Review

- What is a unifier?
- What is Prenex Normal Form?
- What is Skolem Normal Form?
- How do you convert formula to Clausal Normal Form?

Example, cont

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

- Step 3a: Now, skolemize \( w \):
  \[ \forall x. \forall y. ((\neg q(c, z) \lor \neg p(x, y) \land r(y))) \]

- Step 3b: Skolemize \( y \):
  \[ \forall x. \forall y. ((\neg q(c, z) \lor \neg p(x, f(x)) \land r(f(x)))) \]

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 \( x_1, \ldots, x_m \), what is the meaning of this clausal form as a proper FOL formula?

- \( \forall x_1, \ldots, x_m. (l_1 \lor l_2 \ldots \lor l_k) \land \ldots \land (l'_1 \lor l'_2 \ldots \lor l'_{m'}) \)

- Recall: Universal quantifiers distribute over conjuncts:
  \[ \forall x. F_1 \land F_2 \iff \forall x F_1 \land \forall x F_2 \]

- Thus above formula is equivalent to:
  \[ \forall x_1, \ldots, x_m. (l_1 \lor l_2 \ldots \lor l_k) \ldots \land \]

  \[ \forall x_1, \ldots, x_m. (l'_1 \lor l'_2 \ldots \lor l'_{m'}) \]

Example, cont

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

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

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

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

Example, cont

- Convert formula to clausal form:
  \[ \exists w. \forall x. ((\exists z. q(w, z)) \land \exists y. (\neg p(x, y) \land r(y))) \]

- Step 1, 2a: Push negations
  \[ \exists w. \forall x. ((\neg q(w, z)) \lor (\neg p(x, y) \land r(y))) \]

- Step 5: Drop universal quantifiers:

CS389L: Automated Logical Reasoning
Lecture 9: First-Order Resolution
İsıl Dillig
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 \{¬p(z, b)\}, we assume they are renamed as \{p(a, x)\} and \{¬p(z, b)\}

First Order Resolution

- To apply first-order resolution, convert formula to clausal form
- Rename variables to ensure each clause contains different variables
- Resolution:
\[
\frac{\{A, B_1, \ldots, B_k\} \quad \{¬C, D_1, \ldots, D_k\}}{\{B_1, \ldots, B_k, D_1, \ldots, D_k\}\sigma} \quad (\sigma = mgu(A, C))
\]
- What is the result of performing resolution on the following clauses?
  - Clause 1: \{p(a, y), r(g(y))\}
  - Clause 2: \{¬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: \{happy(x), sad(x)\} and \{¬happy(joe), happy(sally)\}
- The first clause says:
- This implies: happy(joe) \lor sad(joe)
- The second clause says:
- 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)
Intuition about First-Order Resolution, summary

- Just like propositional resolution, first-order resolution corresponds to a simple case analysis.
- But more involved due to universal quantifiers.
- To perform deduction, often need to instantiate universal quantifier with something specific like \textit{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?

- Why do we need most general unifiers, not just any unifier?
- Example: Consider clauses: \{happy(x), sad(x)\} \{-sad(y)\}
- Most general unifier:
- Resolvent:
- What does this mean in English?

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} & : \{p(x), p(y)\} \\
  \text{Clause 2} & : \{-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., contradiction.

Incompleteness Example

- What can we deduce using resolution from these clauses?
  
  \begin{align*}
  \text{Clause 1} & : \{p(x), p(y)\} \\
  \text{Clause 2} & : \{-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.
- 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:
  
  \[
  \frac{\{A, B, C_1, \ldots, C_k\}}{A, C_1, \ldots, C_k}\sigma \quad (\sigma = \text{mgu}(A, B))
  \]
- Soundness of factorization: For any clause \(C\) and any substitution \(\sigma\), \(C\sigma\) is always a valid deduction.
- Why?
Revisiting the Example
▶ Consider again the problematic example:

Clause 1: \( \{p(x), p(y)\} \)
Clause 2: \( \{-p(a), -p(b)\} \)

▶ Use factoring on first clause
▶ Mgu for \( p(x) \) and \( p(y) \):
▶ Result of factoring:
▶ Now, do resolution between clause 2 and 3.

Resolution with Implicit Factoring Example
▶ Consider the example we looked at before:

\[
\{p(x), p(y)\} \\
\{-p(a), -p(b)\} \\
\emptyset
\]

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

\[
\{p(x), p(y)\} \\
\{-p(b)\} \\
\emptyset
\]

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. \( \Psi_i \in \Delta \), or
   2. \( \Psi_i \) is resolvent of some \( \Psi_j \) and \( \Psi_k \) such that \( j < i \land k < i \)

Example: Consider clauses
\[
\Delta = \{\text{happy}(x), \text{sad}(x)\}, \{-\text{sad}(y)\}
\]
▶ Here, \( \{\text{happy}(x)\} \) is derivable from \( \Delta \)
▶ If a clause \( C \) is derivable from \( \Delta \), we write \( \Delta \vdash C \)

Resolution with Implicit Factoring
▶ Can formulate resolution and factoring as single inference rule.

