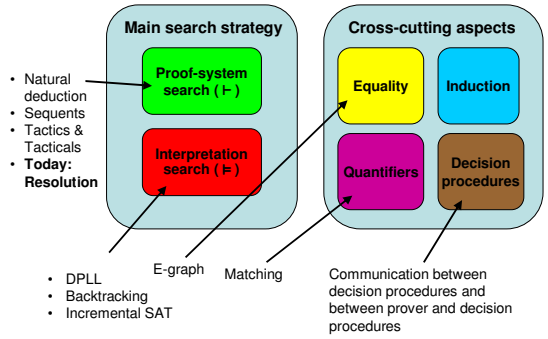
So far



Resolution

- Originally developed by Robinson in 1965
- Most proof systems at the time had aimed at human reasoning.
- They either:
  - Had many axioms (Hilbert style systems)
  - Or many inference rules (Natural deduction, Sequent calculus)
- Robinson wanted to explore the possibility of having a simple calculus
  - with few but powerful axioms and inference rules
  - not necessarily intuitive form a human perspective, but more amenable to automated reasoning.

Resolution

- Resolution calculus is far simpler than any of the other proof systems we have seen so far
  - There is only one axiom and one inference rule
- This simplicity led many researchers to embrace the logic early on
  - One inference rule means that we can put all of our intellectual effort into making this rule efficient
- Resolution is in fact still widely used today
  - Some of the most efficient fully automated theorem provers for first-order logic (E, Gandalf, Spass, Vampire) use variations of the resolution logic

Propositional resolution

$$\frac{\Gamma \vdash \alpha \Rightarrow l \qquad \Gamma \vdash l \Rightarrow \delta}{\Gamma \vdash \alpha \Rightarrow \delta}\ \text{RES}$$

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \neg l \vee \delta}{\Gamma \vdash \alpha \vee \delta \vee \beta \vee \delta}$$

Propositional resolution

$$\frac{\Gamma \vdash \alpha \Rightarrow l \qquad \Gamma \vdash l \Rightarrow \delta}{\Gamma \vdash \alpha \Rightarrow \delta}\ \text{RES}$$

Propositional resolution

$$\frac{\Gamma \vdash \alpha \Rightarrow l \qquad \Gamma \vdash l \Rightarrow \delta}{\Gamma \vdash \alpha \Rightarrow \delta}\ \text{RES}$$

$$\frac{\Gamma \vdash \neg \alpha \vee l \qquad \Gamma \vdash \neg l \vee \delta}{\Gamma \vdash \neg \alpha \vee \delta}\ \text{RES}$$ Expand ⇒

$$\frac{\Gamma \vdash \alpha \vee l \qquad \Gamma \vdash \neg l \vee \delta}{\Gamma \vdash \alpha \vee \delta}\ \text{RES}$$ Generalize

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$ Generalize even more

Propositional resolution

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$

resolvent

Propositional resolution

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$

resolvent

Which direction should we apply this rule in?

Propositional resolution

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$

$$\frac{}{\Gamma, \alpha, \Pi \vdash \alpha}\ \text{Assume}$$

Relation to other inference rules

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$

Relation to other inference rules

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$

- Recall Modus Ponens:

$$\frac{\Gamma \vdash l \qquad \Gamma \vdash l \Rightarrow \delta}{\Gamma \vdash \delta}\ \text{MP}$$

$$\frac{\Gamma \vdash l \qquad \Gamma \vdash \neg l \vee \delta}{\Gamma \vdash \delta}\ \text{MP} \qquad \text{Expand } \Rightarrow$$

MP is a special case of RES (with α, β and γ set false)

Relation to other inference rules

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$

Relation to other inference rules

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}\ \text{RES}$$

- Recall cut rule:

$$\frac{\Gamma \vdash \Delta, l \qquad l, \Pi \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda}\ \text{CUT}$$

$$\frac{\Gamma \vdash \Delta, l \qquad \Pi \vdash \Lambda, \neg l}{\Gamma, \Pi \vdash \Delta, \Lambda}\ \text{CUT}$$

$$\frac{\Gamma \vdash \Delta \vee l \qquad \Pi \vdash \Lambda \vee \neg l}{\Gamma, \Pi \vdash \Delta \vee \Lambda}$$

