## 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 $\longrightarrow$ {True, False}

Example predicates:

$$\text{Feathers}(x) \Rightarrow \text{Bird}(x)$$

$$(\text{Flies}(x) \wedge \text{LaysEggs}(x)) \Rightarrow \text{Bird}(x)$$

$$\neg\text{Feathers (Suzie)}$$

$$\text{Feathers (Suzie)} \Rightarrow \text{Bird(Suzie)}$$

$$\neg\text{Feathers (Suzie)} \vee \text{Bird(Suzie)}$$

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

Scope of variable $x$

# Logic – Propositional Calculus

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

$$
\begin{aligned}
&\textbf{Commutative Laws}: \\
&\qquad E_1 \wedge E_2 \quad \Leftrightarrow \quad E_2 \wedge E_1 \\
&\qquad E_1 \vee E_2 \quad \Leftrightarrow \quad E_2 \vee E_1 \\
&\textbf{Distributive Laws}: \\
&\qquad E_1 \wedge (E_2 \vee E_3) \quad \Leftrightarrow \quad (E_1 \wedge E_2) \vee (E_1 \wedge E_3) \\
&\qquad E_1 \vee (E_2 \wedge E_3) \quad \Leftrightarrow \quad (E_1 \vee E_2) \wedge (E_1 \vee E_3) \\
&\textbf{Associative Laws}: \\
&\qquad E_1 \wedge (E_2 \wedge E_3) \quad \Leftrightarrow \quad (E_1 \wedge E_2) \wedge E_3 \\
&\qquad E_1 \vee (E_2 \vee E_3) \quad \Leftrightarrow \quad (E_1 \vee E_2) \vee E_3 \\
&\textbf{De Morgan's Laws}: \\
&\qquad \neg\,(E_1 \wedge E_2) \quad \Leftrightarrow \quad (\neg E_1) \vee (\neg E_2) \\
&\qquad \neg\,(E_1 \vee E_2) \quad \Leftrightarrow \quad (\neg E_1) \wedge (\neg E_2) \\
&\textbf{Double Negation Law}: \\
&\qquad \neg\,(\neg\, E_1) \quad \Leftrightarrow \quad E_1
\end{aligned}
$$

Precedence of operators in following order:

NOT $\neg$, AND $\wedge$, OR $\vee$, IMPLICATION $\Rightarrow$.

# Logic – 1st Order Predicate Calculus

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

$$
\begin{aligned}
\neg\,\forall x P(x) \quad &\Leftrightarrow \quad \exists x\, \neg P(x) \\
\neg\,\exists x P(x) \quad &\Leftrightarrow \quad \forall x\, \neg P(x)
\end{aligned}
$$

**Term**
– constant
– variable
– function: term $\rightarrow$ term

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

**Atomic formula** = predicate with argument

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

**Well-formed formula** (wff)
– literals
– disjunction: wff $\vee$ wff, conjunction: wff $\wedge$ 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:

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

$$\text{Feathers(Albatross)} \Rightarrow \text{Bird(Albatross)}\,]$$

Sentence?

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

$y$ is **free** variable

**Axioms:**

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

**Theorem:**

$$\text{Bird (Squigs)}$$

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**

$$\text{Feathers(Squigs)} \Rightarrow \text{Bird(Squigs)}$$

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.

$$\begin{array}{c} \text{Feathers (Squigs)} \\ \underline{\text{Feathers(Squigs)} \Rightarrow \text{Bird(Squigs)}} \\ \text{Bird (Squigs)} \end{array}$$

Inference rule: **Resolution**

**Resolution**

| | |
|---|---|
| Axiom 1 | $E_1 \vee E_2$ |
| Axiom 2 | $\neg E_2 \vee E_3$ |
| Resolvent | $E_1 \vee E_3$ |

Modus ponens is a special case of resolution:

| | |
|---|---|
| Axiom 1 | $\neg E_1 \vee E_2$ |
| 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:

Theorem: Bird(Squigs)
Proof:

| | | |
|---|---|---|
| Axiom 1 | Feathers(Squigs) | |
| Specialized Axiom 2 | ¬ Feathers(Squigs) ∨ | Bird(Squigs) |
| Negation of Theorem (step 3) | | ¬ Bird(Squigs) |
| Resolvent of 1 & 2 (step 4) | | Bird(Squigs) |
| Resolvent of steps 3 & 4 | | NIL |

**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 theroem 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."

Not covered in 2023

## 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

Not covered in 2023

## Transformation Example

Axiom:

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

1. Eliminate implications: Use $(E_1 \Rightarrow E_2) \Leftrightarrow (\neg E_1 \vee E_2)$.

$$\forall x\, [\neg Brick(x) \; \vee \; (\exists y\, [On(x,y) \wedge \neg Pyramid(y)] \; \wedge \\ \neg\exists y\, [On(x,y) \wedge On(y,x)] \; \wedge \\ \forall y\, [\neg\neg Brick(y) \; \vee \; \neg Equal(x,y)]\; )]$$

