# Pros and cons of propositional logic

- ☺ Propositional logic is declarative
- ☺ Propositional logic allows partial/disjunctive/negated information
  - (unlike most data structures and databases)
- ☺ Propositional logic is compositional:
  - meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- ☺ Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)
- ☹ Propositional logic has very limited expressive power
  - (unlike natural language)
  - E.g., cannot say "all pits cause breezes in adjacent squares"
    - except by writing one sentence for each square

## This time

- **First-order logic**
  - Syntax
  - Semantics
  - Wumpus world example

- **Ontology** (ont = 'to be'; logica = 'word'): kinds of things one can talk about in the language
- A "controlled" vocabulary to describe objects and relations between them in a formal manner

## Why first-order logic?

- We saw that propositional logic is limited because it only makes the ontological commitment that the world consists of **facts**.

- Difficult to represent even simple worlds like the Wumpus world;

  e.g.,
  “don’t go forward if the Wumpus is in front of you” takes 64 rules

# First-order logic (FOL)

- Ontological commitments:
  - **Objects**: wheel, door, body, engine, seat, car, passenger, driver
  - **Relations**: Inside(car, passenger), Beside(driver, passenger)
  - **Functions**: ColorOf(car)
  - **Properties**: Colored(car), IsOpen(door), IsOn(engine)…

- Functions are relations with single value for each object

## Semantics

there is a correspondence between

- functions, which return values
- predicates, which are true or false

Function: father_of(Mary) = Bill

Predicate: father_of(Mary, Bill)

# Examples:

- "One plus two equals three"

  Objects:

  Relations:

  Properties:

  Functions:

- "Squares neighboring the Wumpus are smelly"

  Objects:

  Relations:

  Properties:

  Functions:

## Examples:

- “One plus two equals three”

  Objects: one, two, three, one plus two

  Relations: equals

  Properties: --

  Functions: plus (“one plus two” is the name of the object obtained by applying function plus to one and two; three is another name for this object)

- “Squares neighboring the Wumpus are smelly”

  Objects: Wumpus, square

  Relations: neighboring

  Properties: smelly

  Functions: --

## FOL: Syntax of basic elements

- **Constant symbols:** 1, 5, A, B, USC, JPL, Alex, Manos, …
- **Predicate symbols:** >, Friend, Student, Colleague, …
- **Function symbols:** +, sqrt, SchoolOf, TeacherOf, ClassOf, …
- **Variables:** *x, y, z, next, first, last, …*
- **Connectives:** $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- **Quantifiers:** $\forall, \exists$
- **Equality:** $=$

## Syntax of Predicate Logic

- Symbol set
  - **constants**
  - **Boolean connectives**
  - variables
  - functions
  - predicates (relations)
  - quantifiers

## Syntax of Predicate Logic

- Terms: a reference to an object
  - variables,
  - constants,
  - functional expressions (can be arguments to predicates)

- Examples:
  - first([a,b,c]), sq_root(9), sq_root(n), tail([a,b,c])

## Syntax of Predicate Logic

- Sentences: make claims about objects
  - (Well-formed formulas, (wffs))
- Atomic Sentences (predicate expressions):
  - loves(John,Mary), brother_of(John,Ted)
- Complex Sentences (Atomic Sentences connected by boolean connectors):
  - loves(John,Mary)
  - brother_of(John,Ted)
  - teases(Ted, John)

## Examples of Terms: Constants, Variables and Functions

- Constants: object constants refer to individuals
  - Alan, Sam, R225, R216
- Variables
  - PersonX, PersonY, RoomS, RoomT
- Functions
  - father_of(PersonX)
  - product_of(Number1,Number2)

## Examples of Predicates and Quantifiers

- Predicates
  - in(Alan,R225)
  - partOf(R225,Pender)
  - fatherOf(PersonX,PersonY)
- Quantifiers
  - All dogs are mammals.
  - Some birds can’t fly.
  - 3 birds can’t fly.

## Semantics

- Referring to individuals
  - Jackie
  - son-of(Jackie), Sam

- Referring to states of the world
  - person(Jackie), female(Jackie)
  - mother(Sam, Jackie)

## FOL: Atomic sentences

AtomicSentence $\rightarrow$ Predicate(Term, …) | Term = Term

Term $\rightarrow$ Function(Term, …) | Constant | Variable

- Examples:
  - SchoolOf(Manos)
  - Colleague(TeacherOf(Alex), TeacherOf(Manos))
  - >((+ x y), x)

# FOL: Complex sentences

Sentence $\rightarrow$ AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable, … Sentence
| $\neg$ Sentence
| (Sentence)

