Overview

- Project 1 tips
- Resolvents
- Resolution in first order logic: example
- Theorem proving strategies
- Uncertainty

Announcement

Lydia will have a Q&A session for the project: HRBB 104, Thu 6–8pm.

Project 1 Tips

- It is normal for certain search strategies to take extremely long (or even fail) on harder problems. Don’t spend more than 5 minutes. If you’re patient, wait up till 10 minutes.
- Compile your code `gcl -compile prog.lsp` and load the `prog.o` file.
- Use the `dupe` function to check for duplicate nodes before expanding. Keep a global list `explored` of explored nodes to check for duplicates.

Resolving Two Clauses: Revisited

Resolving two clauses $C_1$ and $C_2$ with the most general unifier $\sigma$:

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

This is basically:

1. Find the most general unifier $\sigma$.
2. Apply $\sigma$ to both $C_1$ and $C_2$
3. Remove the complementary literal from $C_1$ and $C_2$.

Resolving Two Clauses: Example Revisited

Example: Resolve the following (hint: $\sigma = \{x/a\}$)

\[
\begin{align*}
C_1 &= P(x) \lor Q(x) \quad \text{and} \quad C_2 = \neg P(a) \lor R(y) \\
&= \underbrace{L_1}_{C_1} \quad \underbrace{L_2}_{C_2}
\end{align*}
\]

\[
\begin{align*}
C_1 &= P(x) \lor Q(x) \\
&= \underbrace{L_1}_{C_1} \\
\sigma &= \{x/a\} \\
&= \underbrace{L_2}_{C_2} \\
C_1\sigma &= P(a) \lor Q(a) \\
C_2\sigma &= \neg P(a) \lor R(y)
\end{align*}
\]

1. Remove $L_1\sigma = P(a)$
2. Remove $L_2\sigma = \neg P(a)$

\[
\begin{align*}
(C_1\sigma - L_1\sigma) &= Q(a) \\
(C_2\sigma - L_2\sigma) &= \lor R(y)
\end{align*}
\]
Resolvent: A Full Example

Example: resolve the two clauses
1. \[ C_1 = P(x) \lor P(f(y)) \lor R(g(y)) \] and
2. \[ C_2 = \neg P(f(g(a))) \lor Q(b). \]

1. Get the factor of \( C_1 \):
   \[ C_1 \{x/f(y)\} = P(f(y)) \lor R(g(y)) \]

2. Resolve factor of \( C_1 \) and \( C_2 \):
   \[ P(f(y)) \lor R(g(y)) \text{ vs. } \neg P(f(g(a))) \lor Q(b) \]

3. \[ \sigma = \{ y/g(a) \} \text{ remove} \]
   \[ P(f(g(a))) \lor R(g(a)) \text{ vs. } \neg P(f(g(a))) \lor Q(b) \text{ remove} \]

4. Result:
   \[ R(g(g(a))) \lor Q(b) \]

Property of Resolution for First-Order Logic

- **Complete**: If a set of clauses \( S \) is unsatisfiable, resolution will eventually derive \( F \).
  - *Everything that is true can be proved (eventually).*

- **Sound**: If \( F \) is derived by resolution, then the original set of clauses \( S \) is unsatisfiable.
  - *Everything that is proved is true.*

Weaknesses of Resolution

Basically, resolution tries to derive

\[ \text{Axioms} \land \neg \text{Theorem} = F \]

- Is there a \( F \) in the axioms? If there is, the whole formula will always be unsatisfiable no matter what.

- Can we tell whether axioms alone can derive \( F \)? (generally, this is not the case)

Example Proof Using Resolution

Given: (1) The customs officials searched everyone who entered the country who was not a VIP. (2) Some of the drug dealers entered the country, and they were only searched by drug dealers. (3) No drug dealer was a VIP.

Prove: (4) Some of the customs officials were drug dealers.

\[ \text{Chang & Lee, Example 5.22} \]
Example: Predicates

1. $C(x): x$ is a customs official
2. $E(x): x$ entered the country
3. $V(x): x$ is a VIP
4. $S(x, y): x$ was searched by $y$
5. $D(x): x$ is a drug dealer

Example: English to First Order Logic

(1) The customs officials searched everyone who entered the country who was not a VIP. (2) Some of the drug dealers entered the country, and they were only searched by drug dealers. (3) No drug dealer was a VIP. (4) Some of the customs officials were drug dealers.

1. $\forall x((E(x) \land \neg V(x)) \rightarrow \exists y(S(x, y) \land C(y)))$
2. $\exists x(E(x) \land D(x) \land \forall y(S(x, y) \rightarrow D(y)))$
3. $\forall x(D(x) \rightarrow \neg V(x))$
4. $\exists x(D(x) \land C(x))$

Example: Standard Form (I)

(1) $\forall x((E(x) \land \neg V(x)) \rightarrow \exists y(S(x, y) \land C(y)))$
{\{rm \rightarrow\}} = $\forall x((\neg (E(x) \land \neg V(x)) \lor \exists y(S(x, y) \land C(y)))$
{\{prenex\}} = $\forall x \exists y(\neg E(x) \lor V(x) \lor (S(x, f(x)) \land C(f(x))))$
{\{skol\}} = $\forall x((\neg E(x) \lor V(x)) \lor (S(x, f(x)) \land C(f(x))))$
{\{add\}} = $\forall x((\neg E(x) \lor V(x)) \lor (S(x, f(x)) \land C(f(x))))$
{\{dist\}} = $\forall x((\neg E(x) \lor V(x)) \lor S(x, f(x))$

