RESOLUTION

Topics in AI

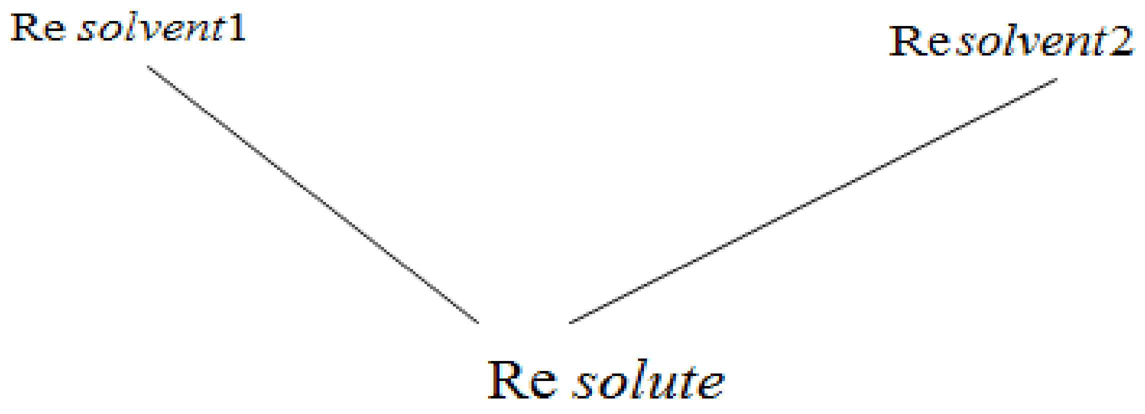
Slides compiled by: Sanghamitra De

Resolution

Resolution is one kind of proof technique that works this way -

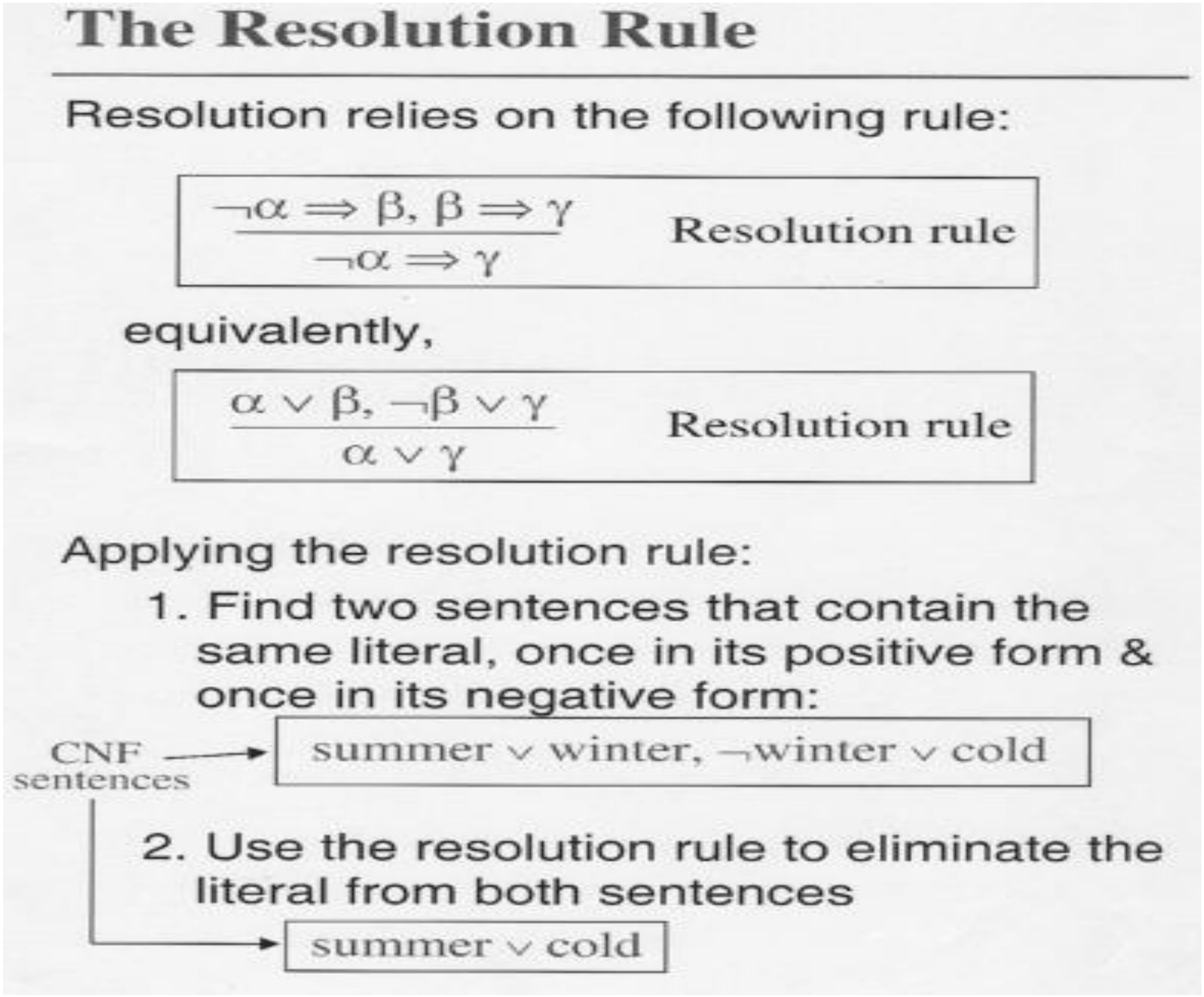
- ✓ select two clauses that contain conflicting terms
- ✓ combine those two clauses and
- ✓ cancel out the conflicting terms.

Resolution Tree



Resolution

- ✓ Simple, efficient, sound, complete... dominates AI reasoning, automated logical proof
- ✓ Produces proofs by refutation: to prove S from KB, add ~ S to KB and show that set of clauses is inconsistent.
- ✓ CNF ("Clause Normal Form" or "Conj. Nor. Form") means simplicity: no issues of how to represent knowledge
- ✓ Resolution only rule: no decisions: simplicity
- ✓ Every FOL sentence convertable to CNF.
- ✓ By keeping track of the substitutions (of values for variables) in the proof, "proved or disproved" can be extended to "question answering" (e.g. Prolog).
- ✓ Search required, however: which two clauses to resolve? Result makes another clause (in general), so the KB typically grows, complicating search... hence generations of research on Resolution Theorem Proving or RTP proof techniques.



The Resolution Rule

Resolution relies on the following rule:

$$\boxed{\frac{\neg\alpha \Rightarrow \beta,\ \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma} \qquad \text{Resolution rule}}$$

equivalently,

$$\boxed{\frac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma} \qquad \text{Resolution rule}}$$

Applying the resolution rule:

1. Find two sentences that contain the same literal, once in its positive form & once in its negative form:

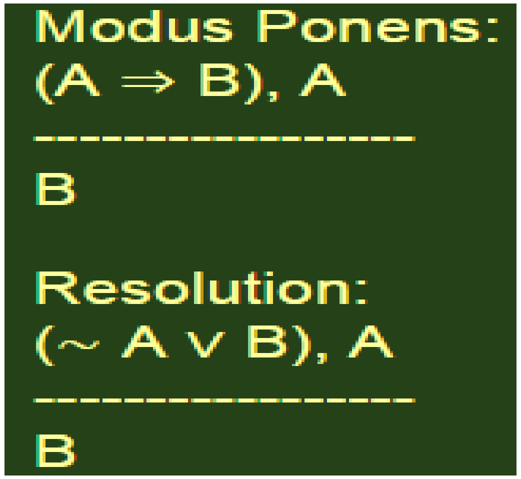
CNF sentences $\rightarrow$ $\boxed{\text{summer} \vee \text{winter},\ \neg\text{winter} \vee \text{cold}}$

2. Use the resolution rule to eliminate the literal from both sentences

$\rightarrow$ $\boxed{\text{summer} \vee \text{cold}}$

Resolution

- ✓ So"unit" resolution produces a new clause with one less term than its longer parent.
- ✓ It is closely related to modus ponens.



