The Resolution Principle for First Order Logic

Summary

- Resolution for FOL [Chang-Lee Ch. 5.5]
- Completeness of the resolution principle [Chang-Lee Ch. 5.6]
- Examples of resolution [Chang-Lee Ch. 5.7]
- Deletion Strategy [Chang-Lee Ch. 5.8]

Resolution Principle for FOL

Brief Recap.

- We introduced resolution as a refutation procedure for prop. logic
- We know how to match literals containing variables using unification and substitutions
- We will see how to use these concepts to obtain a refutation procedure for FOL

Factor

Definition (Factor)

If two ore more literals (with the same sign) in a clause C have a most general unifier σ, then $C\sigma$ is called a factor for C. If $C\sigma$ is a unit clause then it is called a unit factor.

Example

Example (unit factor)

Consider $C = P(x) \vee P(a)$.

- $\sigma = \{a/x\}$ is a MGU for $P(x)$ and $P(a)$.
- $C\sigma = P(a)$ is a unit factor of C

Example II

Example (factor)

Consider $C = P(x) \vee P(f(y)) \vee \neg Q(x)$.

- $\sigma = \{f(y)/x\}$ is a MGU for $P(x)$ and $P(f(y))$.
- $C\sigma = P(f(y)) \vee \neg Q(f(y))$ is a factor of C

Binary Resolvent

Definition (Binary Resolvent)

Given two clauses C_1 and C_2 (called parent clauses) with no variables in common. Let L_1 and L_2 be two literals in C_1 and C_2 respectively. If L_1 and $\neg L_2$ have a MGU σ then the clause

$$(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$$

is a binary resolvent of C_1 and C_2. L_1 and L_2 are the literals solved upon.

Example: Binary Resolvent

Example (Binary Resolvent)

Consider the two clauses $C_1 = P(x) \vee Q(x)$ and $C_2 = \neg P(a) \vee R(x)$.

- Since x appears in both we will rename x with y in $C_2 = P(a) \vee R(y)$
- Choose $L_1 = P(x)$ and $L_2 = \neg P(a)$.
- L_1 and $\neg L_2 = P(a)$ have the MGU $\sigma = a/x$

$$
\begin{aligned}
(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma) &= \\
(\{P(a), Q(a)\} - \{P(a)\}) \cup ((\neg P(a), R(y)) - \{\neg P(a)\}) &= \\
(\{Q(a)\} \cup \{R(y)\} = \{Q(a), R(y)\} = Q(a) \vee R(y)
\end{aligned}
$$

- $Q(a) \vee R(y)$ is the binary resolvent and $P(x)$, $\neg P(a)$ are the literals resolved upon

Resolvent

Definition (Resolvent)

Given two clauses C_1 and C_2 (parent clauses) a resolvent is one of the following binary resolvents:

- a binary resolvent of C_1 and C_2
- a binary resolvent of C_1 and a factor of C_2
- a binary resolvent of a factor of C_1 and C_2
- a binary resolvent of a factor of C_1 and a factor of C_2

Example: Resolvent

Example (Resolvent)

Consider the two clauses $C_1 = P(x) \vee P(f(y)) \vee R(g(y))$ and $C_2 = \neg P(f(g(a))) \vee Q(b)$.

- $C_1' = P(f(y)) \vee R(g(y))$ is a factor of C_1
- $C_r = R(g(g(a))) \vee Q(b)$ is a binary resolvent of C_1' and C_2
- Therefore C_r is a resolvent of C_1 and C_2

Completeness of Resolution

Completeness of resolution

- Resolution is an inference rule that produce resolvents from sets of clauses
- It is more efficient than previous proof procedure (e.g. Gilmore + DPLL)
- Resolution is complete: if the set S of clauses is unsatisfiable using resolution we will always manage to obtain $\Box$

Example

Example (Trapezoid)

Show that alternate interior angles formed by a diagonal of a trapezoid are equal.

- $T(x, y, z, w)$ is true iff $xyzw$ are the vertices of a trapezoid.
- $P(x, y, u, v)$ is true iff line segment xy is parallel to line segment uv.
- $E(x, y, z, u, v, w)$ is true iff the angle xyz is equal to uvw.

Axioms:

- $A_1 \triangleq (\forall x)(\forall y)(\forall u)(\forall v)(T(x, y, u, v) \rightarrow P(x, y, u, v))$
- $A_2 \triangleq (\forall x)(\forall y)(\forall u)(\forall v)(P(x, y, u, v) \rightarrow E(x, y, v, u, v, y))$.
- $A_3 \triangleq T(a, b, c, d)$.

