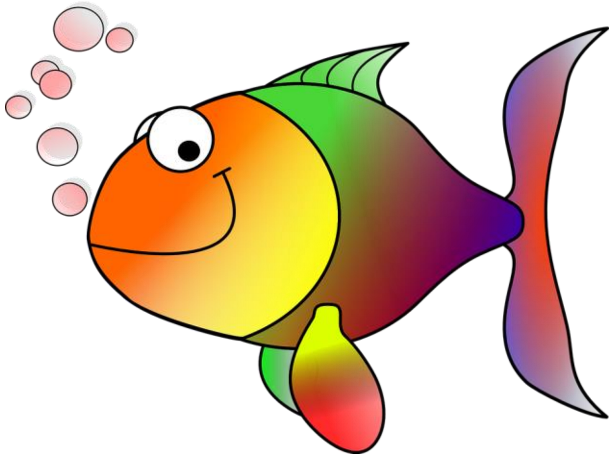
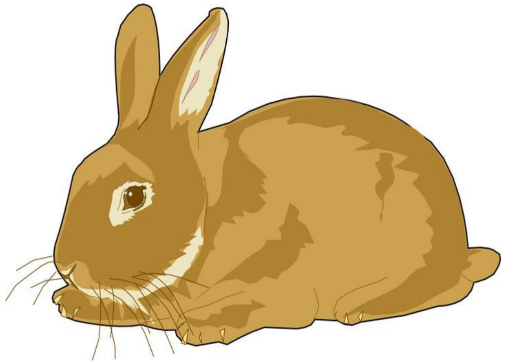
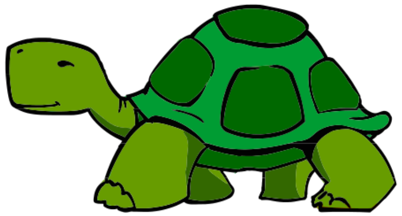
Inference in First-Order Logic







Review: First-Order Logic

First-order logic (like natural language) assumes the world contains:

- Objects: people, animals, numbers, courses, rooms...
- Relations: (between objects, or describing properties) Located(cs156, dh450), Adjacent([1, 1], [1, 2]), Breezy([1, 2])
- Functions: (mapping from object to object):

  BestFriend(...), Instructor(..)

Review: First-Order Logic

Constants represent objects:

[1, 1], 2, SJSU, Anna, Bob, Ryan, Joe, ...

Predicates represent relations:

Adjacent([1, 1], [1, 2]), Sibling(Ryan, Joe), Dating(Anna, Bob)

predicate($term_1$,...,$term_n$) is true

iff the objects referred to by $term_1$,...,$term_n$

are in the relation referred to by the predicate

Functions: represent a mapping between objects:

sqrt, BestFriend

Review: First-Order Logic

Variables also represent objects:

x, y, a, b, student, room, . . .

Connectives (same as in propositional logic)

$\wedge \; \vee \; \neg \; \Rightarrow \; \Leftrightarrow$

Equality: =

Refer to the same object

Quantifiers:

universal: $\forall$

existential: $\exists$

Review: First-Order Logic Examples

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

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

$\forall r\ \forall s\ Pit(r) \wedge Adjacent(r, s) \Rightarrow Breezy(s)$

All rooms adjacent to a pit are breezy

$\forall s\ Breezy(s) \Rightarrow \exists\ r\ Adjacent(r,s) \wedge Pit(r)$

Any breezy room is adjacent to a pit

Review: First-Order Logic Examples

There is at least one instructor:

$\exists x\ Instructor(x)$

There is exactly one instructor:

$\exists x\ Instructor(x) \land (\forall y\ (y \neq x) \Rightarrow \neg Instructor(y))$

There is at most one instructor:

$\forall x\ \forall y\ Instructor(x) \land Instructor(y) \Rightarrow x = y$

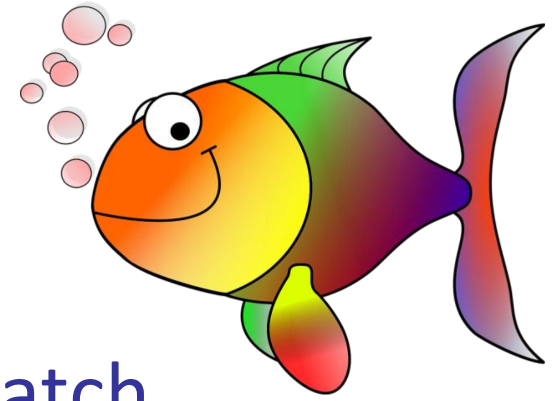
Victor has at least two jobs:

$\exists\ x\ \exists\ y\ Job(Victor, x) \land Job(Victor, y) \land (x \neq y)$

Today: Inference

Goal: define inference rules that work in First-Order Logic

Example:



Given: Fish(Nemo) and $\forall x$ Fish(x) $\Rightarrow$ Swims(x)

How can we infer Swims(Nemo)?

