Inference in first-order logic

Chapter 9

Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward and backward chaining
- Logic programming
- Resolution

Universal instantiation (UI)

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

$$\forall v \ a \ \Rightarrow \ Subst(v/g, a)$$

for any variable $v$ and ground term $g$

E.g., $\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$ yields

- $King(John) \land Greedy(John) \Rightarrow Evil(John)$
- $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$
- $King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John))$

Existential instantiation (EI)

For any sentence $\alpha$, variable $v$, and constant symbol $k$ that does not appear elsewhere in the knowledge base:

$$\exists v \ a \ \Rightarrow \ Subst(v/k, \alpha)$$

E.g., $\exists x \ Crown(x) \land OnHead(x, John)$ yields

- $Crown(C) \land OnHead(C, John)$

provided $C$ is a new constant symbol, called a Skolem constant

Another example: from $\exists x \ d(x^y)/dy = x^y$ we obtain

- $d(e^y)/dy = e^y$

provided $e$ is a new constant symbol

A brief history of reasoning

- 450 B.C.: Stoics - propositional logic, inference (maybe)
- 322 B.C.: Aristotle - "syllogisms" (inference rules), quantifiers
- 1565: Cardano - probability theory (propositional logic + uncertainty)
- 1847: Boole - propositional logic (again)
- 1879: Frege - first-order logic
- 1922: Wittgenstein - proof by truth tables
- 1930: Gödel - complete algorithm for FOL
- 1930: Herbrand - complete algorithm for FOL (reduce to propositional)
- 1931: Gödel - $\neg$ complete algorithm for arithmetic
- 1960: Davis/Plutam - "practical" algorithm for propositional logic
- 1960: Robinson - "practical" algorithm for FOL—resolution

Existential instantiation contd.

UI can be applied several times to add new sentences; the new KB is logically equivalent to the old

EI can be applied once to replace the existential sentence; the new KB is not equivalent to the old, but is satisfiable iff the old KB was satisfiable
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

\[ \text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Evil}(\text{John}), \text{King}(\text{Richard}) \text{ etc.} \]

Reduction contd.

Claim: a ground sentence is entailed by new KB iff entailed by original KB
Claim: every FOL KB can be propositionalized so as to preserve entailment

Idea: propositionalize KB and query, apply resolution, return result

Problem: with function symbols, there are infinitely many ground terms, e.g., \( \text{Father}(\text{Father}(\text{Father}(\text{John}))) \)

Theorem: Herbrand (1930). If a sentence \( \alpha \) is entailed by an FOL KB, it is entailed by a finite subset of the propositional KB

Idea: For \( n = 0 \) to \( \infty \) do
- create a propositional KB by instantiating with depth-\( n \) terms
- see if \( \alpha \) is entailed by this KB

Problem: works if \( \alpha \) is entailed, loops if \( \alpha \) is not entailed

Theorem: Turing (1936), Church (1936), entailment in FOL is semidecidable

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

With function symbols, it gets much much worse!

Unification

We can get the inference immediately if we can find a substitution \( \theta \) such that \( \text{King}(x) \) and \( \text{Greedy}(x) \) match \( \text{King}(\text{John}) \) and \( \text{Greedy}(\text{y}) \)

\( \theta = \{ x/\text{John}, y/\text{John} \} \) works

\[ \text{UNIFY}(\alpha, \beta) = \emptyset \text{ if } c\theta = \emptyset \theta \]

