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.

Resolvent 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

\[
\begin{align*}
\Gamma \vdash a \rightarrow l & \quad \Gamma \vdash l \rightarrow \delta \\
\Gamma \vdash a \rightarrow \delta & \quad \text{RES} \\
\Gamma \vdash \neg \alpha \lor \delta & \\
\Gamma \vdash \neg \alpha \lor l & \\
\Gamma \vdash l \rightarrow \delta & \\
\Gamma \vdash \neg l \lor \delta & \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]

Propositional resolution

\[
\begin{align*}
\Gamma \vdash a \lor l & \quad \Gamma \vdash l \lor \delta & \quad \text{RES} \\
\Gamma \vdash \neg a \lor \delta & \\
\end{align*}
\]
Propositional resolution

\[ \Gamma \vdash \alpha \lor \beta \lor \gamma \lor \delta \]

Which direction should we apply this rule in?

\[ \Gamma \vdash \alpha \lor \beta \lor \gamma \lor \delta \]

\[ \Gamma, \alpha, \Pi \vdash \alpha \]

Assume

\[ \Gamma, \alpha, \Pi \vdash a \]

Relation to other inference rules

\[ \Gamma \vdash a \lor \beta \lor \gamma \lor \delta \]

Recall Modus Ponens:

\[ \Gamma \vdash I \quad \Gamma \vdash I \rightarrow \delta \quad \Gamma \vdash \delta \]

\[ \Gamma \vdash I \quad \Gamma \vdash I \rightarrow \delta \quad \Gamma \vdash \delta \]

\[ \Gamma \vdash I \quad \Gamma \vdash \neg I \lor \delta \quad \Gamma \vdash \delta \]

MP is a special case of RES (with \( \alpha \), \( \beta \) and \( \gamma \) set false)
**Relation to other inference rules**

- Recall cut rule:

\[
\frac{\Gamma \vdash \alpha \lor \beta \lor \gamma \lor \delta}{\Gamma \vdash \alpha \lor \ddot{l} \lor \beta} \quad \text{CUT}
\]

- Resolution rule embodies the same idea as the cut rule from sequents.

**Example Proof**

\[
\frac{\Gamma \vdash \alpha \lor \beta \lor 
\gamma \lor \delta}{\Gamma \vdash \alpha \lor \dot{l} \lor \beta} \quad \text{RES}
\]

- Throw \( l \) to the other side

- “Expand” definition of sequents

**Example Proof**

- Find a derivation of

\[
\{ \neg P \lor Q, S \lor P, \neg Q \lor R, \neg R \} \vdash S
\]

**Proof is not unique**

- Find a derivation of

\[
\{ \neg P \lor Q, S \lor P, \neg Q \lor R, \neg R \} \vdash S
\]

**Another example**

- Find a derivation of:

\[
\emptyset \vdash A \lor \neg A
\]

- 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 \( \text{false} \)

Refutation

- Find a derivation of: \( \{\} \vdash A \lor \neg A \)

\[ \begin{array}{c}
\vdash A \\
\vdash \neg A
\end{array} \]

\[ \begin{array}{c}
\vdash \neg (A \lor \neg A)
\end{array} \]

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

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 \( \text{false} \) not in knowledge base:
    - Choose two formulas to resolve
    - Add resolvant formula to the knowledge base
    - If \( \text{false} \) 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: \( l \) 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

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
Assume:

\( B_1 \lor B_2 \rightarrow A \) (in prolog, written: \( A :- B_1, B_2 \))
\( B_3 \lor B_4 \rightarrow A \) (in prolog, written: \( A :- B_3, B_4 \))

Query: \( A \)

\( \text{Resolvew with } (\sigma) : \Gamma \lor A \)
\( \sigma = (\top) : \quad B_1 \lor B_2 \)

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 \( \neg l \)

