Propositional logic is a weak language

- Hard to identify "individuals." Ex. Mary, 3
- Can't directly talk about properties of individuals or relations between individuals. Ex. "Bill is tall"
- Generalizations, patterns, regularities can't easily be represented. Ex. all triangles have 3 sides

- **First-Order Logic** (abbreviated FOL or FOPC) is expressive enough to concisely represent this kind of situation.
  - FOL adds relations, variables, and quantifiers, e.g.,
    - "Every elephant is gray": $\forall$ x (elephant(x) $\rightarrow$ gray(x))
    - "There is a white elephant": $\exists$ x (elephant(x) ^ white(x))

First-order logic

- First-order logic (FOL) models the world in terms of
  - Objects, which are things with individual identities
  - Properties of objects that distinguish them from other objects
  - Relations that hold among sets of objects
  - Functions, which are a subset of relations where there is only one "value" for any given "input"

Ex:Objects: Students, lectures, companies, cars ...

  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

FOL Syntax

- **Variable symbols**
  - E.g., x, y, John
- **Connectives**: ¬, ∧, ∨, ⇒
  - **Quantifiers**
  - Universal ∀x
  - Existential ∃x

Syntax of First-order logic

Sentence → *Atomicsentence*
| (*Sentence Connective Sentence*)
| *Quantifier Variable*,. . . *Sentence*
| > *Sentence*
AtomicSentence → *Predicate(Term*, . . .)
| (*Term* = *Term*
Term → *Function(Term*, . . .)
l *Constant*
| *Variable*
Connective → ¬, ∧, ∨, ⇒
Quantifier → ∀, ∃
Constant → *A* (Xl (*John* 1 . . .
Variable → *a* | *x* | s | . . .
Predicate → *Before...*
Function → *Mother* | *...*

Atomic Sentences

- Propositions are represented by a **predicate applied to a** tuple of terms. A predicate represents a property or relation between terms that can be true or false:
- Brother(John, Fred), Left-of(Square1, Square2), GreaterThan(plus(1,1), plus(0,1))
- Sentences in logic **state facts** that are true or false.
- In FOL properties and n-ary relations do express that:

  LargerThan(2,3) is false. Brother(Mary,Pete) is false.
- Note: **Functions do not state facts** and form no sentence: Brother(Pete) refers to the object John (his brother) and is neither true nor false.
- Brother(Pete,Brother(Pete)) is True.

Binary relation Function

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies:

constant symbols → objects
predicate symbols → relations
function symbols → functional relations

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

Entailment

- Entailment means that one thing follows from another:

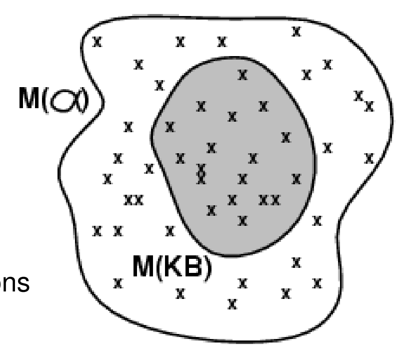
$$KB \models \alpha$$

Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

  – E.g., the KB containing "the Greens won" and "the Reds won" entails "Either the Greens 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
  – entailment: necessary truth of one sentence given another

Models

- 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 α if α is true in *m*
- *M(α)* is the set of all models of α
- Then KB $\models$ α iff *M(KB)* $\subseteq$ *M(*α)
  – E.g. *KB* = Greens won and Reds won
    α = Greens won



- Think of KB and α as collections of constraints and of models m as possible states. M(KB) are the solutions to KB and M(α) the solutions to α. Then, KB $\models$ α when all solutions to KB are also solutions to α.

Nested Quantifiers

- Combinations of universal and existential quantification are possible:

$$\forall x \forall y\, Father(x,y) \equiv \forall y \forall x\, Father(x,y)$$
$$\exists x \exists y\, Father(x,y) \equiv \exists y \exists x\, Father(x,y)$$
$$\forall x \exists y\, Father(x,y) \neq \exists y \forall x\, Father(x,y)$$
$$\exists x \forall y\, Father(x,y) \neq \forall y \exists x\, Father(x,y)$$
$$x,y \in \{All\ \ people\}$$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:
- Ex:

$\forall$x At(x,CU) $\wedge$ Smart(x)
means "Everyone is at CU and everyone is smart"

Yet to say Everyone at CU is smart

$\forall$x At(x,CU) $\Rightarrow$ Smart(x)

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

  $$\exists x \, At(x,CU) \Rightarrow Smart(x)$$

  is true if there is anyone who is smart not at CU.

