



Lecture 12 of 42

First-Order Logic: Syntax and Semantics Discussion: Role of Logic in Planning

Wednesday, 19 September 2007

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/Fall-2007/CIS730>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:

Section 8.1 – 8.2, p.240 – 253, Russell & Norvig 2nd edition



Logical agents apply inference to a knowledge base
to derive new information and make decisions

Basic concepts of logic:

- syntax: formal structure of sentences
- semantics: truth of sentences wrt models
- entailment: necessary truth of one sentence given another
- inference: deriving sentences from other sentences
- soundness: derivations produce only entailed sentences
- completeness: derivations can produce all entailed sentences

Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

Propositional logic suffices for some of these tasks





Predicate Logic and FOL Road Map

- Predicate Logic
 - * Enriching language
 - ⇒ Predicates
 - ⇒ Functions
 - * Syntax and semantics of predicate logic
- First-Order Logic (FOL, FOFC)
 - * Need for quantifiers
 - * Relation to (unquantified) predicate logic
 - * Syntax and semantics of FOL
- Fun with Sentences
- Wumpus World in FOL



Syntax of FOL: Basic Elements

Constants *KingJohn, 2, UCB, ...*
Predicates *Brother, >, ...*
Functions *Sqrt, LeftLegOf, ...*
Variables *x, y, a, b, ...*
Connectives $\wedge \vee \neg \Rightarrow \Leftrightarrow$
Equality $=$
Quantifiers $\forall \exists$

Adapted from slides by
S. Russell, UC Berkeley





Fun with Sentences: Family Feud

- Brothers are Siblings
 - * $\forall x, y. \text{Brother}(x, y) \Leftrightarrow \text{Sibling}(x, y)$
- Siblings (i.e., Sibling Relationships) are Reflexive
 - * $\forall x, y. \text{Sibling}(x, y) \Leftrightarrow \text{Sibling}(y, x)$
- One's Mother is One's Female Parent
 - * $\forall x, y. \text{Mother}(x, y) \Leftrightarrow \text{Female}(x) \wedge \text{Parent}(x, y)$
- A First Cousin Is A Child of A Parent's Sibling
 - * $\forall x, y. \text{First-Cousin}(x, y) \Leftrightarrow$
 $\exists p, ps. \text{Parent}(p, x) \wedge \text{Sibling}(p, ps) \wedge \text{Parent}(ps, y)$

Adapted from slides by
S. Russell, UC Berkeley



Exercise [1]: First-Order Logic Sentences

- "Every Dog Chases Its Own Tail"
 - * $\forall d. \text{Chases}(d, \text{tail-of}(d))$
 - * Alternative Statement: $\forall d. \exists t. \text{Tail-Of}(t, d) \wedge \text{Chases}(d, t)$
 - * Prefigures concept of Skolemization (Skolem variables / functions)
- "Every Dog Chases Its Own (Unique) Tail"
 - * $\forall d. \exists^1 t. \text{Tail-Of}(t, d) \wedge \text{Chases}(d, t) \equiv$
 $\forall d. \exists t. \text{Tail-Of}(t, d) \wedge \text{Chases}(d, t) \wedge [\forall t' \text{Chases}(d, t') \Rightarrow t' = t]$
- "Only The Wicked Flee when No One Pursueth"
 - * $\forall x. \text{Flees}(x) \wedge [\neg \exists y \text{Pursues}(y, x)] \Rightarrow \text{Wicked}(x)$
 - * Alternative : $\forall x. [\exists y. \text{Flees}(x, y)] \wedge [\neg \exists z. \text{Pursues}(z, x)] \Rightarrow \text{Wicked}(x)$
- Offline Exercise: What Is An *m*th Cousin, *m* Times Removed?





Exercise [2]: First-Order Logic Sentences



Validity and Satisfiability

A sentence is **valid** if it is true in **all** models

e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the **Deduction Theorem**:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is **satisfiable** if it is true in **some** model

e.g., $A \vee B$, C

A sentence is **unsatisfiable** if it is true in **no** models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

i.e., prove α by *reductio ad absurdum*





FOL: Atomic Sentences (Atomic Well-Formed Formulae)

Atomic sentence = $predicate(term_1, \dots, term_n)$
or $term_1 = term_2$

Term = $function(term_1, \dots, term_n)$
or *constant* or *variable*

E.g., $Brother(KingJohn, RichardTheLionheart)$
> $(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Adapted from slides by
S. Russell, UC Berkeley



FOL: Complex Sentences (Well-Formed Formulae)

Complex sentences are made from atomic sentences using connectives

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
> $(1, 2) \vee \leq(1, 2)$
> $(1, 2) \wedge \neg >(1, 2)$

