CS389L: Automated Logical Reasoning

Lecture 9: First-Order Resolution

İşil Dillig

Review

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

Clausal Normal Form Example

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

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
  \[ \forall x. \forall y. ((\neg q(c,z) \lor \neg p(x,f(x))) \land (\neg q(c,z) \lor r(f(x))) \]
- Step 5: Drop universal quantifiers:
  \[ (\neg q(c,z) \lor \neg p(x,f(x))) \land (\neg q(c,z) \lor r(f(x))) \]
- Step 6: Finally, write formula as a set of clauses
  \[ \{ (\neg q(c,z)), \neg p(x,f(x)) \} \]
  \[ \{ (\neg q(c,z)), r(f(x)) \} \]

A Word About Clausal Form

- Consider the clausal form \( \{ l_1, l_2, \ldots, l_k \}, \ldots, \{ l'_1, l'_2, \ldots, l'_{k'} \} \)
- Assuming clauses contain variables \( x_1, \ldots, x_n \), what is the meaning of this clausal form as a proper FOL formula?
- \( \forall x_1, \ldots, x_n. (l_1 \lor l_2 \ldots \lor l_k) \land \ldots \land (l'_1 \lor l'_2 \ldots \lor l'_{k'}) \)
- 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:
  \[ \forall x_1, \ldots, x_n. (l_1 \lor l_2 \ldots \lor l_k) \ldots \land \]
  \[ \forall x_1, \ldots, x_n. (l'_1 \lor l'_2 \ldots \lor l'_{k'}) \]

| RAW TEXT START | RAW TEXT END |
A Word About Clausal Form, cont.

\[ \forall x_1, \ldots, x_n . \left( l_1 \lor l_2 \ldots \lor l_k \right) \ldots \land \forall y_1, \ldots, y_n . \left( l'_1 \lor l'_2 \ldots \lor l'_n \right) \]

- **Recall:** If we rename quantified variables, the resulting formula is equivalent to original one

\[ \forall x . F \iff \forall y. F[y/x] \]

- Hence, the above formula is equivalent to:

\[ \forall x_1, \ldots, x_n . \left( l_1 \lor l_2 \ldots \lor l_k \right) \ldots \land \forall y_1, \ldots, y_n . \left( l'_1 \lor l'_2 \ldots \lor l'_n \right)[y/x] \]

- Thus, if two different clauses \( C_1 \) and \( C_2 \) contain same variable \( x \), we can rename \( x \) to some other \( x' \) in one of \( C_1 \) or \( C_2 \)

First Order Resolution

- To apply first-order resolution, convert formula to clausal form
- Rename variables to ensure each clause contains different variables
- **Resolution:**

\[
\frac{\{ A, B_1, \ldots, B_k \} \quad \{-C, D_1, \ldots, D_k \}}{\{ B_1, \ldots, B_k, D_1, \ldots, D_n \}, \sigma} \quad (\sigma = \text{mgu}(A, C))
\]

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) ∨ 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) ∨ sad(joe)

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

Example

Resolution:

\[
\frac{\{ A, B_1, \ldots, B_k \} \quad \{-C, D_1, \ldots, D_k \}}{\{ B_1, \ldots, B_k, D_1, \ldots, D_n \}, \sigma} \quad (\sigma = \text{mgu}(A, C))
\]

- 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, cont.

\[
\frac{\{ A, B_1, \ldots, B_k \} \quad \{-C, D_1, \ldots, D_k \}}{\{ B_1, \ldots, B_k, D_1, \ldots, D_n \}, \sigma} \quad (\sigma = \text{mgu}(A, C))
\]

- What happens if we apply resolution to \{happy(x), sad(x)\} and \{¬happy(joe), happy(sally)\}?

- Instantiate resolution rule with our clauses:

\[
\{ \text{happy}(x), \text{sad}(x) \} \quad \{\neg \text{happy}(joe), \text{happy}(sally)\}
\]

- Same conclusion as before!
Incompleteness Example

- What can we deduce using resolution from these clauses?

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

- 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

Why Most General Unifiers?

