Review

- Last lecture: unification
- Unifier: a substitution that makes two different expressions syntactically identical
- Most general unifier: A unifier \( \sigma \) is an mgu if for any other unifier \( \sigma' \), there exists a substitution \( \delta \) such that \( \sigma \delta = \sigma' \)
- Example: What is the mgu of \( p(x, f(x)) \) and \( p(a, y) \)?
- This lecture: Clausal form and first-order resolution

Clausal Form

- A formula in FOL in said to be in clausal form it 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

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 \( U, I, \{x \mapsto o\} \models \phi \)
- But this is the same as saying \( \phi \) is satisfiable iff \( U, I \models \exists x. \phi \) for some \( U, I \)
- 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 \( \phi \)
### Prenex Normal Form

- A formula is in **prenex normal form (PNF)** if all of its quantifiers appear at the beginning of formula:
  \[ Qx_1, \ldots, Qx_n, F(x_1, \ldots, x_n) \]
  where \( F \) is quantifier-free and \( Q \in \{ \forall, \exists \} \)
- Is \( \forall x.\exists y.(p(x,y) \rightarrow q(x)) \) in PNF? **Yes**
- What about \( \forall x.((\exists y.p(x,y)) \rightarrow q(x)) \) in PNF? **No**

### Conversion to PNF Example

- Convert formula to PNF:
  \[
  \forall x. (\neg(\exists y.(p(x,y) \land p(x,z))) \lor \exists y.p(x,y))
  \]
  1. Convert to NNF:
     \[
     \forall x. ((\forall y.\neg(p(x,y) \lor p(x,z))) \lor \exists y.p(x,y))
     \]
  2. Rename conflicting variables:
     \[
     \forall x. ((\forall y.\neg(p(x,y) \lor p(x,z))) \lor \exists w.p(x,w))
     \]
  3. Move quantifiers to front:
     \[
     \forall x. \exists w. \forall y. ((\neg(p(x,y) \lor p(x,z))) \lor p(x,w))
     \]

### 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:
    \[ \neg \forall x. \phi \equiv \exists x. \neg \phi \]
    \[ \neg \exists x. \phi \equiv \forall x. \neg \phi \]
- **Step 2b:** Rename quantified variables as necessary so no two quantified variables have the same name.
- **Step 2c:** Move quantifiers to front of formula

### Step 3: Skolemization

- After converting formula to PNF, we want to remove all existential quantifiers
- Skolemization produces **equisatisfiable** formula without existential quantifiers
- Suppose an existentially quantified variable \( y \) appears in the scope of universal quantifiers \( x_1, \ldots, x_k \)
  - **Skolemization:** replaces \( y \) with function term: \( f(x_1, \ldots, x_k) \) where \( f \) is a **fresh** function symbol
  - This new function \( f \) called **Skolem function**
  - What happens if \( y \) is not in scope of any quantifiers? introduce fresh constant object

### 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? "Everyone hates someone"
  - Now, let’s replace \( y \) with object constant \( c \): \( \forall x. \text{hates}(x, c) \)
  - What does this formula say? "Everyone hates the same person"
  - 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 Clausal Form Example

- Convert formula to clausal form:
  $\forall y.(p(y) \land \neg(\forall z.(r(z) \rightarrow q(y, z, w)))$
- Step 1: No free variables:
  $\exists w.\forall y.(p(y) \land \neg(\forall z.(r(z) \rightarrow q(y, z, w)))$
- Step 2a: Convert to NNF (necessary for PNF):
  $\exists w.\forall y.(p(y) \land \neg(\forall z.(r(z) \lor q(y, z, w))))$ remove $\rightarrow$
  $\exists w.\forall y.(p(y) \land (\exists z.(r(z) \land r(y)))$ push negations
- Step 2b: Move quantifiers out (necessary for PNF):
  $\exists w.\forall y.\exists z.(p(y) \land (r(z) \land q(y, z, w)))$

