Lecture 13 of 42

FOL: The Frame Problem, Situation Calculus, and First-Order Inference

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:

Section 9.2 – 9.4, p. 275 - 294, Russell & Norvig 2nd edition
Handout, Nilsson & Genesereth, *Logical Foundations of Artificial Intelligence*

Lecture Outline

- Reading for Next Class: Sections 9.2 – 9.4 (p. 275 – 294), R&N 2e
- Last Class: Intro to FOL, Sections 8.1-8.2 (p. 240-253), R&N 2e
  - Elements of logic: ontology and epistemology
  - First-order predicate calculus (FOPC) *aka* first order logic (FOL)
  - Properties of sentences (and sets of sentences, *aka* knowledge bases)
    - entailment vs. provability/derivability
    - validity vs. satisfiability
  - Properties of proof rules
    - soundness: $KB \vdash_i \alpha \Rightarrow KB \models \alpha$ (can prove only true sentences)
    - completeness: $KB \models \alpha \Rightarrow KB \vdash_i \alpha$ (can prove all true sentences)
- Today: KR in FOL, Sections 8.3-8.4 (p. 253-266), 9.1 (p. 272-274), R&N 2e
  - Frame problem: representational (frame axioms) vs. inferential
  - Related inference problems: ramification and qualification
  - Situation calculus
  - First-order inference: Generalized Modus Ponens (GMP)
- This Week: KR and Inference in First-Order Logic (Ch. 8 – 10)

Soundness and Completeness in FOL: Review

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Consequences of KB are a haystack; α is a needle.
Entailment = needle in haystack; inference = finding it