2. Move negations down to atomic formulas:

$$\forall x\, [\neg Brick(x) \; \vee \; (\exists y\, [On(x,y) \wedge \neg Pyramid(y)] \; \wedge \\ \forall y\, [\neg On(x,y) \vee \neg On(y,x)] \; \wedge \\ \forall y\, [Brick(y) \; \vee \; \neg Equal(x,y)]\; )]$$

3. Eliminate existential quantifiers using Skolem functions:

$$\forall x\, [\neg Brick(x) \; \vee \; (On(x, Support(x)) \wedge \neg Pyramid(Support(x)) \; \wedge \\ \forall y\, [\neg On(x,y) \vee \neg On(y,x)] \; \wedge \\ \forall y\, [Brick(y) \; \vee \; \neg Equal(x,y)]\; )]$$

4. Rename variables:

$$\forall x\, [\neg Brick(x) \; \vee \; (On(x, Support(x)) \wedge \neg Pyramid(Support(x)) \; \wedge \\ \forall y\, [\neg On(x,y) \vee \neg On(y,x)] \; \wedge \\ \forall z\, [Brick(z) \; \vee \; \neg Equal(x,z)]\; )]$$

Not covered in 2023

4. Rename variables:

$$\forall x\,[\neg Brick(x) \;\vee\; (On(x, Support(x)) \wedge \neg Pyramid(Support(x)) \;\wedge$$
$$\forall y\,[\neg On(x,y) \vee \neg On(y,x)] \;\wedge$$
$$\forall z\,[Brick(z) \;\vee\; \neg Equal(x,z)]\,)]$$

5. Move universal quantifiers to left:

$$\forall x \forall y \forall z\,[\neg Brick(x) \vee (On(x, Support(x)) \;\wedge\; \neg Pyramid(Support(x))$$
$$\wedge\,(\neg On(x,y) \;\vee\; \neg On(y,x))$$
$$\wedge\,(Brick(z) \;\vee\; \neg Equal(x,z))\,)]$$

6. Move disjunctions down to literals: Use $E_1 \vee (E_2 \wedge E_3) \Leftrightarrow (E_1 \vee E_2) \wedge (E_1 \vee E_3)$.

$$\forall x \forall y \forall z\,[(\neg Brick(x) \vee (On(x, Support(x)) \wedge \neg Pyramid(Support(x))))$$
$$\wedge(\neg Brick(x) \vee \quad (\neg On(x,y) \vee \neg On(y,x)))$$
$$\wedge(\neg Brick(x) \vee \quad (Brick(z) \vee \neg Equal(x,z))\,)]$$

$$\forall x \forall y \forall z\;[\quad (\neg Brick(x) \vee On(x, Support(x)))$$
$$\wedge(\neg Brick(x) \vee \neg Pyramid(Support(x)))$$
$$\wedge(\neg Brick(x) \vee \neg On(x,y) \vee \neg On(y,x))$$
$$\wedge(\neg Brick(x) \vee Brick(z) \vee \neg Equal(x,z))\;]$$

7. Eliminate conjunctions:

$$\forall x\;[\quad \neg Brick(x) \vee On(x, Support(x))]$$
$$\forall x\;[\;\neg Brick(x) \vee \neg Pyramid(Support(x))]$$
$$\forall x \forall y\;[\;\neg Brick(x) \vee \neg On(x,y) \vee \neg On(y,x)]$$
$$\forall x \forall z\;[\;\neg Brick(x) \vee Brick(z) \vee \neg Equal(x,z))\;]$$

Not covered in 2023

7. Eliminate conjunctions:

$$
\begin{aligned}
\forall x \; [ & \quad \neg Brick(x) \lor On(x, Support(x))] \\
\forall x \; [ & \quad \neg Brick(x) \lor \neg Pyramid(Support(x))] \\
\forall x \forall y \; [ & \quad \neg Brick(x) \lor \neg On(x, y) \lor \neg On(y, x)] \\
\forall x \forall z \; [ & \quad \neg Brick(x) \lor Brick(z) \lor \neg Equal(x, z)) \, ]
\end{aligned}
$$

8. Rename variables:

$$
\begin{aligned}
\forall x \; [ & \quad \neg Brick(x) \lor On(x, Support(x))] \\
\forall w \; [ & \quad \neg Brick(w) \lor \neg Pyramid(Support(w))] \\
\forall u \forall y \; [ & \quad \neg Brick(u) \lor \neg On(u, y) \lor \neg On(y, x)] \\
\forall v \forall z \; [ & \quad \neg Brick(v) \lor Brick(z) \lor \neg Equal(v, z)) \, ]
\end{aligned}
$$