Conversion to Clausal Form Example, continued

- In Skolem Normal Form:
  $\forall y.(p(y) \land (r(f(y)) \land r(y, f(y), c)))$
- Step 4: Convert inner formula to CNF (already in CNF)
- Step 5: Drop universal quantifiers:
  $(p(y) \land (r(f(y)) \land r(y, f(y), c)))$
- Step 6: Finally, write formula as a set of clauses
  $\{p(y)\}, \{r(f(y))\}, \{y, f(y), c\}$
- This formula is now in clausal form

Example II

- Convert formula to clausal form:
  $\exists w.\forall z.((\exists z. q(w, z)) \rightarrow \exists y.(\neg p(x, y) \land r(y)))$
- Step 1,2a: No free variables, convert to NNF:
  $\exists w.\forall z.((\exists z. q(w, z)) \lor \exists y.(\neg p(x, y) \land r(y)))$ remove $\rightarrow$
  $\exists w.\forall z.((\exists z. \neg q(w, z)) \lor \exists y.(\neg p(x, y) \land r(y)))$ push negations
- Step 2b: Move quantifiers out (necessary for PNF):
  $\exists w.\forall z.\exists y.((\neg q(w, z)) \lor (\neg p(x, y) \land r(y)))$
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?

\( \forall x_1, \ldots, x_n. (l_1 \lor l_2 \ldots \lor l_k) \land \ldots \land (l'_1 \lor l'_2 \ldots \lor l'_n) \)

Recall: Universal quantifiers distribute over conjunctions:

\( \forall \vec{x}, F_1 \land F_2 \iff \forall \vec{x} F_1 \land \forall \vec{x} F_2 \)

Thus above formula is equivalent to:

\( \forall x_1, \ldots, x_n. (l_1 \lor l_2 \ldots \lor l_k) \ldots \land \forall x_1, \ldots, x_n. (l'_1 \lor l'_2 \ldots \lor l'_n) \)

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 \( \{ \neg p(x, b) \} \), we assume they are renamed as \( \{ p(a, x) \} \) and \( \{ \neg p(z, b) \} \)

In Prenex Normal Form:

\[ \exists w. \exists x. \exists y. ((\neg q(w, z)) \lor (\neg p(x, y) \land r(y))) \]

Step 3a: Now, skolemize \( w \):

\[ \forall x. \exists y. ((\neg q(c, z)) \lor (\neg p(x, f(x)) \land r(f(x)))) \]

Step 4: Convert inner formula to CNF

\[ \{ \neg q(c, z), \neg p(x, f(x)) \} \]

\[ \{ (\neg p(x, f(x)), r(y)) \} \]

Example II, cont

In Skolem Normal Form:

\[ \forall x. \forall z. ((\neg q(c, z)) \lor (\neg p(x, f(x)) \land r(y))) \]

Step 4: Convert inner formula to CNF

\[ \forall x. \forall z. ((\neg q(c, z)) \lor (\neg p(x, f(x))) \land (\neg p(x, f(x)) \lor r(y))) \]

Step 5: Drop universal quantifiers:

\[ (\neg q(c, z) \lor \neg p(x, f(x))) \land (\neg p(x, f(x)) \lor r(y)) \]

Step 6: Finally, write formula as a set of clauses

\[ \{ \neg q(c, z), \neg p(x, f(x)) \} \]

\[ \{ (\neg p(x, f(x)), r(y)) \} \]

First Order Resolution

To apply first-order resolution, convert formula to clausal form

Rename variables to ensure each clause contains different variables

Resolution:

\[
\frac{\{ A, B_1, \ldots, B_n \} \quad \{ \neg C, D_1, \ldots, D_n \}}{\{ B_1, \ldots, B_k, D_1, \ldots, D_n \} \sigma} \quad (\sigma = \text{mgu}(A, C))
\]
Why Most General Unifiers?

- Why do we need most general unifiers, not just any unifier?
- Because want to keep our deductions as general as possible!
- Example: Consider clauses: \{happy(x), sad(x)\} \{¬sad(y)\}
- Most general unifier: \[y \mapsto x\]
- Resolvent: \{happy(x)\}
- What does this mean in English? “Everyone is happy”

Intuition about First-Order Resolution

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

Example

Resolution:
\[
\left\{ A, B_1, \ldots, B_k \right\} \left\{ ¬C, D_1, \ldots, D_k \right\} \left( \sigma = \mbox{mgu}(A, C) \right)
\]

- What is the result of performing resolution on the following clauses?

