Logic and Mechanized Reasoning

First-Order Resolution

Josh Clune

Carnegie Mellon University

Normal Forms

First-Order Resolution

Decision Procedures and Completeness

First-Order Resolution Completeness

Normal Forms

First-Order Resolution

Decision Procedures and Completeness

First-Order Resolution Completeness

Normal Forms

In propositional logic, we've seen various normal forms:

- Negation normal form
- Disjunctive normal form
- Conjunctive normal form

Analogs to these normal forms exist in first-order logic, and there are additional normal forms we can describe to impose constraints on quantifiers

Normal Forms

First-Order Normal Forms:

- Prenex normal form: All quantifiers must appear at the beginning of the formula and range over the whole formula

Normal Forms

First-Order Normal Forms:

- Prenex normal form: All quantifiers must appear at the beginning of the formula and range over the whole formula
- Skolem normal form: Prenex normal form with only universal quantifiers

Normal Forms

First-Order Normal Forms:

- Prenex normal form: All quantifiers must appear at the beginning of the formula and range over the whole formula
- Skolem normal form: Prenex normal form with only universal quantifiers
- Clause normal form: Skolem normal form where the formula is a conjunction of disjunctions of literals

In propositional logic, a literal is a variable or negated variable.
In first-order logic, a literal is a relation or negated relation.

Normal Forms Application

For propositional logic, we saw that the resolution rule requires that formulas first be transformed to conjunctive normal form.

For first-order logic, we will see that the resolution rule requires that formulas first be transformed to clause normal form.

In classical logic, any first-order formula f can be transformed to an equisatisfiable formula f' in clause normal form.

- So if our goal is to determine the validity or satisfiability of arbitrary first-order formulas, converting to clause normal form does not restrict us

Normal Forms

First-Order Resolution

Decision Procedures and Completeness

First-Order Resolution Completeness

First-Order Resolution Example 1

In propositional logic, if we have the clauses p and $\neg p \vee q$, we can resolve them to obtain the clause q.

Since resolution is an effective technique for SAT solving in propositional logic, we want to generalize it to first-order logic.

First-Order Resolution Example 1

In propositional logic, if we have the clauses p and $\neg p \vee q$, we can resolve them to obtain the clause q.

Since resolution is an effective technique for SAT solving in propositional logic, we want to generalize it to first-order logic.

Suppose we have two first-order clauses:

- $\forall x. \forall y. P(f(x), y)$
- $\forall w. \forall z. \neg P(w, g(z)) \vee Q(w, z)$

What might it look like to resolve these clauses?

First-Order Resolution Example

Clauses: $(\forall x. \forall y. P(f(x), y))$, $(\forall w. \forall z. \neg P(w, g(z)) \vee Q(w, z))$

First, we unify $P(f(x), y)$ and $P(w, g(z))$

First-Order Resolution Example

Clauses: $(\forall x. \forall y. P(f(x), y))$, $(\forall w. \forall z. \neg P(w, g(z)) \vee Q(w, z))$

First, we unify $P(f(x), y)$ and $P(w, g(z))$

- The most general unifier (*mgu*) is $\sigma = \{x \mapsto u, y \mapsto g(v), w \mapsto f(u), z \mapsto v\}$
- Applying σ the either term yields $P(f(u), g(v))$

First-Order Resolution Example

Clauses: $(\forall x. \forall y. P(f(x), y))$, $(\forall w. \forall z. \neg P(w, g(z)) \vee Q(w, z))$

First, we unify $P(f(x), y)$ and $P(w, g(z))$

- The most general unifier (*mgu*) is $\sigma = \{x \mapsto u, y \mapsto g(v), w \mapsto f(u), z \mapsto v\}$
- Applying σ the either term yields $P(f(u), g(v))$

Second, we instantiate according to the *mgu* to obtain:

- $P(f(u), g(v))$
- $\neg P(f(u), g(v)) \vee Q(f(u), v)$

First-Order Resolution Example

Clauses: $(\forall x.\forall y.P(f(x),y))$, $(\forall w.\forall z.\neg P(w,g(z)) \vee Q(w,z))$

