Logic and Resolution Proof

R1: IF ?x has feathers THEN ?x is a bird
R2: IF ?x flies ?x lays eggs THEN ?x is a bird

Predicate =
Function: Objects → {True, False}

Example predicates:

Feathers(x) ⇒ Bird(x)

(Flies(x) ∧ LaysEggs(x)) ⇒ Bird(x)

¬Feathers (Suzie)

Feathers (Suzie) ⇒ Bird(Suzie)

¬Feathers (Suzie) ∨ Bird(Suzie)

∀x [ Feathers(x) ⇒ Bird(x) ]

Scope of variable x
Logic – Propositional Calculus

No variables allowed. Only objects, e.g., $E_1, E_2$.

**Commutative Laws:**

$E_1 \land E_2 \Leftrightarrow E_2 \land E_1$

$E_1 \lor E_2 \Leftrightarrow E_2 \lor E_1$

**Distributive Laws:**

$E_1 \land (E_2 \lor E_3) \Leftrightarrow (E_1 \land E_2) \lor (E_1 \land E_3)$

$E_1 \lor (E_2 \land E_3) \Leftrightarrow (E_1 \lor E_2) \land (E_1 \lor E_3)$

**Associative Laws:**

$E_1 \land (E_2 \land E_3) \Leftrightarrow (E_1 \land E_2) \land E_3$

$E_1 \lor (E_2 \lor E_3) \Leftrightarrow (E_1 \lor E_2) \lor E_3$

**De Morgan’s Laws:**

$\neg (E_1 \land E_2) \Leftrightarrow (\neg E_1) \lor (\neg E_2)$

$\neg (E_1 \lor E_2) \Leftrightarrow (\neg E_1) \land (\neg E_2)$

**Double Negation Law:**

$\neg (\neg E_1) \Leftrightarrow E_1$

Precedence of operators in following order:

NOT $\neg$, AND $\land$, OR $\lor$, IMPLICATION $\Rightarrow$.

Logic – 1st Order Predicate Calculus

Variables allowed, e.g., $x$. Variables cannot represent predicates $P$. Existential quantifier $\exists$ and universal quantifier $\forall$.

$\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$

$\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$
**Term**
- constant
- variable
- function: term $\rightarrow$ term

**Predicate**
- function: term $\rightarrow \{\text{True, False}\}$

**Atomic formula** = predicate with argument

**Literal** = atomic formula or negated atomic formula

**Well-formed formula (wff)**
- literals
- disjunction: wff $\lor$ wff, conjunction: wff $\land$ wff, negation: $\neg$ wff, implication: wff $\rightarrow$ wff
- $\forall x$ [wff], $\exists x$ [wff]
- clause = wff consisting of a disjunction of literals

**Sentence** = wff with all variables (if any) within scope

Example of sentences:

$$\forall x \ [ \text{Feathers}(x) \Rightarrow \text{Bird}(x) ]$$

Feathers(Albatross) $\Rightarrow$ Bird(Albatross)

Sentence?

$$\forall x \ [ \text{Feathers}(x) \lor \neg \text{Feathers}(y) ]$$

$y$ is free variable
Axioms:

Feathers (Squigs)
\[ \forall x \ [ \text{Feathers}(x) \Rightarrow \text{Bird}(x) \] 

Theorem:

Bird (Squigs)

A **proof** ties axioms to consequences
A **proof** shows theorem is true given axioms
A **proof** needs inference rules to derive new expressions from axioms
A **proof** needs substitution rules to derive expressions from axioms

Substitution rule: **Specialization**

\[ \text{Feathers}(\text{Squigs}) \Rightarrow \text{Bird}(\text{Squigs}) \]

Inference rule: **Modus Ponens**
If axioms of form \( E_1 \Rightarrow E_2 \) and \( E_1 \) are given, then \( E_2 \) is a new true expression.

\[ \begin{align*}
\text{Feathers (Squigs)} \\
\text{Feathers(Squigs) \Rightarrow Bird(Squigs)} \\
\text{Bird (Squigs)}
\end{align*} \]

Inference rule: **Resolution**
**Resolution**

Axiom 1  \(E_1 \lor E_2\)
Axiom 2  \(\neg E_2 \lor E_3\)
Resolvent  \(E_1 \lor E_3\)

Modus ponens is a special case of resolution:

Axiom 1  \(\neg E_1 \lor E_2\)
Axiom 2  \(E_1\)
Resolvent  \(E_2\)

Contradiction is a special case of resolution:

Axiom 1  \(\neg E_1\)
Axiom 2  \(E_1\)
Resolvent  NIL

Resolution proof = proof by refutation (= show theorem is false)
Show theorem’s negation cannot be true.