  Clause 1 : \{p(a, y), r(g(y))\}
  Clause 2 : \{¬p(x, f(x)), q(g(x))\}

- Mgu for \(p(a, y)\) and \(p(x, f(x))\): \([x \mapsto a, y \mapsto f(a)]\)
- Resolvent: \{r(g(f(a)), q(g(a))}\}

Intuition about First-Order Resolution, cont.

- Instantiate resolution rule with our clauses:

  \[
  \left\{ happy(x), sad(x) \right\} \left\{ ¬happy(joe), happy(sally) \right\}
  \]

- Same conclusion as before!

Intuition about First-Order Resolution, summary

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

Why Most General Unifiers?

- Now, suppose we use a less general unifier, e.g. \([x \mapsto joe, y \mapsto joe]\)
- Resolvent: \(happy(joe)\)
- 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

- The inference rule for resolution so far is sound, but not complete: there are valid deductions it cannot derive.
- Consider the following clauses:
  
  Clause 1: \{p(x), p(y)\}
  
  Clause 2: \{-p(a), -p(b)\}

- What does the first clause say? \(\forall x, y. (p(x) \lor p(y))\)
- Simpler way of saying the same thing: \(\forall x. p(x)\)
- Clearly contradicts the second clause!
- So, we should derive the empty clause, i.e., contradiction

---

Incompleteness Example

- What can we deduce using resolution from these clauses?
  
  Clause 1: \{p(x), p(y)\}
  
  Clause 2: \{-p(a), -p(b)\}

- Use mgu for \(p(x)\) and \(p(a)\), Clause 3: \{p(y), -p(b)\}
- Use mgu for \(p(y)\) and \(p(b)\), Clause 4: \{p(y), -p(a)\}

- More deductions possible using new clauses, but redundant
- 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:
  
  \[ \{A, B, C_1, \ldots, C_k\} \leadsto A, C_1, \ldots, C_k \sigma \]  
  
  \(\sigma = \text{mgu}(A, B)\)

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

- Why? Because can replace universally quantified variable with any term
- Thus, \(\{A, B, C_1, \ldots, C_k\}\sigma\) is a valid deduction

- But why can we omit \(B\) in the conclusion? Because \(A\sigma\) and \(B\sigma\) are identical, so \(B\) redundant

---

Resolution with Implicit Factoring

- Can formulate resolution and factoring as single inference rule.

- Resolution with Implicit Factoring:
  
  \[ \{A_1, A_2, \ldots, A_n, B_1, \ldots, B_k\} \]
  
  \[ \{\neg C, D_1, \ldots, D_k\} \]
  
  \[ \{B_1, \ldots, B_k, D_1, \ldots, D_k\}\sigma \]  
  
  \(\sigma = \text{mgu}(A_1, \ldots, A_n, C)\)

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

---

Resolution with Implicit Factoring Example

- Consider the example we looked at before:
  
  \[ \{p(x), p(y)\} \]
  
  \[ \{-p(a), -p(b)\} \]

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

- Now, apply resolution with implicit factoring one more time:
  
  \[ \{p(x), p(y)\} \]
  
  \[ \{-p(b)\} \]

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

- A clause $C$ is derivable from a set of clauses $\Delta$ if there is a sequence of clauses $\Psi_1, \ldots, \Psi_k$ terminating in $C$ such that:
  1. $\Psi_1 \in \Delta$, or
  2. $\Psi_i$ is resolvent of some $\Psi_j$ and $\Psi_k$ such that $j < i$ and $k < i$.

- Example: Consider clauses
  
  $$\Delta = \{happy(x), sad(x)\}, \{-sad(y)\}$$

  Here, $\{happy(x)\}$ is derivable from $\Delta$.

- If a clause $C$ is derivable from $\Delta$, we write $\Delta \vdash C$.

Resolution Refutation

- The derivation of the empty clause from a set of clauses $\Delta$ is called resolution refutation of $\Delta$.

