Title: Inference in First-Order Logic
AIMA: Chapter 9

Introduction to Artificial Intelligence
CSCE 476-876, Spring 2005
URL: `www.cse.unl.edu/~choueiry/S05-476-876`

Berthe Y. Choueiry (Shu-we-ri)
`choueiry@cse.unl.edu, (402)472-5444`

Outline

- Reducing first order inference to propositional inference: Universal Instantiation, Existential Instantiation, Skolemization, Generalized Modus Ponens
- Unification
- Inference mechanisms in First-Order Logic:
  - Forward chaining
  - Backward chaining
  - Resolution (and CNF)

Universal instantiation (UI)

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

$$\frac{\forall v \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

for any variable v and ground term g

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

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$

$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$

$$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$$

$$\vdots$$

Existential instantiation (EI)

For any sentence α, variable v, and constant symbol k
<u>that does not appear elsewhere</u> in the knowledge base:

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

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

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

provided C_1 is a new constant symbol, called a <u>Skolem constant</u>

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

UI and EI

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 (I)

$\forall x King(x) \land Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

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 propositionalized: proposition symbols are:
$King(John),\ Greedy(John),\ Evil(John), King(Richard)$ etc.

Reduction to propositional inference (II)

- 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., $Father(Father(Father(John)))$

Reduction to propositional inference (III)

- Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a <u>*finite*</u> subset of the propositional KB
- Idea: For $n = 0$ to ∞ do
  create a propositional KB by instantiating with depth-n terms
  see if α is entailed by this KB
- Problem: works if α is entailed, loops if α is not entailed
- Theorem: Turing (1936), Church (1936), entailment in FOL is <u>semidecidable</u>

Problems with propositionalization

Propositionalization generates lots of irrelevant sentences.
E.g., from

$\forall x King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$\forall y Greedy(y)$
$Brother(Richard, John)$

it seems obvious that $Evil(John)$, but propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant

With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations!

Unification

We can get the inference immediately if we can find a substitution θ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

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

Unify$(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$

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

Generalized Modus Ponens (GMP)

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta} \qquad \text{where } p_i{}'\theta = p_i\theta \text{ for all } i$$

$$\begin{array}{rl} p_1{}' \text{ is } King(John) & p_1 \text{ is } King(x) \\ p_2{}' \text{ is } Greedy(y) & p_2 \text{ is } Greedy(x) \\ \theta \text{ is } \{x/John, y/John\} & q \text{ is } Evil(x) \\ q\theta \text{ is } Evil(John) & \end{array}$$

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

... it is a crime for an American to sell weapons to hostile nations:
$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Example of KB (2)

Nono ... has some missiles, i.e., $\exists\, x\; Owns(Nono, x) \wedge Missile(x)$:
$Owns(Nono, M_1)$ and $Missile(M_1)$

... all of its missiles were sold to it by Colonel West
$\forall x Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Missiles are weapons:
$Missile(x) \Rightarrow Weapon(x)$

Example of KB (3)

An enemy of America counts as "hostile":
$Enemy(x, America) \Rightarrow Hostile(x)$

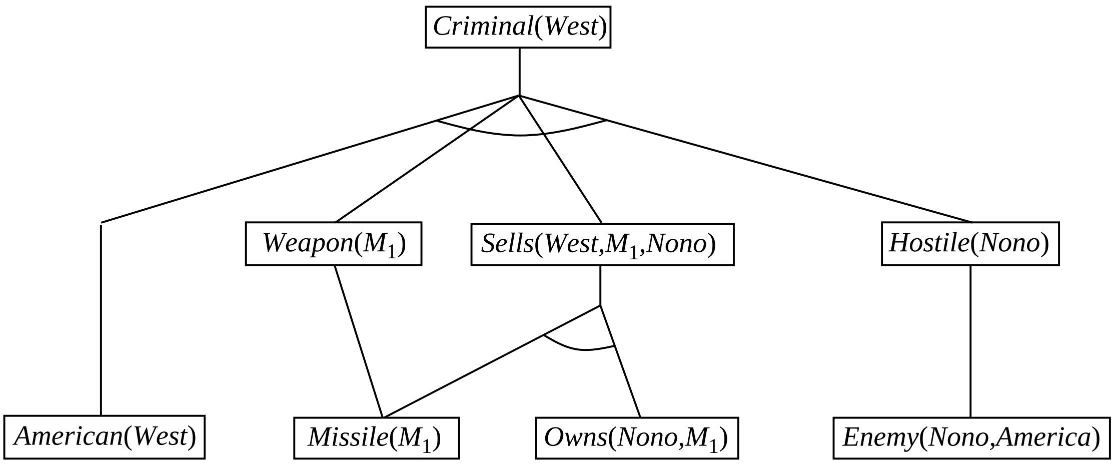
West, who is American ...
$American(West)$

