# Knowledge Representation: Logic

# Knowledge Representation Languages

Mathematical Logic Hierarchy

- Proposition Calculus ($0^{th}$ order)
  - No variables or quantifiers
  - Propositions together with logical connectives
- Predicate Calculus ($1^{st}$ order)
  - Adds Variables for terms, and quantification
- Second Order Logics ($2^{nd}$ order)
  - Adds variables for predicates

# Propositional Calculus

- Propositions are true or false
    - The moon is made of cheese
    - Socrates is a man
    - Men are mortal
    - The population of the world is 3 million
- Questions are not propositions
    - What is the meaning of life, the universe, and everything?
    - Where is the door?

# Propositional Calculus (cont'd)

Propositional Calculus or Propositional Logic is about propositions and their combinations

Combine Propositions with logical connectives:

- $\neg$ not (negation)
- $\vee$ or (disjunction)
- $\wedge$ and (conjunction)
- $\rightarrow$ implies (conditional)
- $\leftrightarrow$ equivalence (bi-conditional)

# Truth tables

How logical are humans?

Truth tables show the truth value of propositions and their combinations

We can use symbols like p and q to represent propositions

| P | Q | P$\rightarrow Q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

# Logical Equivalence

It turns out we don't need all of the connectives

Truth table proves $P \rightarrow Q$ is logically equivalent to $\neg P \vee Q$

There is **no way** to assign values to P or Q to make these two formulas (formulae) different

| P | Q | $P \rightarrow Q$ | $\neg P \vee Q$ |
|---|---|---|---|
| T | T | T | T |
| T | F | F | F |
| F | T | T | T |
| F | F | T | T |

# When P is false

Many of us have some trouble with the last line of the previous truth table:

| P | Q | P$\rightarrow Q$ |
|---|---|---|
| F | F | T |

Our intuition tells us that so much falsity in P and Q cannot result in truth?

Explanation: P->Q is not as strong as our human intuition wants to make it.  It's saying that if P is true, then there is more to the story.  But if P is false, you can stop reading, the story is already finished, in that we know the overall statement is true.

# When P is false (cont'd)

Think of the following P->Q, F -> F statement, which is true:

If 2 < 1 then all humans are extremely wealthy

Our human intuition wants to say, no it's not true, because "making" 2 < 1 true would not cause humans to be wealthy, the concepts are unrelated

After we get used to it, we can see that 2<1 is simply false, and it doesn't make sense to imagine "making" it true. The "making" it true is not part of the statement (it's a weak statement).

Our intuition wants to draw a causal relationship between "making" P true, and causing Q (all humans to be wealthy).

Our intuition wants to say "no such relationship exists" so "false"

Logic says **if** P is **false**, then the statement is **true** and it doesn't matter what Q is

# Why do we care about logic?

- Propositional Calculus is a tool that allows us to derive conclusions from combinations of simpler statements known to be true
- We are seeing the workings of a system where we can
  - Make statements that we know to be true
  - The *logical entailments* of those statements are also true
  - Prolog systematically finds logical entailments

# Time to check your learning!

Let’s see how many key concepts from propositional calculus you recall by filling in the truth table:

| P | Q | $P \rightarrow Q$ | $\neg P \vee Q$ |
|---|---|---|---|
| T | T | | |
| T | F | | |
| F | T | | |
| F | F | | |

# Predicate Calculus

Predicate Calculus (or first-order predicate calculus FOPC or first-order logic FOL) gives us all of propositional calculus, plus the following logical symbols

- Variables to represent terms, or "things in the domain"
- Quantifiers
    - $\forall$ universal quantification, forall
    - $\exists$ existential quantification, exists
- $=$ equality symbol

# First Order Logic equivalents to Prolog statements

| Prolog | FOL |
|---|---|
| ancestor(X,Y) :- parent(X,Y). | $\forall X\ \forall Y parent(X,Y) \rightarrow ancestor(X,Y)$ |
| ancestor(X,Z) :-<br>parent(X,Y),<br>ancestor(Y,Z). | $\forall X \forall Y \forall Z\ parent(X,Y) \land ancestor(Y,Z)$<br>$\rightarrow ancestor(X,Z)$ |
| parent(john,sue). | $parent(john, sue)$ |
| parent(X,sue):-<br>X = john<br>;<br>X = sally. | $\forall X\ X = john \lor X = sally \rightarrow parent(X, sue)$ |

# Predicate Calculus (cont'd)

As well as the additional logical symbols, predicate calculus adds non-logical symbols:

- function symbols of different arity
  - for example: +, -, 0, 1, 2, todd, triangle
- predicate symbols of different arity
  - for example: <, rides_a_bike, triangle_exists,

# First-order terms represent things

With the additional symbols, we can build terms that represent things in our domain of discourse:

**Variables:** Any variable is a term.

**Functions:** Any expression $f(t_1,...,t_n)$ of $n$ arguments, where each argument $t_i$ is a term and $f$ is a function symbol of arity $n$, is a term.

**Constants**: A special case of a function term where the arity is 0

# Term examples

+(3,4) : a term that denotes a number

- 3 is a constant, which strictly speaking is a function that takes no arguments
- 4 is another function of no arguments
- + is a function of arity 2

- temperature_of(mars) : a term that denotes a temperature
    - mars is a constant
    - temperature_of is a function of arity 1

# FOL Formulas represent statements about things

We saw that terms represent ***things***.