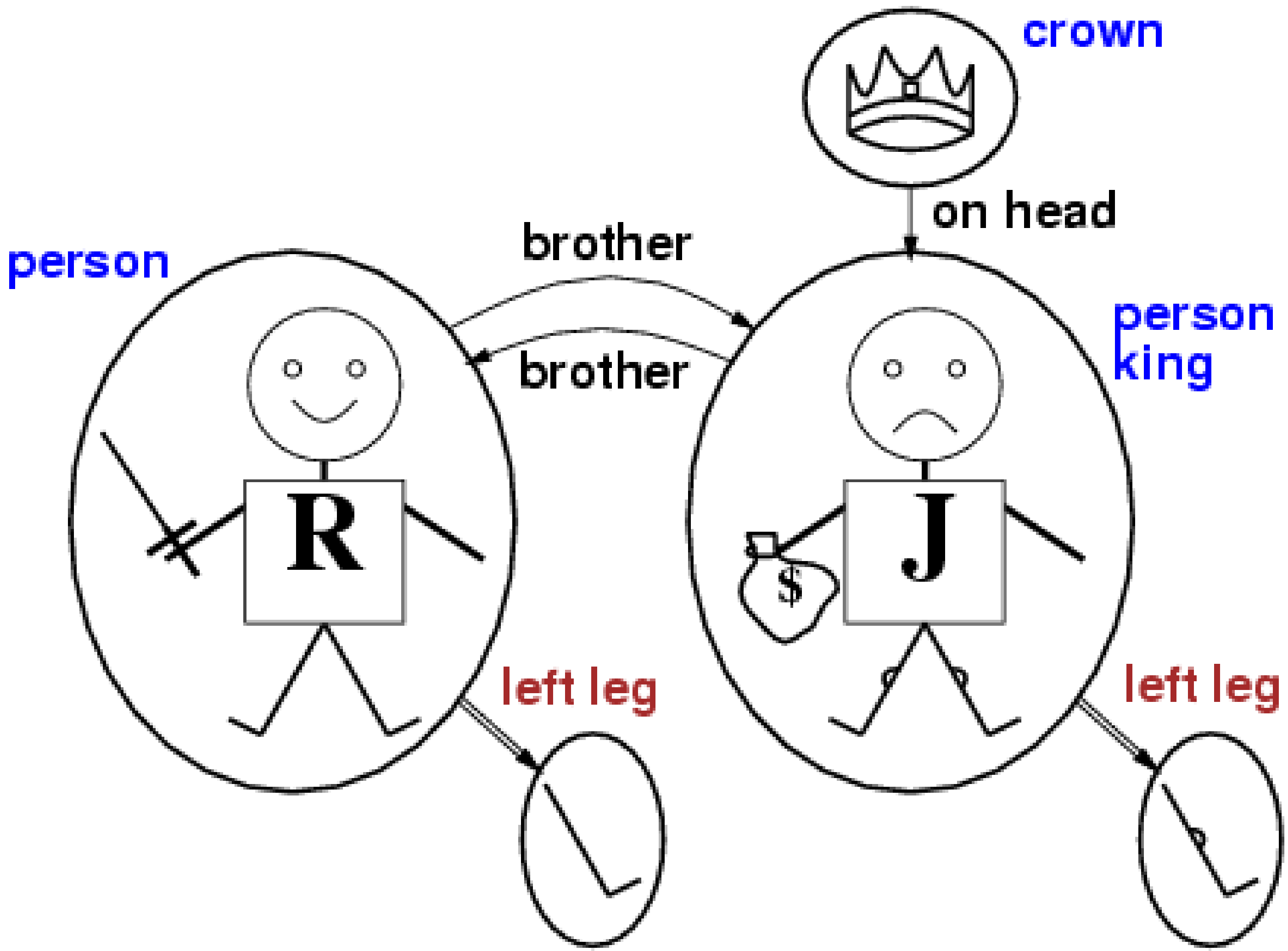
- Examples:
  - $S1 \wedge S2$, $S1 \vee S2$, $(S1 \wedge S2) \vee S3$, $S1 \Rightarrow S2$, $S1 \Leftrightarrow S3$
  - Colleague(Paolo, Maja) $\Rightarrow$ Colleague(Maja, Paolo)
    Student(Alex, Paolo) $\Rightarrow$ Teacher(Paolo, Alex)

## Semantics of atomic sentences

- Sentences in FOL are interpreted with respect to a model and an interpretation
- Model contains objects and relations among them
- Terms: refer to objects (e.g., Door, Alex, StudentOf(Paolo))
  - Constant symbols: refer to objects
  - Predicate symbols: refer to relations
  - Function symbols: refer to functional Relations

- An atomic sentence *predicate*($term_1, \ldots, term_n$) is **true** iff the relation referred to by *predicate* holds between the objects referred to by $term_1, \ldots, term_n$