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Knows}(\text{John}, x)</td>
<td>\text{Knows}(\text{John}, \text{Jane})</td>
<td>{ x/\text{Jane} }</td>
</tr>
<tr>
<td>\text{Knows}(\text{John}, y)</td>
<td>\text{Knows}(\text{y}, \text{OJ})</td>
<td>{ x/\text{OJ}, y/\text{John} }</td>
</tr>
<tr>
<td>\text{Knows}(\text{John}, y)</td>
<td>\text{Knows}(\text{y}, \text{Mother}(y))</td>
<td>{ x/\text{OJ}, y/\text{John} }</td>
</tr>
<tr>
<td>\text{Knows}(\text{John}, y)</td>
<td>\text{Knows}(x, \text{OJ})</td>
<td>{ x/\text{OJ}, y/\text{John} }</td>
</tr>
</tbody>
</table>
We can get the inference immediately if we can find a substitution \( \theta \) such that \( \text{King}(x) \) and \( \text{Greedy}(x) \) match \( \text{King}(\text{John}) \) and \( \text{Greedy}(y) \)

\[ \theta = \{ x/\text{John}, y/\text{John} \} \] works

\[ \text{UNIFY}(\alpha, \beta) = \theta \] if \( a\theta = b\theta \)

| \( p \) | \( q \) | \( \theta \) |
|----------------------------------|
| \text{Knows}(\text{John}, x) | \text{Knows}(\text{John}, \text{Jane}) | \{x/\text{Jane}\} |
| \text{Knows}(\text{John}, x) | \text{Knows}(y, \text{OJ}) | \{x/\text{OJ}, y/\text{John}\} |
| \text{Knows}(\text{John}, x) | \text{Knows}(y, \text{Mother}(y)) | \{y/\text{John}, x/\text{Mother}(\text{John})\} |
| \text{Knows}(\text{John}, x) | \text{Knows}(x, \text{OJ}) | fail |

Standardizing apart eliminates overlap of variables, e.g., \( \text{Knows}(z_7, \text{OJ}) \)

-----

\textbf{Soundness of GMP}

Need to show that

\[ p_1'^i, \ldots, p_n'^i; (p_1 \land \ldots \land p_n \Rightarrow q) \vdash q\theta \]

provided that \( p_i\theta = p_i\theta \) for all \( i \)

Lemma: For any definite clause \( p \), we have \( p \models p\theta \) by UI

1. \( (p_1 \land \ldots \land p_n \Rightarrow q) \models (p_1 \land \ldots \land p_n \Rightarrow q\theta) = (p_1 \land \ldots \land p_n \Rightarrow q\theta) \)

2. \( p_1'^i, \ldots, p_n'^i \models p_1'^i \land \ldots \land p_n'^i = p_1\theta \land \ldots \land p_n\theta \)

3. From 1 and 2, \( q\theta \) follows by ordinary Modus Ponens

-----

\textbf{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

-----

\textbf{Example knowledge base contd.}

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

\[ p_1'^i, \ldots, p_n'^i; (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]

\where \( p_i\theta = p_i\theta \) for all \( i \)

\( p_i'^i \) is \( \text{King}(x) \)

\( p_i \) is \( \text{King}(x) \)

\( p_2'^i \) is \( \text{Greedy}(y) \)

\( p_2 \) is \( \text{Greedy}(y) \)

\( \theta = \{ x/\text{John}, y/\text{John} \} \)

\( q \) is \( \text{Evil}(x) \)

\( q\theta \) is \( \text{Evil}(x) \)

GMP used with KB of definite clauses (exactly one positive literal)

All variables assumed universally quantified
Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( 3 \ x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \): 
\[ \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

Missiles are weapons:
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

An enemy of America counts as "hostile":

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( 3 \ x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

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 ...
\[ \text{Enemy}(\text{Nono}, \text{America}) \]

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( 3 \ x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

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 ...
\[ \text{Enemy}(\text{Nono}, \text{America}) \]

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( 3 \ x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

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 ...
\[ \text{Enemy}(\text{Nono}, \text{America}) \]

Example knowledge base contd.

