Inference in first-order logic

Logical inference in FOL

Logical inference problem:
• Given a knowledge base KB (a set of sentences) and a sentence $\alpha$, does the KB semantically entail $\alpha$?

$$KB \models \alpha$$

• Logical inference problem in the first-order logic is undecidable !!!
  – No procedure that can decide the entailment for all possible input sentences in a finite number of steps.

• Methods:
  – Inference rule approach
  – Resolution refutation
Inference rules for quantifiers

- **Universal elimination**
  \[
  \forall x \phi(x) \quad a - \text{is a constant symbol}
  \]
  - Substitutes a variable with a constant symbol
  \[
  \forall x \, Likes(x, \text{IceCream}) \quad Likes(\text{Ben, IceCream})
  \]

- **Existential elimination.**
  \[
  \exists x \phi(x) \\
  \phi(a)
  \]
  - Substitutes a variable with a constant symbol that does not appear elsewhere in the KB
  \[
  \exists x \, Kill(x, \text{Victim}) \quad Kill(\text{Murderer, Victim})
  \]

Inference rules for quantifiers

- **Universal instantiation (introduction)**
  \[
  \phi \\
  \forall x \phi
  \]
  - Introduces a universal variable which does not affect \( \phi \) or its assumptions
  \[
  \text{Sister(Amy, Jane)} \quad \forall x \, \text{Sister(Amy, Jane)}
  \]

- **Existential instantiation (introduction)**
  \[
  \phi(a) \\
  \exists x \phi(x)
  \]
  - Substitutes a ground term in the sentence with a variable and an existential statement
  \[
  \text{Likes(Ben, IceCream)} \quad \exists x \, \text{Likes(x, IceCream)}
  \]
Unification

- **Problem in inference:** Universal elimination gives us many opportunities for substituting variables with ground terms
  \[ \forall x \, \phi(x) \quad \overset{\phi(a)}{=} \quad a \text{ - is a constant symbol} \]

- **Solution:** avoid making blind substitutions of ground terms
  - Make substitutions that help to advance inferences
    - Use substitutions matching “similar” sentences in KB
  - Make inferences on the variable level
    - Do not substitute ground terms if not necessary

- **Unification** – takes two similar sentences and computes the substitution that makes them look the same, if it exists
  \[ \text{UNIFY}(p, q) = \sigma \text{ s.t. SUBST}(\sigma, p) = \text{SUBST}(\sigma, q) \]

Generalized inference rules

- **Use substitutions that let us make inferences !!!!!**

  **Example: Generalized Modus Ponens**

  - If there exists a substitution \( \sigma \) such that
    \[ \text{SUBST}(\sigma, A_i) = \text{SUBST}(\sigma, A_i') \quad \text{for all } i=1,2, n \]
  
    \[ A_1 \land A_2 \land \ldots A_n \Rightarrow B, \quad A_1', A_2', \ldots A_n' \]

    \[ \text{SUBST}(\sigma, B) \]

  - Substitution that satisfies the generalized inference rule can be build via unification process
  - Advantage of the generalized rules: they are focused
    - only substitutions that allow the inferences to proceed are tried
Resolution inference rule

- **Recall:** Resolution inference rule is sound and complete (refutation-complete) for the **propositional logic** and CNF

\[
A \lor B, \quad \neg A \lor C \\
\hline
B \lor C
\]

- **Generalized resolution rule is sound and refutation complete** for the first-order logic and CNF w/o equalities (if unsatisfiable the resolution will find the contradiction)

\[
\sigma = \text{UNIFY}(\phi_i, \neg \psi_j) \neq \text{fail} \\
\phi_1 \lor \phi_2 \cdots \lor \phi_k, \quad \psi_1 \lor \psi_2 \lor \cdots \lor \psi_n \\
\text{SUBST}(\sigma, \phi_1 \lor \cdots \lor \phi_{i-1} \lor \phi_{i+1} \lor \cdots \lor \phi_k \lor \psi_1 \lor \cdots \lor \psi_{j-1} \lor \psi_{j+1} \lor \cdots \lor \psi_n)
\]

**Example:**

\[
P(x) \lor Q(x), \quad \neg Q(\text{John}) \lor S(y) \\
P(\text{John}) \lor S(y)
\]

Inference with the resolution rule

Matches the KB in CNF

- **Proof by refutation:**
  - Prove that $KB, \neg \alpha$ is unsatisfiable
  - resolution is refutation-complete

