survey of Automatic Theorem Provers

Hyper-resolution: Otter, Gandalf, SPASS, Vampire, \ldots

Model Elimination: Prolog Technology Theorem Prover, SETHEO

Parallel ME: PARTHENON, PARTHEO

Higher-Order Logic: TPS, LEO

Tableau (sequent) based: LeanTAP, 3TAP, \ldots
Approaches to Equality Reasoning

Equality is reflexive, symmetric, transitive

Equality is substitutive over functions, predicates

- Use specialized prover: Knuth-Bendix, ... 
- Assert axioms directly
- Paramodulation rule

\[
\begin{align*}
&\{B[t], A_1, \ldots, A_m\} \quad \{t = u, C_1, \ldots, C_n\} \\
&\{B[u], A_1, \ldots, A_m, C_1, \ldots, C_n\}
\end{align*}
\]
Modal Operators

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

$R$: accessibility relation between worlds

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

$\square A$ means $A$ is necessarily true \quad in all accessible worlds

$\Diamond A$ means $A$ is possibly true

$\neg \Diamond A \equiv \square \neg A$ \quad $A$ cannot be true $\iff$ $A$ must be false

Semantics of Propositional Modal Logic

For a particular frame $(W, R)$

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

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

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

$w \models A \land B \iff w \models A$ and $w \models B$

$w \models \square A \iff v \models A$ for all $v$ such that $R(w, v)$

$w \models \Diamond A \iff v \models A$ for some $v$ such that $R(w, v)$
Truth and Validity in Modal Logic

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

\[ w \models A \] means \(A\) is true in world \(w\)

\[ \models_{W,R,I} A \] means \(w \models A\) for all \(w\) in \(W\)

\[ \models_{W,R} A \] means \(w \models A\) 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

---

A Hilbert-Style Proof System for \(K\)

Extend your favourite propositional proof system with

\[ \text{Dist} \quad \Box (A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B) \]

Inference Rule: *Necessitation*

\[ \frac{A}{\Box A} \]

Treat \(\Diamond\) as a *definition*

\[ \Diamond A \overset{\text{def}}{=} \neg \Box \neg A \]
Variant Modal Logics

Start with pure modal logic, which is called K

Add axioms to constrain the accessibility relation:

\[ \text{T} \quad \square A \rightarrow A \] (reflexive) logic T
\[ \text{4} \quad \square A \rightarrow \square \square A \] (transitive) logic S4
\[ \text{B} \quad A \rightarrow \square \diamond A \] (symmetric) logic S5

And countless others!

We shall mainly look at S4

Extra Sequent Calculus Rules for S4

\[ \frac{A, \Gamma \Rightarrow \Delta}{\square A, \Gamma \Rightarrow \Delta} \quad (\square 1) \]
\[ \frac{\Gamma^* \Rightarrow \Delta^*, A}{\Gamma \Rightarrow \Delta, \square A} \quad (\square r) \]
\[ \frac{A, \Gamma^* \Rightarrow \Delta^*}{\diamond A, \Gamma \Rightarrow \Delta} \quad (\diamond 1) \]
\[ \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \diamond A} \quad (\diamond r) \]

\[ \Gamma^* \overset{\text{def}}{=} \{ \square B \mid \square B \in \Gamma \} \]

Erase non-\(\square\) assumptions

\[ \Delta^* \overset{\text{def}}{=} \{ \diamond B \mid \diamond B \in \Delta \} \]

Erase non-\(\diamond\) goals!
**A Proof of the Distribution Axiom**

\[
\begin{align*}
A \rightarrow B, & \quad A \Rightarrow B \\
\quad & A \rightarrow B, A \Rightarrow B \quad (\rightarrow l) \\
\quad & A \rightarrow B, \Box A \Rightarrow B \quad (\Box l) \\
\Box (A \rightarrow B), & \quad \Box A \Rightarrow B \quad (\Box l) \\
\Box (A \rightarrow B), & \quad \Box A \Rightarrow \Box B \quad (\Box r)
\end{align*}
\]

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

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

---

**Part of an Operator String Equivalence**

\[
\begin{align*}
\Diamond A \Rightarrow \Diamond A \\
\Box \Diamond A \Rightarrow \Diamond A \quad (\Box l) \\
\Diamond \Box \Diamond A \Rightarrow \Diamond A \\
\Box \Diamond \Box \Diamond A \Rightarrow \Diamond A \\
\Box \Diamond \Box \Diamond A \Rightarrow \Box \Diamond A \quad (\Box r)
\end{align*}
\]

In fact, \(\Box \Diamond \Box \Diamond A \simeq \Box \Diamond A\) \quad also \(\Box \Box A \simeq \Box A\)

The S4 operator strings are \(\Box \Diamond \Box \Diamond \Box \Diamond \Box \Box\)
Two Failed Proofs

\[ \begin{align*}
& \Rightarrow A \\
& \Rightarrow \Diamond A \\
& A \Rightarrow \Box \Diamond A
\end{align*} \]

\[ \begin{align*}
& B \Rightarrow A \land B \\
& B \Rightarrow \Diamond (A \land B) \\
& \Diamond A, \Diamond B \Rightarrow \Diamond (A \land B)
\end{align*} \]

