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

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 \( \exists \) complete algorithm for FOL
1930 Herbrand complete algorithm for FOL (reduce to propositional)
1931 Gödel \( \neg \exists \) complete algorithm for arithmetic
1960 Davis/Putnam "practical" algorithm for propositional logic
1965 Robinson "practical" algorithm for FOL—resolution

Universal instantiation (UI)

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

\[
\forall v \alpha \\
\text{Subst}\{\{v/g\}, \alpha\}
\]

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

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

\[
\begin{align*}
\text{King}(John) \land \text{Greedy}(John) & \Rightarrow \text{Evil}(John) \\
\text{King}(Richard) \land \text{Greedy}(Richard) & \Rightarrow \text{Evil}(Richard) \\
\text{King}(\text{Father}(John)) \land \text{Greedy}(\text{Father}(John)) & \Rightarrow \text{Evil}(\text{Father}(John))
\end{align*}
\]
Existential instantiation (EI)

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

$$\exists v \: \alpha$$

Subst($\{v/k\}$,$\alpha$)

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

$Crown (C_1) \land OnHead (C_1, John)$

provided $C_1$ 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

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:

$$\forall x \: King(x) \land Greedy(x) \Rightarrow Evil(x)$$

King(John)

Greedy(John)

Brother(Richard, John)

Instantiating the universal sentence in all possible ways, we have

King(John) \land Greedy(John) \Rightarrow Evil(John)

King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)

King(John)

Greedy(John)

Brother(Richard, John)

The new KB is **propositionalyzed**: proposition symbols are

King(John), Greedy(John), Evil(John), King(Richard) etc.

Reduction contd.

Claim: a ground sentence $\alpha$ is entailed by new KB iff entailed by original KB

Claim: every FOL KB can be propositionalyzed so as to preserve entailment

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

Problem: with function symbols, there are infinitely many ground terms, e.g., Father(Father(Father(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
\[
\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)
\]
\[
King(\text{John})
\]
\[
\forall y \ Greedy(y)
\]
\[
\text{Brother} (\text{Richard, John})
\]

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

With \( p \) \( k \)-ary predicates and \( n \) 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 \( \text{Greedy}(y) \)

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

\( \text{Unify}(\alpha, \beta) = \theta \) if \( \alpha\theta = \beta\theta \)

\[
\begin{array}{lll}
 p & q & \theta \\
\text{Knows}(\text{John}, x) & \text{Knows}(\text{John}, \text{Jane}) & \{ x/\text{Jane} \} \\
\text{Knows}(\text{John}, x) & \text{Knows}(y, \text{OJ}) & \{ x/OJ, y/\text{John} \} \\
\text{Knows}(\text{John}, x) & \text{Knows}(y, \text{Mother}(y)) & \\
\text{Knows}(\text{John}, x) & \text{Knows}(x, \text{OJ}) & \\
\end{array}
\]
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 $\text{Greedy}(y)$

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

$\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha \theta = \beta \theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, x)</td>
<td>Knows(John, Jane)</td>
<td>${x/Jane}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, OJ)</td>
<td>${x/OJ, y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td>${y/\text{John}, x/Mother(\text{John})}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, OJ)</td>
<td>${}$</td>
</tr>
</tbody>
</table>

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, OJ)$

Generalized Modus Ponens (GMP)

$\forall x \text{ King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

$\text{King}(\text{John})$

$\text{Greedy}(y)$

$\text{Brother}(\text{Richard}, \text{John})$

Given $p_1', p_2', \ldots, p_n'$, $(p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)$ where $p_i' \theta = p_i \theta$ for all $i$

$q \theta$ is $\text{Evil}(\text{John})$

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

Soundness of GMP

Need to show that

$p_1', \ldots, p_n' \vdash (p_1 \land \ldots \land p_n \Rightarrow q) \Rightarrow q \theta$

provided that $p_i' \theta = p_i \theta$ for all $i$

Lemma: For any definite clause $p$, we have $p \vdash p \theta$ by UI

1. $(p_1 \land \ldots \land p_n \Rightarrow q) \Rightarrow (p_1 \land \ldots \land p_n \Rightarrow q) \Rightarrow (p_1 \land \ldots \land p_n \Rightarrow q)$
2. $p_1', \ldots, p_n' \vdash p_1' \land \ldots \land p_n' \vdash p_1' \theta \land \ldots \land p_n' \theta$
3. From 1 and 2, $q \theta$ follows by ordinary Modus Ponens
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
Missiles are weapons:

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

An enemy of America counts as “hostile”:

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

West, who is American . . .