The country Nono, an enemy of America ...
$Enemy(Nono, America)$

Forward chaining algorithm

<FOL-FC-Ask, Figure 9.3 page 282>

Forward chaining proof



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 α is not entailed
- This is unavoidable: entailment with definite clauses is semidecidable

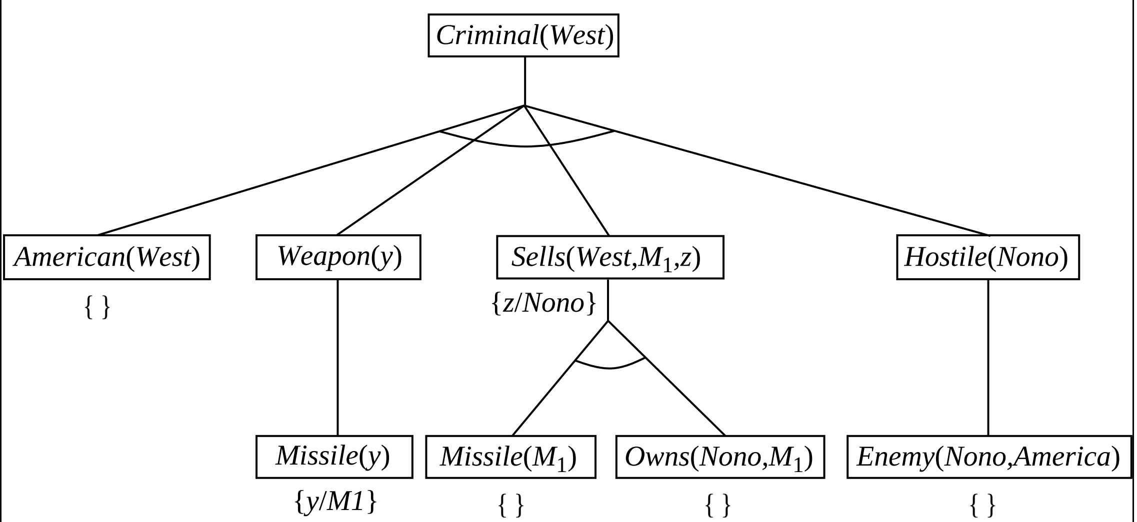
Efficiency of forward chaining

- Simple observation: no need to match a rule on iteration k if a premise wasn't added on iteration $k-1$
  $\Rightarrow$ match each rule whose premise contains a newly added literal
- Matching itself can be expensive
- Database indexing allows $O(1)$ retrieval of known facts
  l 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

<FOL-BC-Ask, Figure 9.6 page 288>

Backward chaining example



Properties of backward chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  $\Rightarrow$ fix by checking current goal against every goal on stack
- Inefficient due to repeated subgoals (both success and failure)
  $\Rightarrow$ fix using caching of previous results (extra space!)
- Widely used (without improvements!) for logic programming

Resolution: brief summary

Full first-order version:

$$\frac{l_1 \vee \cdots \vee l_k, \qquad m_1 \vee \cdots \vee m_n}{(l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where Unify$(l_i, \neg m_j) = \theta$.

For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

Apply resolution steps to $CNF(KB \wedge \neg\alpha)$; complete for FOL

Conversion to CNF (I)

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

1. Eliminate biconditionals and implications

$$\forall x[\neg\forall y \neg Animal(y) \vee Loves(x,y)] \vee [\exists y Loves(y,x)]$$

2. Move $\neg$ inwards: $\neg\forall x, p \equiv \exists x \neg p, \quad \neg\exists x, p \equiv \forall x \neg p$:

$$\forall x[\exists y \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y Loves(y,x)]$$

$$\forall x[\exists y \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$$

$$\forall x[\exists y Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y Loves(y,x)]$$

Conversion to CNF (II)

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

$$\forall x[\exists y Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z Loves(z,x)]$$

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

$$\forall x[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$$

5. Drop universal quantifiers:

$$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$$

6. Distribute $\wedge$ over $\vee$:

$$[Animal(F(x)) \vee Loves(G(x),x)] \wedge [\neg Loves(x,F(x)) \vee Loves(G(x),x)]$$

Resolution proof: definite clauses