- **Main procedure (steps):**
  1. Convert $KB, \neg \alpha$ to CNF with ground terms and universal variables only
  2. Apply repeatedly the resolution rule while keeping track and consistency of substitutions
  3. Stop when empty set (contradiction) is derived or no more new resolvents (conclusions) follow
Conversion to CNF

1. Eliminate implications, equivalences
    \((p \iff q) \rightarrow (\neg p \lor q)\)

2. Move negations inside (DeMorgan’s Laws, double negation)
    \neg(p \land q) \rightarrow \neg p \lor \neg q
    \neg(p \lor q) \rightarrow \neg p \land \neg q

3. Standardize variables (rename duplicate variables)
    \((\forall x P(x)) \lor (\exists x Q(x)) \rightarrow (\forall x P(x)) \lor (\exists y Q(y))\)

4. Move all quantifiers left (no invalid capture possible)
    \((\forall x P(x)) \lor (\exists y Q(y)) \rightarrow \forall x \exists y P(x) \lor Q(y)\)

---

Conversion to CNF

5. Skolemization (removal of existential quantifiers through elimination)
   - If no universal quantifier occurs before the existential quantifier, replace the variable with a new constant symbol also called Skolem constant
     \(\exists y P(A) \lor Q(y) \rightarrow P(A) \lor Q(B)\)
   - If a universal quantifier precedes the existential quantifier replace the variable with a function of the “universal” variable
     \(\forall x \exists y P(x) \lor Q(y) \rightarrow \forall x P(x) \lor Q(F(x))\)

\(F(x)\) - a special function
- called Skolem function
Conversion to CNF

6. Drop universal quantifiers (all variables are universally quantified)

$$\forall x \ P(x) \lor Q(F(x)) \rightarrow P(x) \lor Q(F(x))$$

7. Convert to CNF using the distributive laws

$$p \lor (q \land r) \rightarrow (p \lor q) \land (p \lor r)$$

The result is a CNF with variables, constants, functions

Resolution example

KB

$$\neg A$$

$$\neg P(w) \lor Q(w), \neg Q(y) \lor S(y), \ P(x) \lor R(x), \neg R(z) \lor S(z), \neg S(A)$$
Resolution example

\[
\text{KB} \quad \neg \alpha \\
\neg P(w) \lor Q(w), \neg Q(y) \lor S(y), \ P(x) \lor R(x), \ \neg R(z) \lor S(z), \ \neg S(A)
\]

\[
\{y/w\} \\
\neg P(w) \lor S(w)
\]

Resolution example

\[
\text{KB} \quad \neg \alpha \\
\neg P(w) \lor Q(w), \neg Q(y) \lor S(y), \ P(x) \lor R(x), \ \neg R(z) \lor S(z), \ \neg S(A)
\]

\[
\{y/w\} \\
\neg P(w) \lor S(w)
\]

\[
\{x/w\} \\
S(w) \lor R(w)
\]
Resolution example

\[ \neg \alpha \]

\[ \text{KB} \]

\[ \neg P(w) \lor Q(w), \neg Q(y) \lor S(y), \ P(x) \lor R(x), \neg R(z) \lor S(z), \neg S(A) \]

\[ \neg P(w) \lor S(w), \{y/w\} \]

\[ \neg P(w) \lor S(w), \{x/w\} \]

\[ S(w) \lor R(w), \{z/w\} \]

\[ S(w) \]

\[ \text{Empty resolution} \]
Dealing with equality

- Resolution works for the first-order logic without equalities
- To incorporate equalities we need an additional inference rule
- **Demodulation rule**
  \[ \sigma = \text{UNIFY}(z, t_1) \neq \text{fail} \]
  where \( z \) occurs in \( \phi_i \)
  \[ \phi_1 \lor \phi_2 \lor \ldots \lor \phi_k, \quad t_1 = t_2 \]
  \[ \text{SUB}(\text{SUBST}(\sigma, t_1), \text{SUBST}(\sigma, t_2), \phi_1 \lor \phi_2 \lor \ldots \lor \phi_k) \]
- **Example:** \[ P(f(a)), f(x) = x \]
  \[ P(a) \]
- **Paramodulation rule:** more powerful
- Resolution+paramodulation give a refutation-complete proof theory for FOL
Sentences in Horn normal form

• Horn normal form (HNF) in the propositional logic
  – a special type of a clause with at most one positive literal

\[(A \lor \neg B) \land (\neg A \lor \neg C \lor D)\]

Typically written as: \((B \Rightarrow A) \land ((A \land C) \Rightarrow D)\)

• A clause with one literal, e.g. \(A\), is also called a fact
• A clause representing an implication (with a conjunction of positive literals in antecedent and one positive literal in consequent), is also called a rule

• Resolution rule and modus ponens:
  – Both are complete inference rule for unit inferences for KBs in the Horn normal form.
  – Recall: Not all KBs are convertible to HNF !!!