- Why do we need most general unifiers, not just any unifier?
- Example: Consider clauses: \( \{\text{happy}(x), \text{sad}(x)\} \), \( \{\neg \text{sad}(y)\} \)
- 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:
  - Clause 1: \( \{p(x), p(y)\} \)
  - Clause 2: \( \{\neg p(a), \neg p(b)\} \)

  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

Solution: Factoring

- To ensure we can deduce all valid facts, we need another inference rule for factoring.
  - Factorization:
    \[
    \begin{align*}
    \{A, B, C_1, \ldots, C_k\} & \quad (\sigma = \text{mgu}(A, B)) \\
    \{A, C_1, \ldots, C_k\} & \quad \sigma
    \end{align*}
    \]
  - 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

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 \( \text{joe} \)
- 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: \( \{\text{happy}(x), \text{sad}(x)\} \), \( \{\neg \text{sad}(y)\} \)
  - Most general unifier:
  - Resolvent:
  - What does this mean in English?
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:
  - Now, do resolution between clause 2 and 3.

Resolution with Implicit Factoring

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

  \[ \{A_1, \ldots, A_n, B_1, \ldots, B_k\} \]

  \[
  \{\neg C, D_1, \ldots, D_k\}
  \]

  \[
  \{B_1, \ldots, B_k, D_1, \ldots, D_k\} \sigma 
  \]

  \( (\sigma = \text{mgu}(A_1, \ldots, A_n, C)) \)

- From now on, by “resolution”, we mean resolution with implicit 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 \(\Delta\):
    
    \[
    \{\text{happy}(x), \text{sad}(x)\}
    \]
    
    \[
    \{\neg \text{sad}(y)\}
    \]
    
    \[
    \{\neg \text{happy}(\text{mother}(joe))\}
    \]

  - Resolution refutation of \(\Delta\):

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

    \[
    \{\text{happy}(x)\} \quad \{\text{happy}(x)\}
    \]

    \[
    \{\text{happy}(x)\}
    \]

    \[
    \{\text{happy}(x)\}
    \]

    \[
    \{\text{happy}(x)\}
    \]

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:
- Thus, want to prove validity of:

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

### Example, cont.

- Want to prove negation unsatisfiable:

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

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

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

- Remove outer implication:

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

### Example, cont.

- Push innermost negation in:

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

- Push outermost negation in:

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

### Example, cont.

- Now, move quantifiers to front. Restriction:

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

- Next, skolemize existentially quantified variables:

\[ (\forall u.\forall w.\forall v. \neg(\text{loves}(u,v) \lor \text{loves}(w,u))) \land \neg(\text{loves}(z,t)) \land \text{loves}(z,\text{lovet}(t)) \land \neg(\text{loves}(z,t) \lor \text{loves}(w,u)) \land \neg(\text{loves}(\text{joe},\text{jane})) \]
Example II, cont.

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

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

- Convert to CNF: already in CNF!

- In clausal form:
\[
\{\text{loves}(x, \text{lover}(x))\}
\]
\[
\{\neg \text{loves}(u, v), \text{loves}(w, u)\}
\]
\[
\{\neg \text{loves}\text{(joe, jane)}\}
\]

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

- To convert to NNF, get rid of \(\land\):
\[\exists y.\forall z.(\neg p(z, y) \lor \exists x.(p(z, x) \land p(x, z)))\land
(p(z, y) \lor \exists x.(p(z, x) \land p(x, z)))\]

Example II, cont.

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

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

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

Example II, cont.

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

- Drop quantifiers and convert to CNF:
\[\neg p(z, a) \lor \neg p(z, x) \lor \neg p(x, z)\land
p(z, a) \lor p(z, f(x))\land
p(z, a) \lor p(f(x), z)\]

- In clausal form (with renamed variables):
\[C1 : \{\neg p(z, a), \neg p(z, x), \neg p(x, z)\}\]
\[C2 : \{p(y, a), p(y, f(y))\}\]
\[C3 : \{p(w, a), p(f(w), w))\}\]
### Example II, cont.

\[
\begin{align*}
C_1 & : \{\neg p(z,a), \neg p(z,x), \neg p(x,z)\} \\
C_2 & : \{p(y,a), p(y,f(y))\} \\
C_3 & : \{p(w,a), p(f(w),w)\} \\
C_4 & : \{p(a,f(a))\} \\
C_5 & : \{f(a), a\}
\end{align*}
\]

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

### Resolution and First-Order Theorem Provers

- Resolution (with factorization) forms the basis of most automated first-order theorem provers.
- But with some modifications/improvements:
  - Ordered resolution
  - Removal of useless clauses (tautology, identical clause etc)
  - Built-in equality reasoning