9. Eliminate universal quantifiers:

$$
\begin{gathered}
\neg Brick(x) \lor On(x, Support(x)) \\
\neg Brick(w) \lor \neg Pyramid(Support(w)) \\
\neg Brick(u) \lor \neg On(u, y) \lor \neg On(y, x) \\
\neg Brick(v) \lor Brick(z) \lor \neg Equal(v, z))
\end{gathered}
$$

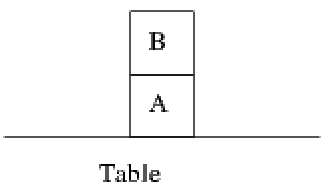
# Planning using Situation Variables

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:

$$On(B, A, S) \quad \wedge \quad On(A, \text{Table}, S)$$




## Goal situation:

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


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:

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

Is there a way to move B onto table? $\rightarrow$ Turn "resolution crank."

**List of Axioms:**

$$On(B, A, S) \quad \wedge \quad On(A, \text{Table}, S)$$

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

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

**Negation of Theorem:**

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

---

**After Transformation into Clause Form:**

$$On(B, A, S) \tag{1}$$

$$On(A, \text{Table}, S) \tag{2}$$

$$On(x, \text{Table}, s_3) \vee On(x, \text{Table}, \text{STORE}(x, s_3)) \tag{3}$$

$$\neg On(y, z, s_4) \vee Equal(z, \text{Table}) \vee \neg On(y, \text{Table}, s_4) \tag{4}$$

$$\neg Equal(B, A) \tag{5}$$

$$\neg Equal(B, \text{Table}) \tag{6}$$

$$\neg Equal(A, \text{Table}) \tag{7}$$

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

**Resolution Proof:**

$$\begin{array}{rcl}
 & & (8) \\
 & & \neg On(B, \text{Table}, s_f) \\
 & & \downarrow \\
(3) & & \\
On(x, \text{Table}, s_3) \quad \vee & & \\
\underline{On(x, \text{Table}, \text{STORE}(x, s_3))} & \longrightarrow & (9) \\
 & & On(B, \text{Table}, s_9) \\
 & & \downarrow \\
(4) & & \\
\neg On(y, z, s_4) \quad \vee & & \\
Equal(z, \text{Table}) \quad \vee & & \\
\underline{\neg On(y, \text{Table}, s_4)} & \longrightarrow & \\
 & & (10) \\
 & & \neg On(B, w, s_{10}) \quad \vee \\
 & & \underline{Equal(w, \text{Table})} \\
 & & \downarrow \\
(7) & & \\
\neg Equal(A, \text{Table}) & \longrightarrow & (11) \\
 & & \neg On(B, A, s_{11}) \\
 & & \downarrow \\
(1) & & \\
On(B, A, S) & \longrightarrow & (12) \\
 & & NIL
\end{array}$$

## How to get to goal situation $s_f$?

Trace situation history.

$$s_f \to \text{STORE}(B, s_3)$$
$$s_3 \to s_9$$
$$s_9 \to s_{10}$$
$$s_{10} \to s_{11}$$
$$s_{11} \to S$$

Tedious $\Rightarrow$

**Green's trick:** Add extra "Answer" term. Not covered in 2023

Not covered in 2023

**Resolution Proof with Green's Trick:**

$$
\begin{array}{rcl}
 & & (8) \\
 & & \neg \mathit{On}(B, \text{Table}, s_f) \lor \mathit{Answer}(s_f) \\
 & & \downarrow \\
(3) & & \\
\mathit{On}(x,\text{Table},s_3) \;\lor & & \\
\underline{\mathit{On}(x,\text{Table},\text{STORE}(x,s_3))} & \longrightarrow & (9) \\
 & & \mathit{On}(B,\text{Table},s_9) \\
 & & \lor \mathit{Answer}(\text{STORE}(B,s_9)) \\
 & & \downarrow \\
(4) & & \\
\neg \mathit{On}(y,z,s_4) \;\lor & & \\
\mathit{Equal}(z,\text{Table}) \;\lor & & \\
\underline{\neg \mathit{On}(y,\text{Table},s_4)} & \longrightarrow & \\
 & & (10) \\
 & & \neg \mathit{On}(B,w,s_{10}) \;\lor \\
 & & \underline{\mathit{Equal}(w,\text{Table})} \\
 & & \lor \mathit{Answer}(\text{STORE}(B,s_{10})) \\
 & & \downarrow \\
(7) & & \\
\neg \mathit{Equal}(A,\text{Table}) & \longrightarrow & (11) \\
 & & \neg \mathit{On}(B,A,s_{11}) \\
 & & \lor \mathit{Answer}(\text{STORE}(B,s_{11})) \\
 & & \downarrow \\
(1) & & \\
\mathit{On}(B,A,S) & \longrightarrow & (12) \\
 & & \mathit{Answer}(\text{STORE}(B,S))
\end{array}
$$

