Example of Computing MGUs

Apply algorithm to find mgu for $p(f(x), f(x))$ and $p(y, f(a))$

Predicates match; unify the arguments.

Unify first arguments $f(x)$ and $y$

Result:

Apply unifier to second arguments $f(x)$ and $f(a)$ (unchanged)

Then, unify $f(x)$ and $f(a)$:

Compose $[y \mapsto f(x)]$ and $[x \mapsto a]$

Final result:

Another Example

Apply algorithm to find mgu for $p(x, x)$ and $p(y, f(y))$

Predicates match; unify the arguments.

Unify first arguments $x$ and $y$; result:

Apply unifier to second arguments $x$ and $f(y)$:

Now unify $y$ and $f(y)$:

Thus $p(x, x)$ and $p(y, f(y))$ not unifiable
Clausal Form

- A formula in FOL in said to be in clausal form if obeys following syntactic restrictions:
  1. Formula should be of the form $\forall x_1, \ldots, x_k. F(x_1, \ldots, x_k)$ (i.e., only universally quantified variables)
  2. The inner formula $F(x_1, \ldots, x_k)$ should be in CNF

The Bad and The Good News

- **Bad News:**
  In general, if $\phi$ is the original formula, there may not be an equivalent formula $\phi'$ that is of this form
- **Good News:**
  But we can always find an equi-satisfiable formula $\phi''$ that is of this form
  Since we are trying to determine satisfiability of $\phi$, this is good enough . . .

Converting Formulas to Equisatisfiable Clausal Form

Given formula $\phi$, there are five steps to convert it to equisatisfiable clausal form:

1. Make sure there are no free variables in $\phi$
2. Convert resulting formula to Prenex normal form
3. Apply skolemization to remove existentially quantified variables (resulting formula called Skolem Normal Form)
4. Since formula obtained after step 3 is of the form $\forall x_1, \ldots, x_k. F(x_1, \ldots, x_k)$, convert inner formula $F$ to CNF
5. Since all variables are universally quantified, drop explicit quantifiers and write formula as set of clauses

Prenex Normal Form

- A formula is in prenex normal form (PNF) if all of its quantifiers appear at the beginning of formula:
  $Qx_1, \ldots, Qx_n. F(x_1, \ldots, x_k)$
  where $F$ is quantifier-free and $Q \in \{\forall, \exists\}$
- Is $\forall x. \exists y. (p(x, y) \rightarrow q(x))$ in PNF?
- What about $\forall x. ((\exists y. p(x, y)) \rightarrow q(x))$ in PNF?

Step 1: Removing Free Variables

- Suppose a formula $\phi$ contains free variable $x$
  - $\phi$ is satisfiable iff $U, I, \sigma \models \phi$ for some variable assignment $\sigma$
  - Thus, $\phi$ is satisfiable iff there exists some $o \in U$ under which $U, I, \{x \rightarrow o\} \models \phi$
  - But this is the same as saying $\phi$ is satisfiable iff $U, I \models \exists x. \phi$
  - If a formula $\phi$ contains free variables $\vec{x}$, a closed formula equisatisfiable to $\phi$
  - Thus, to perform step 1 of transformation, existentially quantify all free variables of $\phi$

Step 2: Conversion to Prenex Normal Form

- **Step 2a:** Convert to NNF.
  - Conversion to NNF is just like in propositional logic, but need new equivalences for distributing negation over quantifiers:
    - $\neg \forall x. \phi \iff \exists x. \neg \phi$
    - $\neg \exists x. \phi \iff \forall x. \neg \phi$
- **Step 2b:** Rename quantified variables as necessary so no two quantified variables have the same name.
- **Step 2c:** Move quantifiers to front of formula $Q_1 x_1, \ldots, Q_n x_n. F'$ such that if $Q_j$ is in the scope of quantifier $Q_i$, then $i < j$.
  - **Claim:** Formula in PNF is equivalent to original formula.
Skolemization: Intuition I
- Consider formula $\exists x. F$
- We know there is some object for which $F$ holds, but we don’t want to make any assumptions about this object
- Thus, we replace $x$ with a fresh object constant $c$ in $F$
- This is the same reason as why we introduced a fresh object in proof rules for semantic argument method
- The formula $F[c/x]$ is equisatisfiable to $\exists x. F$, but not equivalent

Skolemization: Intuition II
- However, if existential variable $x$ is in scope of universally quantified variables, we can’t replace it with object constant
  - Consider formula: $\forall x. \exists y. \text{hates}(x, y)$
  - What does this formula say?
  - Now, let’s replace $y$ with object constant $c$: $\forall x. \text{hates}(x, c)$
  - What does this formula say?
  - Clearly, very different meaning!

