

Resolution

- Clausal form.
 - Resolution works with formulas expressed in clausal form.
 - A **literal** is an atomic formula or the negation of an atomic formula. $\text{dog}(\text{fido})$, $\neg\text{cat}(\text{fido})$
 - A **clause** is a disjunction of literals:
 - $\neg\text{owns}(\text{fido},\text{fred}) \vee \neg\text{dog}(\text{fido}) \vee \text{person}(\text{fred})$
 - We write
 $(\neg\text{owns}(\text{fido},\text{fred}), \neg\text{dog}(\text{fido}), \text{person}(\text{fred}))$
 - A **clausal theory** is a conjunction of clauses.

Resolution Rule for Ground Clauses

- The resolution proof procedure consists of only one simple rule:
 - From the two clauses
 - $(P, Q_1, Q_2, \dots, Q_k)$
 - $(\neg P, R_1, R_2, \dots, R_n)$
 - We infer the new clause
 - $(Q_1, Q_2, \dots, Q_k, R_1, R_2, \dots, R_n)$
 - Example:
 - $(\neg \text{largerThan}(\text{clyde}, \text{cup}), \neg \text{fitsIn}(\text{clyde}, \text{cup}))$
 - $(\text{fitsIn}(\text{clyde}, \text{cup}))$
 - $\Rightarrow \neg \text{largerThan}(\text{clyde}, \text{cup})$

Resolution Proof: Forward chaining

- Logical consequences can be generated from the resolution rule in two ways:
 1. Forward Chaining inference.
 - If we have a sequence of clauses C_1, C_2, \dots, C_k
 - Such that each C_i is either in KB or is the result of a resolution step involving two prior clauses in the sequence.
 - We then have that $KB \vdash C_k$.

Forward chaining is sound so we also have $KB \models C_k$

Resolution Proof: Refutation proofs

2. Refutation proofs.

- We determine if $KB \vdash f$ by showing that a **contradiction** can be generated from $KB \wedge \neg f$.
- In this case a contradiction is an **empty** clause $()$.
- We employ resolution to construct a sequence of clauses C_1, C_2, \dots, C_m such that
 - C_i is in $KB \wedge \neg f$, or is the result of resolving two previous clauses in the sequence.
 - $C_m = ()$ i.e. its the empty clause.

Resolution Proof: Refutation proofs

- If we can find such a sequence $C_1, C_2, \dots, C_m = ()$, we have that
 - $KB \vdash f$.
 - Furthermore, this procedure is sound so
 - $KB \models f$
- And the procedure is also complete so it is capable of finding a proof of any f that is a logical consequence of KB . I.e.
 - If $KB \models f$ then we can generate a refutation from $KB \wedge \neg f$

Resolution Proofs Example

Want to prove $\text{likes}(\text{clyde}, \text{peanuts})$ from:

1. $(\text{elephant}(\text{clyde}), \text{giraffe}(\text{clyde}))$
2. $(\neg \text{elephant}(\text{clyde}), \text{likes}(\text{clyde}, \text{peanuts}))$
3. $(\neg \text{giraffe}(\text{clyde}), \text{likes}(\text{clyde}, \text{leaves}))$
4. $\neg \text{likes}(\text{clyde}, \text{leaves})$

Forward Chaining Proof:

- $3 \& 4 \rightarrow \neg \text{giraffe}(\text{clyde})$ [5.]
- $5 \& 1 \rightarrow \text{elephant}(\text{clyde})$ [6.]
- $6 \& 2 \rightarrow \text{likes}(\text{clyde}, \text{peanuts})$ [7.] ✓

Resolution Proofs Example

1. $(\text{elephant}(\text{clyde}), \text{giraffe}(\text{clyde}))$
2. $(\neg \text{elephant}(\text{clyde}), \text{likes}(\text{clyde}, \text{peanuts}))$
3. $(\neg \text{giraffe}(\text{clyde}), \text{likes}(\text{clyde}, \text{leaves}))$
4. $\neg \text{likes}(\text{clyde}, \text{leaves})$

Refutation Proof:

- $\neg \text{likes}(\text{clyde}, \text{peanuts})$ [5.]
- $5 \& 2 \rightarrow \neg \text{elephant}(\text{clyde})$ [6.]
- $6 \& 1 \rightarrow \text{giraffe}(\text{clyde})$ [7.]
- $7 \& 3 \rightarrow \text{likes}(\text{clyde}, \text{leaves})$ [8.]
- $8 \& 4 \rightarrow ()$ ✓

Resolution Proofs

- Proofs by refutation have the advantage that they are easier to find.
 - They are more focused to the particular conclusion we are trying to reach.
- To develop a complete resolution proof procedure for First-Order logic we need :
 1. A way of converting KB and f (the query) into clausal form.
 2. A way of doing resolution even when we have variables (unification).