**Example:**

<table>
<thead>
<tr>
<th>Theorem: Bird(Squigs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proof:</strong></td>
</tr>
<tr>
<td>Axiom 1</td>
</tr>
<tr>
<td>Specialized Axiom 2</td>
</tr>
<tr>
<td>Negation of Theorem (step 3)</td>
</tr>
<tr>
<td>Resolvent of 1 &amp; 2 (step 4)</td>
</tr>
<tr>
<td>Resolvent of steps 3 &amp; 4</td>
</tr>
</tbody>
</table>
To prove a theorem using resolution:
– Negate theorem
– Add negated theorem to list of axioms
– Transform axioms into clause form
– REAT UNITL there is no resolvable pair of clauses:
  * Find resolvable clauses and resolve them
  * Add results to list of clauses
  * If NIL produced, STOP. Report theorem is TRUE.
– STOP. Report theorem is FALSE.

Strategies to search for resolvable clauses:
– *Unit-preference strategy:* Clauses with smallest # of literals first
– *Set-support strategy:* Only work with resolutions involving negated theorem or clauses derived from it
– *Breadth-first strategy:* First reduce all possible pairs of initial clauses then all pairs of resulting sets with initial set, level by level

Exponential explosion problem

Halting problem:
Completion of proof procedures is “semidecidable” =
* Guaranteed to find proof if theorem logically follows from axioms
* Search is not guaranteed to terminate unless there is a proof

Informally: “While the search is going on, we don’t know if it hasn’t found the proof yet, or there is no proof.”
Prolog

Programming in logic

Early 1970s in Marseille, France
Late 1970s in Edinburgh, UK, Warren and Pereira

Declarative programming:
– What is true?
– What needs to be done?
(versus procedural programming – how to do it)

Based on 1st-order predicative calculus
Syntax, Semantics
Method of computing: resolution
Transformation Example

Axiom:

\[ \forall x [Brick(x) \Rightarrow (\exists y [On(x, y) \land \neg Pyramid(y)] \land \neg \exists y [On(x, y) \land On(y, x)] \land \forall y [\neg Brick(y) \Rightarrow \neg Equal(x, y)]] \]

1. Eliminate implications: Use \((E_1 \Rightarrow E_2) \iff (\neg E_1 \lor E_2)\).

\[ \forall x [\neg Brick(x) \lor (\exists y [On(x, y) \land \neg Pyramid(y)] \land \neg \exists y [On(x, y) \land On(y, x)] \land \forall y [\neg Brick(y) \lor \neg Equal(x, y)]] \]

2. Move negations down to atomic formulas:

\[ \forall x [\neg Brick(x) \lor (\exists y [On(x, y) \land \neg Pyramid(y)] \land \forall y [\neg On(x, y) \lor \neg On(y, x)] \land \forall y [Brick(y) \lor \neg Equal(x, y)]] \]

3. Eliminate existential quantifiers using Skolem functions:

\[ \forall x [\neg Brick(x) \lor (On(x, Support(x)) \land \neg Pyramid(Support(x)) \land \forall y [\neg On(x, y) \lor \neg On(y, x)] \land \forall y [Brick(y) \lor \neg Equal(x, y)]] \]

4. Rename variables:

\[ \forall x [\neg Brick(x) \lor (On(x, Support(x)) \land \neg Pyramid(Support(x)) \land \forall y [\neg On(x, y) \lor \neg On(y, x)] \land \forall z [Brick(z) \lor \neg Equal(x, z)]] \]
Not covered in 2023

4. Rename variables:

\[ \forall x \ [ \neg \text{Brick}(x) \lor (\text{On}(x, \text{Support}(x)) \land \neg \text{Pyramid}(\text{Support}(x)) \land \forall y \ [ \neg \text{On}(x, y) \lor \neg \text{On}(y, x)] \land \forall z \ [ \text{Brick}(z) \lor \neg \text{Equal}(x, z)] ] \]