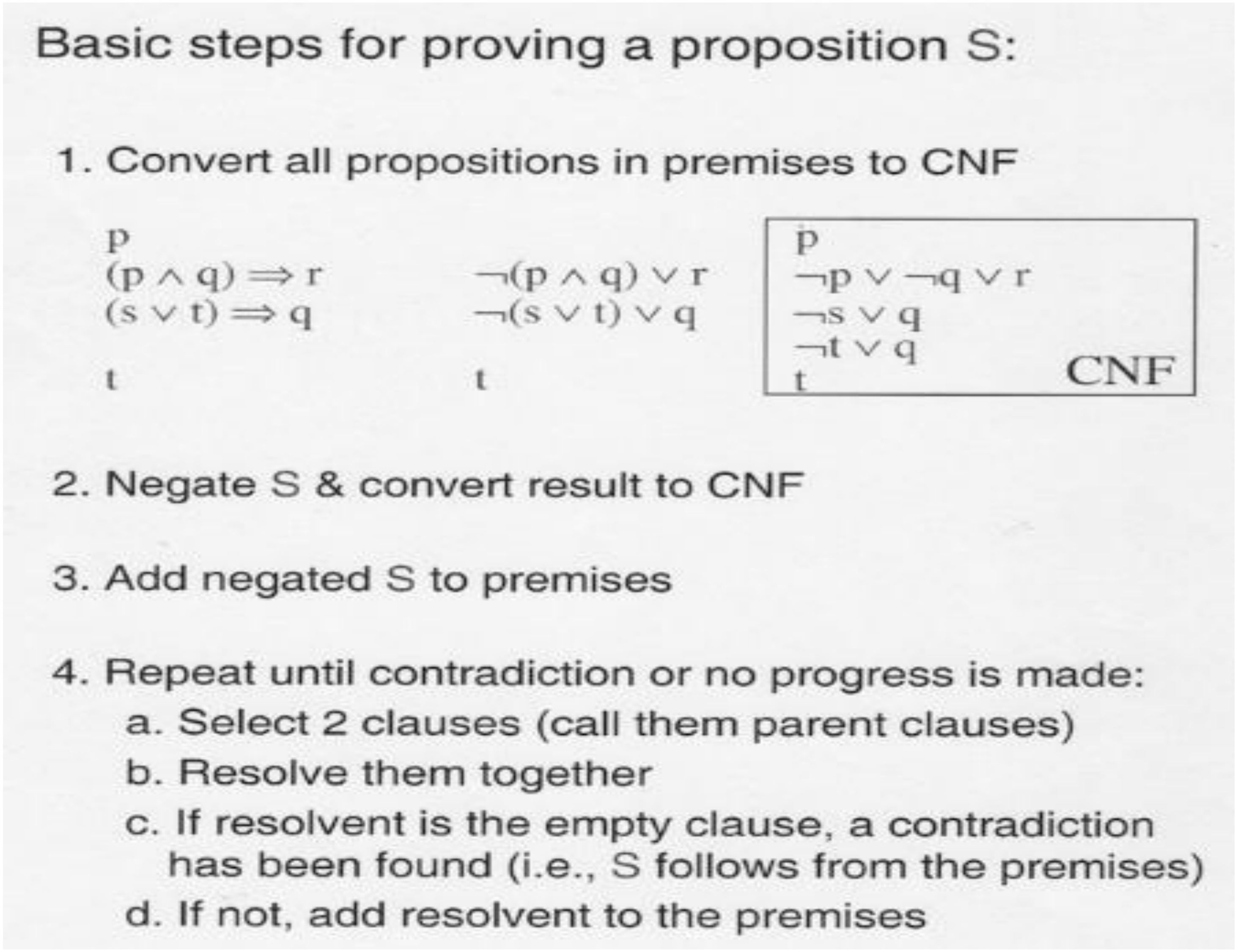
Modus Ponens:
(A ⇒ B), A

B

Resolution:
(~ A v B), A

B

Resolution in Propositional Logic



Basic steps for proving a proposition S:

1. Convert all propositions in premises to CNF

p		p
$(p \wedge q) \Rightarrow r$	$\neg(p \wedge q) \vee r$	$\neg p \vee \neg q \vee r$
$(s \vee t) \Rightarrow q$	$\neg(s \vee t) \vee q$	$\neg s \vee q$
		$\neg t \vee q$
t	t	t

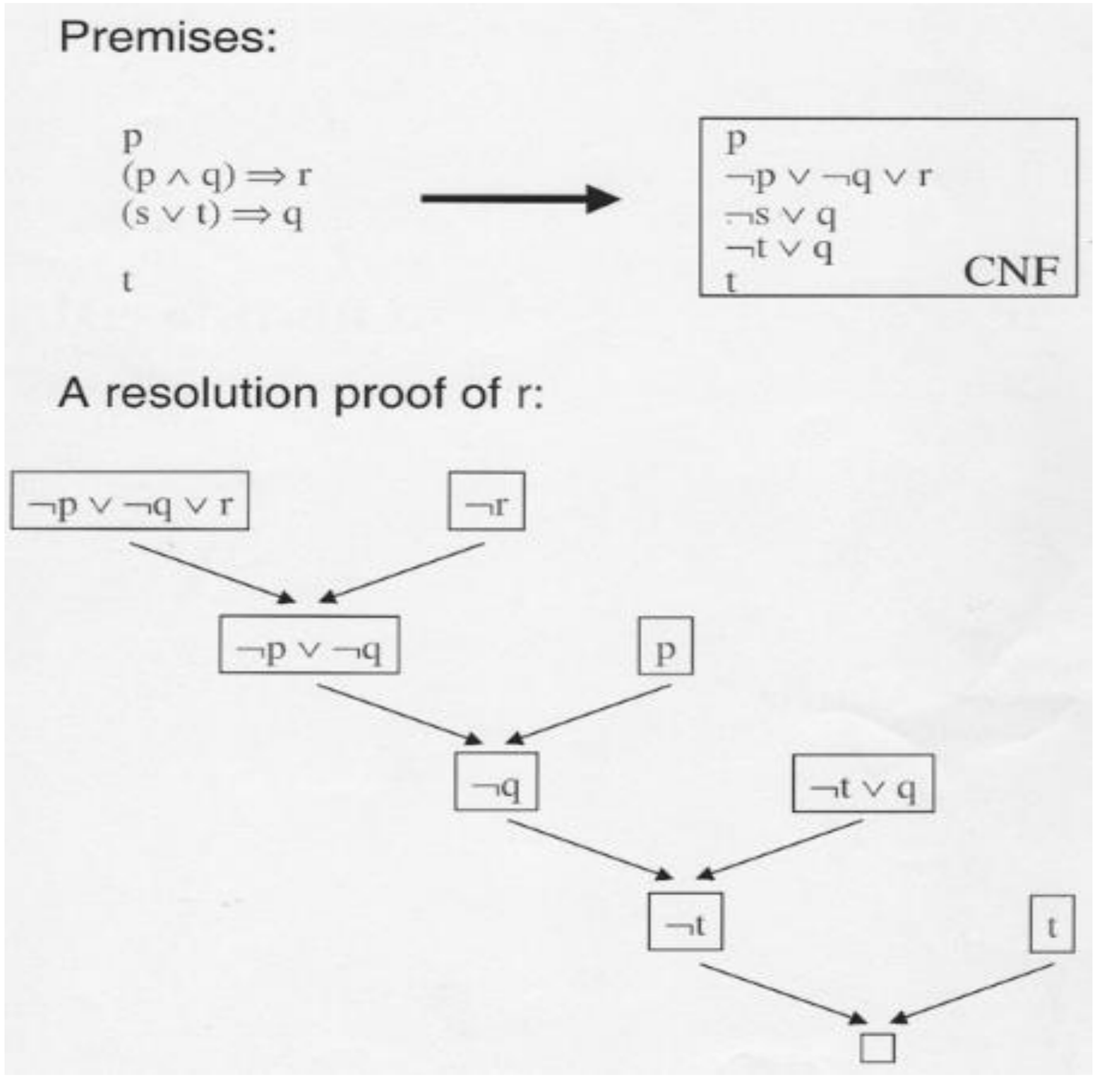
CNF

2. Negate S & convert result to CNF

3. Add negated S to premises

4. Repeat until contradiction or no progress is made:
   a. Select 2 clauses (call them parent clauses)
   b. Resolve them together
   c. If resolvent is the empty clause, a contradiction has been found (i.e., S follows from the premises)
   d. If not, add resolvent to the premises

Resolution in Propositional Logic- Example



Premises:

p
$(p \wedge q) \Rightarrow r$
$(s \vee t) \Rightarrow q$
t

$\longrightarrow$

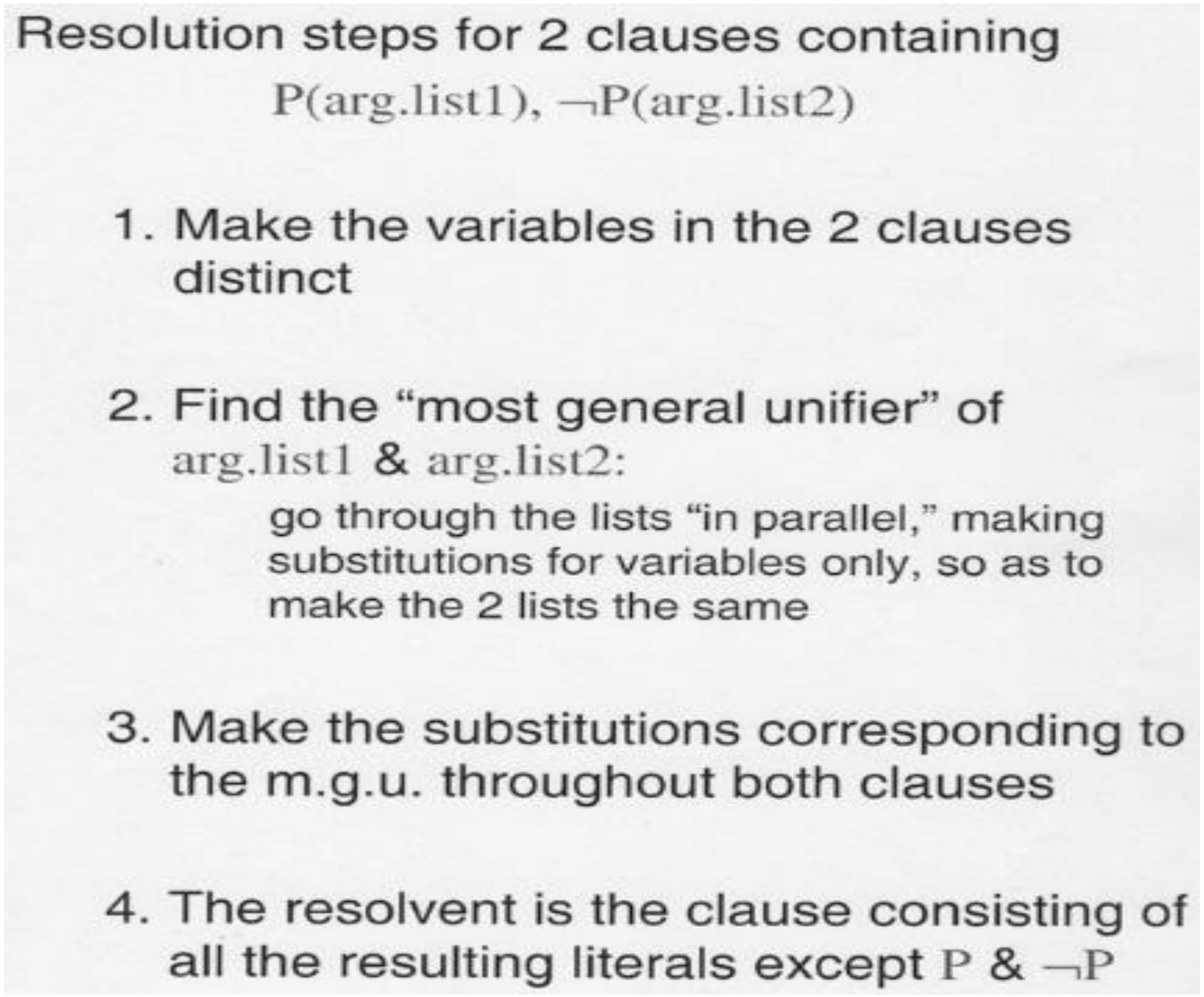
p
$\neg p \vee \neg q \vee r$
$\neg s \vee q$
$\neg t \vee q$
t