  Consider set of clauses $\Delta$:
  
  $$\{happy(x), sad(x)\}, \{-sad(y)\}, \{-happy(mother(joe))\}$$

  Resolution refutation of $\Delta$:
  
  $$\begin{array}{c}
  \{happy(x), sad(x)\} \\
  \{-sad(y)\} \\
  \{-happy(mother(joe))\} \\
  \end{array}$$

  Resolution refutation of $\Delta$:
  
  $$\begin{array}{c}
  \{happy(x), sad(x)\} \\
  \{-sad(y)\} \\
  \{-happy(mother(joe))\} \\
  \end{array}$$

  The derivation of the empty clause from $\Delta$ is indeed unsatisfiable.

  Corollary: If $F$ is unsatisfiable, then there exists a resolution refutation of $F$ using only resolution with factorization.

  This is called the refutational completeness of resolution.

Refutational Soundness and Completeness

- Theorem: Resolution is sound, i.e., if $\Delta \vdash C$, then $\Delta \models C$.

  Corollary: If there is a resolution refutation of $\Delta$, $\Delta$ is indeed unsatisfiable.

  In other words, we cannot conclude a satisfiable formula is unsatisfiable using resolution.

  Resolution with implicit factorization is also complete, i.e., if $\Delta \models C$, then $\Delta \vdash C$.

  Corollary: If $F$ is unsatisfiable, then there exists a resolution refutation of $F$ using only resolution with factorization.

  This is called the refutational completeness of resolution.

Validity Proofs using Resolution

- How to prove validity FOL formula using resolution?

  Use duality of validity and unsatisfiability:

  $$F \text{ is valid iff } \neg F \text{ is unsatisfiable}$$

- We will use resolution to show $\neg F$ is unsatisfiable.

- First, convert $\neg F$ to clausal form $C$.

  If there is a resolution refutation of $C$, then, by soundness, $F$ is valid.

Example

- Everybody loves somebody. Everybody loves a lover. Prove that everybody loves everybody.

  First sentence in FOL: $\forall x. \exists y. \text{loves}(x, y)$

  Second sentence in FOL:

  $\forall u. \forall w. ((\exists v. \text{loves}(u, v)) \rightarrow \text{loves}(w, u))$

  Goal in FOL: $\forall z. \forall t. \text{loves}(z, t)$

  Thus, want to prove validity of:

  $$(\forall z. \exists y. \text{loves}(x, y) \land \forall u. \forall w. ((\exists v. \text{loves}(u, v)) \rightarrow \text{loves}(w, u))) \rightarrow \forall z. \forall t. \text{loves}(z, t)$$

Example, cont.

- Want to prove negation unsatisfiable:

  $$\neg((\forall z. \exists y. \text{loves}(x, y) \land \forall u. \forall w. ((\exists v. \text{loves}(u, v)) \rightarrow \text{loves}(w, u))) \rightarrow \forall z. \forall t. \text{loves}(z, t))$$

  Convert to PNF: in NNF, quantifiers in front.

  Remove inner implication:

  $$\neg((\forall z. \exists y. \text{loves}(x, y) \land \forall u. \forall w. ((\exists v. \text{loves}(u, v)) \lor \text{loves}(w, u))) \rightarrow \forall z. \forall t. \text{loves}(z, t))$$

  Remove outer implication:

  $$\neg((\forall z. \exists y. \text{loves}(x, y) \land \forall u. \forall w. ((\neg(\exists v. \text{loves}(u, v)) \lor \text{loves}(w, u)) \lor \forall z. \forall t. \text{loves}(z, t)))$$
Example, cont.

\(\neg(\forall x.\exists y.\text{loves}(x, y) \land \forall u.\forall w.((\neg(\exists v.\text{loves}(u,v)) \lor \text{loves}(w,u)) \lor \forall z.\forall t.\text{loves}(z,t)))\)

- Push innermost negation in:
  \(\neg(\forall x.\exists y.\text{loves}(x, y) \land \forall u.\forall w.\neg(\text{loves}(u,v) \lor \text{loves}(w,u)) \lor \forall z.\forall t.\text{loves}(z,t)))\)

- Push outermost negation in:
  \(\neg
  (\forall x.\exists y.\text{loves}(x, y) \land \forall u.\forall w.\neg(\exists v.\text{loves}(u,v)) \lor \text{loves}(w,u)) \lor \forall z.\forall t.\text{loves}(z,t))\)

Example, cont.