Horn normal form in FOL

• First-order logic (FOL)
  – adds variables and quantifiers, works with terms, predicates
• HNF in FOL: primitive sentences (propositions) are formed by predicates
• Generalized modus ponens rule:

\[\sigma = a\ \text{substitution}\ ,\ \forall i\ SUBST(\sigma, \phi_i^') = SUBST(\sigma, \phi_i)\]

\[\phi_1^', \phi_2^', \ldots, \phi_n^',\quad \phi_1 \land \phi_2 \land \ldots \land \phi_n \Rightarrow \tau\]

\[\text{SUBST}(\sigma, \tau)\]

Generalized resolution and generalized modus ponens:
  – is complete for unit inferences for the KBs in HN;
  – Not all first-order logic sentences can be expressed in HNF

M. Hauskrecht
Forward and backward chaining

Two inference procedures based on modus ponens for Horn KBs:

- **Forward chaining**
  
  **Idea:** Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.
  
  **Typical usage:** If we want to infer all sentences entailed by the existing KB.

- **Backward chaining (goal reduction)**
  
  **Idea:** To prove the fact that appears in the conclusion of a rule prove the premises of the rule. Continue recursively.
  
  **Typical usage:** If we want to prove that the target (goal) sentence $\alpha$ is entailed by the existing KB.

Both procedures are complete for KBs in Horn form !!!

Forward chaining example

- **Forward chaining**
  
  **Idea:** Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied

  Assume the KB with the following rules:

<table>
<thead>
<tr>
<th>KB:</th>
<th>R1</th>
<th>Steamboat ($x$) $\land$ Sailboat ($y$) $\Rightarrow$ Faster ($x, y$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>$\quad$ Sailboat ($y$) $\land$ RowBoat ($z$) $\Rightarrow$ Faster ($y, z$)</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>$\quad$ Faster ($x, y$) $\land$ Faster ($y, z$) $\Rightarrow$ Faster ($x, z$)</td>
<td></td>
</tr>
</tbody>
</table>

| F1: Steamboat (Titanic) |
| F2: Sailboat (Mistral) |
| F3: RowBoat (PondArrow) |

Theorem: Faster (Titanic, PondArrow) ?
Forward chaining example

KB:

R1:  \( \text{Steamboat}(x) \land \text{Sailboat}(y) \Rightarrow \text{Faster}(x, y) \)
R2:  \( \text{Sailboat}(y) \land \text{RowBoat}(z) \Rightarrow \text{Faster}(y, z) \)
R3:  \( \text{Faster}(x, y) \land \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z) \)

F1:  \( \text{Steamboat}(\text{Titanic}) \)
F2:  \( \text{Sailboat}(\text{Mistral}) \)
F3:  \( \text{RowBoat}(\text{PondArrow}) \)

Rule R1 is satisfied:

F4:  \( \text{Faster}(\text{Titanic}, \text{Mistral}) \)
Forward chaining example

KB:
R1: \( \text{Steamboat}(x) \land \text{Sailboat}(y) \Rightarrow \text{Faster}(x, y) \)
R2: \( \text{Sailboat}(y) \land \text{RowBoat}(z) \Rightarrow \text{Faster}(y, z) \)
R3: \( \text{Faster}(x, y) \land \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z) \)

F1: \( \text{Steamboat}(\text{Titanic}) \)
F2: \( \text{Sailboat}(\text{Mistral}) \)
F3: \( \text{RowBoat}(\text{PondArrow}) \)

Rule R1 is satisfied:
F4: \( \text{Faster}(\text{Titanic}, \text{Mistral}) \)

Rule R2 is satisfied:
F5: \( \text{Faster}(\text{Mistral}, \text{PondArrow}) \)

----------

Forward chaining example

KB:
R1: \( \text{Steamboat}(x) \land \text{Sailboat}(y) \Rightarrow \text{Faster}(x, y) \)
R2: \( \text{Sailboat}(y) \land \text{RowBoat}(z) \Rightarrow \text{Faster}(y, z) \)
R3: \( \text{Faster}(x, y) \land \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z) \)