Resolution rule embodies the same idea as the cut rule from Sequents

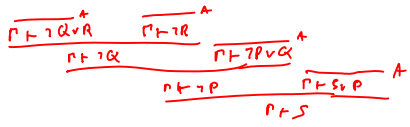
Throw l to the other side

"Expand" definition of sequents

Example Proof

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}$$

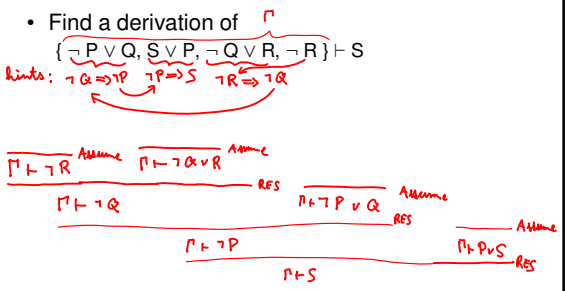
- Find a derivation of
  $\{ \neg P \vee Q, S \vee P, \neg Q \vee R, \neg R \} \vdash S$



Example Proof

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}$$

- Find a derivation of
  $\{ \neg P \vee Q, S \vee P, \neg Q \vee R, \neg R \} \vdash S$

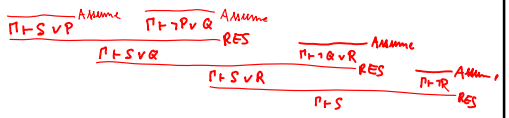


Is the proof unique?

Proof is not unique

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}$$

- Find a derivation of
  $\{ \neg P \vee Q, S \vee P, \neg Q \vee R, \neg R \} \vdash S$



Another example

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}$$

- Find a derivation of: $\{\} \vdash A \vee \neg A$

Another example

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}$$

- Find a derivation of: $\{\} \vdash A \vee \neg A$
- Can't find a derivation
  - There are no clauses to resolve!
- We made the calculus very simple (with only one axiom, and one inference rule)
  - but we also made it incomplete
- What should we do?

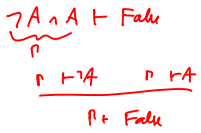
Try refutation

- We have seen refutation in the semantic domain
  - To show that a goal is valid, show that its negation is unsatisfiable
  - We can apply the idea in the proof domain
- In the context of the proof domain:
  - To show that a goal is valid, assume its negation, and derive *false*

Try refutation

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}$$

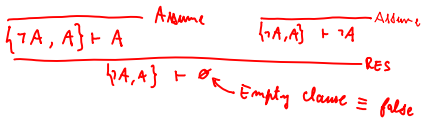
- Find a derivation of: {} ⊢ A ∨ ¬ A



Try refutation

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta}$$

- Find a derivation of: {} ⊢ A ∨ ¬ A



- Refutation worked in this case
- Q: Would it always work if the formula is valid?

Refutation

- A: Yes
- Although resolution is incomplete, it is *refutation* complete, which means that if the formula is valid, then adding its negation to the assumptions makes *false* provable
- More formally…
- Recall defn of completeness:
  - If $\Gamma \models \alpha$ then $\Gamma \vdash \alpha$

Refutation

- A: Yes
- Although resolution is incomplete, it is *refutation* complete, which means that if the formula is valid, then adding its negation to the assumptions makes *false* provable
- More formally…
- Recall defn of completeness:
  - If $\Gamma \models \alpha$ then $\Gamma \vdash \alpha$
- Defn of refutation complete:
  - If $\Gamma \models \alpha$ then $\Gamma, \neg \alpha \vdash \mathit{false}$

Forward refutation-based resolution search

- Keep a knowledge base, which is the set of formulas that have been inferred so far
- Given goal to prove:
  - Add negation of goal to the knowledge base
  - While *false* not in knowledge base:
    - Choose two formulas to resolve
    - Add resolvant formula to the knowledge base
    - If *false* is in the knowledge base, return VALID
- Is this search strategy complete?

Key issue: non-determinism

- Source of non-determinism: need to determine which clauses to resolve
- Two main approaches for handling these non-determinism
  - Simplification strategies
  - Ordering clauses

Simplification strategies

- Simplification strategies remove redundant clauses from the knowledge base
  - Reduces number of choices, but also makes the search more space efficient
- Example 1: remove a clause C if it contains a literal l that is not complementary with any other literal in the remaining clauses
  - Intuition?

Simplification strategies

- Simplification strategies remove redundant clauses from the knowledge base
  - Reduces number of choices, but also makes the search more space efficient
- Example 1: remove a clause C if it contains a literal l that is not complementary with any other literal in the remaining clauses
  - Intuition: l will never get resolved upon, and so resolvents derived from C (directly or indirectly) will therefore at least contain l, and thus cannot possibly be the empty clause

Simplification strategies

- Example 2: tautologies can be removed, where a tautology can be detected by checking if a clause contains both a literal and its negation
- Example 3: remove clauses that are implied by other clauses in the set
  - This is called subsumption
  - Various forms of it, depending on how the implication is tested, and when during the search the test is done

$$l_1 \vee l_2 \Rightarrow l_1 \vee l_2 \vee l_3$$

Ordering (clause selection) strategies

- A good clause selection strategy is critical for finding proofs efficiently
- Many ways to order clauses…
- Just the E theorem prover (which won various automated theorem proving competitions) implements over 60 predefined clause selection schemes

