Inference in First Order Logic

Department of Electrical and Electronics Engineering
Spring 2005
Dr. Afşar Saranlı

Reference: Artificial Intelligence: A Modern Approach, 2nd Ed., Russel & Norvig

A Bit of History for Logic

450B.C.	Stoics	propositional logic, inference (maybe)
322B.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

First-order Inference, How?

- *Brute force*: Reduce to Propositional inference,
- Then apply propositional inference.
- How?

- *Universal Instantiation,*
- *Existential Instantiation.*

Universal Instantiation

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

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

for any variable v and ground term g

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

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

Existential Instantiation

- For any sentence α, variable v, and constant symbol k **that does not appear elsewhere in the knowledge base:**

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

- 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

Properties of Instantiations

- Universal Instantiation can be applied several times to add new sentences,
- The new KB is logically equivalent to the old. (Provided the original universal sentence is preserved)

- Existential instantiation can be applied only once, (and the existential sentence removed)
- The new KB is NOT logically equivalent to the old,
- However: *New KB is satisfiable iff the old KB was satisfiable.*

Now: Reduction to Propos. Logic

- 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 propositionalized: proposition symbols are

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

Reduction Continued…

- 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 Continued…

- Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB,
  it is entailed by a **finite** 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 semidecidable

Problems with Propositionalization

Propositionalization seems to generate lots of irrelevant sentences.
E.g., from

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

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

With function symbols, it gets nuch much worse!

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

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

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

Generalized Modus-Ponens

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

p_1' is $King(John)$	p_1 is $King(x)$
p_2' is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$	q is $Evil(x)$
$q\theta$ is $Evil(John)$	

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

Other Techniques

- Generalized Modus-Ponens leads to *Forward* and *Backward Chaining,*
- Backward chaining widely used: *Logic Programming* (e.g. Prolog Language)
- Resolution for FOL.

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

Conclusions