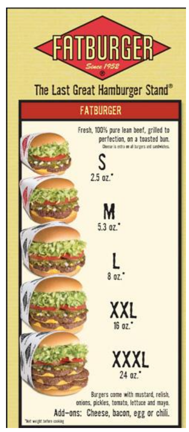
CSE 473

Lecture 12

Chapter 8

First-Order Logic

What's on our menu today?



First-Order Logic

- Definitions
- Universal and Existential Quantifiers
- Skolemization
- Unification

Propositional vs. First-Order

Propositional logic: Deals with facts and propositions (can be true or false):

$P_{1,1}$ "there is a pit in (1,1)"

George_Monkey "George is a monkey"

George_Curious "George is curious"

473student1_Monkey

(George_Monkey $\wedge$ ¬473student1_Monkey) $\vee$ …

3

Propositional vs. First-Order

First-order logic: Deals with objects and relations

Objects: George, 473Student1, Monkey2, Raj, …

Relations: Monkey(George), Curious(George),
Smarter(473Student1, Monkey2)
Smarter(Monkey2, Raj)
Stooges(Larry, Moe, Curly)
PokesInTheEyes(Moe, Curly)
PokesInTheEyes(473Student1, Raj)

4

FOL Definitions

Constants: Name a specific object.
George, Monkey2, Larry, …

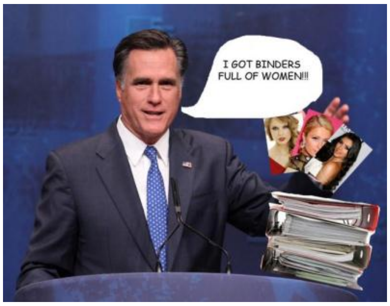
Variables: Refer to an object without naming it.
X, Y, …

Relations (predicates): Properties of or relationships between objects.
Curious, PokesInTheEyes, …

FOL Definitions

Functions: Mapping from objects to objects.
banana-of, grade-of, binders-full-of

Terms: Logical expressions referring to objects
banana-of(George)
grade-of(stdnt1)
binders-full-of(women)
binders-full-of(men)
binders-full-of(monkeys)



More Definitions

Logical connectives: **and, or, not, ⇒, ⇔**

Quantifiers:

- ∀ **For all** **(Universal quantifier)**
- ∃ **There exists** **(Existential quantifier)**

Examples

- **George is a monkey and he is curious**
  Monkey(George) ∧ Curious(George)
- **All monkeys are curious**
  ∀m: Monkey(m) ⇒ Curious(m)
- **There is a curious monkey**
  ∃m: Monkey(m) ∧ Curious(m)

Quantifier / Connective Interaction

M(x) == "x is a monkey"
C(x) == "x is curious"

∀x: M(x) ∧ C(x)

"Everything is a curious monkey"

∀x: M(x) ⇒C(x)

"All monkeys are curious"

∃x: M(x) ∧ C(x)

"There exists a curious monkey"

∃x: M(x) ⇒ C(x)

"There exists an object that is *either* a curious monkey, *or* not a monkey at all"

Nested Quantifiers: Order matters!

$$\forall x \exists y\ P(x,y) \neq \exists y \forall x\ P(x,y)$$

Example

Every monkey has a tail

$\forall m \exists t\ has(m,t)$

Every monkey ***shares*** a tail!

$\exists t \forall m\ has(m,t)$

Try:

Everybody loves somebody *vs.* Someone is loved by everyone

$\forall x \exists y\ loves(x, y)$ $\exists y \forall x\ loves(x, y)$

Semantics

Semantics = **what the arrangement of symbols *means* in the world**

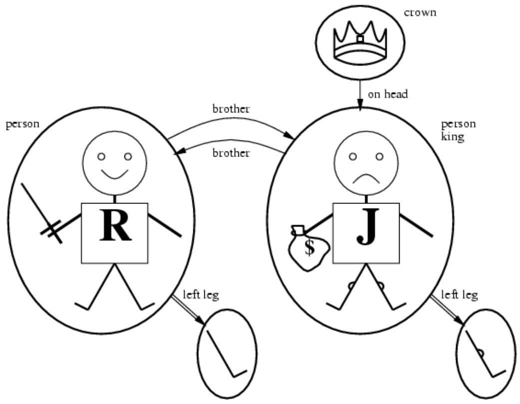
Propositional logic

- **Basic elements are *propositional variables*** e.g., $P_{1,1}$
  (refer to facts about the world)
- **Possible worlds: mappings from variables to T/F**

First-order logic

- **Basic elements are *terms*, e.g.,** George, banana-of(George), binders-full-of(banana-of(George))
  (logical expressions that refer to objects)
- Interpretations: **mappings from terms to real-world elements.**

Example: A World of Kings and Legs



Syntactic elements:

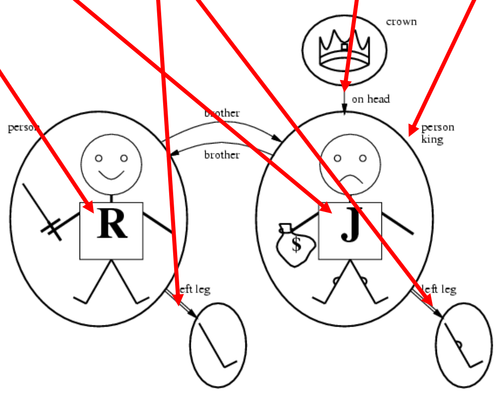
Constants:	Functions:	Relations:
Richard John	LeftLeg(p)	On(x,y) King(p)

11

Interpretation I

Interpretations map syntactic tokens to model elements

Constants:	Functions:	Relations:
Richard John	LeftLeg(p)	On(x,y) King(p)



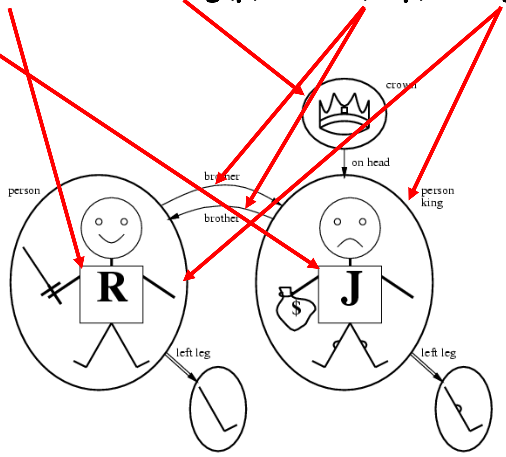
12

Interpretation II

Constants: Richard John

Functions: LeftLeg(p)

Relations: On(x,y) King(p)



How Many Interpretations?

Two constants (and 5 objects in world)

- Richard, John (R, J, crown, RL, JL)

5^2 = 25 object mappings

One unary relation

King(x)

Infinite number of values for x → infinite mappings

Even if we restricted x to: R, J, crown, RL, JL:

2^5 = 32 unary truth mappings

Two binary relations

Leg(x, y); On(x, y)

Infinite. But even restricting x, y to five objects still yields 2^{25} mappings *for each* binary relation

Satisfiability, Validity, & Entailment

S is valid if it is true in all interpretations

S is satisfiable if it is true in some interp

S is unsatisfiable if it is false in all interps

S1 $\models$ S2 (S1 entails S2) if
for all interps where S1 is true,
S2 is also true

Propositional. Logic *vs.* First Order

Ontology	Facts (P, Q,…)	Objects, Properties, Relations
Syntax	Atomic sentences Connectives	Variables & quantification Sentences have structure: terms father-of(mother-of(X)))
Semantics	Truth Tables	Interpretations (Much more complicated)
Inference Algorithm	DPLL, WalkSAT Fast in practice	Unification Forward, Backward chaining Prolog, theorem proving
Complexity	NP-Complete	Semi-decidable May run forever if KB $\not\models \alpha$

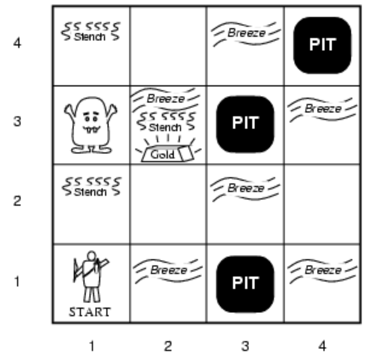
First-Order Wumpus World

Objects

- **Squares, wumpuses, agents,**
- **gold, pits, stinkiness, breezes**

Relations

- **Square topology (adjacency),**
- **Pits/breezes,**
- **Wumpus/stinkiness**



Wumpus World: Squares

- Each square as an object:
  $Square_{1,1}$, $Square_{1,2}$, …,
  $Square_{3,4}$, $Square_{4,4}$
- Square topology relations?
  $Adjacent(Square_{1,1}, Square_{2,1})$
  …
  $Adjacent(Square_{3,4}, Square_{4,4})$

Better: **Squares as lists:**

[1, 1], [1,2], …, [4, 4]

Square topology relations:

$\forall x, y, a, b$: **Adjacent([x, y], [a, b])** $\Leftrightarrow$

[a, b] $\in$ **{[x+1, y], [x-1, y], [x, y+1], [x, y-1]}**

Wumpus World: Pits

- Each pit as an object:
  $Pit_{1,1}$, $Pit_{1,2}$, …,
  $Pit_{3,4}$, $Pit_{4,4}$
- Problem?
  Not all squares have pits

List only the pits we have?

$Pit_{3,1}$, $Pit_{3,3}$, $Pit_{4,4}$

Problem?