Conversion to Clausal Form

To convert the KB into Clausal form we perform the following 8-step procedure:

1. **Eliminate Implications.**
2. **Move Negations inwards (and simplify $\neg\neg$).**
3. **Standardize Variables.**
4. **Skolemize.**
5. **Convert to Prenix Form.**
6. **Distribute conjunctions over disjunctions.**
7. **Flatten nested conjunctions and disjunctions.**
8. **Convert to Clauses.**

C-T-C-F: Eliminate implications

We use this example to show each step:

$$\forall X.p(X) \rightarrow ((\forall Y.p(Y) \rightarrow p(f(X,Y))) \\ \wedge \neg(\forall Y. \neg q(X,Y) \wedge p(Y)))$$

1. Eliminate implications: $A \rightarrow B \rightarrow \neg A \vee B$

$$\forall X. \neg p(X) \\ \vee ((\forall Y. \neg p(Y) \vee p(f(X,Y))) \\ \wedge \neg(\forall Y. \neg q(X,Y) \wedge p(Y)))$$

C-T-C-F: Move \neg Inwards

$$\begin{aligned} &\forall X. \neg p(X) \\ &\quad \vee \left((\forall Y. \neg p(Y) \vee p(f(X, Y))) \right. \\ &\quad \quad \left. \wedge \neg(\forall Y. \neg q(X, Y) \wedge p(Y)) \right) \end{aligned}$$

2. Move Negations Inwards (and simplify $\neg\neg$)

$$\begin{aligned} &\forall X. \neg p(X) \\ &\quad \vee \left((\forall Y. \neg p(Y) \vee p(f(X, Y))) \right. \\ &\quad \quad \left. \wedge (\exists Y. q(X, Y) \vee \neg p(Y)) \right) \end{aligned}$$

C-T-C-F: : \neg continue...

Rules for moving negations inwards

- $\neg(A \wedge B) \rightarrow \neg A \vee \neg B$
- $\neg(A \vee B) \rightarrow \neg A \wedge \neg B$
- $\neg\forall X. f \rightarrow \exists X. \neg f$
- $\neg\exists X. f \rightarrow \forall X. \neg f$
- $\neg\neg A \rightarrow A$

C-T-C-F: Standardize Variables

$$\forall X. \neg p(X) \\ \vee \left((\forall Y. \neg p(Y) \vee p(f(X, Y))) \right. \\ \left. \wedge (\exists Y. q(X, Y) \vee \neg p(Y)) \right)$$

3. Standardize Variables (Rename variables so that each quantified variable is unique)

$$\forall X. \neg p(X) \\ \vee \left((\forall Y. (\neg p(Y) \vee p(f(X, Y)))) \right. \\ \left. \wedge (\exists Z. q(X, Z) \vee \neg p(Z)) \right)$$

C-T-C-F: Skolemize

$$\begin{aligned} &\forall X. \neg p(X) \\ &\vee \left((\forall Y. \neg p(Y) \vee p(f(X, Y))) \right. \\ &\quad \left. \wedge (\exists Z. q(X, Z) \vee \neg p(Z)) \right) \end{aligned}$$

4. Skolemize (Remove existential quantifiers by introducing new function symbols).

$$\begin{aligned} &\forall X. \neg p(X) \\ &\vee \left((\forall Y. \neg p(Y) \vee p(f(X, Y))) \right. \\ &\quad \left. \wedge (q(X, g(X)) \vee \neg p(g(X))) \right) \end{aligned}$$

C-T-C-F: Skolemization

Consider $\exists Y.\text{elephant}(Y) \wedge \text{friendly}(Y)$

- This asserts that there is some individual (binding for Y) that is both an elephant and friendly.
- To remove the existential, we **invent** a name for this individual, say a . This is a new constant symbol **not equal to any previous constant symbols** to obtain:

$\text{elephant}(a) \wedge \text{friendly}(a)$

- This is saying the same thing, since we do not know anything about the new constant a .

