Resolution in Propositional and First-Order Logic

Inference rules

- **Logical inference** creates new sentences that logically follow from a set of sentences (KB)
- An inference rule is **sound** if every sentence X it produces when operating on a KB logically follows from the KB
  - i.e., inference rule creates no contradictions
- An inference rule is **complete** if it can produce every expression that logically follows from (is entailed by) the KB.
  - Note analogy to complete search algorithms

Sound rules of inference

- Here are some examples of sound rules of inference
- Each can be shown to be sound using a truth table

RULE	PREMISE	CONCLUSION
Modus Ponens	A, A → B	B
And Introduction	A, B	A ∧ B
And Elimination	A ∧ B	A
Double Negation	¬¬A	A
Unit Resolution	A ∨ B, ¬B	A
Resolution	**A ∨ B, ¬B ∨ C**	**A ∨ C**

Soundness of modus ponens

A	B	A → B	OK?
True	True	True	√
True	False	False	√
False	True	True	√
False	False	True	√

Resolution

- **Resolution** is a valid inference rule producing a new clause implied by two clauses containing *complementary literals*
  - A literal is an atomic symbol or its negation, i.e., P, ~P
- Amazingly, this is the only interference rule you need to build a sound and complete theorem prover
  - Based on proof by contradiction and usually called resolution refutation
- The resolution rule was discovered by Alan Robinson (CS, U. of Syracuse) in the mid 60s

Resolution

- A KB is actually a set of sentences all of which are true, i.e., a conjunction of sentences.
- To use resolution, put KB into *conjunctive normal form* (CNF), where each sentence written as a disjunction of (one or more) literals

Tautologies
(A→B)↔(~A∨B)
(A∨(B∧C)) ↔(A∨B)∧(A∨C)

Example
- KB: [P→Q , Q→R∧S]
- KB in CNF: [~P∨Q , ~Q∨R , ~Q∨S]
- Resolve KB(1) and KB(2) producing: ~P∨R *(i.e., P→R)*
- Resolve KB(1) and KB(3) producing: ~P∨S *(i.e., P→S)*
- New KB: [~P∨Q , ~Q∨~R∨~S , ~P∨R , ~P∨S]

Soundness of the resolution inference rule

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
False	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*True*	*True*
False	*True*	*False*	*True*	*False*	*False*
False	*True*	*True*	*True*	*True*	*True*
True	*False*	*False*	*True*	*True*	*True*
True	*False*	*True*	*True*	*True*	*True*
True	*True*	*False*	*True*	*False*	*True*
True	*True*	*True*	*True*	*True*	*True*

From the rightmost three columns of this truth table, we can see that

(α ∨ β) ∧ (β ∨ γ) ↔ (α ∨ γ)

is valid (i.e., always true regardless of the truth values assigned to α, β and γ

Resolution

- Resolution is a **sound** and **complete** inference procedure for unrestricted FOL
- Reminder: Resolution rule for propositional logic:
  - $P_1 \vee P_2 \vee \ldots \vee P_n$
  - $\neg P_1 \vee Q_2 \vee \ldots \vee Q_m$
  - Resolvent: $P_2 \vee \ldots \vee P_n \vee Q_2 \vee \ldots \vee Q_m$
- We' ll need to extend this to handle quantifiers and variables

Resolution covers many cases

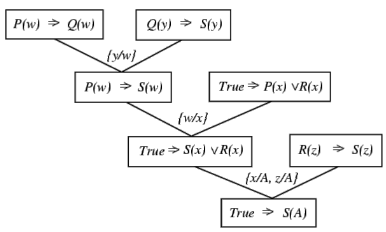
- Modes Ponens
  - from P and P → Q derive Q
  - from P and ¬ P ∨ Q derive Q
- Chaining
  - from P → Q and Q → R derive P → R
  - from (¬ P ∨ Q) and (¬ Q ∨ R) derive ¬ P ∨ R
- Contradiction detection
  - from P and ¬ P derive false
  - from P and ¬ P derive the empty clause (=false)

Resolution in first-order logic

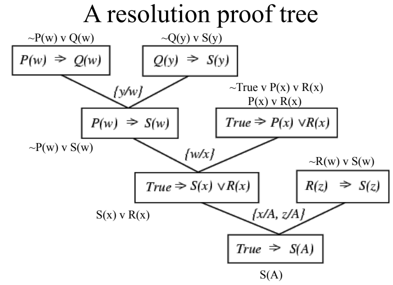
- Given sentences in *conjunctive normal form:*
  - $P_1 \vee \ldots \vee P_n$ and $Q_1 \vee \ldots \vee Q_m$
  - P_i and Q_i are literals, i.e., positive or negated predicate symbol with its terms
- if P_j and $\neg Q_k$ **unify** with substitution list θ, then derive the resolvent sentence:

  $\text{subst}(\theta, P_1 \vee \ldots \vee P_{j-1} \vee P_{j+1} \ldots P_n \vee Q_1 \vee \ldots Q_{k-1} \vee Q_{k+1} \vee \ldots \vee Q_m)$
- Example
  - from clause **P(x, f(a)) ∨ P(x, f(y)) ∨ Q(y)**
  - and clause **¬P(z, f(a)) ∨ ¬Q(z)**
  - derive resolvent **P(z, f(y)) ∨ Q(y) ∨ ¬Q(z)**
  - Using **θ = {x/z}**

A resolution proof tree



A resolution proof tree



Resolution refutation

- Given a consistent set of axioms KB and goal sentence Q, show that KB |= Q
- **Proof by contradiction:** Add ¬Q to KB and try to prove false, i.e.:

  (KB |- Q) ↔ (KB ∧ ¬Q |- False)
- Resolution is **refutation complete:** it can establish that a given sentence Q is entailed by KB, but can't (in general) generate all logical consequences of a set of sentences
- Also, it cannot be used to prove that Q is **not entailed** by KB
- Resolution **won't always give an answer** since entailment is only semi-decidable
  - And you can't just run two proofs in parallel, one trying to prove Q and the other trying to prove ¬Q, since KB might not entail either one

Resolution example

- KB:
  - allergies(X) → sneeze(X)
  - cat(Y) ∧ allergicToCats(X) → allergies(X)
  - cat(felix)
  - allergicToCats(mary)
- Goal:
  - sneeze(mary)

Refutation resolution proof tree