We want to proove that $G \triangleq E(a, b, d, c, d, b)$ holds, given A_1, A_2, A_3. Show that, by using resolution we can refute $A_1 \wedge A_2 \wedge A_3 \wedge \neg G$

Resolution and Semantic trees

Resolution and Semantic trees

- Resolution is deeply related to semantic trees
- Resolution generates clauses that can be used to prune branches of semantic trees
- Semantic trees can be used to prove completeness of resolution

Example

Example (resolution and semantic trees)

Consider the set of clauses $S = \{P, \neg P \vee Q, \neg P \vee \neg Q\}$. We can find a closed semantic tree with 5 nodes. Using resolution we can obtain:

$$\frac{\neg P \vee Q \qquad \neg P \vee \neg Q}{\neg P}$$

Consider the set $S' = S \cup C$, we can find a closed semantic tree with 3 nodes. Using resolution we can obtain:

$$\frac{\neg P \qquad P}{\Box}$$

Consider the set $S'' = S' \cup \Box$ we can find a closed semantic tree with one node.

Semantic tree and completeness of Resolution

Semantic trees and Resolution

- A similar reasoning can be used to prove the completeness of Resoluton
- Given a set of unsatisfiable clauses:
  1. Construct a closed semantic tree
  2. Force the tree to collapse while building a resolution proof.

Lifting lemma

Theorem

Lifting Lemma If C_1' and C_2' are instances of C_1 and C_2 respectively, and if C' is a resolvent of C_1' and C_2', then C' is an instance of C (resolvent of C_1 and C_2).

Example

Consider $C_1 = P(x) \vee Q(x)$ and $C_2 = \neg P(f(y)) \vee \neg P(z) \vee R(y)$.

- $C_1' = P(f(a)) \vee Q(f(a))$ is an instance of C_1
- $C_2' = \neg P(f(a)) \vee R(a)$ is an instance of C_2
- $C_3' = Q(f(a)) \vee R(a)$ is a resolvent for C_1' and C_2'
- Lifting Lemma $\Rightarrow \exists\ C_3$ such that C_3' is an instance of C_3.
- For example, $C_3 = Q(f(y)) \vee R(y)$ is a resolvent for C_1 and C_2 and C_3' is an instance of C_3

Lifting lemma: proof

Lifting Lemma

- If necessary we rename variables in C_1 or C_2 so that variables in C_1 are all different from variables in C_2.
- Let L_1' and L_2' be the literals resolved upon
- $C' = (C_1'\gamma - L_1'\gamma) \cup (C_2'\gamma - L_2'\gamma)$, γ MGU for L_1', L_2'.
- Since C_1' and C_2' are instances of C_1' and C_2' we can write $C_1' = C_1\theta$ and $C_2' = C_2\theta$ where θ is one substitution.
- Let $L_i^1, \cdots, L_i^{R_i}$ denote the literals in C_i corresponding to L_i' (i.e. $L_i^1\theta, \cdots, L_i^{R_i}\theta = L_i'$)

Lifting lemma: proof II

Lifting Lemma

- assume $i > 1$ obtain a MGU λ_i for $L_i^1, \cdots, L_i^{R_i}$. and let $L_i = L_i^1 \lambda_i$ for $i = 1, 2$.
- then L_i is a literal in factor $C_i \lambda_i$ of C_i.
- assume $i = 1$ then $\lambda_i = \epsilon$ and $L_i = L_i^1 \lambda_i$.
- Let $\lambda = \lambda_1 \cup \lambda_2$
- Then L_i' is an instance of L_i
- Since L_1' and L_2' are unifiable then L_1 and L_2 are unifiable.
- Let σ be a MGU of L_1 and L_2

Lifting lemma: proof III

Proof.

(Lifting Lemma)

- Let $C = (C_1(\lambda \circ \sigma) - (\{L_1^1, \cdots, L_1^{R_1}\})(\lambda \circ \sigma)) \cup ((C_2(\lambda \circ \sigma) - (\{L_2^1, \cdots, L_2^{R_2}\})(\lambda \circ \sigma)))$
- Then $C' = (C_1(\theta \circ \gamma) - (\{L_1^1, \cdots, L_1^{R_1}\})(\theta \circ \gamma)) \cup ((C_2(\theta \circ \gamma) - (\{L_2^1, \cdots, L_2^{R_2}\})(\theta \circ \gamma)))$ is an instance of C as $\lambda \circ \sigma$ is a more general unifier than $\theta \circ \gamma$

□

Completeness of Resolution

Theorem (Completeness of Resolution)

A set S of clauses is unsatisfiable iff there is a resolution deduction of the empty clause $\Box$ from S

Completeness of Resolution: proof $\Leftarrow$

$\Leftarrow$.