C-T-C-F: Skolemization

- It is essential that the introduced symbol “a” is **new**. Else we might know something else about “a” in KB.
- If we did know something else about “a” we would be asserting more than the existential.
- In original quantified formula we know nothing about the variable “Y”. Just what was being asserted by the existential formula.

C-T-C-F: Skolemization

Now consider $\forall X \exists Y. \text{loves}(X, Y)$.

- This formula claims that for every X there is some Y that X loves (perhaps a different Y for each X).
- Replacing the existential by a new constant won't work

$\forall X. \text{loves}(X, a)$.

Because this asserts that there is a **particular** individual “a” loved by every X .

- To properly convert existential quantifiers scoped by universal quantifiers we must use **functions** not just constants.

C-T-C-F: Skolemization

- We must use a function that mentions every universally quantified variable that scopes the existential.
- In this case X scopes Y so we must replace the existential Y by a function of X

$$\forall X. \text{loves}(X, g(X)).$$

where g is a **new** function symbol.

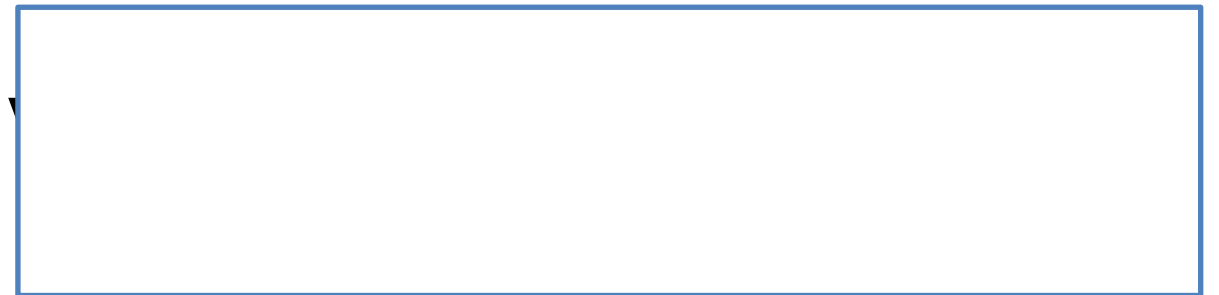
- This formula asserts that for every X there is some individual (given by $g(X)$) that X loves. $g(X)$ can be different for each different binding of X .

C-T-C-F: Skolemization Examples

- $\forall XYZ \exists W. r(X, Y, Z, W) \rightarrow$




- $\forall XY \exists W. r(X, Y, g(W)) \rightarrow$



- $\forall XY \exists W \forall Z. r(X, Y, W) \wedge q(Z, W)$

\rightarrow



C-T-C-F: Convert to prenex

$$\forall X. \neg p(X)$$

$$\vee \left(\forall Y. \neg p(Y) \vee p(f(X, Y)) \right. \\ \left. \wedge q(X, g(X)) \vee \neg p(g(X)) \right)$$

5. Convert to prenex form. (Bring all quantifiers to the front—only universals, each with different name).

$$\forall X \forall Y. \neg p(X)$$

$$\vee \left(\neg p(Y) \vee p(f(X, Y)) \right. \\ \left. \wedge q(X, g(X)) \vee \neg p(g(X)) \right)$$

C-T-C-F: Conjunctions over disjunctions

$$\forall X \forall Y. \neg p(X)$$

$$\vee \left(\left(\neg p(Y) \vee p(f(X, Y)) \right) \right. \\ \left. \wedge \left(q(X, g(X)) \vee \neg p(g(X)) \right) \right)$$

6. Conjunctions over disjunctions

$$A \vee (B \wedge C) \rightarrow (A \vee B) \wedge (A \vee C)$$

$$\forall X Y. \left(\neg p(X) \vee \neg p(Y) \vee p(f(X, Y)) \right) \\ \wedge \left(\neg p(X) \vee q(X, g(X)) \vee \neg p(g(X)) \right)$$