5. Move universal quantifiers to left:

\[ \forall x \forall y \forall z \ [ \neg \text{Brick}(x) \lor (\text{On}(x, \text{Support}(x)) \land \neg \text{Pyramid}(\text{Support}(x)) \land (\neg \text{On}(x, y) \lor \neg \text{On}(y, x)) \land (\text{Brick}(z) \lor \neg \text{Equal}(x, z)) ] \]

6. Move disjunctions down to literals: Use \( E_1 \lor (E_2 \land E_3) \iff (E_1 \lor E_2) \land (E_1 \lor E_3) \).

\[ \forall x \forall y \forall z \ [ (\neg \text{Brick}(x) \lor (\text{On}(x, \text{Support}(x)) \land \neg \text{Pyramid}(\text{Support}(x)))) \land (\neg \text{Brick}(x) \lor (\neg \text{On}(x, y) \lor \neg \text{On}(y, x))) \land (\neg \text{Brick}(x) \lor \text{Brick}(z) \lor \neg \text{Equal}(x, z)) ] \]

7. Eliminate conjunctions:

\[ \forall x \ [ \neg \text{Brick}(x) \lor \text{On}(x, \text{Support}(x))] \]
\[ \forall x \ [ \neg \text{Brick}(x) \lor \neg \text{Pyramid}(\text{Support}(x))] \]
\[ \forall x \forall y \ [ \neg \text{Brick}(x) \lor \neg \text{On}(x, y) \lor \neg \text{On}(y, x)] \]
\[ \forall x \forall z \ [ \neg \text{Brick}(x) \lor \text{Brick}(z) \lor \neg \text{Equal}(x, z)] \]
7. Eliminate conjunctions:

\[ \forall x \ [ \neg \text{Brick}(x) \lor \text{On}(x, \text{Support}(x))] \]

\[ \forall x \ [ \neg \text{Brick}(x) \lor \neg \text{Pyramid}(\text{Support}(x))] \]

\[ \forall x \forall y \ [ \neg \text{Brick}(x) \lor \neg \text{On}(x, y) \lor \neg \text{On}(y, x)] \]

\[ \forall x \forall z \ [ \neg \text{Brick}(x) \lor \text{Brick}(z) \lor \neg \text{Equal}(x, z)) ] \]

8. Rename variables:

\[ \forall x \ [ \neg \text{Brick}(x) \lor \text{On}(x, \text{Support}(x))] \]

\[ \forall w \ [ \neg \text{Brick}(w) \lor \neg \text{Pyramid}(\text{Support}(w))] \]

\[ \forall u \forall y \ [ \neg \text{Brick}(u) \lor \neg \text{On}(u, y) \lor \neg \text{On}(y, x)] \]

\[ \forall v \forall z \ [ \neg \text{Brick}(v) \lor \text{Brick}(z) \lor \neg \text{Equal}(v, z)) ] \]

9. Eliminate universal quantifiers:

\[ \neg \text{Brick}(x) \lor \text{On}(x, \text{Support}(x)) \]

\[ \neg \text{Brick}(w) \lor \neg \text{Pyramid}(\text{Support}(w)) \]

\[ \neg \text{Brick}(u) \lor \neg \text{On}(u, y) \lor \neg \text{On}(y, x) \]

\[ \neg \text{Brick}(v) \lor \text{Brick}(z) \lor \neg \text{Equal}(v, z)) \]
Planning using Situation Variables

Traditional logic: Predicate $On(A, B)$ either true or false.

Here time dependency on value of $On(A, B, s)$ where $s$ describes the situation.

**Initial situation:**

\[
On(B, A, S) \land On(A, \text{Table}, S)
\]

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

\text{Table}

**Goal situation:**

\[
\exists s_f \, [On(B, \text{Table}, s_f)]
\]

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

\text{Table}

STORE($x, s_i$) puts object $x$ on table and creates situation $s_{i+1}$. It is a function with output $s_{i+1}$; not a predicate with output true or false.

