Inference in first-order logic

Chapter 9

Outline

• Reducing first-order inference to propositional inference
• Unification
• Generalized Modus Ponens
• Forward chaining
• Backward chaining
• Resolution
Universal instantiation (UI)

• Every instantiation of a universally quantified sentence is entailed by it:

\[
\forall \nu \alpha \\
\text{Subst}([\nu/g], \alpha)
\]

for any variable \(\nu\) and ground term \(g\)

• E.g., \(\forall x \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)\) yields:

  \(\text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})\)
  \(\text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})\)
  \(\text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))\)


Existential instantiation (EI)

• For any sentence \(\alpha\), variable \(\nu\), and constant symbol \(k\) that does not appear elsewhere in the knowledge base:

\[
\exists \nu \alpha \\
\text{Subst}([\nu/k], \alpha)
\]

• E.g., \(\exists x \text{Crown}(x) \land \text{OnHead}(x, \text{John})\) yields:

\(\text{Crown}(C_1) \land \text{OnHead}(C_1, \text{John})\)

provided \(C_1\) is a new constant symbol, called a Skolem constant
Reduction to propositional inference

Suppose the KB contains just the following:

\[ \forall x \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]
\[ \text{King}(\text{John}) \]
\[ \text{Greedy}(\text{John}) \]
\[ \text{Brother}(\text{Richard}, \text{John}) \]

- Instantiating the universal sentence in all possible ways, we have:
  \[ \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \]
  \[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \]
  \[ \text{King}(\text{John}) \]
  \[ \text{Greedy}(\text{John}) \]
  \[ \text{Brother}(\text{Richard}, \text{John}) \]

- The new KB is propositionalized: proposition symbols are
  \[ \text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Evil}(\text{John}), \text{King}(\text{Richard}), \text{etc.} \]

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.

  E.g., from:

  \[ \forall x \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]
  \[ \text{King}(\text{John}) \]
  \[ \forall y \text{Greedy}(y) \]
  \[ \text{Brother}(\text{Richard}, \text{John}) \]

- It seems obvious that \( \text{Evil}(\text{John}) \), but propositionalization produces lots of facts such as \( \text{Greedy}(\text{Richard}) \) that are irrelevant

- With \( p \)-ary predicates and \( n \) constants, there are \( p \cdot n^k \) instantiations.
Unification

• The UNIFY algorithm takes two sentence and returns a unifier for them if one exists
  – UNIFY(p,q) = θ where SUBST(θ, p) = SUBST(θ, q)

• Examples:
  – UNIFY(Knows(John, x), Knows(John, Jane)) = { x / Jane }
  – UNIFY(Knows(John, x), Knows(y, Bill)) = { x / Bill, y / John }
  – UNIFY(Knows(John, x), Knows(y, Mother(y))) = { y / John, x/Mother(John) }
  – UNIFY(Knows(John, x), Knows(x, Elizabeth)) = fail

• What about
  – UNIFY(Knows(John, x), Knows(y, z))
  – Could be { y / John, x / z} or { y / John, x / John, z / John} …
  – For every pair of unifiable expressions there is a single most general unifier (MGU) that is unique up to renaming of variables

The unification algorithm

function UNIFY(x, y, θ) returns a substitution to make x and y identical
inputs: x, a variable, constant, list, or compound
       y, a variable, constant, list, or compound
       θ, the substitution built up so far
if θ = failure then return failure
else if x = y then return θ
else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFY(ARGS[x], ARG[S[y], UNIFY(OP[x], OP[y], θ)]
else if LIST?(z) and LIST?(y) then
    return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
else return failure
The unification algorithm

\begin{function}
  \textsc{Unify-Var}(\text{var}, x, \theta) \text{ returns a substitution}
  \begin{align*}
    \text{inputs:} & \quad \text{var, a variable} \\
    & \quad x, \text{ any expression} \\
    & \quad \theta, \text{ the substitution built up so far} \\
    \text{if} \{\text{var}/\text{val}\} \in \theta \text{ then return \text{Unify}(\text{val}, x, \theta)} \\
    \text{else if} \{x/\text{val}\} \in \theta \text{ then return \text{Unify}(\text{var}, \text{val}, \theta)} \\
    \text{else if} \text{Occur-Check}(\text{var}, x) \text{ then return failure} \\
    \text{else return add }\{\text{var}/x\} \text{ to } \theta
  \end{align*}
\end{function}