Soundness: i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

Resolution – Algorithm: Review

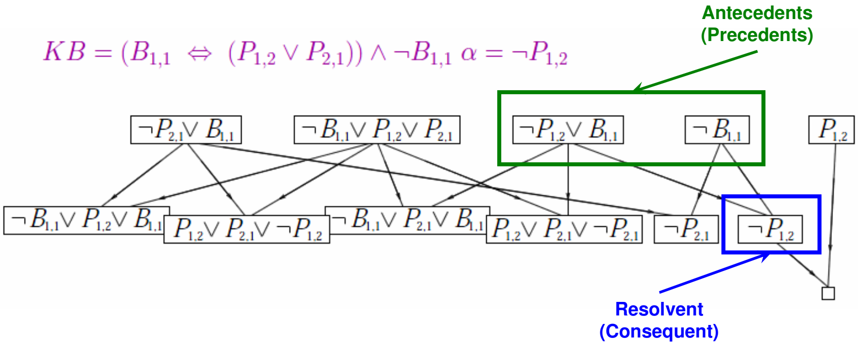
Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

```
function PL-Resolution(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
      for each Ci, Cj in clauses do
         resolvents ← PL-Resolve(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

Resolution – Example: Review

$$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \quad \alpha = \neg P_{1,2}$$



Adapted from slide © 2004 S. Russell & P. Norvig. Reused with permission.

Ontology and Epistemology: Review

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

Ontological commitment – what entities, relationships, and facts exist in world and can be reasoned about

Epistemic commitment – what agents can know about the world

Adapted from slide © 2004 S. Russell & P. Norvig. Reused with permission.

Denotational Semantics of FOL: Review

Sentences are true with respect to a model and an interpretation

Model contains ≥ 1 objects (domain elements) and relations among them

Interpretation specifies referents for
- constant symbols $\rightarrow$ objects
- predicate symbols $\rightarrow$ relations
- function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true
iff the objects referred to by $term_1, \ldots, term_n$
are in the relation referred to by $predicate$

Automated Deduction: Review

Properties of locations:

$\forall x, t \;\; At(Agent, x, t) \land Smelt(t) \;\Rightarrow\; Smelly(x)$

$\forall x, t \;\; At(Agent, x, t) \land Breeze(t) \;\Rightarrow\; Breezy(x)$

Squares are breezy near a pit:

Diagnostic rule—infer cause from effect

$$\forall y \;\; Breezy(y) \;\Rightarrow\; \exists x \;\; Pit(x) \land Adjacent(x, y)$$

Causal rule—infer effect from cause

$$\forall x, y \;\; Pit(x) \land Adjacent(x, y) \;\Rightarrow\; Breezy(y)$$

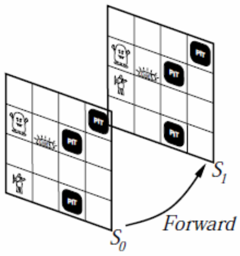
Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Keeping Track of Change

Facts hold in situations, rather than eternally
E.g., $Holding(Gold, Now)$ rather than just $Holding(Gold)$

Situation calculus is one way to represent change in FOL:
  Adds a situation argument to each non-eternal predicate
  E.g., Now in $Holding(Gold, Now)$ denotes a situation

Situations are connected by the $Result$ function
$Result(a, s)$ is the situation that results from doing a in s



Describing Actions [1]: Frame Axioms and the Frame Problem

"Effect" axiom—describe changes due to action
$\forall s \;\; AtGold(s) \;\Rightarrow\; Holding(Gold, Result(Grab, s))$

"Frame" axiom—describe non-changes due to action
$\forall s \;\; HaveArrow(s) \;\Rightarrow\; HaveArrow(Result(Grab, s))$

Frame problem: find an elegant way to handle non-change
  (a) representation—avoid frame axioms
  (b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or ...

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, ...

Describing Actions [2]: Successor-State Axioms

Successor-state axioms solve the representational frame problem

Each axiom is "about" a **predicate** (not an action per se):

P true afterwards ⇔ [an action made P true
∨ P true already and no action made P false]

For holding the gold:

$$\forall\, a, s \;\; Holding(Gold, Result(a, s)) \;\Leftrightarrow$$
$$[(a = Grab \wedge AtGold(s))$$
$$\vee\, (Holding(Gold, s) \wedge a \neq Release)]$$

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Making Plans [1]: Naïve Approach — Theorem Proving

Initial condition in KB:

$At(Agent, [1, 1], S_0)$
$At(Gold, [1, 2], S_0)$

Query: $Ask(KB, \exists\, s \;\; Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Making Plans [2]: A Better Way – Planning Systems

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing p in s

Then the query $Ask(KB, \exists p \;\; Holding(Gold, PlanResult(p, S_0)))$
has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$\forall s \;\; PlanResult([\,], s) = s$

$\forall a, p, s \;\; PlanResult([a|p], s) = PlanResult(p, Result(a, s))$

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Chapter 8: Summary

First-order logic:
- objects and relations are semantic primitives
- syntax: constants, functions, predicates, equality, quantifiers

Increased expressive power: sufficient to define wumpus world

Situation calculus:
- conventions for describing actions and change in FOL
- can formulate planning as inference on a situation calculus KB

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

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

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
that does not appear elsewhere 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 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

Using Universal and Existential Instantiation

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 [1]: Ground Terms

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 to Propositional Inference [2]: Herbrand's Lifting Lemma

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

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: PREVIEW

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)

$$\frac{p_1{}',\ p_2{}',\ \ldots,\ p_n{}',\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

where $p_i{}'\theta = p_i\theta$ 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

Soundness of GMP

Need to show that

$$p_1', \ldots, p_n', (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

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. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta$
2. $p_1', \ldots, p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

Example Knowledge Base: Preview

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

© 2004 S. Russell & P. Norvig. Reused with permission.

CIS 530 / 730
Artificial Intelligence
Lecture 13 of 42
Computing & Information Sciences
Kansas State University

Terminology

- **Resolution: Sound and Complete Inference Rule/Procedure for FOL**
  - Antecedent (*aka* precedent): sentences "above line" in sequent rule
  - Resolvent (*aka* consequent): sentences "below line" in sequent rule
- **Forward Chaining: Systematic Application of Rule to Whole KB**
  - Rete algorithm in production systems for expert systems development
  - Susceptible to high fan-out (branch factor)
- **Backward Chaining: Goal-Directed**
- **Today: Representing States/Actions in FOL; Propositional Logic to FOL**
  - Frame problem
    - representational: proliferation of frame axioms
    - inferential: replication of state
  - Qualification problem: handling "exceptions" (contingencies)
  - Ramification problem: handling "side effects"
  - Situation calculus: FOL-based language; successor state axioms, situations
  - Herbrand's lifting lemma: can generalize from propositional inference
  - Propositionalization: mapping down to ground terms

Summary Points

- **Last Class: Overview of Knowledge Representation (KR) and FOL**
  - Syntax and semantics
  - Ontological vs. epistemological commitments
  - Sentences: entailment vs. provability/derivability, validity vs. satisfiability
  - Proof rules
    - soundness: $KB \vdash_i \alpha \Rightarrow KB \models \alpha$ (can prove only true sentences)
    - completeness: $KB \models \alpha \Rightarrow KB \vdash_i \alpha$ (can prove all true sentences)
- **Today: Representing States and Actions in FOL**
  - Frame problem
    - representational: proliferation of frame axioms
    - inferential: replication of state
  - Relatives: ramification problem and qualification problem
  - Partial solution to representational frame problem: situation calculus
- **Chapter 9: First-Order Inference**
  - Herbrand theory: ground terms, lifting lemma
  - Preview of unification, GMP