C-T-C-F: flatten & convert to clauses

7. Flatten nested conjunctions and disjunctions.

$$(A \vee (B \vee C)) \rightarrow (A \vee B \vee C)$$

8. Convert to Clauses (remove quantifiers and break apart conjunctions).

$$\forall XY. (\neg p(X) \vee \neg p(Y) \vee p(f(X, Y)))$$

$$\wedge (\neg p(X) \vee q(X, g(X)) \vee \neg p(g(X)))$$

a) $\neg p(X) \vee \neg p(Y) \vee p(f(X, Y))$

b) $\neg p(X) \vee q(X, g(X)) \vee \neg p(g(X))$

Unification

- Ground clauses are clauses with no variables in them. For ground clauses we can use syntactic identity to detect when we have a P and $\neg P$ pair.
- What about variables? can the clauses
 - $(P(\text{john}), Q(\text{fred}), R(X))$
 - $(\neg P(Y), R(\text{susan}), R(Y))$Be resolved?

Unification.

- Intuitively, once reduced to clausal form, all remaining variables are universally quantified. So, implicitly $(\neg P(Y), R(\text{susan}), R(Y))$ represents a whole set of ground clauses like
 - $(\neg P(\text{fred}), R(\text{susan}), R(\text{fred}))$
 - $(\neg P(\text{john}), R(\text{susan}), R(\text{john}))$
 - ...
- So there is a “specialization” of this clause that can be resolved with $(P(\text{john}), Q(\text{fred}), R(X))$

Unification.

- We want to be able to match conflicting literals, even when they have variables. This matching process automatically determines whether or not there is a “specialization” that matches.
- We don’t want to over specialize!

Unification.

- $(\neg p(X), s(X), q(\text{fred}))$
- $(p(Y), r(Y))$
- Possible resolvants
 - $(s(\text{john}), q(\text{fred}), r(\text{john})) \{Y=X, X=\text{john}\}$
 - $(s(\text{sally}), q(\text{fred}), r(\text{sally})) \{Y=X, X=\text{sally}\}$
 - $(s(X), q(\text{fred}), r(X)) \{Y=X\}$
- The last resolvent is “**most-general**”, the other two are specializations of it.
- We want to keep the most general clause so that we can use it future resolution steps.

Unification.

- **unification** is a mechanism for finding a “most general” matching.
- First we consider **substitutions**.
 - A substitution is a finite set of equations of the form

$$V = t$$

where V is a variable and t is a term not containing V . (t might contain other variables).

Substitutions.

- We can **apply** a substitution σ to a formula f to obtain a new formula $f\sigma$ by **simultaneously** replacing every variable mentioned in the left hand side of the substitution by the right hand side.

$$p(X, g(Y, Z))[X=Y, Y=f(a)] \rightarrow p(Y, g(f(a), Z))$$

- Note that the substitutions are not applied sequentially, i.e., the first Y is not subsequently replaced by $f(a)$.

Substitutions.

- We can compose two substitutions. θ and σ to obtain a new substitution $\theta\sigma$.

Let $\theta = \{X_1=s_1, X_2=s_2, \dots, X_m=s_m\}$

$\sigma = \{Y_1=t_1, Y_2=t_2, \dots, Y_k=s_k\}$

To compute $\theta\sigma$

$$1. \quad S = \{X_1=s_1\sigma, X_2=s_2\sigma, \dots, X_m=s_m\sigma, Y_1=t_1, Y_2=t_2, \dots, Y_k=s_k\}$$

we apply σ to each RHS of θ and then add all of the equations of σ .

Substitutions.

1. $S = \{X_1=s_1\sigma, X_2=s_2\sigma, \dots, X_m=s_m\sigma, Y_1=t_1, Y_2=t_2, \dots, Y_k=s_k\}$
2. Delete any identities, i.e., equations of the form $V=V$.
3. Delete any equation $Y_i=s_i$ where Y_i is equal to one of the X_j in θ .

The final set S is the composition $\theta\sigma$.

Composition Example.

$$\theta = \{X=f(Y), Y=Z\}, \sigma = \{X=a, Y=b, Z=Y\}$$

$\theta\sigma$

1. $S = \theta$ with σ applied to RH sides followed by σ
 $= \{X=f(Y) \{X=a, Y=b, Z=Y\}, Y=Z \{X=a, Y=b, Z=Y\},$
 $X=a, Y=b, Z=Y\}$
 $= \{X=f(b), Y=Y, X=a, Y=b, Z=Y\}$
2. $\{X=f(b), \cancel{Y=Y}, X=a, Y=b, Z=Y\}$ #delete identities
3. $\{X=f(b), \cancel{X=a}, \cancel{Y=b}, Z=Y\}$ #X and Y were in θ

$$\theta\sigma = \{X = f(b), Z=Y\}$$

Substitutions.