# Models for FOL: Example



# Example model

- Objects: John, James, Mary, Alex, Dan, Joe, Anne, Rich
- Relation: sets of tuples of objects
  {<John, James>, <Mary, Alex>, <Mary, James>, ...}
  {<Dan, Joe>, <Anne, Mary>, <Mary, Joe>, ...}
- E.g.:
  Parent relation -- {<John, James>, <Mary, Alex>, <Mary, James>}

  then Parent(John, James) is true
  Parent(John, Mary) is false

## Quantifiers

- Expressing sentences about **collections** of objects without enumeration (naming individuals)

- E.g., All Trojans are clever

  Someone in the class is sleeping

- Universal quantification (for all): ∀

- Existential quantification (three exists): ∃

## Universal quantification (for all): $\forall$

$\forall$ *<variables> <sentence>*

- *"Every one in the cs460 class is smart"*:
  $\forall\, x\;\; \text{In}(\text{cs460}, x) \Rightarrow \text{Smart}(x)$

- **$\forall$ P corresponds to the conjunction of instantiations of P**
  In(cs460, Manos) $\Rightarrow$ Smart(Manos) $\wedge$
  In(cs460, Dan) $\Rightarrow$ Smart(Dan) $\wedge$
  …
  In(cs460, Bush) $\Rightarrow$ Smart(Bush)

## Universal quantification (for all): $\forall$

- $\Rightarrow$ is a natural connective to use with $\forall$
- Common mistake: to use $\wedge$ in conjunction with $\forall$
  e.g: $\forall\, x \quad \text{In}(\text{cs460}, x) \wedge \text{Smart}(x)$
  means *"every one is in cs460 and everyone is smart"*

## Existential quantification (there exists): $\exists$

$\exists$ *<variables> <sentence>*

- *"Someone in the cs460 class is smart"*:
  $\exists x \;\; In(cs460, x) \wedge Smart(x)$

- **$\exists$ P corresponds to the disjunction of instantiations of P**
  $In(cs460, Manos) \wedge Smart(Manos) \vee$
  $In(cs460, Dan) \wedge Smart(Dan) \vee$
  …
  $In(cs460, Bush) \wedge Smart(Bush)$

## Existential quantification (there exists): $\exists$

- $\wedge$ is a natural connective to use with $\exists$
- Common mistake: to use $\Rightarrow$ in conjunction with $\exists$
  e.g: $\exists\, x \;\; \text{In}(\text{cs460}, x) \Rightarrow \text{Smart}(x)$
  is true if there is anyone that is not in cs460!
  (remember, false $\Rightarrow$ true is valid).

## Properties of quantifiers

$\forall x \; \forall y$ is the same as $\forall y \; \forall x$ (why??)

$\exists x \; \exists y$ is the same as $\exists y \; \exists x$ (why??)

$\exists x \; \forall y$ is not the same as $\forall y \; \exists x$

$\exists x \; \forall y \; Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y \; \exists x \; Loves(x, y)$
"Everyone in the world is loved by at least one person"

Not all by one person but each one at least by one

Quantifier duality: each can be expressed using the other

$\forall x \; Likes(x, IceCream) \qquad \neg\exists x \; \neg Likes(x, IceCream)$ Proof?

$\exists x \; Likes(x, Broccoli) \qquad \neg\forall x \; \neg Likes(x, Broccoli)$

## Proof

- In general we want to prove:

$\forall$ x P(x) <=> $\neg$ $\exists$ x $\neg$ P(x)

- $\forall$ x P(x) = $\neg(\neg(\forall$ x P(x))) = $\neg(\neg$(P(x1) ^ P(x2) ^ … ^ P(xn)) ) = $\neg(\neg$P(x1) v $\neg$P(x2) v … v $\neg$P(xn)) )
- $\exists$ x $\neg$P(x) = $\neg$P(x1) v $\neg$P(x2) v … v $\neg$P(xn)
- **$\neg$**$\exists$ x $\neg$P(x) = $\neg(\neg$P(x1) v $\neg$P(x2) v … v $\neg$P(xn))

## Example sentences

- Brothers are siblings
  - 
- Sibling is transitive
  - 
- One's mother is one's sibling's mother
  - 
- A first cousin is a child of a parent's sibling
  - 

## Example sentences

- Brothers are siblings

  $\forall$ x, y   Brother(x, y) $\Rightarrow$ Sibling(x, y)

- Sibling is transitive

  $\forall$ x, y, z   Sibling(x, y) $\wedge$ Sibling(y, z) $\Rightarrow$ Sibling(x, z)

- One's mother is one's sibling's mother

- A first cousin is defined as a child of a parent's sibling

## Example sentences

- Brothers are siblings

  $\forall$ x, y   Brother(x, y) $\Rightarrow$ Sibling(x, y)