If there is a resolution deduction of the empty clause $\Box$ from S then S is unsatisfiable

- Suppose there is a deduction of $\Box$ from S. Let $R_1, R_2, \cdots, R_k$ be the resolvents in the deduction.
- Assume S is satisfiable then there is $I \models S$.
- Assume R_i is resolvent of C_u and C_v, notice that $I \models S$ therefore $I \models C_u \wedge C_v$
- Since resolution is an inference rule then if $I \models C_u \wedge C_v$ then $I \models R_i$ for all resolvents
- However, one of the resolvents is $\Box$ therefore S must be unsatisfiable.

$\Box$

Completeness of Resolution: proof $\Rightarrow$

$\Rightarrow$

If S is unsatisfiable then there is a resolution deduction of the empty clause $\square$ from S.

- Suppose S is unsatisfiable, and let $A = \{A_1, A_2, A_3, \cdots\}$ be the atome set for S.
- Let T be a complete semantic tree for S.
- By Herbrand's theorem (version I) T has a finite closed sematic tree T'
- If T' consists only of one root node then $\square$ must be in S, because no other clauses can be falsified at the root of a semantic tree, Thus the theorem is true.

Completeness of Resolution: proof $\Rightarrow$

$\Rightarrow$

- Assume T' has more than one node.
- T' must have at least one inference node
- This is because, otherwise, every node would have at least one non failure descendent and thus T' would have an infinite branch (and thus not be a closed tree).
- Let N be an inference node in T', and let N_1 and N_2 be the failure nodes immediately below N.
- Let $I(N) = \{m_1, m_2, \cdots, m_n\}$, $I(N_1) = \{m_1, m_2, \cdots, m_n, m_n + 1\}, I(N_2) = \{m_1, m_2, \cdots, m_n, \neg m_n + 1\}$

Completeness of Resolution: proof $\Rightarrow$

$\Rightarrow$

- Since N not a failure node, there exist C_1' and C_2', ground instances of C_1 and C_2 such that:
  - C_1' and C_2' are both not falsified by $I(N)$
  - C_1' and C_2' are falsified by $I(N_1)$ and $I(N_2)$ respectively.
- Therefore C_1' contains $\neg m_{n+1}$ and C_2' contains m_{n+1}
- Let $L_1' = \neg m_{n+1}$ and $L_2' = m_{n+1}$ and $C' = (C_1' - L_1') \cup (C_2' - L_2')$
- C' must be false in $I(N)$ because both $(C_1' - L_1')$ and $(C_2' - L_2')$ are.
- By the lifting lemma we can then find a resolvent C of C_1 and C_2 such that C' is a (ground) instance of C

Completeness of Resolution: proof $\Rightarrow$

$\Rightarrow$

- Let T'' be the closed semantic tree associated to $S \cup C$.
- T'' is obtained by T' removing all noded which are below the first node where C' is falsified
- T'' has fewer nodes than T'
- We can apply this process until the closed semantic tree consists only of the root node.
- This is possible only when $\square$ is derived, therefore there is deduction of $\square$ from S.

Example I

Example

- $A_1 \triangleq P \to S$
- $A_2 \triangleq S \to U$
- $A_2 \triangleq P$
- $G \triangleq U$

Show that $(A_1 \wedge A_2 \wedge A_3) \models G$.

Example II

Example

- $F \triangleq (\forall x)(\forall y)(P(x, f(y)) \vee P(y, f(x)))$
- $G \triangleq (\exists u)(\exists v)(P(u, f(v)) \wedge P(v, f(u)))$

Show that $F \models G$.

Example III

Example (quack and doctors)

Show that $F_1 \wedge F_2 \models G$, where

- Some patients like all doctors
- $F_1 \triangleq \exists x(P(x) \wedge \forall y(D(y) \to L(x, y))$
- No patient likes any quack
- $F_2 \triangleq \forall x(P(x) \to \forall y(Q(y) \to \neg L(x, y)))$
- No doctor is a quack
- $F_3 \triangleq \forall x D(x) \to \neg Q(x)$

Exercise I

Exercise

Show that $F_1 \wedge F_2 \models G$, where

- $F_1 \triangleq (\forall x)(C(x) \to (W(x) \wedge R(x)))$
- $F_2 \triangleq (\exists x)(C(x) \wedge O(x))$
- $G \triangleq (\exists x)(O(x) \wedge R(x))$

Exercise II

Exercise

Students and votes
Premise: Students are citizens.
Conclusion: Students' votes are citizens votes.