Adapted from slides by
S. Russell, UC Berkeley





Truth in FOL

Sentences are true with respect to a model and an interpretation

Model contains objects and relations among them

Interpretation specifies referents for

constant symbols → objects

predicate symbols → relations

function symbols → functional relations

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

Adapted from slides by
S. Russell, UC Berkeley



Lecture Outline

- Reading for Next Class: Section 8.1 – 8.2, R&N 2e
- Recommended : Nilsson and Genesereth (Chapter 5 online)
- Next Week's: Chapter 8 & first half of Chapter 9, R&N
- Today
 - * Syntax of first-order predicate calculus (FOPC, aka "first-order logic")
 - * Semantics
 - * Role of automated deduction in AI
- This Week
 - * Monday: Propositional Resolution, Soundness, Completeness
 - * Today: First-order logic (FOL): predicates, functions, quantifiers
 - * Friday: Knowledge Engineering (KE) and theorem proving
- Coming Soon
 - * Next week: Resolution, constraint logic, Prolog
 - * Week of 04 Oct 2006: knowledge representation, ontologies





Taking Stock: FOL Inference

- **Previously: Logical Agents and Calculi**
- **FOL in Practice**
 - * Agent “toy” world: Wumpus World in FOL
 - * Situation calculus
 - * Frame problem and variants (see R&N sidebar)
 - ⇒ Representational vs. inferential frame problems
 - ⇒ Qualification problem: “what if?”
 - ⇒ Ramification problem: “what else?” (side effects)
 - * Successor-state axioms
- **FOL Knowledge Bases**
- **FOL Inference**
 - * Proofs
 - * Pattern-matching: unification
 - * Theorem-proving as search
 - ⇒ Generalized Modus Ponens (GMP)
 - ⇒ Forward Chaining and Backward Chaining



Automated Deduction (Chapters 8-10 R&N)

Sound inference: find α such that $KB \models \alpha$.

Proof process is a search, operators are inference rules.

E.g., Modus Ponens (MP)

$$\frac{\alpha, \quad \alpha \Rightarrow \beta}{\beta} \quad \frac{At(Joe, UCB) \quad At(Joe, UCB) \Rightarrow OK(Joe)}{OK(Joe)}$$

E.g., And-Introduction (AI)

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta} \quad \frac{OK(Joe) \quad CSMajor(Joe)}{OK(Joe) \wedge CSMajor(Joe)}$$

E.g., Universal Elimination (UE)

$$\frac{\forall x \alpha}{\alpha\{x/\tau\}} \quad \frac{\forall x At(x, UCB) \Rightarrow OK(x)}{At(Pat, UCB) \Rightarrow OK(Pat)}$$

τ must be a ground term (i.e., no variables)

Adapted from slides by
S. Russell, UC Berkeley





Example Proof

● Bob is a buffalo	1. $Buffalo(Bob)$
● Pat is a pig	2. $Pig(Pat)$
Buffaloes outrun pigs	3. $\forall x, y \text{ Buffalo}(x) \wedge Pig(y) \Rightarrow Faster(x, y)$
Bob outruns Pat	
● Apply Sequent Rules to Generate New Assertions	
AI 1 & 2	4. $Buffalo(Bob) \wedge Pig(Pat)$
UE 3, $\{x/Bob, y/Pat\}$	5. $Buffalo(Bob) \wedge Pig(Pat) \Rightarrow Faster(Bob, Pat)$
MP 6 & 7	6. $Faster(Bob, Pat)$

$$\frac{\alpha, \alpha \Rightarrow \beta}{\beta}$$

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta}$$

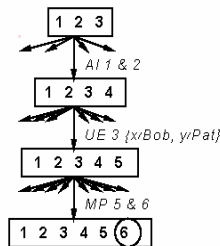
$$\frac{\forall x \alpha}{\alpha\{x/\tau\}}$$

- Modus Ponens
 - And Introduction
 - Universal Elimination
- Adapted from slides by S. Russell, UC Berkeley



Search with Primitive Inference Rules

Operators are inference rules
States are sets of sentences
Goal test checks state to see if it contains query sentence



AI, UE, MP is a common inference pattern

Problem: branching factor huge, esp. for UE

Idea: find a substitution that makes the rule premise match some known facts
⇒ a single, more powerful inference rule

Adapted from slides by S. Russell, UC Berkeley



A Brief History of Reasoning: Chapter 8 End Notes, R&N

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

Adapted from slides by
S. Russell, UC Berkeley



Knowledge Engineering

- **KE: Process of**
 - * Choosing logical language (basis of KR)
 - * Building KB
 - * Implementing proof theory
 - * Inferring new facts
- **Analogy: Programming Languages / Software Engineering**
 - * Choosing programming language (basis of software engineering)
 - * Writing program
 - * Choosing / writing compiler
 - * Running program
- **Example Domains**
 - * Electronic circuits (Section 8.3 R&N)
 - * Exercise
 - ⇒ Look up, read about [protocol analysis](#)
 - ⇒ Find example and think about KE process for your project domain