First, we unify $P(f(x),y)$ and $P(w,g(z))$

- The most general unifier (*mgu*) is $\sigma = \{x \mapsto u, y \mapsto g(v), w \mapsto f(u), z \mapsto v\}$
- Applying σ the either term yields $P(f(u),g(v))$

Second, we instantiate according to the *mgu* to obtain:

- $P(f(u),g(v))$
- $\neg P(f(u),g(v)) \vee Q(f(u),v)$

Third, we resolve to obtain $Q(f(u),v)$

First-Order Resolution Example

Clauses: $(\forall x.\forall y.P(f(x),y))$, $(\forall w.\forall z.\neg P(w,g(z)) \vee Q(w,z))$

First, we unify $P(f(x),y)$ and $P(w,g(z))$

- The most general unifier (*mgu*) is $\sigma = \{x \mapsto u, y \mapsto g(v), w \mapsto f(u), z \mapsto v\}$
- Applying σ the either term yields $P(f(u),g(v))$

Second, we instantiate according to the *mgu* to obtain:

- $P(f(u),g(v))$
- $\neg P(f(u),g(v)) \vee Q(f(u),v)$

Third, we resolve to obtain $Q(f(u),v)$

Finally, we generalize to obtain the result: $\forall u.\forall v.Q(f(u),v)$

First-Order Resolution Definition

Definition (First-Order Resolution)

Let C_1 and C_2 be two first-order clauses such that:

- $C_1 = \forall x_1 \ldots \forall x_i . l \vee l_1 \vee \ldots \vee l_m$
- $C_2 = \forall y_1 \ldots \forall y_j . l' \vee l'_1 \vee \ldots \vee l'_n$
- l is a positive literal and l' is a negative literal
- There exists an *mgu* σ for the relations in l and l'
- σ maps all variables in C_1 and C_2 to terms containing only the variables z_1 through z_k

First-Order Resolution Definition

Definition (First-Order Resolution)

Let C_1 and C_2 be two first-order clauses such that:

- $C_1 = \forall x_1 \ldots \forall x_i . l \vee l_1 \vee \ldots \vee l_m$
- $C_2 = \forall y_1 \ldots \forall y_j . l' \vee l'_1 \vee \ldots \vee l'_n$
- l is a positive literal and l' is a negative literal
- There exists an *mgu* σ for the relations in l and l'
- σ maps all variables in C_1 and C_2 to terms containing only the variables z_1 through z_k

Then resolving C_1 and C_2 on literals l and l' yields
$\forall z_1 \ldots \forall z_k . \sigma(l_1 \vee \ldots l_m \vee l'_1 \vee \ldots \vee l'_n)$

First-Order Resolution Definition

A minor addendum to the previous definition:

It is possible that resolving C_1 and C_2 on literals l and l' yields a result in which there is some $i \in [1, m]$ such that $\sigma(l_i) = l$ (meaning after σ is applied to C_1, l appears multiple times)

If this happens, in addition to removing l and l' from the result, l_i should also be removed (likewise, any literals in C_2 that become l' after applying σ should also be removed)

Some presentations of first-order resolution separate this rule from resolution itself and call it factoring, other presentations include this elimination as part of the resolution rule itself

In section 14.1 of the textbook, there is an example (the barber paradox) that showcases why this is necessary

Resolving on Different Literals

In propositional logic, if it is ever possible to resolve a pair of clauses in two ways, the result will always be a tautology:

- Let $C_1 = p \vee q \vee \ldots$
- Let $C_2 = \bar{p} \vee \bar{q} \vee \ldots$
- Resolving C_1 and C_2 on p yields $q \vee \bar{q} \vee \ldots$
- Resolving C_1 and C_2 on q yields $p \vee \bar{p} \vee \ldots$
- Either way, the result of the resolution is a tautology and therefore useless

Resolving on Different Literals

In first-order logic, it may be possible to resolve a pair of clauses in multiple ways without the results being tautologies

Resolving on Different Literals

In first-order logic, it may be possible to resolve a pair of clauses in multiple ways without the results being tautologies

- Let $C_1 = \forall x.\forall y.P(f(x), y) \vee Q(x, f(y))$
- Let $C_2 = \forall w.\forall z.\neg P(w, g(z)) \vee \neg Q(g(w), z)$