- The empty substitution $\varepsilon = \{\}$ is also a substitution, and it acts as an identity under composition.
- More importantly substitutions when applied to formulas are associative:

$$(f\theta)\sigma = f(\theta\sigma)$$

- Composition is simply a way of converting the sequential application of a series of substitutions to a single simultaneous substitution.

Unifiers.

- A **unifier** of two formulas f and g is a substitution σ that makes f and g **syntactically identical**.
- Not all formulas can be unified—substitutions only affect variables.

$$p(f(X),a) \quad p(Y,f(w))$$

- This pair cannot be unified as there is no way of making $a = f(w)$ with a substitution.
- Note we typically use UPPER CASE to denote variables, lower case for constants.

MGU.

- A substitution σ of two formulas f and g is a **Most General Unifier (MGU)** if
 1. σ is a unifier.
 2. For every other unifier θ of f and g there must exist a third substitution λ such that
$$\theta = \sigma\lambda$$
- This says that every other unifier is “more specialized than σ . The MGU of a pair of formulas f and g is unique up to renaming.

MGU.

$$p(f(X),Z) \quad p(Y,a)$$

1. $\sigma = \{Y = f(a), X=a, Z=a\}$ is a unifier.

$$\begin{aligned} p(f(X),Z)\sigma &= p(f(a),a) \\ p(Y,a)\sigma &= p(f(a),a) \end{aligned}$$

But it is not an MGU.

2. $\theta = \{Y=f(X), Z=a\}$ is an MGU.

$$\begin{aligned} p(f(X),Z) \theta &= p(f(X),a) \\ p(Y,a) \theta &= p(f(X),a) \end{aligned}$$

MGU.

$$p(f(X),Z) \quad p(Y,a)$$

3. $\sigma = \theta\lambda$, where $\lambda = \{X=a\}$

$$\sigma = \{Y = f(a), X=a, Z=a\}$$

$$\theta = \{Y=f(X), Z=a\}$$

$$\lambda = \{X=a\}$$

$$\theta\lambda = \{Y=f(a), Z=a, X=a\}$$

MGU.

- The MGU is the “least specialized” way of making clauses with universal variables match.
- We can compute MGUs.
- Intuitively we line up the two formulas and find the first sub-expression where they disagree. The pair of subexpressions where they **first** disagree is called the **disagreement set**.
- The algorithm works by successively fixing disagreement sets until the two formulas become syntactically identical.

MGU.

To find the MGU of two formulas f and g .

1. $k = 0$; $\sigma_0 = \{\}$; $S_0 = \{f, g\}$
2. If S_k contains an identical pair of formulas stop, and return σ_k as the MGU of f and g .
3. Else find the disagreement set $D_k = \{e_1, e_2\}$ of S_k
4. If $e_1 = V$ a variable, and $e_2 = t$ a term not containing V (or vice-versa) then let

$\sigma_{k+1} = \sigma_k \{V=t\}$ (Compose the additional substitution)

$S_{k+1} = S_k \{V=t\}$ (Apply the additional substitution)

$k = k+1$
GOTO 2

5. Else stop, f and g cannot be unified.

MGU Example 1.

$$S_0 = \{p(f(a), g(X)) ; p(Y, Y)\}$$

$$k=0, \sigma_0 = \{\}$$

1. $D_0 = \{f(a), Y\}$ - ok we have a variable Y and a term not containing Y .

$$2. \sigma_1 = \{\} \{Y=f(a)\} = \{Y=f(a)\}$$

$$S_1 = S_0 \{Y=f(a)\} = \{p(f(a), g(X)) ; p(f(a), f(a))\}$$

$$3. k=1$$

1. $D_1 = \{g(X), f(a)\}$ - fail we do not have a variable and a term not containing a new variable

Not unifiable!

