



## Lecture 13 of 42

### First-Order Logic: KE and Theorem Proving Discussion: Review of Models, Theorem Proving

Friday, 22 September 2006

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-2006/CIS730>

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

#### Reading for Next Class:

Section 8.3 – 8.5, p. 240 – 268, Russell & Norvig 2<sup>nd</sup> edition

Section 9.1, p. 272 – 275, Russell & Norvig 2<sup>nd</sup> edition



## 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: Knowledge Engineering and Theorem Proving
- Next Week
  - \* Resolution
  - \* Constraint logic
  - \* Prolog
- Week of 04 Oct 2006
  - \* Knowledge representation
  - \* Ontologies





## Logical Agents: Review

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

Adapted from slides by  
S. Russell, UC Berkeley



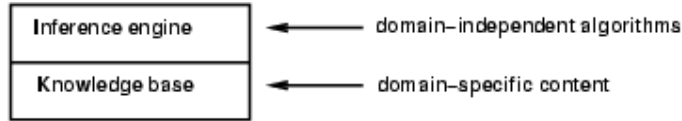
## Predicate Logic and FOL Road Map

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





## Knowledge bases: Review



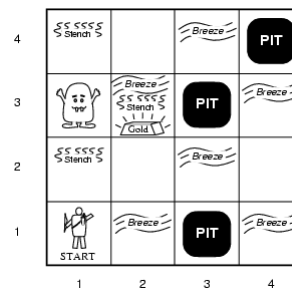
- Knowledge base = set of **sentences** in a **formal language**
- **Declarative** approach to building an agent (or other system):
  - \* **Tell it what it needs to know**
- Then it can **Ask** itself what to do - answers should follow from **KB**
- Agents can be viewed at the **knowledge level**
  - i.e., what they know, regardless of how implemented
- Or at the **implementation level**
  - \* i.e., data structures in KB and algorithms that manipulate them

© 2004 S. Russell, UC Berkeley



## Wumpus World – PEAS Description: Review

- **Performance measure**
  - \* gold +1000, death -1000
  - \* -1 per step, -10 for using the arrow
- **Environment**
  - \* Squares adjacent to wumpus are smelly
  - \* Squares adjacent to pit are breezy
  - \* Glitter iff gold is in the same square
  - \* Shooting kills wumpus if you are facing it
  - \* Shooting uses up the only arrow
  - \* Grabbing picks up gold if in same square
  - \* Releasing drops the gold in same square



- **Sensors:** Stench, Breeze, Glitter, Bump, Scream
- **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot

© 2004 S. Russell, UC Berkeley





## Wumpus world – characterization: Review

- **Fully Observable** No – only **local** perception
- **Deterministic** Yes – outcomes exactly specified
- **Episodic** No – sequential at the level of actions
- **Static** Yes – Wumpus and Pits do not move
- **Discrete** Yes
- **Single-agent?** Yes – Wumpus is essentially a natural feature

© 2004 S. Russell, UC Berkeley



## Logic in General: Review

Logics are formal languages for representing information such that conclusions can be drawn

Syntax defines the sentences in the language

Semantics define the “meaning” of sentences; i.e., define truth of a sentence in a world

E.g., the language of arithmetic

$x + 2 \geq y$  is a sentence;  $x^2 + y >$  is not a sentence

$x + 2 \geq y$  is true iff the number  $x + 2$  is no less than the number  $y$

$x + 2 \geq y$  is true in a world where  $x = 7, y = 1$

$x + 2 \geq y$  is false in a world where  $x = 0, y = 6$

Adapted from slides by  
S. Russell, UC Berkeley





## Entailment: Review

- **Entailment** means that one thing **follows from** another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence  $\alpha$  if and only if  $\alpha$  is true in all worlds where *KB* is true

- \* E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
- \* E.g.,  $x+y = 4$  entails  $4 = x+y$
- \* Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**

© 2004 S. Russell, UC Berkeley



## DPLL algorithm: Review

**function** DPLL-SATISFIABLE?(*s*) **returns** *true* or *false*

**inputs:** *s*, a sentence in propositional logic

*clauses* ← the set of clauses in the CNF representation of *s*

*symbols* ← a list of the proposition symbols in *s*

**return** DPLL(*clauses*, *symbols*, [])

**function** DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

**if** every clause in *clauses* is true in *model* **then return** *true*

**if** some clause in *clauses* is false in *model* **then return** *false*

*P*, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)

**if** *P* is non-null **then return** DPLL(*clauses*, *symbols*-*P*, [*P* = *value* | *model*])

*P*, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)