Resolving on Different Literals

In first-order logic, it may be possible to resolve a pair of clauses in multiple ways without the results being tautologies

- Let $C_1 = \forall x.\forall y.P(f(x),y) \vee Q(x,f(y))$
- Let $C_2 = \forall w.\forall z.\neg P(w,g(z)) \vee \neg Q(g(w),z)$
- If we resolve on the first literal, we get *mgu* $\sigma = \{x \mapsto u, y \mapsto g(v), w \mapsto f(u), z \mapsto v\}$ yielding the result $\forall u.\forall v.Q(u,f(g(v))) \vee \neg Q(g(f(u)),v)$

Resolving on Different Literals

In first-order logic, it may be possible to resolve a pair of clauses in multiple ways without the results being tautologies

- Let $C_1 = \forall x.\forall y.P(f(x), y) \vee Q(x, f(y))$
- Let $C_2 = \forall w.\forall z.\neg P(w, g(z)) \vee \neg Q(g(w), z)$
- If we resolve on the first literal, we get *mgu* $\sigma = \{x \mapsto u, y \mapsto g(v), w \mapsto f(u), z \mapsto v\}$ yielding the result $\forall u.\forall v.Q(u, f(g(v))) \vee \neg Q(g(f(u)), v)$
- If we resolve on the second literal, we get *mgu* $\sigma = \{x \mapsto g(u), y \mapsto v, w \mapsto u, z \mapsto f(v)\}$ yielding the result $\forall u.\forall v.P(f(g(u)), v) \vee \neg P(u, g(f(v)))$

Resolving on Different Literals

In first-order logic, it may be possible to resolve a pair of clauses in multiple ways without the results being tautologies

- Let $C_1 = \forall x.\forall y.P(f(x), y) \lor Q(x, f(y))$
- Let $C_2 = \forall w.\forall z.\neg P(w, g(z)) \lor \neg Q(g(w), z)$
- If we resolve on the first literal, we get *mgu* $\sigma = \{x \mapsto u, y \mapsto g(v), w \mapsto f(u), z \mapsto v\}$ yielding the result $\forall u.\forall v.Q(u, f(g(v))) \lor \neg Q(g(f(u)), v)$
- If we resolve on the second literal, we get *mgu* $\sigma = \{x \mapsto g(u), y \mapsto v, w \mapsto u, z \mapsto f(v)\}$ yielding the result $\forall u.\forall v.P(f(g(u)), v) \lor \neg P(u, g(f(v)))$

Neither of these results are tautologies

Normal Forms

First-Order Resolution

Decision Procedures and Completeness

First-Order Resolution Completeness

Decision Procedures

Definition (Decision Procedure)

A decision procedure is an algorithm that takes in problems from some class of yes/no questions and determines the answer in finitely many steps

Decision Procedures

Definition (Decision Procedure)

A decision procedure is an algorithm that takes in problems from some class of yes/no questions and determines the answer in finitely many steps

Since propositional logic is decidable, the following (equivalent) questions all have decision procedures:

- Is P valid? ($\models P$)
- Is P provable? ($\vdash P$)
- Is $\neg P$ unsatisfiable?
- Is $\neg P$ refutable? ($\neg P \vdash \bot$)

Decision Procedures

Definition (Decision Procedure)

A decision procedure is an algorithm that takes in problems from some class of yes/no questions and determines the answer in finitely many steps

Since propositional logic is decidable, the following (equivalent) questions all have decision procedures:

- Is P valid? ($\models P$)
- Is P provable? ($\vdash P$)
- Is $\neg P$ unsatisfiable?
- Is $\neg P$ refutable? ($\neg P \vdash \bot$) $\longleftarrow$ You made this in HW 5

First-Order Logic is Undecidable

Since first-order logic is, in general, undecidable, none of the following (equivalent) questions have decision procedures:

- Is $\exists \overrightarrow{x}.A(\overrightarrow{x})$ valid? ($\models \exists \overrightarrow{x}.A(\overrightarrow{x})$)
- Is $\exists \overrightarrow{x}.A(\overrightarrow{x})$ provable? ($\vdash \exists \overrightarrow{x}.A(\overrightarrow{x})$)
- Is $\forall \overrightarrow{x}.\neg A(\overrightarrow{x})$ unsatisfiable?