Overview of unification

- Given two terms or formulas \( x \) and \( y \), \( \text{unify}(x, y) \) returns a substitution \( \theta \) such that \( \theta(x) \) is syntactically the same as \( \theta(y) \). If no such \( \theta \) exists, the unification fails.
- Examples
  - \( \text{unify}(P(f(x), y), P(z, g(s))) = [x \rightarrow f(x), y \rightarrow g(s)] \)
  - \( \text{unify}(P(x, y), P(y, x)) = [x \rightarrow y] \)
  - \( \text{unify}(P(x, g(y)), P(f(y), x)) = \text{failed} \)
  - \( \text{unify}(P(x), P(f(x))) = \text{failed} \)
Overview of unification

- Given two terms or formulas x and y, unify(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↦f(x), y↦y(s)]
  - unify(P(x,y), P(y,x)) = []
  - unify(P(x, g(y)), P(f(y), x)) = FAIL
  - unify(P(x), P(f(x))) = FAIL

Example

- Suppose we want to show:
  - ∀x. P(x) ⊢ ∀x. P(f(x))
  - p(y) ⊢ p(f(0))

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)

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**

\[ \sigma = \text{unify}(\alpha, \gamma) \]

\[
\frac{\Gamma \vdash \alpha \lor \beta \lor \gamma \lor \delta}{\Gamma \vdash \alpha \lor \beta \lor \gamma \lor \delta} \quad \text{RES}
\]

\[
\quad \theta = \text{unify}(l_1, l_2) \quad \frac{\Gamma \vdash \alpha \lor \beta \lor \gamma \lor \delta}{\Gamma \vdash \alpha \lor \beta \lor \gamma \lor \delta} \quad \text{GEN-RES}
\]

\[
\forall x. P(x, f(x)) \Rightarrow \forall x. \exists y. R(x, y)
\]

\[
\forall x. R(x, f(x)) \Rightarrow \forall x. \exists y. R(x, y)
\]

\[
\text{Negate: } \forall x. R(x, f(x)) \Rightarrow \forall x. \exists y. R(x, y)
\]

\[
\text{Substitution: } \forall x. R(x, f(x)) \Rightarrow \forall y. R(y, y)
\]

\[
\text{Proof from: } \forall x. \forall y. R(x, f(x)) \Rightarrow \forall y. R(y, y)
\]

\[
\text{Introduction uses: } R(x, f(x)), \quad \forall y. R(y, y)
\]

\[
\text{Substitution: } v_i \mapsto x_i, \quad v_j \mapsto f(x_i, x_j)
\]

**Simple example**

- \( P(a, b) \lor \neg Q(a, b, c) \), \( \neg P(s, t) \lor R(s) \)

\[
\forall x. (P(x, f(x)) \land \neg Q(x, f(x))) \land \neg R(x, f(x))
\]

\[
\forall x. (P(x, f(x)) \land \neg Q(x, f(x))) \land \neg R(x, f(x))
\]

**Complete example**

\[
\forall x. R(x, f(x)) \Rightarrow \forall x. \exists y. R(x, y)
\]

\[
\forall x. R(x, f(x)) \Rightarrow \forall x. \exists y. R(x, y)
\]

\[
\text{Negate: } \forall x. R(x, f(x)) \Rightarrow \forall x. \exists y. R(x, y)
\]

\[
\text{Substitution: } \forall x. R(x, f(x)) \Rightarrow \forall y. R(y, y)
\]

\[
\text{Proof from: } \forall x. \forall y. R(x, f(x)) \Rightarrow \forall y. R(y, y)
\]

\[
\text{Introduction uses: } R(x, f(x)), \quad \forall y. R(y, y)
\]

\[
\text{Substitution: } v_i \mapsto x_i, \quad v_j \mapsto f(x_i, x_j)
\]
Proof-system search ($\vdash \vdash \vdash \vdash$)  
Interpretation search ($\models \models \models \models$)  

Main search strategy review

More human friendly, Less automatable
- Natural deduction  
- Sequents  
- Resolution

Less human friendly, More automatable
- DPLL  
- Backtracking  
- Incremental SAT