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

The country Nono, an enemy of America . . .

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

**Example knowledge base contd.**

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

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

Nono . . . has some missiles, i.e., \( \exists x \ \text{Owns} \left( \text{Nono}, x \right) \land \text{Missile} \left( x \right) \):

\[ \text{Owns} \left( \text{Nono}, M_1 \right) \land \text{Missile} \left( M_1 \right) \]

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

\[ \forall x \ \text{Missile} \left( x \right) \land \text{Owns} \left( \text{Nono}, x \right) \Rightarrow \text{Sells} \left( \text{West}, x, \text{Nono} \right) \]

Missiles are weapons:

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

An enemy of America counts as “hostile”:

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

**Example knowledge base contd.**

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

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

Nono . . . has some missiles, i.e., \( \exists x \ \text{Owns} \left( \text{Nono}, x \right) \land \text{Missile} \left( x \right) \):

\[ \text{Owns} \left( \text{Nono}, M_1 \right) \land \text{Missile} \left( M_1 \right) \]

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

\[ \forall x \ \text{Missile} \left( x \right) \land \text{Owns} \left( \text{Nono}, x \right) \Rightarrow \text{Sells} \left( \text{West}, x, \text{Nono} \right) \]

Missiles are weapons:

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

**Forward chaining algorithm**

function FOL-FC-Ask(KB,\( \alpha \)) returns a substitution or false

repeat until new is empty

new\( \leftarrow \{\}\)

for each sentence \( r \) in KB do

\( (p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow \text{Standardize-Apart} \left( r \right) \)

for each \( \theta \) such that \( (p_1 \land \ldots \land p_n)\theta = (p'_1 \land \ldots \land p'_m)\theta \)

for some \( p'_1, \ldots, p'_m \) in KB

\( q' \leftarrow \text{Subst} \left( \theta, q \right) \)

if \( q' \) is not a renaming of a sentence already in KB or new then do

add \( q' \) to new

\( \phi \leftarrow \text{Unify} \left( q', \alpha \right) \)

if \( \phi \) is not fail then return \( \phi \)

add new to KB

return false
**Forward chaining proof**

- Enemy(Nono, America)
- Owns(Nono, M1)
- Missile(M1)
- American(West)
- Weapon(M1)
- Sells(West, M1, Nono)
- Hostile(Nono)

**Properties of forward chaining**

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

Datalog = first-order definite clauses + no functions (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 $k - 1$  

⇒ 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 $Missile(x)$ retrieves $Missile(M_1)$.

Matching conjunctive premises against known facts is NP-hard.

Forward chaining is widely used in deductive databases.

---

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 ($\theta$ already applied)

$\theta$, the current substitution, initially the empty substitution $\{}$

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

if $goals$ is empty then return $\{\theta\}$

$q' \leftarrow \text{Subst}(\theta, \text{First}(goals))$

for each sentence $r$ in $KB$

where $\text{Standardize-Apart}(r) = (p_1 \land \ldots \land p_n \Rightarrow q)$

and $\theta' \leftarrow \text{Unify}(q, q')$ succeeds

$new Goals \leftarrow [p_1, \ldots, p_n | \text{Rest}(goals)]$

answers $\leftarrow$ FOL-BC-Ask($KB, new Goals, \text{Compose}(\theta', \theta)$) $\cup$ answers

return answers

---

Hard matching example

\[
\begin{align*}
\text{Diff}(wa, nt) & \land \text{Diff}(wa, sa) \land \\
\text{Diff}(nt, q) & \land \text{Diff}(nt, sa) \land \\
\text{Diff}(q, nsw) & \land \text{Diff}(q, sa) \land \\
\text{Diff}(nsw, v) & \land \text{Diff}(nsw, sa) \land \\
\text{Diff}(v, sa) & \Rightarrow \text{Colorable}() \\
\text{Diff}(\text{Red}, \text{Blue}) & \land \text{Diff}(\text{Red}, \text{Green}) \\
\text{Diff}(\text{Green}, \text{Red}) & \land \text{Diff}(\text{Green}, \text{Blue}) \\
\text{Diff}(\text{Blue}, \text{Red}) & \land \text{Diff}(\text{Blue}, \text{Green})
\end{align*}
\]

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

CSPs include 3SAT as a special case, hence matching is NP-hard.