F1: \( \text{Steamboat}(\text{Titanic}) \)
F2: \( \text{Sailboat}(\text{Mistral}) \)
F3: \( \text{RowBoat}(\text{PondArrow}) \)

Rule R1 is satisfied:
F4: \( \text{Faster}(\text{Titanic}, \text{Mistral}) \)

Rule R2 is satisfied:
F5: \( \text{Faster}(\text{Mistral}, \text{PondArrow}) \)

Rule R3 is satisfied:
F6: \( \text{Faster}(\text{Titanic}, \text{PondArrow}) \)
Backward chaining example

- **Backward chaining (goal reduction)**
  
  **Idea:** To prove the fact that appears in the conclusion of a rule prove the antecedents (if part) of the rule & repeat recursively.

**KB:**

R1: \(\text{Steamboat}(x) \land \text{Sailboat}(y) \Rightarrow \text{Faster}(x, y)\)

R2: \(\text{Sailboat}(y) \land \text{RowBoat}(z) \Rightarrow \text{Faster}(y, z)\)

R3: \(\text{Faster}(x, y) \land \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z)\)

**F1:** \(\text{Steamboat}(\text{Titanic})\)

**F2:** \(\text{Sailboat}(\text{Mistral})\)

**F3:** \(\text{RowBoat}(\text{PondArrow})\)

**Theorem:** \(\text{Faster}(\text{Titanic}, \text{PondArrow})\)

---

**Backward chaining example**

- **F1:** \(\text{Steamboat}(\text{Titanic})\)
- **F2:** \(\text{Sailboat}(\text{Mistral})\)
- **F3:** \(\text{RowBoat}(\text{PondArrow})\)

\(\text{Steamboat}(x) \land \text{Sailboat}(y) \Rightarrow \text{Faster}(x, y)\)

\(\text{Faster}(\text{Titanic}, \text{PondArrow})\)

\(\{x / \text{Titanic}, y / \text{PondArrow}\}\)
**Backward chaining example**

\[
\text{Faster(Titanic, PondArrow)}
\]

\[
\text{Steamboat(Titanic)}: \text{R1}
\]

\[
\text{Sailboat(Titanic)}: \text{R2}
\]

\[
\text{Sailboat(PondArrow)}: \text{R3}
\]

\[
\text{RowBoat(PondArrow)}
\]

\[
\text{Faster}(y, z) \implies \text{Faster}(y, z) \implies \text{Faster(Titanic, PondArrow)}
\]

\[
\{y / \text{Titanic}, z / \text{PondArrow}\}
\]

**Backward chaining example**

\[
\text{Faster(x, y)} \land \text{Faster(y, z)} \implies \text{Faster(x, z)}
\]

\[
\text{Faster}(x, y, z) \implies \text{Faster}(x, z) \implies \text{Faster(Titanic, PondArrow)}
\]

\[
\{x / \text{Titanic}, z / \text{PondArrow}\}
\]
Backward chaining example

F1: Steamboat(Titanic)
F2: Sailboat(Mistral)
F3: RowBoat(PondArrow)

Steamboat(Titanic)

R1

Faster(Titanic, PondArrow)

Sailboat(PondArrow)

R2

RowBoat(PondArrow)

R3

Faster(Titanic, PondArrow)

y

Faster(Titanic, Faster)

y

M. Hauskrecht

Backward chaining example

y must be bound to the same term

Faster(Mistral, PondArrow)

Sailboat(PondArrow)

R1

RowBoat(PondArrow)

R2

Faster(Titanic, Mistral)

R3

Steamboat(Titanic)

Sailboat(Mistral)

M. Hauskrecht
Backward chaining example

- Faster(Titanic, PondArrow)
- Steamboat(Titanic)
- Sailboat(PondArrow)
- RowBoat(PondArrow)
- Faster(Titanic, Mistral)
- MistralTitanicFaster
- Sailboat(Mistral)
- RowBoat(PondArrow)
- Faster(Mistral, PondArrow)
- TitanicSteamboat
- R1
- R2
- R3

Backward chaining

- Faster(Titanic, PondArrow)
- Steamboat(Titanic)
- Sailboat(PondArrow)
- RowBoat(PondArrow)
- Faster(Titanic, Mistral)
- MistralTitanicFaster
- Sailboat(Mistral)
- RowBoat(PondArrow)
- Faster(Mistral, PondArrow)
- TitanicSteamboat
- R1
- R2
- R3