**if** *P* is non-null **then return** DPLL(*clauses*, *symbols*-*P*, [*P* = *value* | *model*])

*P* ← FIRST(*symbols*); *rest* ← REST(*symbols*)

**return** DPLL(*clauses*, *rest*, [*P* = *true* | *model*]) **or**

DPLL(*clauses*, *rest*, [*P* = *false* | *model*])

© 2004 S. Russell, UC Berkeley

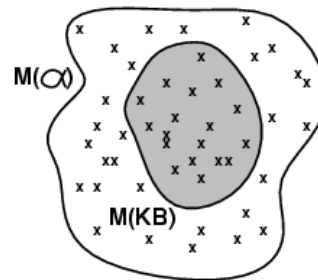
Figure 7.16 p. 222 R&N 2e





## Models: Review

- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated
- We say  $m$  is a **model** of a sentence  $\alpha$  if  $\alpha$  is true in  $m$
- $M(\alpha)$  is the set of all models of  $\alpha$
- Then  $KB \vdash \alpha$  iff  $M(KB) \subseteq M(\alpha)$ 
  - \* E.g.  $KB =$  Giants won and Reds won  $\alpha =$  Giants won
- See: definitions on p. 201, 203 R&N 2e

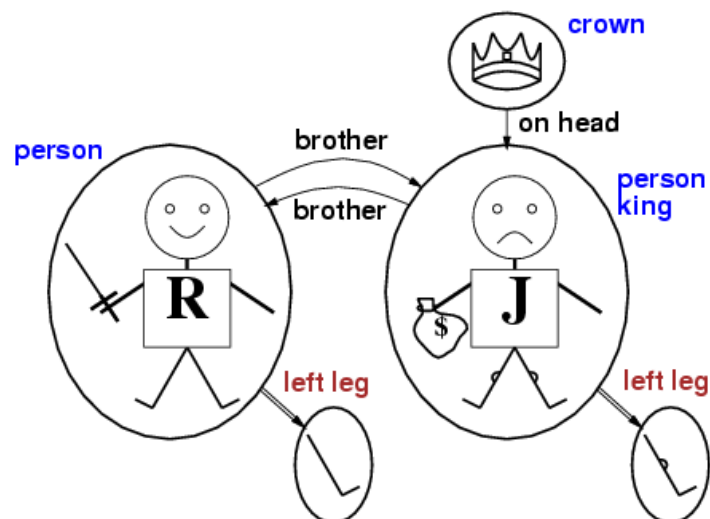


Adapted from slide  
© 2004 S. Russell, UC Berkeley

See also: S. 7.3, p. 200 – 204  
S. 8.2, p. 245 – 253



## Models for FOL: Example





## Types of Logic: Review

Logics are characterized by what they commit to as “primitives”

Ontological commitment: what exists—facts? objects? time? beliefs?

Epistemological commitment: what states of knowledge?

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 0..1
Fuzzy logic	degree of truth	degree of belief 0..1

Adapted from slides by  
S. Russell, UC Berkeley

Figure 8.1 p. 244 R&N 2e



## FOL – Atomic Sentences (Atoms): Review

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 (WFFs): Review

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: Review

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





## Automated Deduction (Chapters 8-10): Review

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

Proof process is a search, operators are inference rules.

E.g., Modus Ponens (MP)

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

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)

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

$\tau$  must be a ground term (i.e., no variables)

Adapted from slides by  
S. Russell, UC Berkeley



## Example Proof

- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li>● Bob is a buffalo</li> <li>● Pat is a pig</li> <li>● Buffaloes outrun pigs</li> <li>● Bob outruns Pat</li> </ul> | <ol style="list-style-type: none"> <li>1. <i>Buffalo</i>(Bob)</li> <li>2. <i>Pig</i>(Pat)</li> <li>3. <math>\forall x, y \text{ Buffalo}(x) \wedge \text{Pig}(y) \Rightarrow \text{Faster}(x, y)</math></li> </ol> |
|--|--|

- Apply Sequent Rules to Generate New Assertions

- |  |   |
|--|---|
| <p>AI 1 &amp; 2</p> <p>UE 3, <math>\{x/Bob, y/Pat\}</math></p> <p>MP 6 &amp; 7</p> | <ol style="list-style-type: none"> <li>4. <i>Buffalo</i>(Bob) <math>\wedge</math> <i>Pig</i>(Pat)</li> <li>5. <i>Buffalo</i>(Bob) <math>\wedge</math> <i>Pig</i>(Pat) <math>\Rightarrow</math> <i>Faster</i>(Bob, Pat)</li> <li>6. <i>Faster</i>(Bob, Pat)</li> </ol> |
|--|---|