Definition of STORE:

\[
\forall s \forall x \, [\neg On(x, \text{Table}, s) \Rightarrow On(x, \text{Table}, \text{STORE}(x, s))]
\]

Axiom about something not on table:

\[
\forall s \forall y \forall z \, [On(y, z, s) \land \neg Equal(z, \text{Table}) \Rightarrow \neg On(y, \text{Table}, s)]
\]

Is there a way to move B onto table? $\rightarrow$ Turn “resolution crank.”
List of Axioms:

\[ On(B, A, S) \land On(A, \text{Table}, S) \]
\[ \forall s \forall x \left[ \neg On(x, \text{Table}, s) \Rightarrow On(x, \text{Table}, \text{STORE}(x, s)) \right] \]
\[ \forall s \forall y \forall z \left[ On(y, z, s) \land \neg Equal(z, \text{Table}) \Rightarrow \neg On(y, \text{Table}, s) \right] \]

Negation of Theorem:

\[ \neg \exists s_f \left[ On(B, \text{Table}, s_f) \right] \]

After Transformation into Clause Form:

\[ On(B, A, S) \quad (1) \]
\[ On(A, \text{Table}, S) \quad (2) \]
\[ On(x, \text{Table}, s_3) \lor On(x, \text{Table}, \text{STORE}(x, s_3)) \quad (3) \]
\[ \neg On(y, z, s_4) \lor Equal(z, \text{Table}) \lor \neg On(y, \text{Table}, s_4) \quad (4) \]
\[ \neg Equal(B, A) \quad (5) \]
\[ \neg Equal(B, \text{Table}) \quad (6) \]
\[ \neg Equal(A, \text{Table}) \quad (7) \]
\[ \neg On(B, \text{Table}, s_f) \quad (8) \]
Resolution Proof:

(8) \neg On(B, \text{Table}, s_f) \\
\downarrow

(3) On(x, \text{Table}, s_3) \lor
On(x, \text{Table}, \text{STORE}(x, s_3)) \rightarrow (9) On(B, \text{Table}, s_9) \\
\downarrow

(4) \neg On(y, z, s_4) \lor
Equal(z, \text{Table}) \lor
\neg On(y, \text{Table}, s_4) \rightarrow (10) \neg On(B, w, s_{10}) \lor
Equal(w, \text{Table}) \\
\downarrow

(7) \neg Equal(A, \text{Table}) \rightarrow (11) \neg On(B, A, s_{11}) \\
\downarrow

(1) On(B, A, S) \rightarrow (12) NIL
How to get to goal situation $s_f$?

Trace situation history.

$$s_f \rightarrow \text{STORE}(B, s_3)$$

$$s_3 \rightarrow s_9$$

$$s_9 \rightarrow s_{10}$$

$$s_{10} \rightarrow s_{11}$$

$$s_{11} \rightarrow S$$

Tedious $\Rightarrow$

**Green’s trick:** Add extra “Answer” term.  Not covered in 2023
Resolution Proof with Green’s Trick:

\[
\neg On(B, \text{Table}, s_f) \lor \text{Answer}(s_f)
\]
\[
\downarrow
\]
\[
On(x, \text{Table}, s_3) \lor
On(x, \text{Table}, \text{STORE}(x, s_3)) \rightarrow
\]
\[
On(B, \text{Table}, s_9) \lor \text{Answer} (\text{STORE}(B, s_9))
\]
\[
\downarrow
\]
\[
\neg On(y, z, s_4) \lor
Equal(z, \text{Table}) \lor
\neg On(y, \text{Table}, s_4) \rightarrow
\]
\[
\neg On(B, w, s_{10}) \lor
Equal(w, \text{Table}) \lor \text{Answer} (\text{STORE}(B, s_{10}))
\]
\[
\downarrow
\]
\[
\neg Equal(A, \text{Table}) \rightarrow
\]
\[
\neg On(B, A, s_{11}) \lor \text{Answer} (\text{STORE}(B, s_{11}))
\]
\[
\downarrow
\]
\[
On(B, A, S) \rightarrow
\]
\[
\text{Answer} (\text{STORE}(B, S))
\]
New goal situation:

\[ \exists s_f \ [ \text{On}(B, \text{Table}, s_f) \land \text{On}(A, \text{Table}, s_f)] \]