Generalized Modus Ponens (GMP)

(p₁ \land p₂ \land \cdots \land pₙ \Rightarrow q)_{\theta}

\begin{align*}
  p₁' & \text{ is } \text{King}(\text{John}) \\
  p₂' & \text{ is } \text{Greedy}(\text{y}) \\
  \theta & \text{ is } \{x/\text{John}, y/\text{John}\} \\
  q' & \text{ is } \text{Evil}(\text{John})
\end{align*}

- GMP used with KB of definite clauses (exactly one positive literal)
- All variables assumed universally quantified
Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

- Prove that Col. West is a criminal

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:

\[ \text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x,y,z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( \exists x \) Owns(Nono,x) \( \wedge \) Missile(x):

\[ \text{Owns}(\text{Nono},M_1) \wedge \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West

\[ \text{Missile}(x) \wedge \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono}) \]

Missiles are weapons:

\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

An enemy of America counts as "hostile":

\[ \text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x) \]

West, who is American ...

\[ \text{American}(\text{West}) \]

The country Nono, an enemy of America ...

\[ \text{Enemy}(\text{Nono},\text{America}) \]
Forward Chaining Algorithm

Given new predicate P:
Add P to KB
For all rules in the KB, if LHS is true then
- Unify variables
- Instantiate RHS
- Repeat with RHS

Forward chaining algorithm

```
function FOL-FC-Ask(KB, α) returns a substitution or false
repeat until new is empty
    new ← ∅
    for each sentence r in KB do
        (p₁ ∧ ... ∧ pₙ ⇒ q) ← STANDARDIZE-Apart(r)
        for each θ such that (p₁ ∧ ... ∧ pₙ)θ = (p₁' ∧ ... ∧ pₙ')θ
            for some p₁', ..., pₙ' in KB
                q' ← SUBST(θ, q)
                if q' is not a renaming of a sentence already in KB or new then do
                    add q' to new
                    φ ← UNIFY(q', α)
                    if φ is not fail then return φ
    add new to KB
return false
```
Forward chaining proof
Forward chaining proof

Efficiency of forward chaining

Incremental forward chaining: no need to match a rule on iteration \( k \) if a premise wasn't added on iteration \( k-1 \)

\[ \Rightarrow \] match each rule whose premise contains a newly added positive literal

Matching itself can be expensive:
Database indexing allows \( O(1) \) retrieval of known facts

- e.g., query \( \text{Missile}(x) \) retrieves \( \text{Missile}(M_1) \)

Forward chaining is widely used in deductive databases
Hard matching example

\[ \text{Diff}(wa,nt) \land \text{Diff}(wa,sa) \land \text{Diff}(nt,q) \land \]
\[ \text{Diff}(nt,sa) \land \text{Diff}(q,nsw) \land \text{Diff}(q,sa) \land \]
\[ \text{Diff}(nsw,v) \land \text{Diff}(nsw,sa) \land \text{Diff}(v,sa) \Rightarrow \]
\[ \text{Colorable()} \]

\[ \text{Diff}(\text{Red,Blue}) \land \text{Diff}(\text{Red,Green}) \]
\[ \text{Diff}(\text{Green,Red}) \land \text{Diff}(\text{Green,Blue}) \]
\[ \text{Diff}(\text{Blue,Red}) \land \text{Diff}(\text{Blue,Green}) \]

- \text{Colorable()} is inferred iff the CSP has a solution
- CSPs include 3SAT as a special case, hence matching is NP-hard

Backwards Chaining

Given a knowledge base in Horn Clause Format:

\[ F_1 \rightarrow F_2 \]
\[ F_3 \land F_5 \rightarrow F_4 \]
\[ F_2 \land F_4 \rightarrow F_6 \]
\[ F_{10} \ldots F_{11} \rightarrow F_{12} \]