MGU Example 2.

$$S_0 = \{p(a, X, h(g(Z))) ; p(Z, h(Y), h(Y))\}$$

$$k=0, \sigma_0 = \{\}$$

1. $D_0 = \{a, Z\}$ - ok we have a variable Z and a term not containing Z .

$$2. \sigma_1 = \{\} \{Z=a\} = \{Z=a\}$$

$$S_1 = S_0 \{Z=a\} = \{p(a, X, h(g(a))); p(a, h(Y), h(Y))\}$$

$$3. k=1$$

$$1. D_1 = \{X, h(Y)\} - \text{ok}$$

$$2. \sigma_2 = \{Z=a\} \{X=h(Y)\} = \{Z=a, X=h(Y)\}$$

$$S_2 = S_1 \{X=h(Y)\} = \{p(a, h(Y), h(g(a))); p(a, h(Y), h(Y))\}$$

$$3. k=2$$

MGU Example 2.

$$S_0 = \{p(a, X, h(g(Z))) \ ; \ p(Z, h(Y), h(Y))\}$$

1. $D_2 = \{g(a), Y\}$ - ok

2. $\sigma_3 = \{Z=a, X=h(Y)\} \{Y=g(a)\} = \{Z=a, X=h(g(a)), Y=g(a)\}$

$$S_3 = S_2 \{Y = g(a)\} = \{p(a, h(g(a)), h(g(a))); \\ p(a, h(g(a)), h(g(a)))\}$$

3. $k=3$

1. $D_3 = \{\}$ - done!

$$\text{MGU} = \sigma_3 = \{Z=a, X=h(g(a)), Y=g(a)\}$$

MGU Example 3.

$$S_0 = \{p(X,X) ; p(Y,f(Y))\}$$

$$k = 0, \sigma_0 = \{\}$$

1. $D_0 = \{X, Y\}$ - ok we have a variable X and a term(Y) not containing X .

$$2. \sigma_1 = \{\} \{X=Y\} = \{X=Y\}$$

$$S_1 = S_0 \{X=Y\} = \{p(Y,Y), p(Y,f(Y))\}$$

$$3. k = 1$$

1. $D_1 = \{Y, f(Y)\}$ - fail! We have a variable Y but the term $f(Y)$ contains Y !

Not Unifiable

Non-Ground Resolution

- Resolution of non-ground clauses. From the two clauses
 $(L, Q_1, Q_2, \dots, Q_k)$
 $(\neg M, R_1, R_2, \dots, R_n)$

Where there exists σ a MGU for L and M .

We infer the new clause

$$(Q_1\sigma, \dots, Q_k\sigma, R_1\sigma, \dots, R_n\sigma)$$

Non-Ground Resolution E.G.

1. $(p(X), q(g(X)))$
2. $(r(a), q(Z), \neg p(a))$

$$L=p(X); M=p(a)$$
$$\sigma = \{X=a\}$$

3. $R[1a,2c]\{X=a\} (q(g(a)), r(a), q(Z))$

The notation is important. You will need to use this notation on the exam!

- “R” means resolution step.
- “1a” means the **first** (**a**-th) literal in the first clause i.e. $p(X)$.
- “2c” means the **third** (**c**-th) literal in the second clause, $\neg p(a)$.
 - 1a and 2c are the “clashing” literals.
- $\{X=a\}$ is the substitution applied to make the clashing literals identical.

Resolution Proof Example

“Some patients like all doctors. No patient likes any quack. Therefore no doctor is a quack.”

Resolution Proof Step 1.

Pick symbols to represent these assertions.

$p(X)$: X is a patient

$d(X)$: X is a doctor

$q(X)$: X is a quack

$l(X, Y)$: X likes Y

Resolution Proof Example

Resolution Proof Step 2.

Convert each assertion to a first-order formula.

1. Some patients like all doctors.

F1. $\exists X. p(X) \wedge \forall Y. (d(Y) \rightarrow I(X, Y))$

Resolution Proof Example

2. No patient likes any quack

F2. $\neg (\exists X. p(X) \wedge \exists Y. q(Y) \wedge I(X, Y))$

3. Therefore no doctor is a quack.

Query. $\forall X. d(X) \rightarrow \neg q(X)$

Resolution Proof Example

Resolution Proof Step 3.