\((\forall x.\exists y.\text{loves}(x, y) \land \forall u.\forall w.\neg(\text{loves}(u,v) \lor \text{loves}(w,u)))\)

- Now, move quantifiers to front.
  \(\exists z.\forall t.\neg(\text{loves}(z,t))\)

- Next, skolemize existentially quantified variables:
  \(\forall u.\forall w.\forall z.\text{loves}(x, \text{lover}(x)) \land (\neg(\text{loves}(u,v) \lor \text{loves}(w,u)) \land \neg\text{loves}(z,t))\)

Example, cont.

- Finally, we can do resolution:
  \(\{
  \neg(\text{loves}(x, \text{lover}(x)))
  \}
  \{
  \neg\text{loves}(u,v), \neg\text{loves}(w,u)
  \}
  \{
  \neg\text{loves}(\text{joe}, \text{jane})
  \}\)

- Resolve first and second clauses. MGU:
  \([u \mapsto x, v \mapsto \text{lover}(x)]\)

- Resolvent: \(\{\text{loves}(w,x)\}\)

- Resolve new clause with third clause.

- Mguc: \([w \mapsto \text{joe}, z \mapsto \text{jane}]\)

- Resolvent: \(\{\}\)

- Thus, we have proven the formula valid.

Example II

- Use resolution to prove validity of formula:
  \(\neg(\exists y.\forall z.(p(z,y) \leftrightarrow \exists x.(p(z,x) \land p(x,z))))\)

- Convert negation to clausal form:
  \(\exists y.\forall z.(p(z,y) \leftrightarrow \exists x.(p(z,x) \land p(x,z)))\)

- To convert to CNF, get rid of \(\leftrightarrow\):
  \(\exists y.\forall z.(\neg p(z,y) \lor \exists x.(p(z,x) \land p(x,z))) \land (p(z,y) \lor \exists x.(p(z,x) \land p(x,z)))\)
Example II, cont.

\[
\forall z. \forall x. \neg p(z, a) \lor (\neg p(z, x) \lor \neg p(x, z)) \land 
\neg p(z, a) \lor (p(z, f(z)) \land p(f(z), z))
\]

- Push negations in:

\[
\forall y. \forall z. (\neg p(z, y) \lor \forall x. (\neg p(z, x) \lor \neg p(x, z)) \land 
p(z, y) \lor \exists w. (p(z, w) \land p(w, z))
\]

- Rename quantified variables:

\[
\forall y. \forall z. (\neg p(z, y) \lor \forall x. (\neg p(z, x) \lor \neg p(x, z)) \land 
p(z, y) \lor \exists w. (p(z, w) \land p(w, z))
\]

- In clausal form (with renamed variables):

\[
C_1 : \{ \neg p(z, a), \neg p(z, x), \neg p(x, z) \}
C_2 : \{ p(y, a), p(y, f(y)) \}
C_3 : \{ p(w, a), p(f(w), w) \}
C_4 : \{ p(a, f(a)) \}
\]

- Now, resolve \( C_1 \) and \( C_3 \) (using factoring).

- What is the MGU for \( p(z, a), p(z, x), p(x, z), p(w, a) \)?
  \[ x \mapsto a, z \mapsto a, w \mapsto a \]

- Resolvent: \( \{ p(f(a), a) \} \)

Example II, cont.

\[
\forall y. \forall z. (\neg p(z, y) \lor \forall x. (\neg p(z, x) \lor \neg p(x, z)) \land 
p(z, y) \lor \exists w. (p(z, w) \land p(w, z))
\]

- In PNF:

\[
\exists y. \forall z. (\neg p(z, y) \lor (\neg p(z, x) \lor \neg p(x, z)) \land 
p(z, x) \lor (p(z, w) \land p(w, z)))
\]