Problem: Fish(x) and Fish(Nemo) don't exactly match.

New concepts:

- Substitution/Instantiation
- Unification

Substitution

A substitution is a mapping from variables to constant symbols or other variables.

$Fish(x) \Rightarrow Swims(x)$

Substitution: $\theta = \{x/Nemo\}$

$Subst(\{x/Nemo\}, Fish(x) \Rightarrow Swims(x))$ is:

$Fish(Nemo) \Rightarrow Swims(Nemo)$

Also written as: $(Fish(x) \Rightarrow Swims(x))\ \theta$

Substitution

Example of a substitution: {x/A, y/F(B), z/w}

Properties of a substitution:

- Each variable is associated with at most one expression.
- No variable with an associated expression occurs within any associated expression (that is, no 'left side' appears on a 'right side')

Is { x/A, y/F(x), z/y} a valid substitution?

A. True

B. False

Universal Instantiation

Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall\, v\ \alpha}{\text{Subst}(\{v/g\},\ \alpha)}$$

for any variable v and ground term g.

A ground term is a term that does not contain variables.

$\forall x\ \text{Fish}(x) \Rightarrow \text{Swims}(x)$

$\text{Subst}(\{x/\text{Nemo}\},\ \text{Fish}(x) \Rightarrow \text{Swims}(x))$ gives us:

$\text{Fish}(\text{Nemo}) \Rightarrow \text{Swims}(\text{Nemo})$

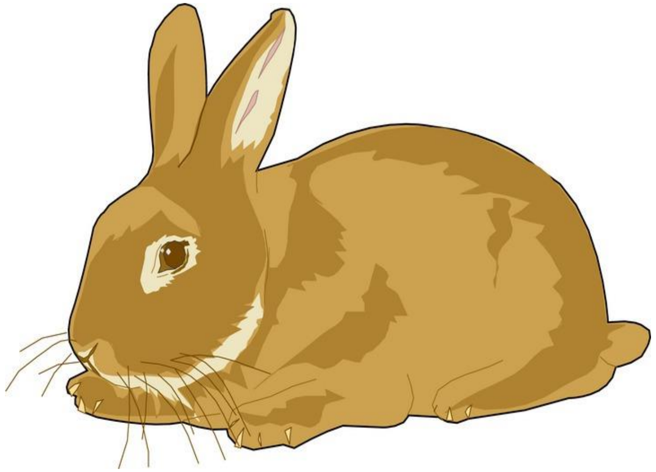
Universal Instantiation

∀ x, ∀ y Rabbit(x) ∧ Turtle(y) ⇒ Faster(x,y)

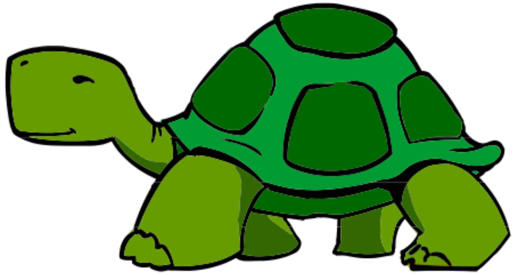
Substitution: {x/Cocoa, y/Speedy}

Replace x with Cocoa and y with Speedy:

Rabbit(Cocoa) ∧ Turtle(Speedy) ⇒ Faster(Cocoa, Speedy)



Cocoa



Speedy

Universal Instantiation

$\forall$ x King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

Substitution: {x/John}:

King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)

Substitution: {x/Richard}:

King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)

Ground term may contain a function:

Substitution: {x/Father(John)}:

King(Father(John)) $\wedge$ Greedy(Father(John)) $\Rightarrow$ Evil(Father(John))

Universal Instantiation

- Universal Instantiation can be applied several times to add new sentences
- The new KB is logically equivalent to the old

Existential Instantiation

An instantiation of an existentially quantified sentence α is entailed by it:

$$\frac{\exists v\ \alpha}{\mathrm{Subst}(\{v/k\},\ \alpha)}$$

for any variable *v*, and constant symbol *k* that does not appear elsewhere in the knowledge base.