CNF

A resolution proof of r:

$$\begin{tikzcd}
\boxed{\neg p \vee \neg q \vee r} \arrow[dr] & & \boxed{\neg r} \arrow[dl] & & & \\
& \boxed{\neg p \vee \neg q} \arrow[dr] & & \boxed{p} \arrow[dl] & & \\
& & \boxed{\neg q} \arrow[dr] & & \boxed{\neg t \vee q} \arrow[dl] & \\
& & & \boxed{\neg t} \arrow[dr] & & \boxed{t} \arrow[dl] \\
& & & & \square &
\end{tikzcd}$$

Resolution Steps



Resolution steps for 2 clauses containing

P(arg.list1), ¬P(arg.list2)

1. Make the variables in the 2 clauses distinct

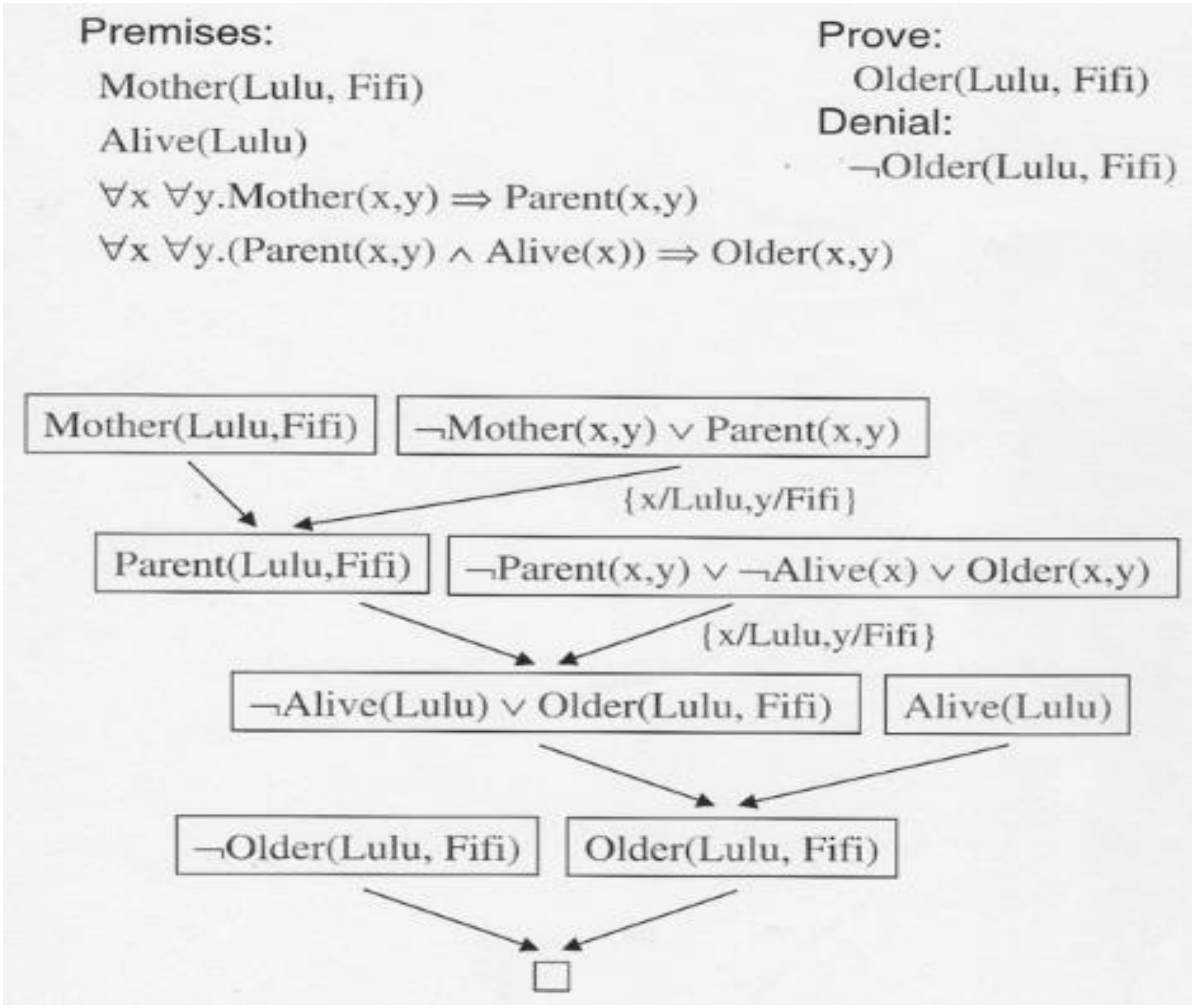
2. Find the "most general unifier" of arg.list1 & arg.list2:

   go through the lists "in parallel," making substitutions for variables only, so as to make the 2 lists the same

3. Make the substitutions corresponding to the m.g.u. throughout both clauses

4. The resolvent is the clause consisting of all the resulting literals except P & ¬P

Resolution Examples



Premises:

Mother(Lulu, Fifi)

Alive(Lulu)

$\forall x\ \forall y.\text{Mother}(x,y) \Rightarrow \text{Parent}(x,y)$

$\forall x\ \forall y.(\text{Parent}(x,y) \land \text{Alive}(x)) \Rightarrow \text{Older}(x,y)$

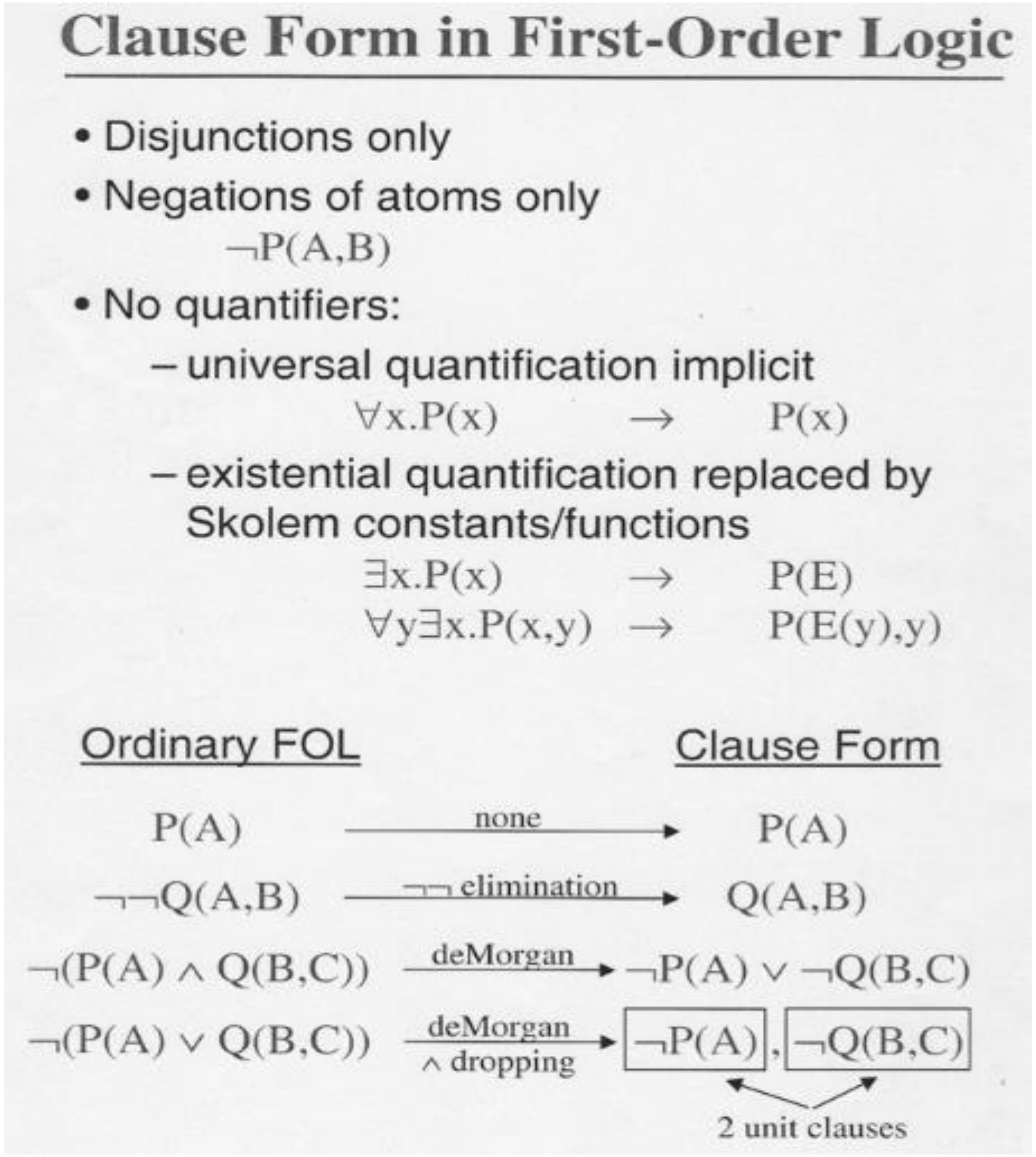
Prove:

Older(Lulu, Fifi)

Denial:

$\neg\text{Older}(\text{Lulu}, \text{Fifi})$

$$\begin{tikzcd}
\boxed{\text{Mother(Lulu,Fifi)}} \arrow[d] & & \boxed{\neg\text{Mother}(x,y) \lor \text{Parent}(x,y)} \arrow[dll, "\{x/\text{Lulu},y/\text{Fifi}\}"] & \\
\boxed{\text{Parent(Lulu,Fifi)}} \arrow[dr] & & \boxed{\neg\text{Parent}(x,y) \lor \neg\text{Alive}(x) \lor \text{Older}(x,y)} \arrow[dl, "\{x/\text{Lulu},y/\text{Fifi}\}"] & \\
& \boxed{\neg\text{Alive(Lulu)} \lor \text{Older(Lulu, Fifi)}} \arrow[dr] & & \boxed{\text{Alive(Lulu)}} \arrow[dl] \\
& \boxed{\neg\text{Older(Lulu, Fifi)}} \arrow[d] & \boxed{\text{Older(Lulu, Fifi)}} \arrow[dl] & \\
& \square & &
\end{tikzcd}$$



Clause Form in First-Order Logic

- Disjunctions only
- Negations of atoms only

  $\neg P(A,B)$
- No quantifiers:
  - universal quantification implicit

    $\forall x.P(x) \quad \rightarrow \quad P(x)$
  - existential quantification replaced by Skolem constants/functions

    $\exists x.P(x) \quad \rightarrow \quad P(E)$

    $\forall y \exists x.P(x,y) \quad \rightarrow \quad P(E(y),y)$

<u>Ordinary FOL</u> <u>Clause Form</u>

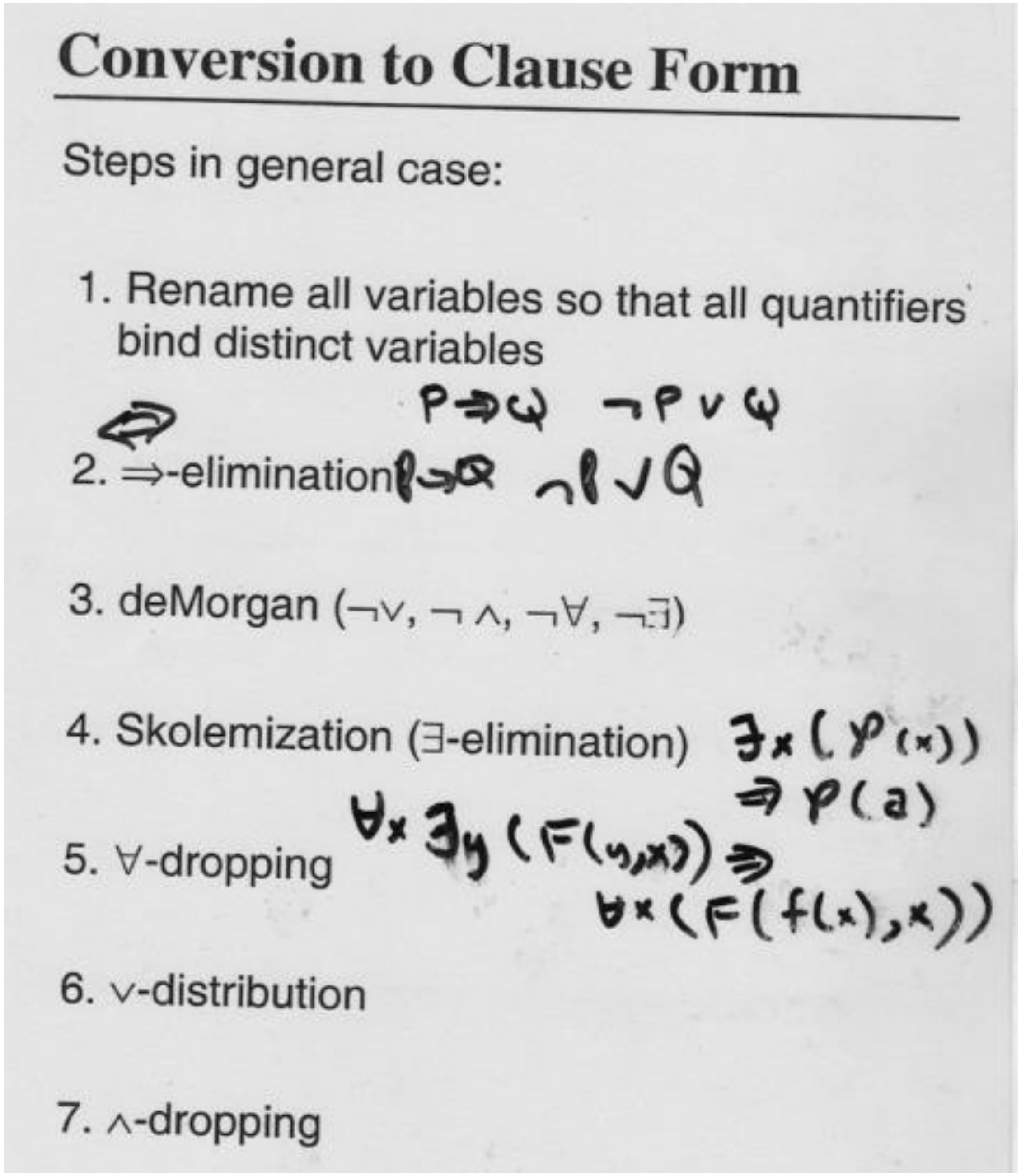
$P(A) \xrightarrow{\text{none}} P(A)$

$\neg\neg Q(A,B) \xrightarrow{\neg\neg\text{ elimination}} Q(A,B)$

$\neg(P(A) \land Q(B,C)) \xrightarrow{\text{deMorgan}} \neg P(A) \lor \neg Q(B,C)$

$\neg(P(A) \lor Q(B,C)) \xrightarrow[\land\text{ dropping}]{\text{deMorgan}} \boxed{\neg P(A)}, \boxed{\neg Q(B,C)}$

2 unit clauses



Conversion to Clause Form

Steps in general case:

1. Rename all variables so that all quantifiers bind distinct variables

$\Leftrightarrow$ $P \Rightarrow Q \quad \neg P \vee Q$

2. $\Rightarrow$-elimination $P \Rightarrow Q \quad \neg P \vee Q$

3. deMorgan ($\neg\vee$, $\neg\wedge$, $\neg\forall$, $\neg\exists$)