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( 3 \ x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

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

function FOL-FC-Ask(KB, φ) returns a substitution or false
repeat until new is empty
  new = {} 
  for each sentence r in KB do
    (p_1 \ldots p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-APART}(r)
    for each θ such that (p_1 \land \ldots \land p_n)\theta = (p'_1 \land \ldots \land p'_{n'})
    for some p_1\ldots p_n in KB
      q' \leftarrow \text{SUBST}(θ, q)
      if q' is not a renaming of a sentence already in KB or new then do
        add q' to new
        φ \leftarrow \text{UNIFY}(q', φ)
        if φ is not fail then return φ
    add new to KB
  return false
Forward chaining proof

**Properties of forward chaining**

Sound and complete for first-order definite clauses
(proof similar to propositional proof)

\[ \text{Datalog} = \text{first-order definite clauses + no functions} \text{ (e.g., crime KB)} \]

FC terminates for Datalog in poly iterations: at most \( p \cdot n^k \) literals

May not terminate in general if \( \alpha \) is not entailed

This is unavoidable: entailment with definite clauses is semidecidable

---

Efficiency of forward chaining

Simple observation: no need to match a rule on iteration \( k \)
if a premise wasn’t added on iteration \( k - 1 \)

\[ \Rightarrow \text{match each rule whose premise contains a newly added literal} \]

Matching itself can be expensive

Database indexing allows \( O(1) \) retrieval of known facts

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

Matching conjunctive premises against known facts is NP-hard

Forward chaining is widely used in deductive databases

---

Hard matching example

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

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

\( \text{Colorable()} \) is inferred iff the CSP has a solution

CSPs include 3SAT as a special case, hence matching is NP-hard
Backward chaining algorithm

```
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions

inputs: KB, a knowledge base

goals, a list of conjuncts forming a query (θ already applied)

θ, the current substitution, initially the empty substitution { }

local variables: answers, a set of substitutions, initially empty

if goals is empty then return {θ}

φ' ← Substitute(θ, First(goals))

for each sentence r in KB where Standardize-Apart(r) = \{ p_1 ∧ \ldots ∧ p_n \implies q \}

and θ' ← Unify(φ, φ') succeeds

e new_goals ← \{ p_1, \ldots, p_n \} Rest(goals)

answers ← FOL-BC-Ask(KB, new_goals, Compose(θ', θ)) ⋃ answers

return answers
```

Backward chaining example

1. **Criminal(West)**
2. **Weapon(y)**
3. **Sells(x,y,z)**
4. **Hostile(z)**

Setting: \{x/West\} { }

Backward chaining example

1. **Criminal(West)**
2. **Weapon(y)**
3. **Sells(x,y,z)**
4. **Hostile(z)**

Setting: \{x/West, y/M1\}

Backward chaining example

1. **Criminal(West)**
2. **Weapon(y)**
3. **Sells(x,y,z)**
4. **Hostile(z)**

Setting: \{x/West\} { }
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 (without improvements!) for logic programming

Backward chaining example

Sound bite: computation as inference on logical KBs

Logic programming

Logic programming

Properties of backward chaining

Depth-first search from a start state X:

Prolog systems

Prolog examples
Resolution: brief summary

Full first-order version:
\[
\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_l
\]
\[
(\ell_1 \lor \cdots \lor \ell_{k-1} \lor \ell_k \lor m_1 \lor \cdots \lor m_{l-1} \lor m_l \lor \cdots \lor m_l)\theta
\]

where \( UNIFY(\ell_i, \neg m_j) = \emptyset \).

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

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

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

Conversion to CNF

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

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

2. Move \( \neg \) inwards:
\[
\neg \forall x, \ p \equiv \exists x, \ \neg p, \quad \neg \exists x, \ p \equiv \forall x \ \neg p.
\]
\[
\forall x \left[ \exists y \ (\neg (\neg \text{Animal}(y) \lor \text{Loves}(x, y)) \lor \exists y \ (\text{Loves}(y, x))) \right]
\]

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

Conversion to CNF contd.

4. Skolemize: a more general form of existential instantiation.
Each existenial 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:
\[
[\text{Animal}(F(x)) \land \neg \text{Loves}(x, F(x))] \lor \text{Loves}(G(x), x)
\]

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