Given predicate P to prove or ask:
- If P is known to be True in the KB, return true
- Find clause with P on the RHS
- Repeat with every clause on the LHS unifying any variables
- If all clauses true, return true, else return false
Backward chaining algorithm

function FOL-BC-Ask(KB, goals, \( \theta \)) returns a set of substitutions
inputs: KB, a knowledge base
goals, a list of conjuncts forming a query
\( \theta \), the current substitution, initially the empty substitution \( \{ \} \)
local variables: ans, a set of substitutions, initially empty
if goals is empty then return \( \{ \theta \} \)
\( q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals)) \)
for each \( r \) in KB where \text{STANDARDIZE-APART}(r) = (p_1 \land \ldots \land p_n \Rightarrow q)
and \( \theta' \leftarrow \text{UNIFY}(q, q') \) succeeds
\( ans \leftarrow \text{FOL-BC-Ask}(KB, [p_1, \ldots, p_n] \cup \text{REST}(goals), \text{COMPOSE}(\theta, \theta')) \cup ans \)
return ans

\( \text{SUBST(COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p)) \)

Backward chaining example

\( \square \text{normal Weir} / \)
Backward chaining example

Diagram:

1. Criminal\(\text{Wear}\)
2. (to Wear)
3. American\(\text{West}\)
4. Weapon\(\text{W}\)
5. Sell\(\text{W}, \text{Y}\)
6. Miss\(\text{W}\)
Backward chaining example
Backward chaining example
Backward chaining example

Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  - fix using caching of previous results (extra space)
- Widely used for logic programming
Resolution: brief summary

• Full first-order version:
  \[ l_1 \lor \cdots \lor l_k, \quad m_1 \lor \cdots \lor m_l \]
  
  \[
  (l_1 \lor \cdots \lor l_{i-1} \lor l_i \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_j \lor \cdots \lor m_l) \theta
  \]

  where \( \text{Unify}(l_i, \neg m_j) = \theta \).

• The two clauses are assumed to be standardized apart so that they share no variables.

• For example,
  \[
  \neg \text{Rich}(x) \lor \text{Unhappy}(x) \\
  \text{Rich}(\text{Ken})
  \]

  with \( \theta = \{x/\text{Ken}\} \)

• Apply resolution steps to \( \text{CNF}(\text{KB} \land \neg \alpha) \); complete for FOL

Conversion to CNF

• Everyone who loves all animals is loved by someone:
  \[
  \forall x [\forall y \text{Animal}(y) \Rightarrow \text{Loves}(x,y)] \Rightarrow [\exists y \text{Loves}(y,x)]
  \]

• 1. Eliminate biconditionals and implications
  \[
  \forall x [\neg \forall y \neg \text{Animal}(y) \lor \text{Loves}(x,y)] \lor [\exists y \text{Loves}(y,x)]
  \]

• 2. Move \( \neg \) inwards:
  \[
  \forall x \exists y \neg \text{Animal}(y) \lor \text{Loves}(x,y) \lor [\exists y \text{Loves}(y,x)]
  \]

  \[
  \forall x \exists y \neg \text{Animal}(y) \land \neg \text{Loves}(x,y) \lor [\exists y \text{Loves}(y,x)]
  \]

  \[
  \forall x \exists y \text{Animal}(y) \land \neg \text{Loves}(x,y) \lor [\exists y \text{Loves}(y,x)]
  \]

  \[
  \forall x \exists y \text{Animal}(y) \land \neg \text{Loves}(x,y) \lor [\exists y \text{Loves}(y,x)]
  \]
Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one
\[ \forall x \left[ \exists y \text{Animal}(y) \land \neg \text{Loves}(x,y) \right] \lor \left[ \exists z \text{Loves}(z,x) \right] \]

4. Skolemize: a more general form of existential instantiation.

Each existential variable is replaced by a Skolem function of the enclosing
universally quantified variables:
\[ \forall x \left[ \text{Animal}(F(x)) \land \neg \text{Loves}(x,F(x)) \right] \lor \text{Loves}(G(x),x) \]

5. Drop universal quantifiers:
\[ \left[ \text{Animal}(F(x)) \land \neg \text{Loves}(x,F(x)) \right] \lor \text{Loves}(G(x),x) \]

6. Distribute \( \lor \) over \( \land \):
\[ \left[ \text{Animal}(F(x)) \lor \text{Loves}(G(x),x) \right] \land \left[ \neg \text{Loves}(x,F(x)) \lor \text{Loves}(G(x),x) \right] \]

Resolution proof: definite clauses