Ordering (clause selection) strategies

- Favor small clauses first, an instance of which is the idea of favoring one-literal clauses (unit resolution)
- Favor old clauses
  - Corresponds to a FIFO order and leads to a breadth first ordering
- Opposite strategy: always resolve the newest resolvent, which leads to a depth first search
  - Such strategies are called linear strategies because they create a linear chain of resolvents, each produced from the previous one
  - One such strategy, called SLD-resolution is at the core of Prolog

Example of SLD-resolution

Assume:
B_1 ①
B_2 ②
③ $B_1 \wedge B_2 \Rightarrow A$ (in prolog, written: A :- B_1, B_2) $\neg B_1 \vee \neg B_2 \vee A$
$B_3 \wedge B_4 \Rightarrow A$ (in prolog, written: A :- B_3, B_4) $\neg B_3 \vee \neg B_4 \vee A$

Query: A

$\neg A$
Resolve with ③ : $\neg B_1 \vee \neg B_2$
" " ② : $\neg B_1$
" " ① : False

Example of SLD-resolution

Assume:
B_1 ①
B_2 ②
$B_1 \wedge B_2 \Rightarrow A$ (in prolog, written: A :- B_1, B_2) $\neg B_1 \vee \neg B_2 \vee A$ ③
$B_3 \wedge B_4 \Rightarrow A$ (in prolog, written: A :- B_3, B_4) $\neg B_3 \vee \neg B_4 \vee A$ ④

Query: A

SLD-RES: Keep track of goal, and always try resulting goal.

Negate $\neg A$ ← goal
Resolve with ④: $\neg B_3 \vee \neg B_4$ ← goal 2
Resolve with ③: $\neg B_1 \vee \neg B_2$ ← goal
Resolve with ②: $\neg B_1$ ← goal
Resolve with ①: False

can't resolve goal anymore → go to most recent goal that can be resolved

Compare this to a backward search using MP.

Another issue: finding complementary lits

- Efficient data structures have been devised for efficiently finding complementary terms
- Graph based data structure of Kowalski
  - Complementary literals connected with graph edges
  - When a resolvent is added, use existing edges in the graph to add the appropriate new edges
- Indexing is another technique for efficiently determining which clauses to resolve
  - For example, answer queries such as: given a literal l, return all clauses that contain literals that unify with ¬l

First-order resolution

- So far, we've seen propositional resolution
- Now we'll take look at the first-order case
- However, before we do that, let's take a detour to see a new technique for handling universal quantifiers in axioms: unification

Universals with unification

- Given a universal axiom, our goal is to instantiate the universal with the right term t
- Key idea
  - Instead of determining immediately what term to instantiate the universal, introduce a new fresh variable v which stands for the term t
  - At later stages in the proof, the variable v is gradually constrained until either its value is known, in which case the proof succeeds, or the system determines that no value for v is appropriate, in which case the proof fails
  - The constraints arise from unification

Overview of unification

- Given two terms or formulas x and y, unifiy(x,y) returns a substitution θ such that θ(x) is syntactically the same as θ(y). If no such θ exists, the unification fails
- Examples
  - unify(P(f(x),y), P(z, g(s)) = $[z \rightarrow f(x), y \rightarrow g(s)]$
  - unify(P(x,y), P(y,x)) = $[x \rightarrow y]$
  - unify(P(x, g(y)), P(f(y), x) = FAILS
  - unify(P(x), P(f(x)) = ~~$[x \rightarrow f(x)]$~~ FAILS

Overview of unification

- Given two terms or formulas x and y, unifiy(x,y) returns a substitution θ such that θ(x) is syntactically the same as θ(y). If no such θ exists, the unification fails
- Examples
  - unify(P(f(x),y), P(z, g(s)) = $[z \mapsto f(x), y \mapsto g(1)]$
  - unify(P(x,y), P(y,x)) = $[x \mapsto y]$ OR $[y \mapsto x]$
  - unify(P(x, g(y)), P(f(y), x) = FAIL
  - unify(P(x), P(f(x)) = FAIL

Example

- Suppose we want to show:
  - $\forall x . P(x) \vdash \forall x . P(f(x))$

$P(v) \vdash P(f(c))$

Example

- Suppose we want to show:
  - $\forall x . P(x) \vdash \forall x . P(f(x))$

Skolemize:

$\forall x . P(x) \vdash P(f(c))$

Introduce variable v:

$P(v) \vdash P(f(c))$

unify with $[v \mapsto f(c)]$

$\Rightarrow$ Proof succeeds with $v \mapsto f(c)$ indicating that $f(c)$ is the right term to instantiate $\forall$.