No reason to distinguish pits (same properties)

Better: **pit as unary predicate**

Pit(x)

Pit([3,1]), Pit([3,3]), Pit([4,4]) will be true

Wumpus World: Breezes

- Represent breezes like pits, as unary predicates:
  Breezy(x)

"Squares next to pits are breezy":

$\forall$c, d, a, b:

Pit([c, d]) $\wedge$ Adjacent([c, d], [a, b]) $\Rightarrow$ Breezy([a, b])

Wumpus World: Wumpuses

- Wumpus as object:
  Wumpus
- Wumpus home as unary predicate:
  WumpusIn(x)

Better: **Wumpus's home as a function:**

Home(Wumpus) references the wumpus's home square.

FOL Reasoning: Outline

Basics of FOL reasoning

Classes of FOL reasoning methods

- **Forward & Backward Chaining**
- **Resolution**
- **Compilation to SAT**

Basics: Universal Instantiation

Universally quantified sentence:

- $\forall x$: Monkey(x) $\Rightarrow$ Curious(x)

Intutively, x can be anything:

- Monkey(George) $\Rightarrow$ Curious(George)
- Monkey(473Student1) $\Rightarrow$ Curious(473Student1)
- Monkey(DadOf(George)) $\Rightarrow$ Curious(DadOf(George))

Formally:

$$\frac{\forall x \quad S}{\text{Subst}(\{x/p\}, S)}$$

x is replaced with p in S, and the quantifier removed

Example:

$$\frac{\forall x \quad \text{Monkey}(x) \rightarrow \text{Curious}(x)}{\text{Monkey(George)} \rightarrow \text{Curious(George)}}$$

x is replaced with George in S, and the quantifier removed

23

Basics: Existential Instantiation

Existentially quantified sentence:

$\exists x$: Monkey(x) $\wedge$ $\neg$Curious(x)

Intutively, x must name something. But what?

Can we conclude:

Monkey(George) $\wedge$ $\neg$Curious(George) ???

No! S might not be true for George!

Use a *Skolem Constant* and draw the conclusion:

Monkey(K) $\wedge$ $\neg$Curious(K)

where K is a completely new **symbol** (stands for the monkey for which the statement is true)

Formally:

$$\frac{\exists x \quad S}{\text{Subst}(\{x/K\}, S)}$$

K is called a Skolem constant

24

Basics: Generalized Skolemization

What if our existential variable is nested?

$\forall x \; \exists y$: Monkey(x) $\Rightarrow$ HasTail(x, y)

Can we conclude:

$\forall x$: Monkey(x) $\Rightarrow$ HasTail(x, K_Tail) ???

Nested existential variables can be replaced by Skolem functions

- Args to function are all surrounding $\forall$ vars

$\forall x$: Monkey(x) $\Rightarrow$ HasTail(x, f(x))

"tail-of" function

25

Motivation for Unification

What if we want to use modus ponens?

Propositional Logic:

$$\frac{a \wedge b, \qquad a \wedge b \Rightarrow c}{c}$$

In First-Order Logic?

$\forall x$ Monkey(x) $\Rightarrow$ Curious(x)

Monkey(George)

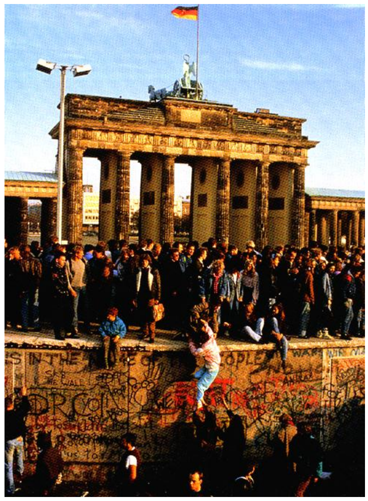
????

Must "*unify*" x with George:

Need to substitute {x/George} in Monkey(x) $\Rightarrow$ Curious(x) to infer Curious(George)

26

What is Unification?



Not this kind of unification...

27

What is Unification?

Match up expressions by *finding variable values that make the expressions identical*

Unify(x, y) returns most general unifier (MGU).

MGU places fewest restrictions on values of variables

Examples:

- Unify(city(x), city(seattle)) returns {x/seattle}
- Unify(PokesInTheEyes(Moe,x), PokesInTheEyes(y,z)) returns {y/Moe,z/x}
  - {y/Moe,x/Moe,z/Moe} possible but not MGU

28

Unification and Substitution

Unification produces a mapping from variables to values (e.g., {x/seattle,y/tacoma})

Substitution: Subst(mapping,sentence) returns new sentence with variables replaced by values

- Subst({x/seattle,y/tacoma}),connected(x, y)),
  returns connected(seattle, tacoma)

Next Time

Reasoning with FOL
- Chaining
- Resolution
- Compilation to SAT

To Do:
- Project #2
- Read Chapters 8-9