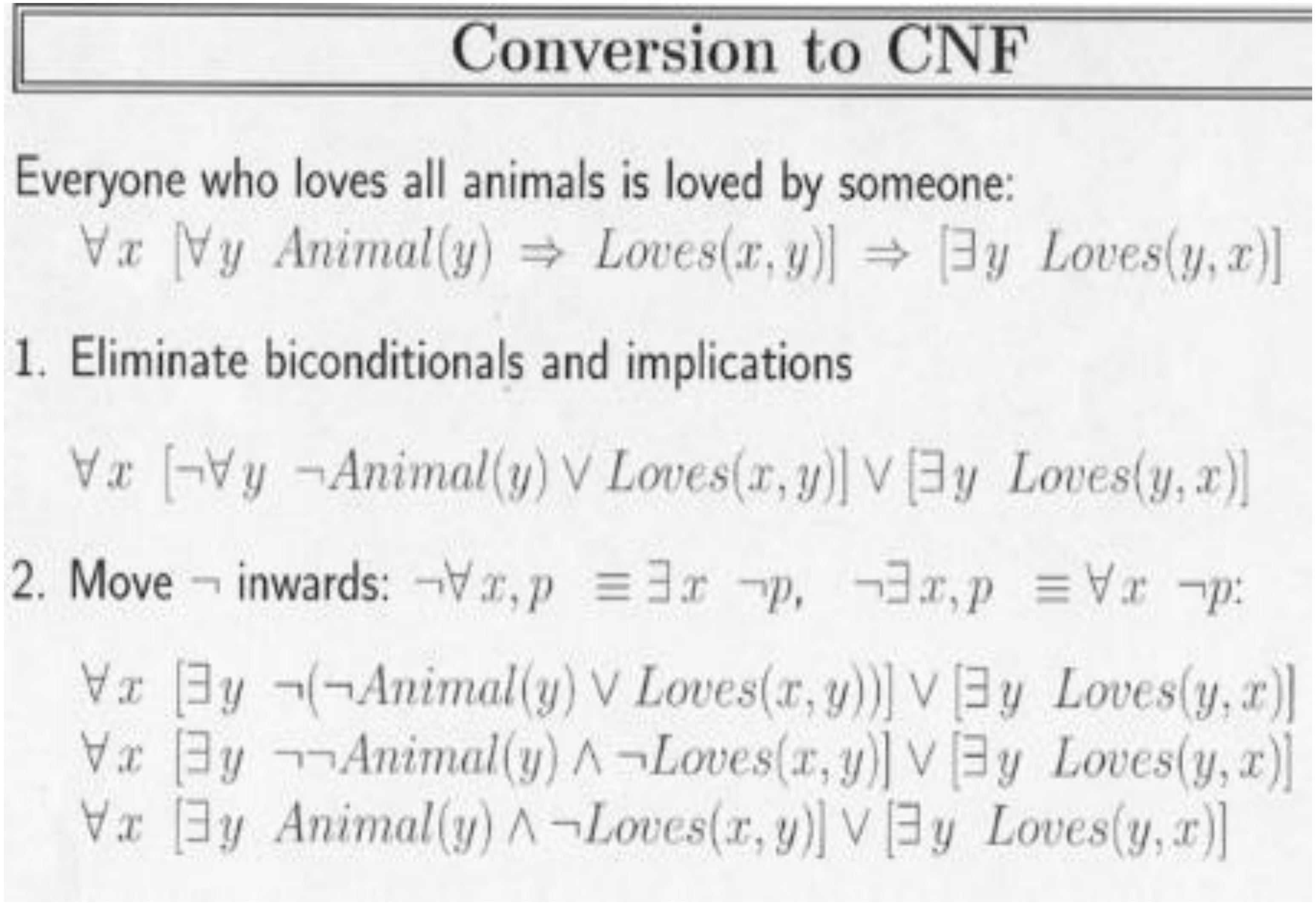
4. Skolemization ($\exists$-elimination) $\exists x(\varphi(x)) \Rightarrow \varphi(a)$

5. $\forall$-dropping $\forall x \exists y (F(y,x)) \Rightarrow \forall x(F(f(x),x))$

6. $\vee$-distribution

7. $\wedge$-dropping

Clause Form Conversion Example



Conversion to CNF

Everyone who loves all animals is loved by someone:

$$\forall x\ [\forall y\ \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ \ Loves(y,x)]$$

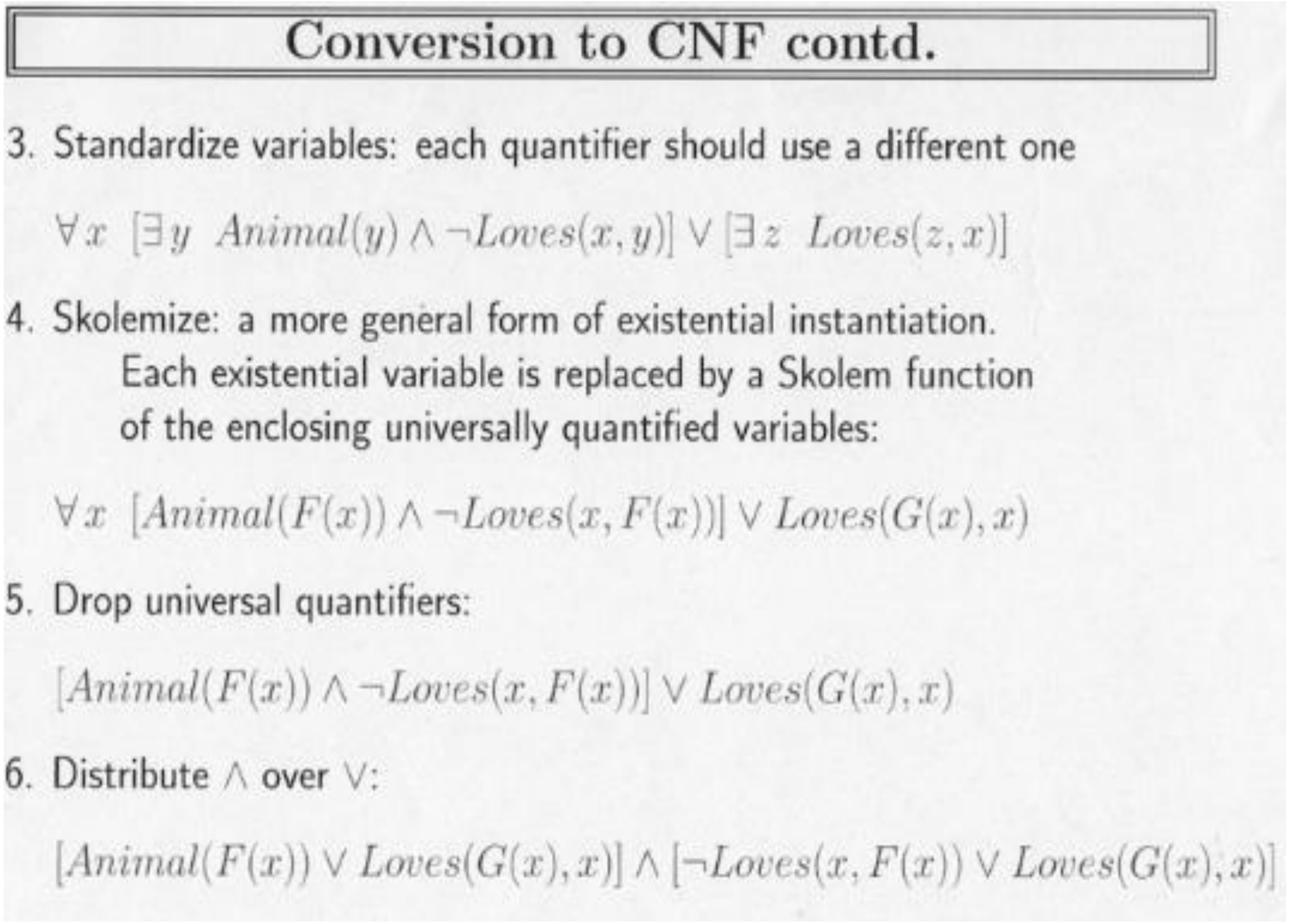
1. Eliminate biconditionals and implications

$$\forall x\ [\neg\forall y\ \ \neg Animal(y) \lor Loves(x,y)] \lor [\exists y\ \ Loves(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x, p\ \equiv \exists x\ \ \neg p,\quad \neg\exists x, p\ \equiv \forall x\ \ \neg p$:

$$\forall x\ [\exists y\ \ \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y\ \ Loves(y,x)]$$
$$\forall x\ [\exists y\ \ \neg\neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y\ \ Loves(y,x)]$$
$$\forall x\ [\exists y\ \ Animal(y) \land \neg Loves(x,y)] \lor [\exists y\ \ Loves(y,x)]$$

Clause Form Conversion Example (contd.)



Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

$$\forall x \ [\exists y \ Animal(y) \land \neg Loves(x,y)] \lor [\exists z \ Loves(z,x)]$$

4. Skolemize: a more general form of existential instantiation.
   Each existential variable is replaced by a Skolem function
   of the enclosing universally quantified variables:

$$\forall x \ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$$

5. Drop universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$$

6. Distribute $\land$ over $\lor$:

$$[Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)]$$

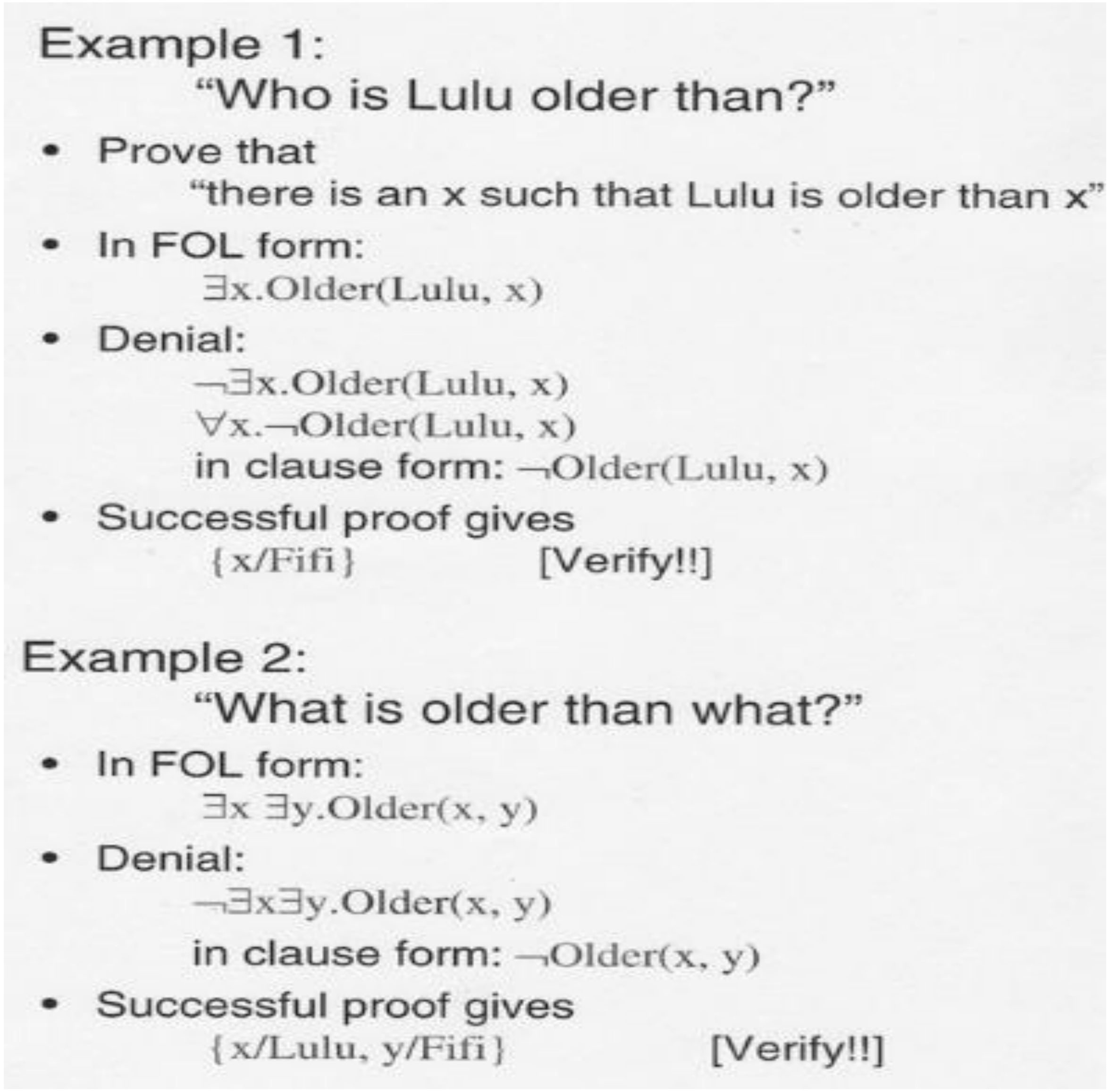
EXISTENTIAL ELIMINATION: SKOLEMIZATION

- ✓ Existential Quantification asserts at least one individual exists that satisfies the predicate. The trick is just to make up a name for that individual and keep track of him like any other individual. If someone asks ``who is that?'', the answer is ``the one who exists according to the Existentially Quantified statement.''
- ✓ $\exists$ x (P(x)) $\rightarrow$ P(Jerry)
  This *Skolem constant* idea also works for $\exists$ acting over $\forall$.
  $\exists$ x $\forall$ y (P(x,y)) $\rightarrow$ $\forall$ y P(Fred,y)
  Extending the idea to a *Skolem function* deals with cases of $\forall$ acting over $\exists$, in which case the particular individual who exists depends on just who gets instantiated in the "forall"
- ✓ E.g. "Everyone has a biological father" needs a Skolem function to assure that the father making the sentence true depends on who gets substituted in for 'Everyone': Bob's father may not be Alice's.
  $\forall$ x $\exists$ y (Father(y,x)) $\rightarrow$
  $\forall$ x Father(father-of(x), x)
  Note that such a function has to exist because it is guaranteed by the $\exists$ in the premise. However we may have no idea what any of its values are (do I know your father?).

RTP Recap

- ✓ Fact: Resolution is universal, sound, complete.
- ✓ Fact: Using it, we can detect inconsistent set of clauses in finite time.
- ✓ So…Given set of FOL sentences, (∀, ∃, &rHarr, etc.)
- ✓ Convert sentences to CNF
- ✓ Assert negative of desired conclusion, add to clauses. (not in other order!)
- ✓ Resolve your head off
- ✓ Use Heuristics, prayer?
- ✓ Find ∅ (you hope)

Using Resolution in Q-Ans/Answer Extraction



Example 1:

"Who is Lulu older than?"

- Prove that
  "there is an x such that Lulu is older than x"
- In FOL form:
  $\exists x.Older(Lulu, x)$
- Denial:
  $\neg\exists x.Older(Lulu, x)$
  $\forall x.\neg Older(Lulu, x)$
  in clause form: $\neg Older(Lulu, x)$
- Successful proof gives
  $\{x/Fifi\}$ [Verify!!]

Example 2:

"What is older than what?"

- In FOL form:
  $\exists x\ \exists y.Older(x, y)$
- Denial:
  $\neg\exists x\exists y.Older(x, y)$
  in clause form: $\neg Older(x, y)$
- Successful proof gives
  $\{x/Lulu, y/Fifi\}$ [Verify!!]

Converting sentences to CNF

1. Eliminate all $\leftrightarrow$ connectives

   $(P \leftrightarrow Q) \Rightarrow ((P \rightarrow Q) \wedge (Q \rightarrow P))$

2. Eliminate all $\rightarrow$ connectives

   $(P \rightarrow Q) \Rightarrow (\neg P \vee Q)$

3. Reduce the scope of each negation symbol to a single predicate

   $\neg\neg P \Rightarrow P$

   $\neg(P \vee Q) \Rightarrow \neg P \wedge \neg Q$

   $\neg(P \wedge Q) \Rightarrow \neg P \vee \neg Q$

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

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

4. Standardize variables: rename all variables so that each quantifier has its own unique variable name

Converting sentences to clausal form
Skolem constants and functions

5. Eliminate existential quantification by introducing Skolem constants/functions

   $(\exists x)P(x) \Rightarrow P(C)$

   C is a Skolem constant (a brand-new constant symbol that is not used in any other sentence)

   $(\forall x)(\exists y)P(x,y) \Rightarrow (\forall x)P(x, f(x))$

   since $\exists$ is within scope of a universally quantified variable, use a **Skolem function f** to construct a new value that **depends on** the universally quantified variable

   f must be a brand-new function name not occurring in any other sentence in the KB

   E.g., $(\forall x)(\exists y)loves(x,y) \Rightarrow (\forall x)loves(x,f(x))$

   In this case, f(x) specifies the person that x loves

   a better name might be **oneWhoIsLovedBy**(x)

Converting sentences to clausal form

6. Remove universal quantifiers by (1) moving them all to the left end; (2) making the scope of each the entire sentence; and (3) dropping the "prefix" part

   Ex: $(\forall x)P(x) \Rightarrow P(x)$

7. Put into conjunctive normal form (conjunction of disjunctions) using distributive and associative laws

   $(P \wedge Q) \vee R \Rightarrow (P \vee R) \wedge (Q \vee R)$

   $(P \vee Q) \vee R \Rightarrow (P \vee Q \vee R)$

8. Split conjuncts into separate clauses

9. Standardize variables so each clause contains only variable names that do not occur in any other clause

Conversion to CNF: Example

Everyone who loves all animals is loved by someone:

$$\forall x\ [\forall y\ Animal(y) \implies Loves(x,y)] \implies [\exists y\ Loves(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x\ [\neg\forall y\ \neg Animal(y) \lor Loves(x,y)] \lor [\exists y\ Loves(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x, p\ \equiv \exists x\ \neg p$, $\neg\exists x, p\ \equiv \forall x\ \neg p$:

$$\forall x\ [\exists y\ \neg(\neg Animal(y) \lor Loves(x,y))] \lor [\exists y\ Loves(y,x)]$$
$$\forall x\ [\exists y\ \neg\neg Animal(y) \land \neg Loves(x,y)] \lor [\exists y\ Loves(y,x)]$$
$$\forall x\ [\exists y\ Animal(y) \land \neg Loves(x,y)] \lor [\exists y\ Loves(y,x)]$$

Conversion to CNF: Example (contd.)

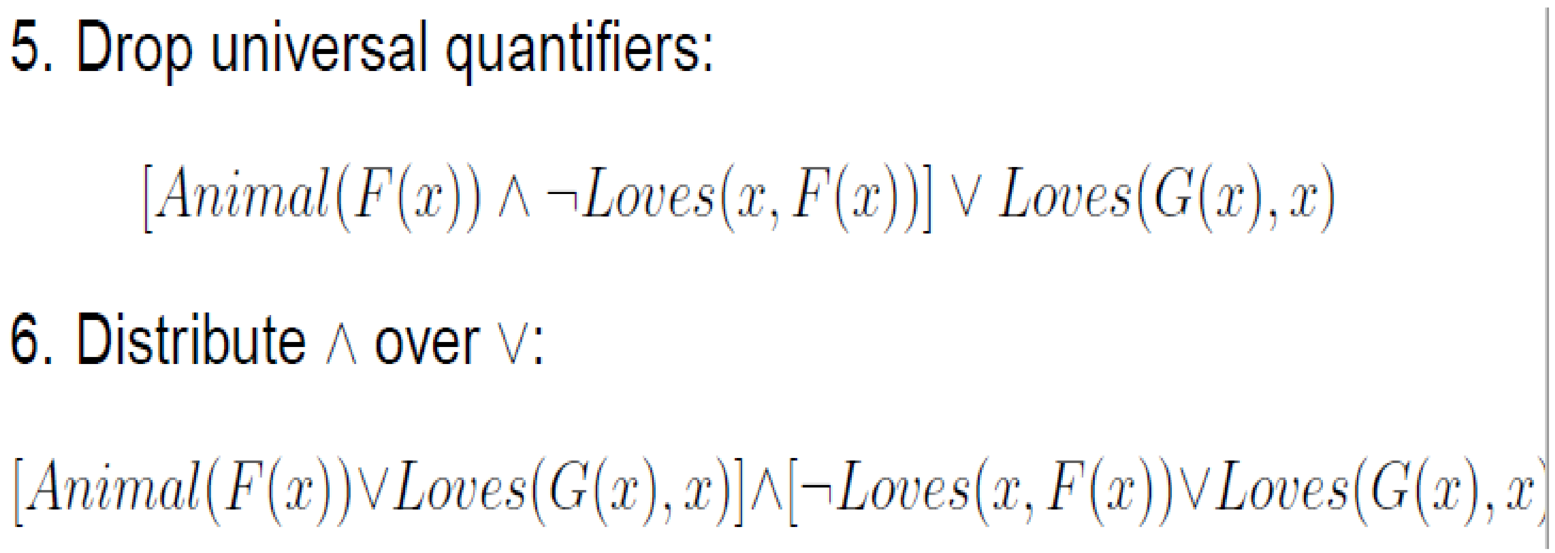
3. Standardize variables: each quantifier should use a different one

$$\forall x \;\; [\exists y \;\; Animal(y) \land \neg Loves(x,y)] \lor [\exists z \;\; Loves(z,x)]$$

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$$\forall x \;\; [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$$

Conversion to CNF: Example (contd.)