Convert to Clausal form.

$$F1. \exists X. p(X) \wedge \forall Y. (d(Y) \rightarrow I(X, Y))$$

$$\exists X. p(X) \wedge \forall Y. (\neg d(Y) \vee I(X, Y))$$

$$\forall Y. p(a) \wedge (\neg d(Y) \vee I(a, Y)) \text{ \#Skolem constant}$$

$$1. p(a)$$

$$2. \neg d(Y) \vee I(a, Y)$$

$$F2. \neg (\exists X. p(X) \wedge \exists Y. q(Y) \wedge I(X, Y))$$

$$\forall X. \neg p(X) \vee \forall Y. \neg q(Y) \vee \neg I(X, Y)$$

$$\forall X, Y. \neg p(X) \vee \neg q(Y) \vee \neg I(X, Y)$$

$$3. \neg p(X) \vee \neg q(Y) \vee \neg I(X, Y)$$

Resolution Proof Example

Resolution Proof Step 3.

Negation of Query.

$$\neg(\forall X. d(X) \rightarrow \neg q(X))$$

$$\neg(\forall X. \neg d(X) \vee \neg q(X))$$

$$\exists X. d(X) \wedge q(X)$$

$$d(b) \wedge q(b). \text{ \#Skolem constant}$$

$$4. d(b)$$

$$5. q(b)$$

Resolution Proof Example

Resolution Proof Step 4.

Resolution Proof from the Clauses.

1. $p(a)$
2. $(\neg d(Y), l(a, Y))$
3. $(\neg p(X), \neg q(Y), \neg l(X, Y))$
4. $d(b)$
5. $q(b)$

If we have to resolve two clauses that have the same variable, we always rename the variables in one clause so that there are no shared variables

Resolution Proof Example

Resolution Proof Step 4.

Resolution Proof from the Clauses.

1. $p(a)$
2. $(\neg d(Y), l(a, Y))$
3. $(\neg p(Z), \neg q(R), \neg l(Z, R))$
4. $d(b)$
5. $q(b)$
6. $R[3b, 5] \{R=b\} (\neg p(Z), \neg l(Z, b))$
7. $R[6a, 1] \{Z=a\} \neg l(a, b)$
8. $R[7, 2b] \{Y=b\} \neg d(b)$
9. $R[8, 4]() \text{ \#contradiction proving query is true.}$

Answer Extraction.

- The previous example shows how we can answer true-false questions. With a bit more effort we can also answer “fill-in-the-blanks” questions (e.g., what is wrong with the car?).
- As in Prolog we use free variables in the query where we want the fill in the blanks. We simply need to keep track of the binding that these variables received in proving the query.
 - `parent(art, jon)` –is art one of jon’s parents?
 - `parent(X, jon)` -who is one of jon’s parents?

Answer Extraction.

- A simple bookkeeping device is to use a predicate symbol $\text{answer}(X, Y, \dots)$ to keep track of the bindings automatically.
- To answer the query $\text{parent}(X, \text{jon})$, we construct the clause
$$(\neg \text{parent}(X, \text{jon}), \text{answer}(X))$$
- Now we perform resolution until we obtain a clause consisting of only answer literals (previously we stopped at empty clauses).

Answer Extraction: Example 1

1. $\text{father}(\text{art}, \text{jon})$
2. $\text{father}(\text{bob}, \text{kim})$
3. $(\neg \text{father}(Y, Z), \text{parent}(Y, Z))$
i.e. *all fathers are parents*
4. $(\neg \text{parent}(X, \text{jon}), \text{answer}(X))$
i.e. the query is: who is parent of jon?

Here is a resolution proof:

5. $R[4a, 3b] \{Y=X, Z=\text{jon}\}$
 $(\neg \text{father}(X, \text{jon}), \text{answer}(X))$
6. $R[5, 1] \{X=\text{art}\}$ **answer(art)**
so **art** is parent of jon

Answer Extraction: Example 2

1. $(\text{father}(\text{art}, \text{jon}), \text{father}(\text{bob}, \text{jon}))$ //either bob or art is parent of jon
2. $\text{father}(\text{bob}, \text{kim})$
3. $(\neg \text{father}(Y, Z), \text{parent}(Y, Z))$ //i.e. all fathers are parents
4. $(\neg \text{parent}(X, \text{jon}), \text{answer}(X))$ //i.e. query is $\text{parent}(X, \text{jon})$

Here is a resolution proof:

5. $R[4a, 3a] \{Y=X, Z=\text{jon}\} (\neg \text{father}(X, \text{jon}), \text{answer}(X))$
6. $R[5a, 1a] \{X=\text{art}\} (\text{father}(\text{bob}, \text{jon}), \text{answer}(\text{art}))$
7. $R[6a, 3a] \{Y=\text{bob}, Z=\text{jon}\}$
 $(\text{parent}(\text{bob}, \text{jon}), \text{answer}(\text{art}))$
8. $R[7a, 4a] \{X=\text{bob}\} (\text{answer}(\text{bob}), \text{answer}(\text{art}))$

A disjunctive answer: either bob or art is parent of jon.