- Students are citizens
- $F_1 \triangleq (\forall y)(S(y) \rightarrow C(y))$
- Students' votes are citizens votes
- $F_2 \triangleq (\forall x)((\exists y)(S(y) \wedge V(x, y)) \rightarrow (\exists z)(C(z) \wedge V(x, z)))$

Deleting Clauses

Need for deleting clauses

- Resolution is complete (Binary resolution + factorisation)
- Resolution is more efficient than earlier methods (e.g., Gilmore + DPLL)
- computational issue: Repeated application of resolution generates irrelevant and redundant clauses

Applying Resolution

Computing resolvents

- Need a deterministic method to apply resolution
- Deterministic strategy to compute resolvents
- Straightforward strategy:
  - compute resolvents for all possible pairs
  - add resolvents to S
  - repeat until □ appears
- Called Level Saturation

Level Saturation

Level Saturation Definition

- Generate the sequence $S^0, S^1, S^2, \cdots$
- $S^0 = S$
- $S^i = \{\text{Resolvents of } C_1 \text{ and } C_2 | C_1 \in (S^0 \cup \cdots \cup S^{i-1}) \text{ and } C_2 \in S^{i-1}\}, i = 1, 2, 3, \cdots$

Level Saturation: Procedure

Level Saturation Procedure

At every step $i > 0$

- List all clauses in $\{S^0 \cup \cdots \cup S^{i-1}\}$ in order
- compute all resolvents by comparing every clause $C_1 \in \{S^0 \cup \cdots \cup S^{i-1}\}$ with a clause $C_2 \in S^{i-1}$ that is listed after C_1.
- append computed resolvents to the end of the list

Example: Level Saturation

Example (Level Saturation)

Consider the set of clauses
$S = \{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$

Level Saturation: Problems

Problems with Level Saturation

- Generation of many irrelevant and redundant clauses
  - Tautologies
  - Clauses repeatadly generated
- Tautologies have no impact on satisfiability
  - Tautologies are true in every interpretations
  - If S is unsatisfiable, S' obtained from S removing tautologies is unsatisfiable
  - Tautologies can create other irrelevant clauses
- We need a deletion strategy that maintains completeness

Subsumption

Definition (Subsumption)

A clause C subsumes a clause D iff there is a substitution σ such that $C\sigma \subseteq D$. D is called a subsumed clause.

Example (Subsumption)

Consider the two clauses $C = P(x)$ and $D = P(a) \vee Q(a)$.

- Consider the substitution $\sigma = \{a/x\}$.
- $C\sigma = P(a)$ therefore $C\sigma \subseteq D$
- C subsumes D.

Note

If C is identical to D or if C is an instance of D then C subsumes D.

Deletion Strategy

A deletion strategy

Delete any tautology and any subsumed clause whenever possible

A Complete deletion strategy

The above deletion strategy is complete if it is used with the level saturation method

For each step $i > 0$:

1. List clauses in $S^0 \cdots S^{i-1}$ in order
2. Compute resolvents by comparing any clause in $C_1 \in S^0 \cdots S^{i-1}$ with a clause $C_2 \in S^{i-1}$ which il listed after C_1
3. When a resolvent C is computed, append C to the list only if C is not a tautology and C is not subsumed by any claus in the list. Otherwise delete C.

Example: Level Saturation deleting clauses

Example (Level Saturation Deleting clauses)

Consider the set of clauses
$S = \{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$

Checking redundant clauses

Redundant clauses

Need to check:

1. whether a clause is a tautology Easy
2. whether a clause is subsumed by another clause need an algorithm

Checking tautology

- Directly check whether there is a complementary pair in the clause.
- No substitutions involved.