First-Order Logic is Undecidable

Since first-order logic is, in general, undecidable, none of the following (equivalent) questions have decision procedures:

- Is $\exists \overrightarrow{x}.A(\overrightarrow{x})$ valid? ($\models \exists \overrightarrow{x}.A(\overrightarrow{x})$)
- Is $\exists \overrightarrow{x}.A(\overrightarrow{x})$ provable? ($\vdash \exists \overrightarrow{x}.A(\overrightarrow{x})$)
- Is $\forall \overrightarrow{x}.\neg A(\overrightarrow{x})$ unsatisfiable?

Definition (Refutation-Completeness)

A set of inference rules is refutation-complete if every unsatisfiable formula can be refuted using just those inferences. In other words, for every unsatisfiable formula A, refutation-completeness requires that $A \vdash \bot$

Consequence of Refutation-Completeness

Resolution is sound, meaning $A \vdash \bot$ entails that A is unsatisfiable. So if resolution is refutation-complete, then "Is A refutable?" is equivalent to "Is A unsatisfiable?"

Consequence of Refutation-Completeness

Resolution is sound, meaning $A \vdash \bot$ entails that A is unsatisfiable. So if resolution is refutation-complete, then "Is A refutable?" is equivalent to "Is A unsatisfiable?"

Note that this does NOT mean that there is a decision procedure for determining whether A is refutable

Search Nontermination

Example

- Consider the clause $\forall x. \neg P(x) \vee P(f(x))$.

Search Nontermination

Example

- Consider the clause $\forall x. \neg P(x) \vee P(f(x))$. If we resolve this clause with itself, we obtain $\forall y. \neg P(y) \vee P(f(f(y)))$. If we resolve this new clause with itself, we obtain $\forall z. \neg P(z) \vee P(f(f(f(f(z)))))$

Search Nontermination

Example

- Consider the clause $\forall x. \neg P(x) \vee P(f(x))$. If we resolve this clause with itself, we obtain $\forall y. \neg P(y) \vee P(f(f(y)))$. If we resolve this new clause with itself, we obtain $\forall z. \neg P(z) \vee P(f(f(f(f(z)))))$
- You can prove by induction that there are infinitely many clauses that you can generate via resolution in this manner

Search Nontermination

Example

- Consider the clause $\forall x. \neg P(x) \vee P(f(x))$. If we resolve this clause with itself, we obtain $\forall y. \neg P(y) \vee P(f(f(y)))$. If we resolve this new clause with itself, we obtain $\forall z. \neg P(z) \vee P(f(f(f(f(z)))))$
- You can prove by induction that there are infinitely many clauses that you can generate via resolution in this manner
- But the clause $\forall x. \neg P(x) \vee P(f(x))$ is satisfiable (just consider a model where no elements satisfy P)

Search Nontermination

Example

- Consider the clause $\forall x. \neg P(x) \lor P(f(x))$. If we resolve this clause with itself, we obtain $\forall y. \neg P(y) \lor P(f(f(y)))$. If we resolve this new clause with itself, we obtain $\forall z. \neg P(z) \lor P(f(f(f(f(z)))))$
- You can prove by induction that there are infinitely many clauses that you can generate via resolution in this manner
- But the clause $\forall x. \neg P(x) \lor P(f(x))$ is satisfiable (just consider a model where no elements satisfy P)

So first-order resolution is not a decision procedure. If a formula is satisfiable, proof search can either terminate or go on forever

Normal Forms

First-Order Resolution

Decision Procedures and Completeness

First-Order Resolution Completeness

First-Order Resolution Completeness

Theorem

Resolution is a refutation-complete calculus for first-order clause normal form formulas. So if C is an unsatisfiable first-order clause normal form formula, then $C \vdash \bot$

First-Order Resolution Completeness

Theorem

Resolution is a refutation-complete calculus for first-order clause normal form formulas. So if C is an unsatisfiable first-order clause normal form formula, then $C \vdash \bot$

Last week, we saw that any first-order formula can be transformed into an equisatisfiable Skolem normal form formula. And any Skolem normal form formula can be transformed into an equivalent clause normal form formula