Unification: Definitions and Idea Sketch

A substitution σ unifies atomic sentences p and q if $p\sigma = q\sigma$

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

Idea: Unify rule premises with known facts, apply unifier to conclusion

E.g., if we know q and $Knows(John, x) \Rightarrow Likes(John, x)$
 then we conclude $Likes(John, Jane)$
 $Likes(John, OJ)$
 $Likes(John, Mother(John))$

Adapted from slides by
S. Russell, UC Berkeley



Generalized Modus Ponens

$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\sigma}$ where $p_i'\sigma = p_i\sigma$ for all i

E.g. $p_1' = \text{Faster}(\text{Bob}, \text{Pat})$

$p_2' = \text{Faster}(\text{Pat}, \text{Steve})$

$p_1 \wedge p_2 \Rightarrow q = \text{Faster}(x, y) \wedge \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z)$

$\sigma = \{x/\text{Bob}, y/\text{Pat}, z/\text{Steve}\}$

$q\sigma = \text{Faster}(\text{Bob}, \text{Steve})$

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

either a single atomic sentence or

(conjunction of atomic sentences) \Rightarrow (atomic sentence)

All variables assumed universally quantified

Adapted from slides by
S. Russell, UC Berkeley





Soundness of GMP

Need to show that

$$p_1', \dots, p_n', (p_1 \wedge \dots \wedge p_n \Rightarrow q) \models q\sigma$$

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

Lemma: For any definite clause p , we have $p \models p\sigma$ by UE

1. $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\sigma = (p_1\sigma \wedge \dots \wedge p_n\sigma \Rightarrow q\sigma)$
2. $p_1', \dots, p_n' \models p_1' \wedge \dots \wedge p_n' \models p_1'\sigma \wedge \dots \wedge p_n'\sigma$
3. From 1 and 2, $q\sigma$ follows by simple MP

Adapted from slides by S. Russell, UC Berkeley



Forward Chaining

When a new fact p is added to the KB
for each rule such that p unifies with a premise
if the other premises are known
then add the conclusion to the KB and continue chaining

Forward chaining is data-driven
e.g., inferring properties and categories from percepts

Adapted from slides by S. Russell, UC Berkeley



Example: Forward Chaining

Add facts 1, 2, 3, 4, 5, 7 in turn.

Number in \square = unification literal; \surd indicates rule firing

1. $Buffalo(x) \wedge Pig(y) \Rightarrow Faster(x, y)$
2. $Pig(y) \wedge Slug(z) \Rightarrow Faster(y, z)$
3. $Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z)$
4. $Buffalo(Bob)$ $\square_{1a, \times}$
5. $Pig(Pat)$ $\square_{1b, \surd}$ \rightarrow 6. $Faster(Bob, Pat)$ $\square_{3a, \times}$, $\square_{3b, \times}$
 $\square_{2a, \times}$
7. $Slug(Steve)$ $\square_{2b, \surd}$
 \rightarrow 8. $Faster(Pat, Steve)$ $\square_{3a, \times}$, $\square_{3b, \surd}$
 \rightarrow 9. $Faster(Bob, Steve)$ $\square_{3a, \times}$, $\square_{3b, \times}$

Adapted from slides by S. Russell, UC Berkeley



Backward Chaining

When a query q is asked

if a matching fact q' is known, return the unifier

for each rule whose consequent q' matches q

attempt to prove each premise of the rule by backward chaining

(Some added complications in keeping track of the unifiers)

(More complications help to avoid infinite loops)

Two versions: find any solution, find all solutions

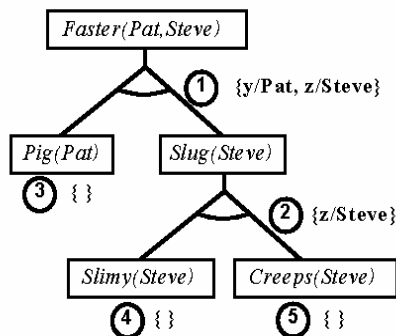
Backward chaining is the basis for logic programming, e.g., Prolog

Adapted from slides by S. Russell, UC Berkeley



Example: Backward Chaining

1. $Pig(y) \wedge Slug(z) \Rightarrow Faster(y, z)$
2. $Slimy(z) \wedge Creeps(z) \Rightarrow Slug(z)$
3. $Pig(Pat)$ 4. $Slimy(Steve)$ 5. $Creeps(Steve)$



Adapted from slides by S. Russell, UC Berkeley



Review: Backward Chaining

When a query q is asked
 if a matching fact q' is known, return the unifier
 for each rule whose consequent q' matches q
 attempt to prove each premise of the rule by backward chaining

(Some added complications in keeping track of the unifiers)

(More complications help to avoid infinite loops)

Two versions: find any solution, find all solutions

Backward chaining is the basis for logic programming, e.g., Prolog

● Question: How Does This Relate to Proof by Refutation?