Factoring

1. $(p(X), p(Y))$ $// \forall X. \forall Y. \neg p(X) \rightarrow p(Y)$
2. $(\neg p(V), \neg p(W))$ $// \forall V. \forall W. p(V) \rightarrow \neg p(W)$

- These clauses are intuitively contradictory, but following the strict rules of resolution only we obtain:

3. $R[1a, 2a](X=V) (p(Y), \neg p(W))$
Renaming variables: $(p(Q), \neg p(Z))$
4. $R[3b, 1a](X=Z) (p(Y), p(Q))$

No way of generating empty clause!

Factoring is needed to make resolution over non-ground clauses complete,
without it resolution is incomplete!

Factoring.

- If two or more literals of a clause C have an mgu θ , then $C\theta$ with all duplicate literals removed is called a **factor** of C .
- $C = (p(X), p(f(Y)), \neg q(X))$
 $\theta = \{X=f(Y)\}$
 $C\theta = (p(f(Y)), p(f(Y)), \neg q(f(Y))) \rightarrow (p(f(Y)), \neg q(f(Y)))$ is a factor

Adding a factor of a clause can be a step of proof:

1. $(p(X), p(Y))$
2. $(\neg p(V), \neg p(W))$
3. $f[1ab]\{X=Y\} p(Y)$
4. $f[2ab]\{V=W\} \neg p(W)$
5. $R[3,4]\{Y=W\} ()$.

Review: One Last Example!

Consider the following English description

- Whoever can read is literate.
 - Dolphins are not literate.
 - Flipper is an intelligent dolphin.
-
- Who is intelligent but cannot read.

Example: pick symbols & convert to first-order formula

- Whoever can read is literate.
 - $\forall X. \text{read}(X) \rightarrow \text{lit}(X)$
- Dolphins are not literate.
 - $\forall X. \text{dolph}(X) \rightarrow \neg \text{lit}(X)$
- Flipper is an intelligent dolphin
 - $\text{dolph}(\text{flipper}) \wedge \text{intell}(\text{flipper})$
- Who is intelligent but cannot read?
 - $\exists X. \text{intell}(X) \wedge \neg \text{read}(X).$

Example: convert to clausal form

- $\forall X. \text{read}(X) \rightarrow \text{lit}(X)$
 - $(\neg \text{read}(X), \text{lit}(X))$
- Dolphins are not literate.
 $\forall X. \text{dolph}(X) \rightarrow \neg \text{lit}(X)$
 - $(\neg \text{dolph}(X), \neg \text{lit}(X))$
- Flipper is an intelligent dolphin.
 - $\text{dolph}(\text{flipper})$
 - $\text{intell}(\text{flipper})$
- who are intelligent but cannot read?
 - $\exists X. \text{intell}(X) \wedge \neg \text{read}(X).$
 - $\rightarrow \forall X. \neg \text{intell}(X) \vee \text{read}(X)$
 - $\rightarrow (\neg \text{intell}(X), \text{read}(X), \text{answer}(X))$

Example: do the resolution proof

1. $(\neg \text{read}(X), \text{lit}(X))$
2. $(\neg \text{dolp}(X), \neg \text{lit}(X))$
3. $\text{dolp}(\text{flip})$
4. $\text{intell}(\text{flip})$
5. $(\neg \text{intell}(X), \text{read}(X), \text{answer}(X))$

6. $R[5a,4] \text{ } X=\text{flip}. (\text{read}(\text{flip}), \text{answer}(\text{flip}))$
7. $R[6a,1a] \text{ } X=\text{flip}. (\text{lit}(\text{flip}), \text{answer}(\text{flip}))$
8. $R[7a,2b] \text{ } X=\text{flip}. (\neg \text{dolp}(\text{flip}), \text{answer}(\text{flip}))$
9. $R[8a,3] \text{ } \text{answer}(\text{flip})$

so flip is intelligent but cannot read!