Transformation of negated theorem into clause form:

\[ \neg \exists s_f \ [ \text{On}(B, \text{Table}, s_f) \land \text{On}(A, \text{Table}, s_f)] \]

\[ \forall s_f \ [ \neg (\text{On}(B, \text{Table}, s_f) \land \text{On}(A, \text{Table}, s_f))] \]

\[ \forall s_f \ [ \neg \text{On}(B, \text{Table}, s_f) \lor \neg \text{On}(A, \text{Table}, s_f)] \]

\[ \neg \text{On}(B, \text{Table}, s_f) \lor \neg \text{On}(A, \text{Table}, s_f) \]

List of axioms and negated theorem in clause form:

- \( \text{On}(B, A, S) \)
- \( \text{On}(A, \text{Table}, S) \)
- \( \text{On}(x, \text{Table}, s_3) \lor \text{On}(x, \text{Table}, \text{STORE}(x, s_3)) \)
- \( \neg \text{On}(y, z, s_4) \lor \text{Equal}(z, \text{Table}) \lor \neg \text{On}(y, \text{Table}, s_4) \)
  - \( \neg \text{Equal}(A, B) \)
  - \( \neg \text{Equal}(B, \text{Table}) \)
  - \( \neg \text{Equal}(A, \text{Table}) \)
- \( \neg \text{On}(B, \text{Table}, s_f) \lor \neg \text{On}(A, \text{Table}, s_f) \)

Resolution Proof:

\[ \neg \text{On}(B, \text{Table}, s_f) \]
\[ \lor \neg \text{On}(A, \text{Table}, s_f) \]
\[ \downarrow \]

\[ \text{On}(x, \text{Table}, s_3) \lor \text{On}(x, \text{Table}, \text{STORE}(x, s_3)) \]
\[ \longrightarrow \]

\[ \text{On}(B, \text{Table}, s_9) \]
\[ \lor \neg \text{On}(A, \text{Table}, \text{STORE}(B, s_9)) \]
Resolution procedure gets stuck:

\(\neg \text{On}(B, \text{Table}, s_f) \lor \neg \text{On}(A, \text{Table}, s_f)\)

\(\downarrow\)

\(\text{On}(x, \text{Table}, s_3) \lor \text{On}(x, \text{Table}, \text{STORE}(x, s_3))\)

\(\downarrow\)

\(\text{On}(B, \text{Table}, s_9) \lor \neg \text{On}(A, \text{Table}, \text{STORE}(B, s_9))\)

\(\downarrow\)

\(\neg \text{On}(y, z, s_4) \lor \text{Equal}(z, \text{Table}) \lor \neg \text{On}(y, \text{Table}, s_4)\)

\(\downarrow\)

\(\neg \text{On}(B, w, s_{10}) \lor \text{Equal}(w, \text{Table}) \lor \neg \text{On}(A, \text{Table}, \text{STORE}(B, s_{10}))\)

\(\downarrow\)

\(\neg \text{Equal}(A, \text{Table})\)

\(\downarrow\)

\(\text{On}(B, A, S) \rightarrow \neg \text{On}(A, \text{Table}, \text{STORE}(B, S))\)

\(\downarrow\)

\(\text{On}(A, \text{Table}, S) \rightarrow \text{NIL}\)
Not covered in 2023

Solution:

**Frame axioms** = statements about how predicates “survive” operations

If $x$ is on $y$ before STORE operation, then $x$ remains on $y$ afterward, as long as $x$ was not the object put on the table:

$$\forall s \forall x \forall y \forall z [On(x, y, s) \land \neg \text{Equal}(x, z) \Rightarrow On(x, y, \text{STORE}(z, s))]$$

Convert to frame axiom:

$$\neg On(p, q, s_0) \lor \text{Equal}(p, r) \lor On(p, q, \text{STORE}(r, s_0))$$

Previously stuck at (12):

$$\neg On(A, \text{Table}, \text{STORE}(B, S))$$

Resolve (12) with frame axiom:

$$\neg On(A, \text{Table}, S) \lor \text{Equal}(A, B) \quad (13)$$

Resolve (13) with (5) $\neg \text{Equal}(A, B)$:

$$\neg On(A, \text{Table}, S) \quad (14)$$

Resolve (14) with (2) $On(A, \text{Table}, S)$:

$$NIL$$