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.

Commutative Laws :

$$E_1 \wedge E_2 \quad \Leftrightarrow \quad E_2 \wedge E_1$$
$$E_1 \vee E_2 \quad \Leftrightarrow \quad E_2 \vee E_1$$

Distributive Laws :

$$E_1 \wedge (E_2 \vee E_3) \quad \Leftrightarrow \quad (E_1 \wedge E_2) \vee (E_1 \wedge E_3)$$
$$E_1 \vee (E_2 \wedge E_3) \quad \Leftrightarrow \quad (E_1 \vee E_2) \wedge (E_1 \vee E_3)$$

Associative Laws :

$$E_1 \wedge (E_2 \wedge E_3) \quad \Leftrightarrow \quad (E_1 \wedge E_2) \wedge E_3$$
$$E_1 \vee (E_2 \vee E_3) \quad \Leftrightarrow \quad (E_1 \vee E_2) \vee E_3$$

De Morgan′s Laws :

$$\neg (E_1 \wedge E_2) \quad \Leftrightarrow \quad (\neg E_1) \vee (\neg E_2)$$
$$\neg (E_1 \vee E_2) \quad \Leftrightarrow \quad (\neg E_1) \wedge (\neg E_2)$$

Double Negation Law :

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

Precedence of operators in following order:

$\neg, \ \wedge, \ \vee, \ \Rightarrow.$

Truth Table:

A	B	$A \Rightarrow B \iff (\neg A \vee B)$
T	T	T
T	F	F
F	T	T
F	F	T

Logic – 1st Order Predicate Calculus

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

Order Matters.

$$\forall x \, \exists y \quad Loves(x, y)$$
$$\exists y \, \forall x \quad Loves(x, y)$$

Obey **De Morgan′s** rules

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

Term
– constant **(object)**
– 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
– wff $\vee$ wff, wff $\wedge$ wff, $\neg$ wff, 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:

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

Sentence?

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

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	$E_1 \lor E_2$
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:

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."

Example

Axiom:

$$\forall x\, \forall y\, [\mathrm{On}(x, y) \Rightarrow \mathrm{Above}(x, y)]$$

$$\forall x\, \forall y\, \forall z\, [\mathrm{Above}(x, y) \wedge \mathrm{Above}(y, z) \Rightarrow \mathrm{Above}(x, z)]$$

$$\mathrm{On}(B, A)$$

$$\mathrm{On}(A, Table)$$

Theorem:

$$\mathrm{Above}(B, Table)$$

Wednesday, December 4, 2024 12:41

Transform axioms into clause forms

- Eliminate implications ($E_1 \Rightarrow E_2 \Leftrightarrow \neg E_1 \vee E_2$)
- Move negations down to atomic formulas
- Eliminate conjunctions (De Morgen's Law)
- Move universal qualifiers to the left. rename if necessary, eliminate

- Transform Axioms :

$\neg On(u, v) \vee Above(u, v)$ ····· (1)

$\neg Above(x, y) \vee \neg Above(y, z) \vee Above(x, z)$ ··· (2)

$On(B, A)$ ···· (3)

$On(A, Table)$ ···· (4)

- Negate Theorem:

$\neg Above(B, Table)$ ··· (5)

- Specialize (2) with $x \to B$, $z \to Table$:

$\neg Above(B, y) \vee \neg Above(y, Table) \vee \cancel{Above(B, Table)}$ (2)

$\neg \cancel{Above(B, Table)}$ (5)

$\neg Above(B, y) \vee \neg Above(y, Table)$ ···· (6)

- Specialize (1) with $u \to B$, $v \to y$:

$\neg On(B, y) \vee \cancel{Above(B, y)}$ (1)

$\neg \cancel{Above(B, y)} \vee \neg Above(y, Table)$ (6)

$\neg On(B, y) \vee \neg Above(y, Table)$ ··· (7)

- Specialize (1) with $u \to y$, $v \to Table$,

$\neg On(y, Table) \vee \cancel{Above(y, Table)}$ (1)

$\neg On(B, y) \vee \neg \cancel{Above(y, Table)}$ (7)

$\neg On(y, Table) \vee \neg On(B, y)$ ··· (8)

- Specialize (8) with $y \to A$:

$\neg On(A, Table) \vee \neg On(B, A)$ (8)

$On(B, A)$ (3)

$\neg On(A, Table)$ ··· (9)

$On(A, Table)$ (4)

NIL ∎

Learning outcomes:

- know the termology covered in the previous pages
- be able to prove a theorem by refutation with the given axioms
- be able to transform a well-formed formula into clause form

Extra:

The example in the following pages demonstrates step by step how to transform a *complicated* axiom into clause form. Students are encouraged to read and understand how it was done. Note those *existential qualifiers* included in the axiom and how they are handled.

Example

Transform the following Axiom into the clause form

$$\begin{aligned}\forall x\,[Brick(x) \Rightarrow\ & (\exists y\,[On(x,y) \land \neg Pyramid(y)]\ \land \\ & \neg\exists y\,[On(x,y) \land On(y,x)]\ \land \\ & \forall y\,[\neg Brick(y) \Rightarrow \neg Equal(x,y)]\ )]\end{aligned}$$

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

$$\begin{aligned}\forall x\,[\neg Brick(x)\ \lor\ & (\exists y\,[On(x,y) \land \neg Pyramid(y)]\ \land \\ & \neg\exists y\,[On(x,y) \land On(y,x)]\ \land \\ & \forall y\,[\neg\neg Brick(y)\ \lor\ \neg Equal(x,y)]\ )]\end{aligned}$$

2. Move negations down to atomic formulas:

$$\begin{aligned}\forall x\,[\neg Brick(x)\ \lor\ & (\exists y\,[On(x,y) \land \neg Pyramid(y)]\ \land \\ & \forall y\,[\neg On(x,y) \lor \neg On(y,x)]\ \land \\ & \forall y\,[Brick(y)\ \lor\ \neg Equal(x,y)]\ )]\end{aligned}$$

Skolem functions are helper functions. You can name them whatever way you like, preferably meaningful though. In this example, we have x on y so we consider y is a support of x; hence the name.

3. Eliminate existential quantifiers using Skolem functions:

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

4. Rename variables:

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

4. Rename variables:

$$\begin{aligned}\forall x\,[\neg Brick(x) \ \vee\ (On(x, Support(x)) \wedge \neg Pyramid(Support(x))\ \wedge \\ \forall y\,[\neg On(x,y) \vee \neg On(y,x)]\ \wedge \\ \forall z\,[Brick(z)\ \vee\ \neg Equal(x,z)]\,)]\end{aligned}$$

5. Move universal quantifiers to left:

$$\begin{aligned}\forall x \forall y \forall z\,[\neg Brick(x) \vee (On(x, Support(x))\ \wedge\ \neg Pyramid(Support(x)) \\ \wedge (\neg On(x,y)\ \vee\ \neg On(y,x)) \\ \wedge (Brick(z)\ \vee\ \neg Equal(x,z))\,)]\end{aligned}$$

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

$$\begin{aligned}\forall x \forall y \forall z\,[(\neg Brick(x) \vee\ (On(x, Support(x)) \wedge \neg Pyramid(Support(x)))) \\ \wedge (\neg Brick(x) \vee\ (\neg On(x,y) \vee \neg On(y,x))) \\ \wedge (\neg Brick(x) \vee\ (Brick(z) \vee \neg Equal(x,z))\,)]\end{aligned}$$

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

7. Eliminate conjunctions:

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

7. Eliminate conjunctions:

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

8. Rename variables:

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

9. Eliminate universal quantifiers:

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