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

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

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

● Modus Ponens

And Introduction

Universal Elimination

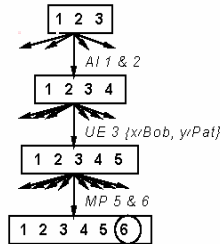
Adapted from slides by  
S. Russell, UC Berkeley





## 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  
 $\Rightarrow$  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  $\sigma$  unifies atomic sentences  $p$  and  $q$  if  $p\sigma = q\sigma$

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

**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

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

E.g.  $p_1' = \text{Faster}(\text{Bob}, \text{Fat})$   
 $p_2' = \text{Faster}(\text{Pat}, \text{Steve})$   
 $p_1 \wedge p_2 \Rightarrow q = \text{Faster}(x, y) \wedge \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z)$   
 $\sigma = \{x/\text{Bob}, y/\text{Pat}, z/\text{Steve}\}$   
 $q\sigma = \text{Faster}(\text{Bob}, \text{Steve})$

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.  $Pig(Pat) \square_{1b, \surd} \rightarrow \square_{2a, \times} \rightarrow \square_{3a, \times}, \square_{3b, \times}$
7.  $Slug(Steve) \square_{2b, \surd}$   
 $\rightarrow \square_{3a, \times}, \square_{3b, \surd}$   
 $\rightarrow \square_{3a, \times}, \square_{3b, \times}$

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

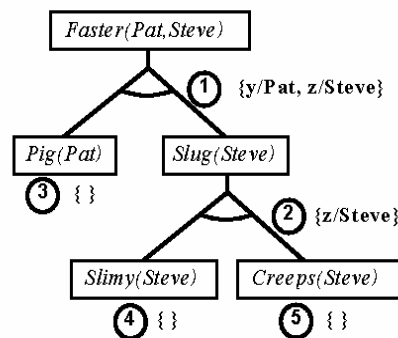
Backward chaining is the basis for logic programming, e.g., Prolog

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





## 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

- Answer
  - \* Suppose ¬Query, For The Sake Of Contradiction (FTSOC)
  - \* Attempt to prove that  $KB \wedge \neg Query \vdash \perp$



## 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,

$$\frac{\neg Rich(x) \vee Unhappy(x)}{Rich(Me)} \quad \frac{}{Unhappy(Me)}$$

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: *checking solution*)
    - ⇒ *Closed under complementation* (consequence?)
  - \* Recursive enumerable but not recursive (RE - REC)
  - \* Not recursive ( $\not\in$  RE)
- What Are FOL-VALID, FOL-NOT-SAT, FOL-SAT, FOL-NOT-VALID?



## Universal quantification

- $\forall$  <variables> <sentence>

Everyone at K-State is smart:

$$\forall x \text{ At}(x, \text{K-State}) \Rightarrow \text{Smart}(x)$$

- $\forall x P$  is true in a model  $m$  iff  $P$  is true with  $x$  being each possible object in the model
- Roughly speaking, equivalent to the **conjunction** of **instantiations** of  $P$

$$\begin{aligned} & \text{At}(\text{KingJohn}, \text{K-State}) \Rightarrow \text{Smart}(\text{KingJohn}) \\ \wedge & \text{At}(\text{Richard}, \text{K-State}) \Rightarrow \text{Smart}(\text{Richard}) \\ \wedge & \text{At}(\text{K-State}, \text{K-State}) \Rightarrow \text{Smart}(\text{K-State}) \\ \wedge & \dots \end{aligned}$$





## A common mistake to avoid

- Typically,  $\Rightarrow$  is the main connective with  $\forall$
- Common mistake: using  $\wedge$  as the main connective with  $\forall$ :  
 $\forall x \text{ At}(x, \text{K-State}) \wedge \text{Smart}(x)$   
means “Everyone is at K-State and everyone is smart”



## Existential quantification

- $\exists \langle \text{variables} \rangle \langle \text{sentence} \rangle$
- Someone at KU is smart:
- $\exists x \text{ At}(x, \text{KU}) \wedge \text{Smart}(x)$
- $\exists x P$  is true in a model  $m$  iff  $P$  is true with  $x$  being some possible object in the model
- Roughly speaking, equivalent to the **disjunction** of **instantiations** of  $P$ 
  - $\text{At}(\text{KingJohn}, \text{KU}) \wedge \text{Smart}(\text{KingJohn})$
  - $\vee \text{At}(\text{Richard}, \text{KU}) \wedge \text{Smart}(\text{Richard})$
  - $\vee \text{At}(\text{KU}, \text{KU}) \wedge \text{Smart}(\text{KU})$
  - $\vee \dots$





## 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)