Skolemization: Intuition III
- Consider a formula of the form $\forall x. \exists y. F$
- We know that for each object $o$, there exists some object $o'$ for which $F$ holds
- But for different $o$’s, the $o$’s can be different
- For $\forall x. \exists y. \text{hates}(x, y)$, it is possible that Joe and David hate different people
- Thus, we replace $y$ with $f(x)$
- Observe: The formula $\forall x. \text{hates}(x, f(x))$ doesn’t imply that Joe and David have to hate the same person

Skolem Normal Form
- The formula after performing skolemization looks like this:
  $$\forall x_1, \ldots, x_n. F(x_1, \ldots, x_n)$$
- This form is called Skolem Normal Form
- Resulting formula not equivalent to original formula, but equisatisfiable

Conversion to PNF Example
- Convert formula to PNF:
  $$\forall x. (\neg (\exists y. (p(x, y) \land p(x, z))) \lor \exists y. p(x, y))$$
Conversion to Clausal Form Example

Example II

In Skolem Normal Form:
\[ \forall y. (p(y) \land (r(f(y)) \land \neg q(y, f(y), c))) \]

Step 4: Convert inner formula to CNF (already in CNF)

Step 5: Drop universal quantifiers:
\[ (p(y) \land (r(f(y)) \land \neg q(y, f(y), c))) \]

Step 6: Finally, write formula as a set of clauses
\[ \{p(y), \{r(f(y))\}, \{q(y, f(y), c)\} \] This formula is now in **clausal form**

Example II, cont

In Skolem Normal Form:
\[ \exists w \exists x \exists y \exists z. ((\neg q(w, z)) \lor (\neg p(x, y) \land r(y))) \]

Step 1, 2a: No free variables, convert to NNF:
\[ \exists w \forall x \forall z. (\neg q(w, z)) \lor \exists y. (\neg p(x, y) \land r(y)) \]

Step 2b: Move quantifiers out (necessary for PNF):
\[ \exists w \forall x \forall z. (\neg q(w, z)) \lor (\neg p(x, y) \land r(y)) \]

In Prenex Normal Form:
\[ \exists w \forall x \forall y. \exists z. (p(y) \land (r(z) \land \neg q(y, z, w))) \]

Step 1: Remove free variables:
\[ \exists w \forall y. (p(y) \land \neg (\forall z. (r(z) \land \neg q(y, z, w)))) \]

Step 2a: Convert to NNF (necessary for PNF):
\[ \exists w \forall y. (p(y) \land (\exists z. (r(z) \land \neg q(y, z, w)))) \]

Push negations

Step 2b: Move quantifiers out (necessary for PNF):
\[ \exists w \forall y. (p(y) \land (\exists z. (r(z) \land \neg q(y, z, w)))) \]

In Skolem Normal Form:
\[ \forall y. (p(y) \land (r(f(y)) \land \neg q(y, f(y), c))) \]

Step 4: Convert inner formula to CNF (already in CNF)

Step 5: Drop universal quantifiers:
\[ (p(y) \land (r(f(y)) \land \neg q(y, f(y), c))) \]

Step 6: Finally, write formula as a set of clauses
\[ \{p(y), \{r(f(y))\}, \{q(y, f(y), c)\} \]

This formula is now in **clausal form**

Conversion to Clausal Form Example

In Prenex Normal Form:
\[ \exists w \forall y. (p(y) \land (r(z) \land \neg q(y, z, w))) \]

Step 1: Remove free variables:
\[ \exists w \forall y. (p(y) \land \neg (\forall z. (r(z) \land \neg q(y, z, w)))) \]

Step 2a: Convert to NNF (necessary for PNF):
\[ \exists w \forall y. (p(y) \land (\exists z. (r(z) \land \neg q(y, z, w)))) \]

Push negations

Step 2b: Move quantifiers out (necessary for PNF):
\[ \exists w \forall y. (p(y) \land (\exists z. (r(z) \land \neg q(y, z, w)))) \]

In Skolem Normal Form:
\[ \forall y. (p(y) \land (r(f(y)) \land \neg q(y, f(y), c))) \]

Step 4: Convert inner formula to CNF (already in CNF)

Step 5: Drop universal quantifiers:
\[ (p(y) \land (r(f(y)) \land \neg q(y, f(y), c))) \]

Step 6: Finally, write formula as a set of clauses
\[ \{p(y), \{r(f(y))\}, \{q(y, f(y), c)\} \] This formula is now in **clausal form**