∃x Bird(x) ∧ ¬ Flies(x)

Subst({x/C1}, Bird(x) ∧ ¬ Flies(x)) gives us:

Bird(C1) ∧ ¬ Flies(C1)

C1 is a new constant symbol, called a Skolem constant

Existential Instantiation?

$\forall x\ (\exists y\ Faculty(y) \wedge Advises(y, x))$

If we just replace the existential variable y with a constant: {y/C}:

$\forall x\ (Faculty(C) \wedge Advises(C, x))$

There is a faculty member who advises everyone?

Skolemization

Skolemization is a more general form of existential instantiation.

Each existential variable is replaced by a Skolem function of the enclosing universally quantified variable:

$$\frac{\forall\ x\ \exists v\ \alpha}{\forall\ x\ \mathrm{Subst}(\{v/F(x)\},\ \alpha)}$$

F is the Skolem function.

Skolemization

$$\frac{\forall\, x\, \exists v\, \alpha}{\forall\, x\, \mathrm{Subst}(\{v/F(x)\}, \alpha)}$$

Example:

$\forall$ x ($\exists$y Faculty(y) $\wedge$ Advises(y, x))

Subst({y/F(x)}, Faculty(y) $\wedge$ Advises(y, x))) gives us:

Faculty(F(x)) $\wedge$ Advises(F(x), x)   where F(x)could be the Advisor function.

Now we can write:

$\forall$ x Faculty(F(x)) $\wedge$ Advises(F(x), x)

Unification

Unification is the process of determining whether two expressions can be made identical by appropriate substitutions for their variables.

$\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

Example: Consider the expressions P(A, y, z) and P(x, B, z). To unify the two expressions, we look for a substitution that makes them identical.

P(A, y, z){x/A, y/B, z/C} = P(A, B, C}

P(x, B, z){x/A, y/B, z/C} = P(A, B, C}

The substitution {x/A, y/B, z/C} unifies the expressions P(A, y, z) and P(x, B, z).

Unification

Unification is the process of determining whether two expressions can be made identical by appropriate substitutions for their variables.

Unify(α, β) = θ if $\alpha\theta = \beta\theta$

α	β	θ
Knows(John, x)	Knows(John, Jane)	{x/Jane}
Knows(John, x)	Knows(y, Anna)	{x/Anna, y/John}
Knows(John, x)	Knows(y, Mother(y))	{y/John, x/Mother(John)}
Fish(x) $\land$ Swims(x)	Fish(Wanda) $\land$ Swims(y)	{x/Wanda, y/Wanda}
Knows(John, x)	Knows(x, Anna)	?

Unification

Unification is the process of determining whether two expressions can be made identical by appropriate substitutions for their variables.

Unify(α, β) = θ if $\alpha\theta = \beta\theta$ – Eliminate overlap of variables – Standardize apart

α	**β**	**θ**
Knows(John, x)	Knows(John, Jane)	{x/Jane}
Knows(John, x)	Knows(y, Anna)	{x/Anna, y/John}
Knows(John, x)	Knows(y, Mother(y))	{y/John, x/Mother(John)}
Fish(x) ∧ Swims(x)	Fish(Wanda) ∧ Swims(y)	{x/Wanda, y/Wanda}
Knows(John, x)	Knows(y, Anna)	?

Unification

Unification is the process of determining whether two expressions can be made identical by appropriate substitutions for their variables.

Unify(α, β) = θ if $\alpha\theta = \beta\theta$ – Eliminate overlap of variables – Standardize apart

α	β	θ
Knows(John, x)	Knows(John, Jane)	{x/Jane}
Knows(John, x)	Knows(y, Anna)	{x/Anna, y/John}
Knows(John, x)	Knows(y, Mother(y))	{y/John, x/Mother(John)}
Fish(x) ∧ Swims(x)	Fish(Wanda) ∧ Swims(y)	{x/Wanda, y/Wanda}
Knows(John, x)	Knows(y, Anna)	{x/Anna, y/John}

Unification

If there is one unifier for a set of expressions, there are usually many.

Some are more general than others.

Consider the two expressions: Knows(John,x) and Knows(y,z)

The following substitutions make them identical:

{y/John, x/z }: Knows(John,z) and Knows(John,z)

or

{y/John, x/John, z/John}: Knows(John,John) and Knows(John,John)

The first unifier ({y/John, x/z }) is more general than the second ({y/John, x/John, z/John}) .