Not covered in 2023

**New goal situation:**

$$\exists s_f \ [\mathrm{On}(\mathrm{B}, \mathrm{Table}, s_f) \wedge \mathrm{On}(\mathrm{A}, \mathrm{Table}, s_f)]$$

**Transformation of negated theorem into clause form:**

$$\begin{array}{ll} \neg \exists s_f \ [\ \mathrm{On}(\mathrm{B},\mathrm{Table},s_f) & \wedge \ \mathrm{On}(\mathrm{A},\mathrm{Table},s_f)] \\ \quad \forall s_f \ [\ \neg(\ \mathrm{On}(\mathrm{B},\mathrm{Table},s_f) & \wedge \ \mathrm{On}(\mathrm{A},\mathrm{Table},s_f))] \\ \quad \forall s_f \ [\ \neg \ \mathrm{On}(\mathrm{B},\mathrm{Table},s_f) & \vee \ \neg \ \mathrm{On}(\mathrm{A},\mathrm{Table},s_f)] \\ \qquad \quad \neg \ \mathrm{On}(\mathrm{B},\mathrm{Table},s_f) & \vee \ \neg \ \mathrm{On}(\mathrm{A},\mathrm{Table},s_f) \end{array}$$

**List of axioms and negated theorem in clause form:**

$$On(B, A, S)$$
$$On(A, \mathrm{Table}, S)$$
$$On(x, \mathrm{Table}, s_3) \vee On(x, \mathrm{Table}, \mathrm{STORE}(x, s_3))$$
$$\neg On(y, z, s_4) \vee Equal(z, \mathrm{Table}) \vee \neg On(y, \mathrm{Table}, s_4)$$
$$\neg Equal(A, B)$$
$$\neg Equal(B, \mathrm{Table})$$
$$\neg Equal(A, \mathrm{Table})$$
$$\neg On(B, \mathrm{Table}, s_f) \vee \neg On(A, \mathrm{Table}, s_f)$$

**Resolution Proof:**

(8)
$\neg On(B, \mathrm{Table}, s_f)$
$\vee \neg On(A, \mathrm{Table}, s_f)$
$\downarrow$

(3) $On(x, \mathrm{Table}, s_3) \ \vee$
$\underline{On(x, \mathrm{Table}, \mathrm{STORE}(x, s_3))}$ $\longrightarrow$ (9)
$On(B, \mathrm{Table}, s_9)$
$\vee \neg On(A, \mathrm{Table}, \mathrm{STORE}(B, s_9))$

Not covered in 2023

**Resolution procedure gets stuck:**

$$
\begin{array}{rcl}
 & & (8) \\
 & & \neg On(B, \text{Table}, s_f) \\
 & & \vee \neg On(A, \text{Table}, s_f) \\
 & & \downarrow \\
(3) \quad On(x, \text{Table}, s_3) \quad \vee & & \\
\underline{On(x, \text{Table}, \text{STORE}(x, s_3))} & \longrightarrow & (9) \\
 & & \underline{On(B, \text{Table}, s_9)} \\
 & & \vee \neg On(A, \text{Table}, \text{STORE}(B, s_9)) \\
 & & \downarrow \\
(4) \quad \neg On(y, z, s_4) \quad \vee & & \\
Equal(z, \text{Table}) \quad \vee & & (10) \\
\underline{\neg On(y, \text{Table}, s_4)} & \longrightarrow & \neg On(B, w, s_{10}) \\
 & & \vee \underline{Equal(w, \text{Table})} \\
 & & \vee \neg On(A, \text{Table}, \text{STORE}(B, s_{10})) \\
 & & \downarrow \\
(7) \quad \neg Equal(A, \text{Table}) & \longrightarrow & (11) \\
 & & \underline{\neg On(B, A, s_{11})} \\
 & & \vee \neg On(A, \text{Table}, \text{STORE}(B, s_{11})) \\
 & & \downarrow \\
(1) \quad On(B, A, S) & \longrightarrow & (12) \\
 & & \neg On(A, \text{Table}, \text{STORE}(B, S)) \\
\textbf{Cannot make step}: & & \\
 & & \downarrow \\
(2) \quad On(A, \text{Table}, S) & \longrightarrow & NIL
\end{array}
$$

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:

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

Convert to frame axiom:

$$\neg On(p, q, s_0) \vee Equal(p, r) \vee On(p, q, \mathrm{STORE}(r, s_0))$$

Previously stuck at (12):

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

Resolve (12) with frame axiom:

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

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

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

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

$$NIL$$