- Skolemize existentials:

\[
\exists y. \forall z. (\neg p(z, y) \lor (\neg p(z, x) \lor \neg p(x, z)) \land 
p(z, x) \lor (p(z, f(z)) \land p(f(z), z)))
\]

Example II, cont.

\[
\forall z. \forall x. (\neg p(z, a) \lor (\neg p(z, x) \lor \neg p(x, z)) \land 
p(z, a) \lor (p(z, f(z)) \land p(f(z), z))
\]

- Drop quantifiers and convert to CNF:

\[
(\neg p(z, a) \lor (\neg p(z, x) \lor \neg p(x, z)) \land 
p(z, a) \lor p(z, f(z)) \land 
p(z, a) \lor p(f(z), z))
\]

- In clausal form (with renamed variables):

\[
C_1 : \{ \neg p(z, a), \neg p(z, x), \neg p(x, z) \}
C_2 : \{ p(y, a), p(y, f(y)) \}
C_3 : \{ p(w, a), p(f(w), w) \}
C_4 : \{ p(a, f(a)) \}
\]

- What is the MGU for \( p(z, a), p(z, x), p(x, z), p(y, a) \)?
  \[ x \mapsto a, z \mapsto a, y \mapsto a \]

- Resolvent: \( \{ p(a, f(a)) \} \)
Example II, cont.

C1 : \{-\neg p(z,a), \neg p(z,x), \neg p(x,z)\}
C2 : \{p(y,a), p(y,f(y))\}
C3 : \{p(w,a), p(f(w),w))\}
C4 : \{p(a,f(a))\}
C5 : \{p(f(a),a)\}
C6 : \{\neg p(a,f(a))\}

- Finally, resolve C4 and C6.
- Resolvent: \{\}
- Thus, the original formula is valid.

Resolution and First-Order Theorem Provers

- Resolution (with factorization) forms the basis of most automated first-order theorem provers.
- However, to make relational refutation more efficient, there are typically two main improvements:
  - Ordered resolution
  - Removal of useless clauses

Motivation for Ordered Resolution

- **Motivation:** In ordinary resolution, search space is very large.
- At a given step in the proof, there may be many clauses between which we can perform resolution
- Example:
  
  \[
  C1 : \{p(a,y), \ldots\} \\
  C2 : \{\neg p(y,z), \ldots\} \\
  C3 : \{\neg p(w,a), \ldots\} \\
  C4 : \{p(a,v), \ldots\}
  \]
- At least four pairs of clauses that we can resolve
- So far, we were free to choose any pair of clauses; but some of these may be unnecessary

Ordered Resolution, cont.

- Doing ordered resolution can substantially reduce search space compared to ordinary resolution
- Furthermore, if we are careful about how we define an order on clauses, can still guarantee refutational completeness of ordered resolution
- If there is refutation of a formula using ordinary resolution, there is also a refutation of that formula using ordered resolution
- Theorem provers typically perform ordered resolution

Removal of Useless Clauses

- When we do resolution, we keep all resolvent clauses in addition to original clauses
- The number of clauses can become really large!
- Thus, we want to get rid of any resolvent clauses that are redundant or lead to useless deductions
- Theorem provers use strategies to eliminate some of the resolvents
Removal of Useless Clauses, cont

- **Identical Clause Elimination**: If two clauses are identical up to variable renaming, one of them is redundant

- **Example**: \{p(x)\} and \{p(y)\} are identical, thus one of them can be eliminated

- **Tautology Elimination**: If a clause contains \( p(\vec{x}) \) and \( \neg p(\vec{x}) \), it can be eliminated

- **Example**: \{q(a, y), p(x), \neg q(a, y)\} is a tautology

- **Pure Literal Elimination**: If a literal \( p(x) \) occurs in some clause, but \( \neg p(x) \) does not occur in any other clause, \( p(x) \) is called *pure literal*

- If a clause consists of a single pure literal, it can be eliminated

---

Summary

- First-order resolution basis of first-order theorem provers

- Theorem provers use ordered resolution and employ strategies to eliminate useless or redundant clauses

- In addition, theorem provers also typically provide native support for reasoning about equality

- This is last lecture on standard first-order logic

- **Next lecture**: First-order theories