---

Backward chaining example

\[
\text{Criminal}(\text{West})
\]
Backward chaining example

Criminal(West)
{x/West}

American(x)
Weapon(y)
Sells(x,y,z)
Hostile(z)

Backward chaining example

Criminal(West)
{x/West}

American(West)
Weapon(y)
Sells(x,y,z)
Hostile(z)

Missile(y)

Backward chaining example

Criminal(West)
{x/West}

American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)

Backward chaining example

Criminal(West)
{x/West, y/M1}

American(West)
{ }
Weapon(y)
Sells(x,y,z)
Hostile(z)

Missile(y)
{ y/M1 }
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

Logic programming

Sound bite: computation as inference on logical KBs

Logic programming   Ordinary programming
1. Identify problem  Identify problem
2. Assemble information Assemble information
3. Tea break Figure out solution
4. Encode information in KB Program solution
5. Encode problem instance as facts Encode problem instance as data
6. Ask queries Apply program to data
7. Find false facts Debug procedural errors

Should be easier to debug Capital(NewYork,US) than $x := x + 2$!
Prolog systems

Basis: backward chaining with Horn clauses + bells & whistles
Widely used in Europe, Japan (basis of 5th Generation project)
Compilation techniques ⇒ approaching a billion LIPS
Program = set of clauses = head :- literal₁, ... literalₙ.
    criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
Efficient unification by open coding
Efficient retrieval of matching clauses by direct linking
Depth-first, left-to-right backward chaining
Built-in predicates for arithmetic etc., e.g., X is Y*Z+3
Closed-world assumption ("negation as failure")
   e.g., given alive(X) :- not dead(X).
   alive(joe) succeeds if dead(joe) fails

Resolution: brief summary

Full first-order version:
\[ \ell₁ \lor \cdots \lor \ellₖ, \ m₁ \lor \cdots \lor mₙ \]
\[ (\ell₁ \lor \cdots \lor \ellᵢ-1 \lor \ellᵢ \lor \ellᵢ+1 \lor \cdots \lor \ellₖ \lor m₁ \lor \cdots \lor mᵢ-1 \lor mᵢ+1 \lor \cdots \lor mₙ)\theta \]
where \( \text{UNIFY}(\ellᵢ, \neg mᵢ) = \theta \).

For example,
\[ \neg\text{Rich}(x) \lor \text{Unhappy}(x) \]
\[ \text{Rich(Ken)} \]
\[ \frac{}{\text{Unhappy(Ken)}} \]
with \( \theta = \{x/\text{Ken}\} \)

Apply resolution steps to \( \text{CNF}(KB \land \neg α) \); 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: \( \forall x \ [\exists y \ \neg (-\text{Animal}(y) \lor \text{Loves}(x, y))] \lor [\exists y \ \text{Loves}(y, x)] \)
\[ \forall x \ [\exists y \ \neg \text{Animal}(y) \land \neg \text{Loves}(x, y)] \lor [\exists y \ \text{Loves}(y, x)] \]
\[ \forall x \ [\exists y \ \text{Animal}(y) \land \neg \text{Loves}(x, y)] \lor [\exists y \ \text{Loves}(y, x)] \]
Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

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

4. Skolemize: a more general form of existential instantiation.
   Each existential variable is replaced by a Skolem function
   of the enclosing universally quantified variables:

\[ \forall x \; [\text{Animal}(F(x)) \land \neg \text{Loves}(x, F(x))] \lor \text{Loves}(G(x), x) \]

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 \( \lor \):

\[ [\text{Animal}(F(x)) \lor \text{Loves}(G(x), x)] \land [\neg \text{Loves}(x, F(x)) \lor \text{Loves}(G(x), x)] \]

Resolution proof: definite clauses

[Diagram of logical expressions and clauses]