Conjunctive Normal Form (CNF)

We’re almost ready to talk about resolution for first-order logic.

As with propositional logic, we’ll need to convert our Knowledge base to Conjunctive Normal Form first.

We’ll need a couple of properties that will help us deal with quantifiers.

Properties of Quantifiers

$\neg\forall x, p \equiv \exists x \neg p$

Not everyone likes chocolate

$\neg\forall x\ Likes(x, chocolate)$

There is someone who does not like chocolate

$\exists x \neg\ Likes(x, chocolate)$

Properties of Quantifiers

$\neg\exists\, x, p \equiv \forall\, x\, \neg p$

No one likes spiders

$\neg\, \exists\, x$ Likes(x, spiders)

Everyone does not like spiders:

$\forall\, x\, \neg$ Likes(x, spiders)

Conjunctive Normal Form (CNF)

A first order sentence without equality can be converted to an equivalent CNF sentence.

A CNF sentence is a conjunction of clauses, where each clause is a disjunction of literals.

$$\text{conjunction of } \underbrace{\text{disjunctions of literals}}_{\text{clauses}}$$

Literals can contain variables, which are assumed to be universally quantified.

Conjunctive Normal Form (CNF)

Everyone who loves all animals is loved by someone:

$\forall\, x\, (\forall\, y\, Animal(y) \Rightarrow Loves(x, y)) \Rightarrow (\exists\, y\, Loves(y, x))$

Step 1: Eliminate biconditionals and implications.

Replace $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$

$\forall\, x\, (\forall\, y\, Animal(y) \Rightarrow Loves(x, y)) \Rightarrow (\exists\, y\, Loves(y, x))$

$\forall\, x\, \neg\, (\forall\, y\, Animal(y) \Rightarrow Loves(x, y)) \vee (\exists\, y\, Loves(y, x))$

$\forall\, x\, \neg\, (\forall\, y\, Animal(y) \Rightarrow Loves(x, y)) \vee (\exists\, y\, Loves(y, x))$

$\forall\, x\, \neg\, (\forall\, y\, \neg Animal(y) \vee Loves(x, y)) \vee (\exists\, y\, Loves(y, x))$

Conjunctive Normal Form (CNF)

$\forall x \neg (\forall y \neg Animal(y) \lor Loves(x, y)) \lor (\exists y\ Loves(y, x))$

Step 2: Move ¬ inwards.

$\neg\forall y, p \equiv \exists y \neg p$

$\forall x (\exists y \neg(\neg Animal(y) \lor Loves(x, y))) \lor (\exists y\ Loves(y, x))$

$\neg(p \lor q) \equiv \neg p \land \neg q$ - De Morgan’s law

$\forall x (\exists y \neg\neg Animal(y) \land \neg Loves(x, y)) \lor (\exists y\ Loves(y, x))$

$\neg\neg p \equiv p$ - Double Negation Elimination

$\forall x (\exists y\ Animal(y) \land \neg Loves(x, y)) \lor (\exists y\ Loves(y, x))$

Conjunctive Normal Form (CNF)

$\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y\ Loves(y, x))$

Step 3: Standardize variables. Each quantifier should use a different variable.

$\forall x\ (\exists y\ Animal(y) \wedge \neg Loves(x, y)) \vee (\exists z\ Loves(z, x))$

Conjunctive Normal Form (CNF)

$\forall$ x ($\exists$ y Animal(y) $\wedge$ ¬Loves(x, y)) $\vee$ ($\exists$ z Loves(z, x))

Step 4: Skolemize. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables.

$\forall$ x ($\exists$ y Animal(y) $\wedge$ ¬Loves(x, y)) $\vee$ ($\exists$ z Loves(z, x))

$\forall$ x (Animal(F(x)) $\wedge$ ¬Loves(x, F(x))) $\vee$ Loves(G(x), x)

Conjunctive Normal Form (CNF)

$\forall x\ (Animal(F(x)) \land \neg Loves(x, F(x))) \lor Loves(G(x), x)$

Step 5: Drop universal quantifiers.

$(Animal(F(x)) \land \neg Loves(x, F(x))) \lor Loves(G(x), x)$

Conjunctive Normal Form (CNF)

(Animal(F(x)) ∧ ¬Loves(x, F(x))) ∨ Loves(G(x), x)

Step 6: Distribute ∨ over ∧.

(Animal(F(x)) ∨ Loves(G(x), x)) ∧ (¬Loves(x, F(x)) ∨ Loves(G(x), x))