5. Drop universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

6. Distribute $\land$ over $\lor$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x$$

- ✓ Original sentence is now in CNF form – can apply same ideas to all sentences in KB to convert into CNF
- ✓ Also need to include negated query
- ✓ Then use resolution to attempt to derive the empty clause which show that the query is entailed by the KB

Resolution Algorithm

```
procedure resolution-refutation(KB, Q)
   ;; KB is a set of consistent, true FOL sentences
   ;; Q is a goal sentence that we want to derive
   ;; return success if KB |- Q, and failure otherwise
   KB = union(KB, ~Q)
   while false not in KB do
        pick 2 sentences, S1 and S2, in KB that contain
        literals that resolve(if none, return "failure")

        resolvent = resolution-rule(S1, S2)

        KB = union(KB, resolvent)

   return "success"
```

Theorem Proving using Resolution

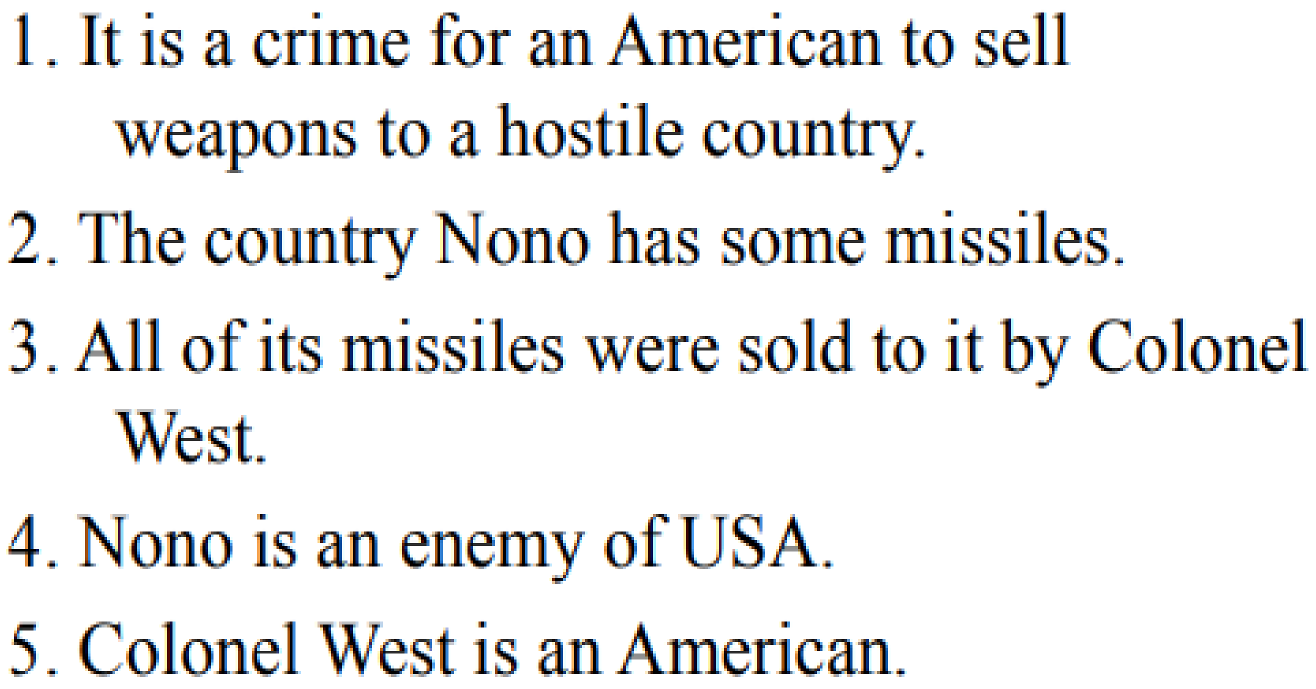
1. Obtain $CNF(KB \wedge \neg\alpha)$
2. Apply resolution steps to the CNF
3. If the empty clause is generated, then $KB \models \alpha$.

Theorem: Resolution is a refutationally complete inference system for $KB \models \alpha$.

Resolution & Entailment

- $\alpha \models \beta$ means α entails β ie. β follows logically from α, where α and β are sentences.
- Mathematically, $\alpha \models \beta$ if and only if in every model in which α is true, β is also true.
- Another way: if α is true, then β must also be true.

Prove if: Colonel West is a criminal



1. It is a crime for an American to sell weapons to a hostile country.
2. The country Nono has some missiles.
3. All of its missiles were sold to it by Colonel West.
4. Nono is an enemy of USA.
5. Colonel West is an American.

. . . it is a crime for an American to sell weapons to hostile nations:

$American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)$

Nono . . . has some missiles, i.e.,

$\exists x\ Owns(Nono, x) \land Missile(x)$:

$Owns(Nono, M_1)$ and $Missile(M_1)$

. . . all of its missiles were sold to it by Colonel West

$\forall x\ \ Missile(x) \land Owns(Nono, x) \implies Sells(West, x, Nono)$

Missiles are weapons:

$Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as “hostile”:

Prove if: Colonel West is a criminal

- An enemy of America counts as “hostile”:

  Enemy(x,America) Hostile(x).
- West is American:

  American(West)

Sentences in CNF:

$\neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x, y, z) \lor \neg Hostile(z) \lor Criminal(x)$

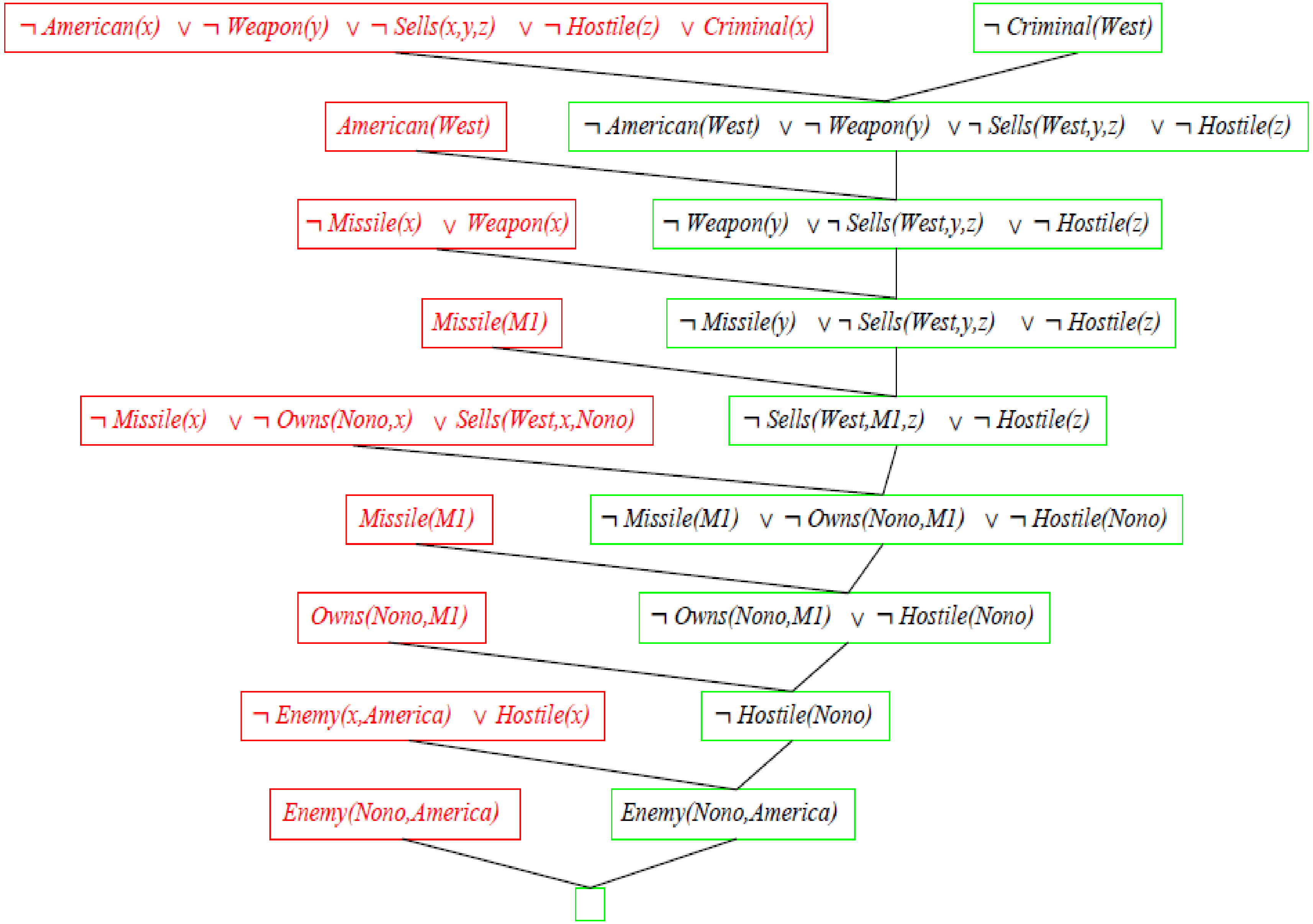
$\neg Missile(x) \lor \neg Owns(Nono, x) \lor Sells(West, x, Nono)$

$\neg Enemy(x, America) \lor Hostile(x)$

$\neg Missile(x) \lor Weapon(x)$

$Owns(Nono, M_1)$ $\quad$ $Missile(M_1)$

$American(West)$ $\quad$ $Enemy(Nono, America)$.