Formulas, or well-formed formulas (formulae), or wwfs, are built up from predicates (and equality) that take terms as arguments, and ***make true or false statements about things***

for example:

- =(+(1,1),2)
- <(temperature_of(mars),3)
- rides_a_bike(todd)

# Prolog terminology vs Predicate Calculus terminology

Unfortunately, the terminology differs between the Predicate Calculus and Prolog:

- term in FOL
  - is a function applied to zero or more arguments
  - represents a thing,
    - for example +(3,4) with two arguments represents a number, seven
    - 4 with zero arguments represents a number, four
    - mother_of(bob) with one argument represents a person, bob's mother
- Atomic Well-formed formula (WWF), also called "atom", in FOL
  - is a predicate applied to zero or more arguments which are terms
  - represents a true-or-false statement about zero or more terms (things)

# Prolog terminology vs Predicate Calculus terminology (cont'd)

- Non-atomic WWF in FOL
  - Is also a statement
  - Involves logical connectives like ¬,∧,∨ (not, and, or)

# Prolog terminology vs Predicate Calculus terminology

- term in Prolog is any structure:
  - Functions applied to arguments
    - +(3,4)
    - mother_of(bob)
  - Predicates applied to arguments
    - >(4,3)
    - parent(bill,bob)
  - Everything is a term, even something such as  a:-b,c
    - :-(a,',',(b,c))

- atom in Prolog is a function of no arguments, such as a, todd, etc
- numbers in prolog are also functions of no arguments, such as 3, 66, etc

# FOL Formulas (cont'd)

We can also use the logical connectives and other logical symbols in formulas

These are examples of statements which may be true or false:

Example1: For all x, there exists a y such that y is greater than x:

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

Example2: It's not the case that there exists an x such that forall y, y is greater than x:

$$\neg (\exists x \forall y \ > (y, x))$$

# Formulas (cont'd)

Let's look closer at the ordering of the quantifiers in statements like these.

True statement: It's not the case that there exists an x such that every y is greater than x:

$$\neg\left(\exists x \forall y > (y, x)\right)$$

This one below looks similar but says something completely different:

$$\neg\,(\forall x \exists y > (y, x))$$

False statement: it's not the case that forall x, there exists a y such that y is greater than x

# More reader friendly with infix notation of greater-than?

Often we find infix notation for > is easier to read than prefix.

True statement: It's not the case that there exists an x such that every y is greater than x:

$$\neg\left(\exists x \forall y\,(y > x)\right)$$

This looks similar but says something completely different:

$$\neg\,(\forall x \exists y\,(y > x))$$

False statement: it's not the case that forall x, there exists a y such that y is greater than x

# Now, without the negation

Let's look at the two statements on the previous slide, but change them by removing the negation symbol:

Example1: False statement: there exists an x such that every y is greater than x:

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

Example2: This looks similar but says something completely different:

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

True statement: forall x, there exists a y such that y is greater than x (think of y = x + 1 where no matter what x is, y is greater)

# More intuitive example?

Ordering of the quantifiers is important.

Consider mother_of(x,y) to mean "x is the mother of y"

In infix notation: x mother_of y

Everyone has a mother, and it's the **same** mother:

$$\exists x \forall y\ (x\ mother\_of\ y)$$

Now, same variables, different order of quantifiers:

Everyone has a possibly different mother

$$\forall y \exists x\ (x\ mother\_of\ y)$$

# Order of universal quantifiers?

In the previous slides, the ordering of exists and forall does affect meaning.

The ordering of universal quantifiers (forall) does not matter

$$\forall x \forall y\ (x\ fellow\_human\_of\ y)$$

Everybody is fellow_human_of everybody

$$\forall y \forall x\ (x\ fellow\_human\_of\ y)$$

These two statements mean the same thing (logically equivalent)

# Free variables

A variable that is not bound to any quantifier in a formula is called a ***free*** variable

We don't want to write logical sentences with free variables because those variable values depend on the interpretation

We are trying to say things that are true **regardless** of interpretation, and free variables prevent that

So, when we see free variables in Sitcalc or Prolog, they aren't actually free, they are implicitly **prenex universally quantified**

# Prenex universal quantification

A formula is in ***prenex normal form*** when all the quantifiers are together on the left side of the formula, in what's called the ***prefix***.

A related idea is ***prenex universal quantification***, which is when a variable is universally quantified and the quantifier is in the prefix.

If there are no free variables in a formula, and all quantifiers in the prefix are universal, then we can drop the prefix if we assume all resulting free variables are prenex universally quantified

# FOL Axioms

- We can “create” a new world by making statements that are true in that world, very much like an author writing a novel makes statements about what is true in that novel world.
- Unlike a novelist, we make our statements using first-order logic sentences instead of (for example) English sentences.
- The statements we make in first-order logic, to specify a world, are called axioms
- Axioms are taken to be true without proof **because** they are stated, similarly to how a novelist makes statements about the characters of a novel
- Axioms must be consistent, because a contradiction can be used to prove anything by contradiction (inconsistent axioms are useless to us).

# Time to check your learning!

Let's see how many key concepts from first-order predicate calculus you recall by answering the following questions!

Which of the following are FOL terms or not FOL terms:

- $\rightarrow x$
- $x$
- $x > y$
- todd
- temperature_of(todd)
- loves(cathy,joseph)

# Higher-order logic

Second-order (and higher-order) logic involves quantifying over not just variables (terms), but also predicates

These systems are more expressive, but things get complicated

Their model-theoretic properties are less well-behaved than those of first-order logic

In this course, we will limit our scope to First Order Logic (FOL) in order to benefit from the nice model-theoretic properties of FOL

First Order Logic can be converted to clausal form

All Prolog programs are sets of Horn clauses, a specific clausal form

# Time to check your learning!

Let's see how many key concepts from Higher Order Logic you recall by answering the following questions!

If Higher-Order logics are more expressive, why do we limit ourselves to First-Order logic in this course?