Example II, cont

In Skolem Normal Form:
\[ \forall z. ((\neg q(c, z)) \lor (\neg p(x, f(x)) \land r(y))) \]

Step 4: Convert inner formula to CNF
\[ \forall z. ((\neg q(c, z)) \lor (\neg p(x, f(x)) \land r(y))) \]

Step 5: Drop universal quantifiers:
\[ (\neg q(c, z)) \lor (\neg p(x, f(x)) \land r(y)) \]

Step 6: Finally, write formula as a set of clauses
\[ \{\neg q(c, z), \neg p(x, f(x))\} \]
A Word About Clausal Form

- Consider the clausal form \{l_1, l_2, \ldots, l_k\} \ldots \{l'_1, l'_2, \ldots, l'_n\}
- Assuming clauses contain variables \(x_1, \ldots, x_n\), what is the meaning of this clausal form as a proper FOL formula?
- Recall: Universal quantifiers distribute over conjuncts:
  \[
  \forall x_1, \ldots, x_n. (l_1 \lor l_2 \ldots \lor l_k) \land \ldots \land (l'_1 \lor l'_2 \ldots \lor l'_n) 
  \]
- Thus above formula is equivalent to:
  \[
  \forall x_1, \ldots, x_n. (l_1 \lor l_2 \ldots \lor l_k) \ldots \land (l'_1 \lor l'_2 \ldots \lor l'_n) 
  \]

Clausal Form and Renaming Variables

- In rest of lecture, we assume that we rename variables in each clause so different clauses contain different variables.
- This is necessary to ensure that we don’t get conflicting names as we do resolution.
- For instance, if we have two clauses \{p(a,x)\} and \{\neg p(b,y)\}, we assume they are renamed as \{\tilde{p}(a,x)\} and \{\neg \tilde{p}(b,y)\}

First Order Resolution

- To apply first-order resolution, convert formula to clausal form
- Rename variables to ensure each clause contains different variables
- Resolution:
  \[
  \frac{\{A, B_1, \ldots, B_k\}, \{\neg C, D_1, \ldots, D_n\}, \{\neg C, D_1, \ldots, D_n\}}{\{B_1, \ldots, B_k, D_1, \ldots, D_n\} \sigma} (\sigma = mgu(A, C)) 
  \]
- What is the result of performing resolution on the following clauses?
  - Clause 1: \{p(a, y), r(g(y))\}
  - Clause 2: \{\neg p(x, f(x)), q(g(x))\}
  - Mgu for \(p(a,y)\) and \(p(x,f(x))\):
  - Resolvent:

Example

Resolution:
\[
\begin{align*}
  \{A, B_1, \ldots, B_k\} & \quad \{\neg C, D_1, \ldots, D_n\} \quad (\sigma = mgu(A, C)) \\
  \{B_1, \ldots, B_k, D_1, \ldots, D_n\} \sigma
\end{align*}
\]
- What is the result of performing resolution on the following clauses?
  - Clause 1: \{p(a, y), r(g(y))\}
  - Clause 2: \{\neg p(x, f(x)), q(g(x))\}
- Mgu for \(p(a,y)\) and \(p(x,f(x))\):
- Resolvent:

Intuition about First-Order Resolution

- Intuition: Consider two clauses: \{\text{happy}(x), \text{sad}(x)\} and \{\neg \text{happy}(\text{joe}), \text{happy}(\text{sally})\}
- The first clause says:
  - This implies: \text{happy}(\text{joe}) \lor \text{sad}(\text{joe})
- The second clause says:
  - Two possibilities: Either Joe is happy or not.
    - If \text{happy}(\text{joe}), second clause implies \text{happy}(\text{sally})
    - If \neg \text{happy}(\text{joe}), then we have \text{sad}(\text{joe})
- In either case, we have \text{happy}(\text{sally}) \lor \text{sad}(\text{joe})
Intuition about First-Order Resolution, cont.

\[
\begin{align*}
\{A, B_1, \ldots, B_k\} & \vdash \{\neg C, D_1, \ldots, D_k\} \\
\{B_1, \ldots, B_k, D_1, \ldots, D_k\} \sigma & (\sigma = \text{mgu}(A, C))
\end{align*}
\]

Why Most General Unifiers?

Why do we need most general unifiers, not just any unifier?

Example: Consider clauses: \{happy(x), sad(x)\} \quad \{\neg sad(y)\}

Most general unifier: 

\{\text{happy}(x), \text{sad}(x)\} \quad \{\neg \text{happy}(\text{Joe}), \text{happy(sally)}\}

Instantiate resolution rule with our clauses:

\{\text{happy}(x), \text{sad}(x)\} \quad \{\neg \text{happy}(\text{Joe}), \text{happy(sally)}\}

\{\text{sad}(x), \text{happy(sally)}\}[x \mapsto \text{Joe}][\text{sad}(\text{Joe}), \text{happy(sally)}]

Same conclusion as before!