Careful!

- Don't confuse skolemization with the introduction of variables for later unification
  - Skolemization applies to universals that we are trying to prove, and it introduces *constants*
  - The introduction of fresh variables for unification applies to universals in assumptions, and it introduces *variables*
- Although in some cases the difference between fresh variables and fresh constants is irrelevant, in the context of unification, the difference is important:
  - A variable can be unified with any constant, whereas a constant can only be unified with the exact same constant
- For example, try showing f(0) ⊢ ∀ x . f(x)

$f(0) \vdash f(c)$

Careful!

- Don't confuse skolemization with the introduction of variables for later unification
  - Skolemization applies to universals that we are trying to prove, and it introduces *constants*
  - The introduction of fresh variables for unification applies to universals in assumptions, and it introduces *variables*
- Although in some cases the difference between fresh variables and fresh constants is irrelevant, in the context of unification, the difference is important:
  - A variable can be unified with any constant, whereas a constant can only be unified with the exact same constant
- For example, try showing f(0) ⊢ ∀ x . f(x)

$f(0) \vdash f(c)$ does not unify !

First-order resolution

- We take the goal, negate it, and then:
  - We place the formula in prenex normal form, where quantifiers are on the outside
  - We remove existentials with skolemization (assumed existentials can be skolemized)
- We are only left with universals, for which we introduce fresh variables, in the hope of doing unification later on

First-order resolution

$\theta = unify(l_1, l_2)$

$$\frac{\Gamma \vdash \alpha \vee l_1 \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l_2 \vee \delta}{\Gamma \vdash \theta(\alpha \vee \beta \vee \gamma \vee \delta)} \text{ RES}$$

First-order resolution

$$\frac{\Gamma \vdash \alpha \vee l \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l \vee \delta}{\Gamma \vdash \alpha \vee \beta \vee \gamma \vee \delta} \text{ RES}$$

$$\frac{\theta = unify(l_1, l_2) \qquad \Gamma \vdash \alpha \vee l_1 \vee \beta \qquad \Gamma \vdash \gamma \vee \neg l_2 \vee \delta \qquad \alpha \vee l_1 \vee \beta \text{ and } \gamma \vee \neg l_2 \vee \delta \text{ have no common vars}}{\Gamma \vdash \theta(\alpha \vee \beta \vee \gamma \vee \delta)} \text{ GEN-RES}$$

$\forall x. \forall y\ P(x,y), \neg P(y,x) \wedge x \neq y$

$P(x), \neg P(x)$

Simple example

- P(a,b) ∨ ¬ Q(a,b,c) , ¬ P(s,t) ∨ R(s)

$\neg Q(s,t,c) \vee R(s)$

$\neg Q(a,b,c) \vee R(a)$

Simple example

- P(a,b) ∨ ¬ Q(a,b,c) , ¬ P(s,t) ∨ R(s)

$\neg Q(s,t,c) \vee R(s)$

OR $\neg Q(a,b,c) \vee R(a)$

Complete example

∀ x. R(x, f(x,x)) ⇒ ∀ x. ∃ y. R(x, y)

$\forall x \forall y \neg R(x, f(x,x)) \wedge \neg R(x_0, y)$

$R(v_1, f(v_1,v_2))$, $\neg R(x_0, v_2)$

$v_1 \mapsto x_0$

$v_2 \mapsto f(x_0, x_0)$ False

Complete example

∀ x. R(x, f(x,x)) ⇒ ∀ x. ∃ y. R(x, y)

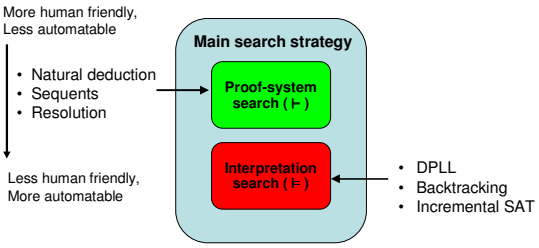
Negate: $\forall x.\ R(x, f(x,x)) \wedge \neg \underbrace{[\forall x. \exists y. R(x,y)]}_{\exists x. \forall y. \neg R(x,y)}$

Skolemize ∃: $\forall x.\ R(x, f(x,x)) \wedge \forall y. \neg R(x_0, y)$

Prenex form: $\forall x \forall y\ R(x, f(x,x)) \wedge \neg R(x_0, y)$

Introduce vars: $\underbrace{R(v_1, f(v_1,v_1))}$, $\neg R(x_0, v_2)$

unify: $v_1 \mapsto x_0$

$v_2 \mapsto f(x_0, x_0)$

Main search strategy review