Checking Subsumption

Preliminaries

- Consider two clauses C and D.
- Let $\theta = \{a_1/x_1, \cdots a_n/x_n\}$ where: $\{x_1, \cdots, x_n\}$ are all variables occurring in D and $\{a_1, \cdots, a_n\}$ are new distinct constants not occurring in C or D.
- Suppose $D = L_1 \vee L_2 \vee \cdots L_m$ then $D\theta = L_1\theta \vee L_2\theta \vee \cdots \vee L_m\theta$
- Note that $D\theta$ is a ground clause.
- $\neg D\theta = \neg L_1\theta \wedge \cdots \wedge \neg L_m\theta$ (using de morgan's law)

Subsumption Algorithm

Algorithm

1. Let $W = \{\neg L_1\theta \cdots \neg L_m\theta\}$
2. Set $k = 0$ and $U^0 = \{C\}$
3. If U^k contains $\square$
   - Yes: terminate; C subsumes D
   - Otw: let $U^{k+1} = \{$ Resolvents of C_1 and $C_2 | C_1 \in U^k$ and $C_2 \in W\}$
4. If U^{k+1} is empty
   - Yes: terminate; C does not subsume D
   - Otw: $k = k + 1$ go to step 3.

Example: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = \neg P(x) \vee Q(f(x), a)$
- $D = \neg P(h(y)) \vee Q(f(h(y)), a) \vee \neg P(c)$

Subsumption Algorithm: termination

Termination

Subsumption algorithm always terminates.

- Each clause CU^{k+1} is always one litteral smaller than clauses in U^k for $k = 0, 1, \cdots$
- This is because U^{k+1} is obtained by computing the resolvents of clauses in U^k and W, therefore, if a resolvent exists it will always be one literal smaller than the parent clauses. Otw U^{k+1} is empty.
- Therefore for some k we will have $\Box \in U^k$ or U^k is empty.

Example II: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = P(x, x)$
- $D = P(f(x), y) \lor P(y, f(x))$

Subsumption Algorithm: correctness

Theorem (Correctness)

C subsumes D iff subsumption algorithm terminates in step 3.

Proof $\Rightarrow$

$\Rightarrow$.

If C subsumes D then subsumption algorithm terminates in step 3

- If C subsumes D then there is σ such that $C\sigma \subseteq D$
- Hence $C(\sigma \circ \theta) \subseteq D\theta$
- Therefore literals in $C(\sigma \circ \theta)$ can be resolved by using unit gound clauses in W
- But $C(\sigma \circ \theta)$ is an instance of C
- Therefore literals in C can be resolved away by using unit clauses in W
- Therefore we will eventually find a U^k such that $\Box \in U^k$ and the algorithm will terminate at step 3.

$\Box$

Proof $\Leftarrow$

$\Leftarrow$.

If the subsumption algorithm terminates in step 3 then C subsumes D

- If algorithm terminates at step 3 then we have a refutation of $\Box$.
- Indicates with R_i, B_i the parent clauses, where $B_i \in W$, and with $R_0 = C$; Indicates with R_{i+1} the resolvent obtained at each step for $i = 0, 1, \cdots, r$
- Let σ_i be the most general unifier for each resolution step.
- Then $C(\sigma_0 \circ \sigma_1 \circ \cdots \circ \sigma_r) = \{\neg B_0, \neg B_1, \cdots \neg B_r\} \subseteq D\theta$
- Let $\lambda = \sigma_0 \circ \cdots \circ \sigma_r$ then $C\lambda \subseteq D\theta$.
- Let σ be the substitution obtained by replacing a_i with x_i in each component of λ for $i = 1, \cdots, n$
- Then $C\sigma \subseteq D$ therefore C subsumes D.

Example III: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = P(x, y) \vee Q(z)$
- $D = Q(a) \vee P(b, b) \vee R(u)$

Check whether C subsumes D

Example III: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = P(x, y) \vee Q(z)$
- $D = Q(a) \vee P(b, b) \vee R(u)$

Check whether C subsumes D

Sol

C Subsumes D

Example IV: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = P(x, y) \vee R(y, x)$
- $D = P(a, y) \vee R(z, b)$

Check whether C subsumes D

Example IV: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = P(x, y) \vee R(y, x)$
- $D = P(a, y) \vee R(z, b)$

Check whether C subsumes D

Sol

C Does not subsumes D

Example V: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = \neg P(x) \vee P(f(x))$
- $D = \neg P(x) \vee P(f(f(x)))$

Check whether C subsumes D and whether $C \models D$

Example V: Subsumption algorithm

Example (Subsumption algorithm)

Consider the two clauses:

- $C = \neg P(x) \vee P(f(x))$
- $D = \neg P(x) \vee P(f(f(x)))$

Check whether C subsumes D and whether $C \models D$

Sol

C Does not subsumes D but $C \models D$