Can extract a countermodel from the proof attempt
Simplifying the Sequent Calculus

7 connectives (or 9 for modal logic):

\[ \neg \land \lor \rightarrow \leftrightarrow \forall \exists (\Box \Diamond) \]

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

Idea! Work in Negation Normal Form

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

Sequents need one side only!

\[
\begin{align*}
\neg A, A, \Gamma & \Rightarrow \quad \text{(basic)} \\
A, \Gamma & \Rightarrow \quad \text{(cut)} \\
A, \land B, \Gamma & \Rightarrow \quad \text{(^\land 1)} \\
A, \lor B, \Gamma & \Rightarrow \quad \text{(\lor t)} \\
A[t/x], \Gamma & \Rightarrow \quad \text{(\forall t)} \\
\forall x A, \Gamma & \Rightarrow \quad \text{(^\forall t)} \\
A, \Gamma & \Rightarrow \quad \text{(^\exists t)} \\
\exists x A, \Gamma & \Rightarrow \quad \text{(\exists t)}
\end{align*}
\]

Rule (\exists t) holds provided \( x \) is not free in the conclusion!
Left-Only Sequent Rules for \( S4 \)

\[
\begin{align*}
A, \Gamma &\Rightarrow \quad (\square l) \\
\square A, \Gamma &\Rightarrow \quad (\square l) \\
\end{align*}
\]

\[
\Gamma^* \overset{\text{def}}{=} \{ \square B \mid \square B \in \Gamma \} \quad \text{Erase non-\( \square \) 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

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:

\[
\begin{align*}
P \land \exists y \neg Q(y), \forall x (\neg P \lor Q(x)) &\Rightarrow \\
\end{align*}
\]

\[
\begin{align*}
P, \neg Q(y), \neg P &\Rightarrow \\
P, \neg Q(y), Q(y) &\Rightarrow \quad (\lor l) \\
P, \neg Q(y), \neg P \lor Q(y) &\Rightarrow \quad (\forall l) \\
P, \neg Q(y), \forall x (\neg P \lor Q(x)) &\Rightarrow \quad (\exists l) \\
P, \exists y \neg Q(y), \forall x (\neg P \lor Q(x)) &\Rightarrow \quad (\land l) \\
P \land \exists y \neg Q(y), \forall x (\neg P \lor Q(x)) &\Rightarrow \\
\end{align*}
\]
Adding Unification

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

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

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!

Skolemization from NNF

Follow tree structure; don’t pull out quantifiers!

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

Better to push quantifiers in (called miniscoping)

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

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: $\forall x P(x) \land \neg P(a)$
A Proof of $\exists x \forall y [P(x) \rightarrow P(y)]$

\[
\begin{align*}
  & y \mapsto f(z) \\
  & \quad P(y), \neg P(f(y)), P(z), \neg P(f(z)) \Rightarrow \quad \text{(basic)} \\
  & \quad P(y), \neg P(f(y)), P(z) \land \neg P(f(z)) \Rightarrow \quad \text{(\&I)} \\
  & \quad P(y) \land \neg P(f(y)), \forall x [P(x) \land \neg P(f(x))] \Rightarrow \quad \text{(\&I)} \\
  & \quad \forall x [P(x) \land \neg P(f(x))] \Rightarrow \quad \text{(\&I)} \\
\end{align*}
\]

Unification chooses the term for (\&I)

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

\[
\begin{align*}
  & y \mapsto a \\
  & \quad \neg P(a), \neg Q(b), P(y) \Rightarrow \quad \neg P(a), \neg Q(b), Q(y) \Rightarrow \quad \text{(\lor I)} \\
  & \quad \neg P(a), \neg Q(b), P(y) \lor Q(y) \Rightarrow \quad \text{(\lor I)} \\
  & \quad \neg P(a), \neg Q(b), \forall x [P(x) \lor Q(x)] \Rightarrow \quad \text{(\&I)} \\
  & \quad \neg P(a) \land \neg Q(b), \forall x [P(x) \lor Q(x)] \Rightarrow \quad \text{(\&I)} \\
\end{align*}
\]
prove((A,B),UnExp,Lits,FreeV,VarLim) :- !,
    prove(A,[B|UnExp],Lits,FreeV,VarLim).
prove((A;B),UnExp,Lits,FreeV,VarLim) :- !,
    prove(A,UnExp,Lits,FreeV,VarLim),
    prove(B,UnExp,Lits,FreeV,VarLim).
prove(all(X,Fml),UnExp,Lits,FreeV,VarLim) :- !,
    \+ 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],_,_) :-
    (Lit = -Neg; -Lit = Neg) ->
        (unify(Neg,L); prove(Lit,[],Lits,_,_)).
prove(Lit,[Next|UnExp],Lits,FreeV,VarLim) :-
    prove(Next,UnExp,[Lit|Lits],FreeV,VarLim).