Yet to say: there exists someone in CU that is smart

$$\exists x \, At(x,CU) \wedge Smart(x)$$

Properties of quantifiers

$\forall x \ \forall y$ is the same as $\forall y \ \forall x$
$\exists x \ \exists y$ is the same as $\exists y \ \exists x$

$\exists x \ \forall y$ 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"

- Quantifier duality: each can be expressed using the other

Exp.	Negation
$\forall x$ Likes(x,IceCream)	$\exists x \neg$Likes(x,IceCream)
$\exists x$ Likes(x,Broccoli)	$\forall x \neg$Likes(x,Broccoli)

Equality

Equality:

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

FOPC can include equality as a primitive predicate or require it to be as identity relation

Equal(x,y) or x=y

Examples:

to say "that Mary is taking two courses", you need to insure that x,y are different

$\exists x\ \exists y$ (takes(Mary,x) ^ takes (Mary,y) ^ ~ (x=y))

To say "Everyone has exactly one father"

$\forall x\ \exists y$ father(y,x) ^ $\forall z$ father(z,x) ➔ y=z

Higher Order Logic

- FOPC is called first order because it allows quantifiers to rang only over objects (terms).

  $\forall x, \forall y\ [x=y \text{ or } x>y \text{ or } y>x]$

- **Second-Order Logic** allows quantifiers to range over predicates and functions as well

  $\forall f, \forall g\ [f=g \iff (\forall x\ f(x)=g(x))]$

- **Third-Order Logic** allows quantifiers to range over predicates of predicates,
- .. etc

Examples of FOPC

- Brothers are siblings

  $\forall x, \forall y\ \textit{Brother(x,y)} \Rightarrow \textit{Sibling(x,y)}$

- One's mother is one's female parent

  $\forall m, \forall c\ \textit{Mother(c)} = m \Leftrightarrow (\textit{Female(m)} \wedge \textit{Parent(m,c)})$

- "Sibling" is symmetric

  $\forall x, \forall y\ \textit{Sibling(x,y)} \Leftrightarrow \textit{Sibling(y,x)}$

Translating English to FOL

- Every gardener likes the sun.

  (∀x) gardener(x) => likes(x,Sun)

Translating English to FOL

- Every gardener likes the sun.

  `(∀x) gardener(x) => likes(x,Sun)`
- You can fool some of the people all of the time.

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

Translating English to FOL

- Every gardener likes the sun.

  `(∀x) gardener(x) => likes(x,Sun)`
- You can fool some of the people all of the time.

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

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

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

Translating English to FOL

- Every gardener likes the sun.

  `(∀x) gardener(x) => likes(x,Sun)`
- You can fool some of the people all of the time.

  `(∃x) person(x) ^ ((∀t) time(t)) => can-fool(x,t))`
- You can fool all of the people some of the time.

  `(∀x) person(x) => ((∃t) time(t) ^ can-fool(x,t))`
- All purple mushrooms are poisonous.

  `(∀ x) (mushroom(x) ^ purple(x)) => poisonous(x)`

Translating English to FOL

- Every gardener likes the sun.

  `(∀x) gardener(x) => likes(x,Sun)`
- You can fool some of the people all of the time.

  `(∃x) person(x) ^ ((∀t) time(t)) => can-fool(x,t))`
- You can fool all of the people some of the time.

  `(∀x) person(x) => ((∃t) time(t) ^ can-fool(x,t))`
- All purple mushrooms are poisonous.

  `(∀ x) (mushroom(x) ^ purple(x)) => poisonous(x)`
- No purple mushroom is poisonous.

  `~(∃x) purple(x) ^ mushroom(x) ^ poisonous(x)`

  or, equivalently,

  `(∀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) v (y=z))
```

Inference in FOL chapter 9 in Russel

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

i.e. deriving sentences from other sentences

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

i.e. derivations produce only entailed sentences ***(no wrong inferences, but maybe not all inferences)***

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

i.e. derivations can produce all entailed sentences (***all inferences can be made, but maybe some wrong extra ones as well)***

Validity and satisfiability

- A sentence is **valid** if it is true in **all models**,
- e.g., ***True***, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the following:
$\boldsymbol{KB} \models \alpha$ if and only if $(\boldsymbol{KB} \Rightarrow \alpha)$ 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:
$\boldsymbol{KB} \models \alpha$ if and only if $(\boldsymbol{KB} \wedge \neg \alpha)$ is unsatisfiable
(there is no model for which KB=true and is false)

Proof Methods in FOL

Major Families:

- GMP
- Reduction
- Resolution
- Forward chaining
- Backward chaining

Some Other inference tools:

Entailment/ Unification/

Proof Methods in FOL

- GMP: Using the generalized form of Modus Ponens
- Reduction: Reduce all FOL sentences to propositional Calculus then use inference in propositional calculus
- Resolution – Refutation
  - Negate goal
  - Convert all pieces of knowledge into clausal form (disjunction of literals)
  - See if contradiction indicated by null clause $\square$ can be derived
- Forward chaining
  - Given P, $P \rightarrow Q$, to infer Q
  - P, match $L.H.S$ of
  - Assert Q from $R.H.S$
- Backward chaining
  - Q, Match $R.H.S$ of $P \rightarrow Q$
  - assert P
  - Check if P exists