● Answer

* Suppose \neg Query, For The Sake Of Contradiction (FTSOC)

Adapted from slides by S. Russell, UC Berkeley



Completeness Redux

Procedure i is complete if and only if

$$KB \vdash_i \alpha \text{ whenever } KB \models \alpha$$

Forward and backward chaining are complete for Horn KBs
but incomplete for general first-order logic

E.g., from

$$PhD(x) \Rightarrow HighlyQualified(x)$$

$$\neg PhD(x) \Rightarrow EarlyEarnings(x)$$

$$HighlyQualified(x) \Rightarrow Rich(x)$$

$$EarlyEarnings(x) \Rightarrow Rich(x)$$

should be able to infer $Rich(Me)$, but FC/BC won't do it

Does a complete algorithm exist?

Adapted from slides by S. Russell, UC Berkeley



Completeness in FOL

Entailment in first-order logic is only semidecidable:

can find a proof of α if $KB \models \alpha$

cannot always prove that $KB \not\models \alpha$

Cf. Halting Problem: proof procedure may be about to terminate with success or failure, or may go on for ever

Resolution is a refutation procedure:

to prove $KB \models \alpha$, show that $KB \wedge \neg\alpha$ is unsatisfiable

Resolution uses $KB, \neg\alpha$ in CNF (conjunction of clauses)

Resolution inference rule combines two clauses to make a new one:



Inference continues until an empty clause is derived (contradiction)

Adapted from slides by
S. Russell, UC Berkeley





Resolution Inference Rule

Basic propositional version:

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or equivalently} \quad \frac{\neg\alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Full first-order version:

$$\frac{\begin{array}{c} p_1 \vee \dots \vee p_j \dots \vee p_m, \\ q_1 \vee \dots \vee q_k \dots \vee q_n \end{array}}{(p_1 \vee \dots \vee p_{j-1} \vee p_{j+1} \dots \vee p_m \vee q_1 \dots \vee q_{k-1} \vee q_{k+1} \dots \vee q_n)\sigma}$$

where $p_j\sigma = \neg q_k\sigma$

For example,

$$\frac{\begin{array}{c} \neg Rich(x) \vee Unhappy(x) \\ Rich(Me) \end{array}}{Unhappy(Me)}$$

with $\sigma = \{x/Me\}$

Adapted from slides by
S. Russell, UC Berkeley



Digression: Decidability and Formal Languages

- See: Hopcroft and Ullman 2e, Lewis and Papadimitriou 3e
- Formal Languages (See: CIS 540, Other Automata Theory Course)
 - * Member of [Turing hierarchy](#)
 - ⇒ [Finite state automata](#): regular languages
 - ⇒ [Pushdown automata](#): context-free languages
 - ⇒ [Linear bounded automata](#): context-sensitive languages
 - ⇒ [Turing machines](#): recursive languages
 - * Recursive languages
 - ⇒ \exists computational model for [decision problem](#), halts in finite number of steps
 - ⇒ [REC](#): set of all recursive languages
 - ⇒ Example: finite searches (convert to decision problem of *checking solution*)
 - ⇒ *Closed under complementation* (consequence?)
 - * Recursive enumerable but not recursive (RE - REC)
 - * Not recursive (\notin RE)
- What Are FOL-VALID, FOL-NOT-SAT, FOL-SAT, FOL-NOT-VALID?





Summary Points

- Applications of Knowledge Bases (KBs) and Inference Systems
- “Industrial Strength” KBs
 - * Building KBs
 - * Components
 - ⇒ Ontologies
 - ⇒ Fact and rule bases
 - ⇒ Knowledge Engineering (KE) and protocol analysis
 - ⇒ Inductive Logic Programming (ILP) and other machine learning techniques
 - * Using KBs
- Systems of Sequent Rules: GMP/AI/UE, Resolution
- Methodology of Inference
 - * Inference as search
 - * Forward and backward chaining
 - * Fan-in, fan-out



Terminology

- Logical Frameworks
 - * Knowledge Bases (KB)
 - * Logic in general: representation languages, syntax, semantics
 - * Propositional logic
 - * First-order logic (FOL, FOPL)
 - * Model theory, domain theory: possible worlds semantics, entailment
- Normal Forms
 - * Conjunctive Normal Form (CNF)
 - * Disjunctive Normal Form (DNF)
 - * Horn Form
- Proof Theory and Inference Systems
 - * Sequent calculi: rules of proof theory
 - * Derivability or provability
 - * Properties
 - ⇒ Soundness (derivability implies entailment)
 - ⇒ Completeness (entailment implies derivability)