Resolution with Implicit Factorization:
\[
\{A_1, \ldots, A_n, B_1, \ldots, B_k\} \\
\{-C, D_1, \ldots, D_k\} \\
\{B_1, \ldots, B_k, D_1, \ldots, D_k\} \sigma
\]

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

Resolution Refutation
▶ The derivation of the empty clause from a set of clauses \( \Delta \) is called resolution refutation of \( \Delta \)

▶ Consider set of clauses \( \Delta \):

\[
\{\text{happy}(x), \text{sad}(x)\} \\
\{-\text{sad}(y)\} \\
\{-\text{happy}(\text{mother}(joe))\}
\]

▶ Resolution refutation of \( \Delta \):

\[
\{\text{happy}(x), \text{sad}(x)\} \\
\{-\text{sad}(y)\} \\
\{-\text{happy}(\text{mother}(joe))\}
\]

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:
- Second sentence in FOL:
- Goal in FOL:

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

Example, cont.

- Want to prove negation unsatisfiable:

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

- Convert to PNF: in NNF, quantifiers in front
- Remove inner implication:

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

- Remove outer implication:

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

Example, cont.

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

- Eliminate double negation:

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

- Push negation on second line in:

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

- Now, move quantifiers to front. Restriction:

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

- Next, skolemize existentially quantified variables:

\[ \forall u.\forall w.\forall z.\exists x.\text{loves}(x,\text{lover}(z)) \land (\neg\text{loves}(v,u) \lor \text{loves}(w,u)) \land \neg\text{loves}(x,\text{jane}) \]
Example II, cont.

\[
\forall u, \forall w, \forall v, \forall x. \\
\text{loves}(x, \text{lover}(x)) \land (\neg\text{loves}(u, v) \lor \text{loves}(w, u)) \\
\land \neg\text{loves}(\text{joe}, \text{jane})
\]

- Now, drop quantifiers:

\[
\text{loves}(x, \text{lover}(x)) \land (\neg\text{loves}(u, v) \lor \text{loves}(w, u)) \\
\land \neg\text{loves}(\text{joe}, \text{jane})
\]

- Convert to CNF: already in CNF!

- In clausal form:

\[
\{\text{loves}(x, \text{lover}(x))\} \\
\{\neg\text{loves}(u, v), \text{loves}(w, u)\} \\
\{\neg\text{loves}(\text{joe}, \text{jane})\}
\]

Example II

- Use resolution to prove validity of formula:

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

- Convert negation to clausal form:

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

- To convert to NNF, get rid of \(\leftrightarrow\):

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

Example II, cont.

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

- Push negations in:

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

- Rename quantified variables:

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

Example II, cont.

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

- Drop quantifiers and convert to CNF:

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

- In clausal form (with renamed variables):

\[
C1: \{\neg p(z, a), \neg p(z, x), \neg p(x, z)\} \\
C2: \{y(a), p(y, f(y))\} \\
C3: \{p(w, a), p(f(w), w))\}
\]
Example II, cont.

\[
C_1 : \{\neg p(z, a), \neg p(z, x), \neg p(x, z)\}
\]

\[
C_2 : \{p(y, a), p(y, f(y))\}
\]

\[
C_3 : \{p(w, a), p(f(w), w)\}
\]

\[
C_4 : \{p(a, f(a))\}
\]

\[
C_5 : \{p(f(a), a)\}
\]

- Resolve \(C_1\) and \(C_5\) (using factoring).
- What is the MGU of \(p(z, a), p(z, x)\) and \(p(f(a), a)\)?
- Resolvent:

Example II, cont.

\[
C_1 : \{\neg p(z, a), \neg p(z, x), \neg p(x, z)\}
\]

\[
C_2 : \{p(y, a), p(y, f(y))\}
\]

\[
C_3 : \{p(w, a), p(f(w), w)\}
\]

\[
C_4 : \{p(a, f(a))\}
\]

- Resolve \(C_1\) and \(C_2\) using factoring.
- What is the MGU for \(p(z, a), p(z, x), p(x, z), p(y, a)\)?
- Resolvent:

Example II, cont.

\[
C_1 : \{\neg p(z, a), \neg p(z, x), \neg p(x, z)\}
\]

\[
C_2 : \{p(y, a), p(y, f(y))\}
\]

\[
C_3 : \{p(w, a), p(f(w), w)\}
\]

\[
C_4 : \{p(a, f(a))\}
\]

- Now, resolve \(C_1\) and \(C_3\) (using factoring).
- What is the MGU for \(p(z, a), p(z, x), p(x, z), p(w, a)\)?
- Resolvent:

Example II, cont.

\[
C_1 : \{\neg p(z, a), \neg p(z, x), \neg p(x, z)\}
\]

\[
C_2 : \{p(y, a), p(y, f(y))\}
\]

\[
C_3 : \{p(w, a), p(f(w), w)\}
\]

\[
C_4 : \{p(a, f(a))\}
\]

\[
C_5 : \{p(f(a), a)\}
\]

\[
C_6 : \{\neg p(a, f(a))\}
\]

- Finally, resolve \(C_4\) and \(C_6\).
- Resolvent: \{
- Thus, the original formula is valid.

Summary

- First-order theorem provers work by converting to clausal form and trying to find resolution refutation.
- But there are no termination guarantees – may diverge if formula is satisfiable.
- Next lecture: First-order theories