Incompleteness

The inference rule for resolution so far is sound, but not complete: there are valid deductions it cannot derive.

Consider the following clauses:

\begin{align*}
\text{Clause 1:} & \quad \{p(x), p(y)\} \\
\text{Clause 2:} & \quad \{\neg p(a), \neg p(b)\}
\end{align*}

What does the first clause say?

Simpler way of saying the same thing:

Clearly contradicts the second clause!

So, we should derive the empty clause, i.e., \textit{contradiction}

Intuition about First-Order Resolution, summary

\begin{itemize}
\item Just like propositional resolution, first-order resolution corresponds to a simple case analysis
\item But it is more involved due to (implicit) universal quantifiers
\item In particular, to perform deduction, we might need to instantiate universal quantifier with something specific like \textit{Joe}
\item The use of unifiers in resolution corresponds to \textit{instantiation} of universally quantifiers
\item Quantifier instantiation is demand-driven; we only unify when it is possible to perform deduction
\end{itemize}

Why Most General Unifiers?

Clausess: \{\text{happy}(x), \text{sad}(x)\} \quad \{\neg \text{sad}(y)\}

Now, suppose we use a less general unifier, e.g. 

\begin{align*}
[x \mapsto \text{Joe}, y \mapsto \text{Joe}]
\end{align*}

Resolvent:

Since "Everyone is happy" implies "Joe is happy", former deduction is much better!

Using most general unifiers ensures our deductions are as general as possible

Otherwise, we might generate many useless deductions or miss important ones

Incompleteness Example

What can we deduce using resolution from these clauses?

\begin{align*}
\text{Clause 1:} & \quad \{p(x), p(y)\} \\
\text{Clause 2:} & \quad \{\neg p(a), \neg p(b)\}
\end{align*}

Using mgu for \{p(x)\} and \{p(a)\},

Using mgu for \{p(x)\} and \{p(b)\},

Using mgu for \{p(y)\}, \{p(a)\},

Using mgu for \{p(y)\}, \{p(b)\},

More deductions possible using new clauses, but redundant

\textit{Conclusion}: Using inference rule for resolution alone, we cannot derive the empty clause
Solution: Factoring

- To ensure we can deduce all valid facts, we need another inference rule for factoring.

Factorization:

\[ \{ A, B, C_1, \ldots, C_k \} \rightarrow \{ A, C_1, \ldots, C_k \} \sigma \quad (\sigma = \text{mgu}(A, B)) \]

- Soundness of factorization: For any clause \( C \) and any substitution \( \sigma \), \( C\sigma \) is always a valid deduction

- Why?

Thus, \( \{ A, B, C_1, \ldots, C_k \} \sigma \) is a valid deduction

- But why can we omit \( B \) in the conclusion?

Revisiting the Example

- Consider again the problematic example:

**Clause 1:** \{\( p(x) \), \( p(y) \)\}

**Clause 2:** \{\( \neg p(a) \), \( \neg p(b) \)\}

- Use factoring on first clause

- Mgu for \( p(x) \) and \( p(y) \):

- Result of factoring:

- Finally, do resolution between clause 2 and 3.

Resolution with Implicit Factoring

- Can formulate resolution and factoring as single inference rule.

Resolution with Implicit Factorization:

\[ \{ A_1, \ldots, A_n, B_1, \ldots, B_k \} \rightarrow \{ \neg C, D_1, \ldots, D_k \} \rightarrow \{ B_1, \ldots, B_k, D_1, \ldots, D_k \} \sigma \quad (\sigma = \text{mgu}(A_1, \ldots, A_n, C)) \]

- From now on, by "resolution", we mean resolution with implicit factorization

Resolution with Implicit Factoring Example

- Consider the example we looked at before:

\[
\begin{align*}
\{ p(x), p(y) \} & \rightarrow \{ \neg p(a), \neg p(b) \} \\
\{ \neg p(b) \} & \rightarrow \{} 
\end{align*}
\]

\( (\sigma = \text{mgu}(p(x), p(y), p(a))) \)

- Now, apply resolution with implicit factoring one more time:

\[
\begin{align*}
\{ p(x), p(y) \} & \rightarrow \{ \neg p(b) \} \\
\{ \neg p(b) \} & \rightarrow \{} 
\end{align*}
\]

\( (\sigma = \text{mgu}(p(x), p(y), p(b))) \)