- Sibling is transitive

  $\forall$ x, y, z   Sibling(x, y) $\wedge$ Sibling(y, z) $\Rightarrow$ Sibling(x, z)

- One’s mother is one’s sibling’s mother

  $\forall$ m, c, d   Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)

- A first cousin is defined as a child of a parent’s sibling

## Example sentences

- Brothers are siblings

  $\forall$ x, y  Brother(x, y) $\Rightarrow$ Sibling(x, y)

- Sibling is transitive

  $\forall$ x, y, z  Sibling(x, y) $\wedge$ Sibling(y, z) $\Rightarrow$ Sibling(x, z)

- One's mother is one's sibling's mother

  $\forall$ m, c, d  Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)

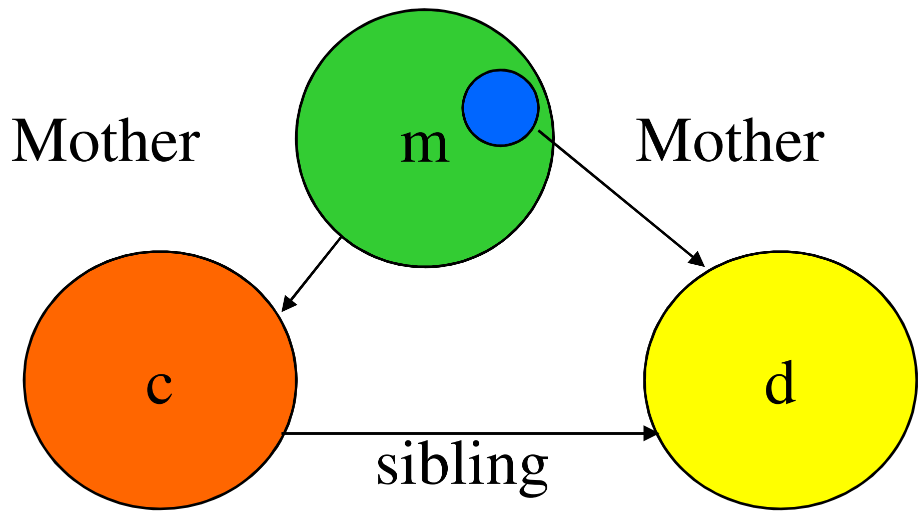
- A first cousin is defined as a child of a parent's sibling

  $\forall$ c, d  FirstCousin(c, d) $\Leftrightarrow$
  $\exists$ p, ps  Parent(p, d) $\wedge$ Sibling(p, ps) $\wedge$ Parent(ps, c)

# Example sentences

- One’s mother is one’s sibling’s mother
  $\forall$ m, c,d   Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)

- $\forall$ c,d $\exists$**m** Mother(m, c) $\wedge$ Sibling(c, d) $\Rightarrow$ Mother(m, d)



## Translating English to FOL

- Every gardener likes the sun.
  $\forall$ **x gardener(x) => likes(x,Sun)**

- You can fool some of the people all of the time.
  $\exists$ **x** $\forall$ **t (person(x) ^ time(t)) => can-fool(x,t)**

## Translating English to FOL

- You can fool all of the people some of the time.

- All purple mushrooms are poisonous.

## Translating English to FOL

- You can fool all of the people some of the time.

  **∀ x ∃ t (person(x) ^ time(t) => can-fool(x,t)**

- All purple mushrooms are poisonous.

## Translating English to FOL

- You can fool all of the people some of the time.

  $\forall$ **`x`** $\exists$ **`t (person(x) ^ time(t) =>`**
  **`can-fool(x,t)`**

- All purple mushrooms are poisonous.

  $\forall$ **`x (mushroom(x) ^ purple(x)) =>`**
  **`poisonous(x)`**

## Translating English to FOL…

- No purple mushroom is poisonous.

## Translating English to FOL…

- No purple mushroom is poisonous.

$\neg (\exists x)$ **purple(x) ^ mushroom(x) ^ poisonous(x)**

or, equivalently,

$(\forall x)$ **(mushroom(x) ^ purple(x)) => ¬poisonous(x)**

## Translating English to FOL…

- There are exactly two purple mushrooms.

```
(∃ x)(∃ y) mushroom(x) ∧ purple(x) ∧
  mushroom(y) ∧ purple(y) ∧ ¬(x=y) ∧
   [(∀ z) (mushroom(z) ∧ purple(z)) =>
            ((x=z) ∨ (y=z))]
```

- Deb is not tall.

```
¬tall(Deb)
```

## Translating English to FOL…

- X is above Y if (and only if) X is on directly on top of Y or else there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

```
(∀ x)(∀ y) above(x,y) <=> (on(x,y) v (∃ z)
  (on(x,z) ^ above(z,y)))
```