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
Enemy(Nono,America)

Second Example

- **KB:**
  - ✓ Everyone who loves all animals is loved by someone
  - ✓ Anyone who kills animals is loved by no-one
  - ✓ Jack loves all animals
  - ✓ Either Curiosity or Jack killed the cat, who is named Tuna
- **Query:**

  Did Curiosity kill the cat?
- **Inference Procedure:**

  Express sentences in FOL

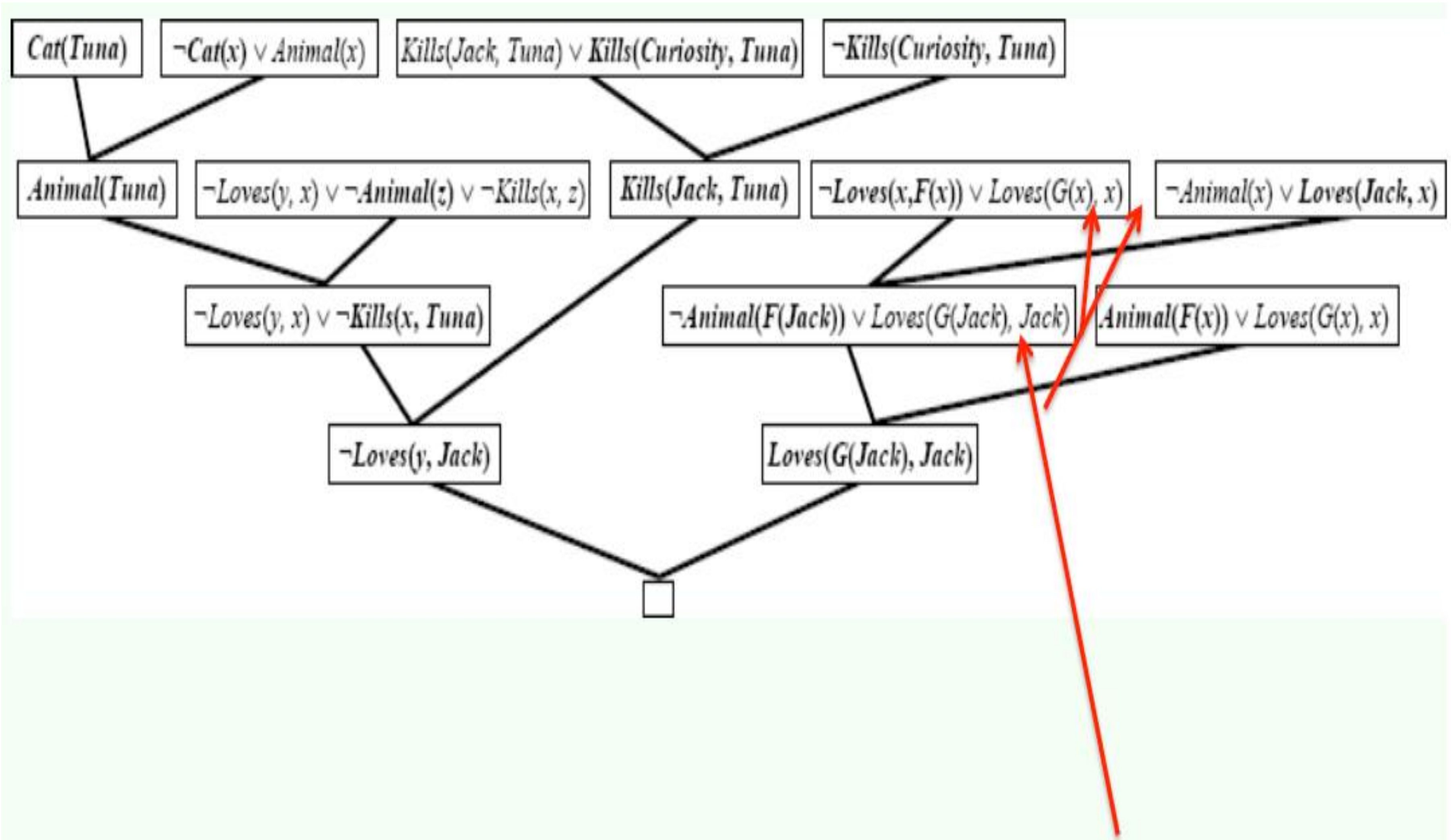
  Convert to CNF form and negated query

First, we express the original sentences, some background knowledge, and the negated goal G in first-order logic:

A.	$\forall x\ [\forall y\ \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ \ Loves(y,x)]$
B.	$\forall x\ [\exists z\ \ Animal(z) \wedge Kills(x,z)] \Rightarrow [\forall y\ \ \neg Loves(y,x)]$
C.	$\forall x\ \ Animal(x) \Rightarrow Loves(Jack,x)$
D.	$Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$
E.	$Cat(Tuna)$
F.	$\forall x\ \ Cat(x) \Rightarrow Animal(x)$
¬G.	$\neg Kills(Curiosity,Tuna)$

Now we apply the conversion procedure to convert each sentence to CNF:

A1.	$Animal(F(x)) \vee Loves(G(x),x)$
A2.	$\neg Loves(x,F(x)) \vee Loves(G(x),x)$
B.	$\neg Loves(y,x) \vee \neg Animal(z) \vee \neg Kills(x,z)$
C.	$\neg Animal(x) \vee Loves(Jack,x)$
D.	$Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$
E.	$Cat(Tuna)$
F.	$\neg Cat(x) \vee Animal(x)$
¬G.	$\neg Kills(Curiosity,Tuna)$



Cat(Tuna)
¬Cat(x) ∨ Animal(x)
Kills(Jack, Tuna) ∨ Kills(Curiosity, Tuna)
¬Kills(Curiosity, Tuna)
Animal(Tuna)
¬Loves(y, x) ∨ ¬Animal(z) ∨ ¬Kills(x, z)
Kills(Jack, Tuna)
¬Loves(x,F(x)) ∨ Loves(G(x), x)
¬Animal(x) ∨ Loves(Jack, x)
¬Loves(y, x) ∨ ¬Kills(x, Tuna)
¬Animal(F(Jack)) ∨ Loves(G(Jack), Jack)
Animal(F(x)) ∨ Loves(G(x), x)
¬Loves(y, Jack)
Loves(G(Jack), Jack)
□

Confusing because the sentences
Have not been standardized apart…

Completeness Of Resolution

- Resolution is **refutation-complete**
  - ✓ … *if* a set of sentences COMPLETENESS is unsatisfiable, then resolution will always be able to derive a contradiction.
- Resolution cannot be used to generate all logical consequences of a set of sentences, but it can be used to establish that a given sentence is entailed by the set of sentences.
- So it can be used to find all answers to a given question, Q(x), by proving that KB ∧ ¬Q(x) is unsatisfiable.

Completeness Of Resolution

- Any sentence in first-order logic (without equality) can be rewritten as a set of clauses in CNF.
- This can be proved by induction on the form of the sentence, using atomic sentences as the base case.
- One only needs to prove : *if* S *is an unsatisfiable set of clauses, then the application of a finite number of resolution steps to* S *will yield a contradiction.*

Rules for the Proof

- First, observe that if S is unsatisfiable, then there exists a particular set of *ground instances* of the clauses of S such that this set is also unsatisfiable (Herbrand's theorem).
- Then appeal to the **ground resolution theorem**, which states that propositional resolution is complete for ground sentences.
- Then use a **lifting lemma** to show that, for any propositional resolution proof using the set of ground sentences, there is a corresponding first-order resolution proof using the first-order sentences from which the ground sentences were obtained.
- Since there is always a resolution proof involving some finite subset of the Herbrand base of S, the next step is to show that there is a resolution proof using the clauses of S itself, which are not necessarily ground clauses. For this, a single application of the resolution rule is considered.

Concepts Required for Proof

- **Lifting Lemma**:

  Let C1 and C2 be two clauses with no shared variables, and let C1' and C2' be ground instances of C1 and C2. If C' is a resolvent of C1' and C2', then there exists a clause C such that (1) C is a resolvent of C1 and C2 and (2) C' is a ground instance of C.

- This is called a **lifting lemma**, because it lifts a proof step from ground clauses up to general first-order clauses.

- **Herbrand's theorem:**

  If a set S of clauses is unsatisfiable, then there exists a finite subset of $H_S(S)$ that is also unsatisfiable.

Sources

- https://www.cs.rochester.edu/~brown/173/lectures/logic/formal_logic/Resolution.html

Practice ……. !!!