$\land (\neg E(x) \lor V(x) \lor C(f(x))))$

Clauses:
(1a) $\neg E(x) \lor V(x) \lor S(x, f(x))$
(1b) $\neg E(x) \lor V(x) \lor C(f(x))$

Example: Standard Form (II)

(2) $\exists x(E(x) \land D(x) \land \forall y(S(x, y) \rightarrow D(y)))$
{\{rm \rightarrow\}} = $\exists x(E(x) \land D(x) \land \forall y(\neg S(x, y) \lor D(y)))$
{\{prenex\}} = $\exists x \forall y(E(x) \land D(x) \land (\neg S(x, y) \lor D(y)))$
{\{skol\}} = $\forall y(E(a) \land D(a) \land (\neg S(a, y) \lor D(y))))$

Clauses:
(2a) $E(a)$
(2b) $D(a)$
(2c) $\neg S(a, y) \lor D(y)$
Example: Standard Form (III)

(3) \( \forall x(D(x) \rightarrow \neg V(x)) \)
\{\text{rm} \rightarrow \} = \forall x(\neg D(x) \lor V(x))

Clause:
(3) \( \neg D(x) \lor V(x) \)

(4) \( \exists x(D(x) \land C(x)) \)
\{\text{negate}\} \Rightarrow \neg(\exists x(D(x) \land C(x)))
\{\text{prenex}\} = \forall x(\neg D(x) \land \neg C(x))
\{C N F\} = \forall x(\neg D(x) \lor \neg C(x))

Example: Clauses

(1a) \( \neg E(x) \lor V(x) \lor S(x, f(x)) \)
(1b) \( \neg E(x) \lor V(x) \lor C(f(x)) \)
(2a) \( E(a) \)
(2b) \( D(a) \)
(2c) \( S(a, y) \lor D(y) \)
(3) \( \neg D(x) \lor V(x) \)
(4) \( \neg D(x) \lor \neg C(x) \)

Note: The input to your theorem prover will be in a standard form like the above.

Exercise 1: rewrite the above in LISP representation.

Exercise 2: use resolution to derive \( F \).

Basic Theorem Proving Algorithm

Level saturation resolution method (or two-pointer method)

Generate all possible resolvents:

- Generate sequences of clauses \( S^0, S^1, S^2, \ldots \), where
  \[
  S^0 = S \quad (\text{original set of clauses})
  \]
  \[
  S^n = \{ \text{all possible resolvents of clauses} \}
  \]
  \[
  C_1 \in (S^0 \cup \ldots S^{n-1}) \text{ and } C_2 \in S^{n-1} \}
  \]
- This is basically a breadth first search method, and it can be extremely inefficient except for small problems.
- The problem is that irrelevant derivations are made: in generating an n-step proof, we also generate all possible derivations of n-1 steps.

Deletion Strategy

To reduce the huge number of generated clauses, we would like to delete clauses whenever possible. We can delete:

1. Any tautology, e.g. \( P(a) \lor \neg P(a) \lor Q(x) \).
2. Any clause which duplicates an existing clause.
3. Any clause which is subsumed by an existing clause.

A clause \( C \) subsumes a clause \( D \) iff there is a substitution \( \sigma \) such that \( C \sigma \subseteq D \) (recall that a clause can be represented as a set of literals). \( D \) is called a subsumed clause.

Deletion strategy will be complete if it is used with certain resolution algorithms (such as level saturation).
Subsumed Clause: Example (I)

Example:

- \( C = P(x) \)
- \( D = P(a) \lor Q(a) \)
- If \( \sigma = \{ x/a \} \), then
  \[
  C\sigma = P(a) = \{ P(a) \} \subseteq \{ P(a), Q(a) \} = P(a) \lor Q(a) = D.
  \]
- Since \( C\sigma \subseteq Q \), \( C \) subsumes \( D \), and \( D \) can be deleted.

Advantages and Disadvantages of Resolution

- **Advantages**: (1) Resolution is universally applicable to problems which can be described in first-order logic. (2) The theorem proving engine can be decoupled from any particular domain.

- **Disadvantage**: (1) Resolution is too inefficient to be generally applicable. (2) This is partly because resolution is purely syntactic, and it does not consider what the predicates mean. For this reason, developing a domain-dependent heuristic is impossible. (3) A contradiction in the axiom set may allow anything to be proved. (4) It is difficult for a human to understand proof by resolution prover.

Strategies to Improve Resolution

1. **Deletion strategy**: remove tautology, duplicates, and subsumed clauses.

2. **Unit preference**: resolve with clauses with the fewest literals.

3. **Set of support**: begin with set \( T \) consisting of the clauses from the negated conclusion. Each resolution step must involve a member of \( T \), and the result is added to \( T \).

4. **Linear resolution** (Depth First): Each step must be a resolution step involving the clause produced by the last step.

Key Points

- resolvent: definition, and how to derive
- properties of resolution: sound and complete
- theorem proving algorithm: level saturation (two pointer method)
- theorem proving: strategies for efficient resolution
- advantages and disadvantages of resolution.
Next Time

- Solutions to exercises
- Application of predicate calculus: question answering
- Uncertainty: chapter 14