First-Order Resolution Completeness

Theorem

Resolution is a refutation-complete calculus for first-order clause normal form formulas. So if C is an unsatisfiable first-order clause normal form formula, then $C \vdash \bot$

Last week, we saw that any first-order formula can be transformed into an equisatisfiable Skolem normal form formula. And any Skolem normal form formula can be transformed into an equivalent clause normal form formula

Corollary

Resolution + skolemization + clausification is a refutation-complete calculus for first-order logic

Herbrand's Theorem

Before the actual proof of first-order resolution's refutational completeness, we need to establish two important helper facts. The first is called Herbrand's Theorem

Herbrand's Theorem

Before the actual proof of first-order resolution's refutational completeness, we need to establish two important helper facts. The first is called Herbrand's Theorem

Theorem (Herbrand's Theorem)

Let $C = \forall x_1 \ldots \forall x_n . C_1 \wedge \ldots \wedge C_m$ be a clause normal form formula with constant and function symbols from Σ. Let Σ' be the set of closed terms that can be made from symbols in Σ.

C is unsatisfiable if and only if there is a finite set Γ where:

- *Each element in Γ is a clause $C_i[t_1/x_1, \ldots t_n/x_n]$ where $1 \leq i \leq m$ and $t_1 \ldots t_n \in \Sigma'$*
- *If each distinct literal in Γ is interpreted as a unique propositional variable, then Γ is unsatisfiable in propositional logic*

Lifting Lemma

The second helper fact we need is called the Lifting Lemma

Lifting Lemma

The second helper fact we need is called the Lifting Lemma

Lemma (Lifting Lemma)

Let $C = \forall x_1 \ldots \forall x_n . C_1 \wedge \ldots \wedge C_m$ *be a clause normal form formula with constant and function symbols from* Σ*. Let* Σ' *be the set of closed terms that can be made from symbols in* Σ*.*

Let Γ *be a set where each element is a clause* $C_i[t_1/x_1, \ldots t_n/x_n]$ *where* $1 \leq i \leq m$ *and* $t_1 \ldots t_n \in \Sigma'$

If each distinct literal in Γ *is interpreted as a unique propositional variable, then any propositional resolution refutation of* Γ *can be transformed into a first-order resolution refutation of* C

First-Order Resolution Completeness

Theorem

Resolution is a refutation-complete calculus for first-order clause normal form formulas. So if C is an unsatisfiable first-order clause normal form formula, then $C \vdash \bot$

First-Order Resolution Completeness Proof

Proof.
Let C be an unsatisfiable clause normal form formula with constant and function symbols from Σ. Since C is in clause normal form, C can be written $\forall x_1 \ldots \forall x_n . C_1 \wedge \ldots \wedge C_m$

First-Order Resolution Completeness Proof

Proof.

Let C be an unsatisfiable clause normal form formula with constant and function symbols from Σ. Since C is in clause normal form, C can be written $\forall x_1 \ldots \forall x_n . C_1 \wedge \ldots \wedge C_m$

Let Σ' be the set of closed terms that can be made from symbols in Σ. By Herbrand's Theorem, there is an unsatisfiable conjunction of clauses $C_i[t_1/x_1, \ldots t_n/x_n]$ where $1 \leq i \leq m$ and $t_1 \ldots t_n \in \Sigma'$. Call this conjunction Γ

First-Order Resolution Completeness Proof

Proof.

Let C be an unsatisfiable clause normal form formula with constant and function symbols from Σ. Since C is in clause normal form, C can be written $\forall x_1 \ldots \forall x_n . C_1 \wedge \ldots \wedge C_m$

Let Σ' be the set of closed terms that can be made from symbols in Σ. By Herbrand's Theorem, there is an unsatisfiable conjunction of clauses $C_i[t_1/x_1, \ldots t_n/x_n]$ where $1 \leq i \leq m$ and $t_1 \ldots t_n \in \Sigma'$. Call this conjunction Γ

Since resolution is complete for propositional logic and Γ is unsatisfiable, there exists a propositional resolution proof that $\Gamma \vdash \bot$. By the lifting lemma, this propositional proof can be transformed into a first-order resolution proof that $C \vdash \bot$

□