Technische Hochschule Rosenheim

Modul - Introduction to AI (AI1)

Bachelor Programme AAI

08 - Propositional Logic and First-Order Logic

Prof. Dr. Marcel Tilly

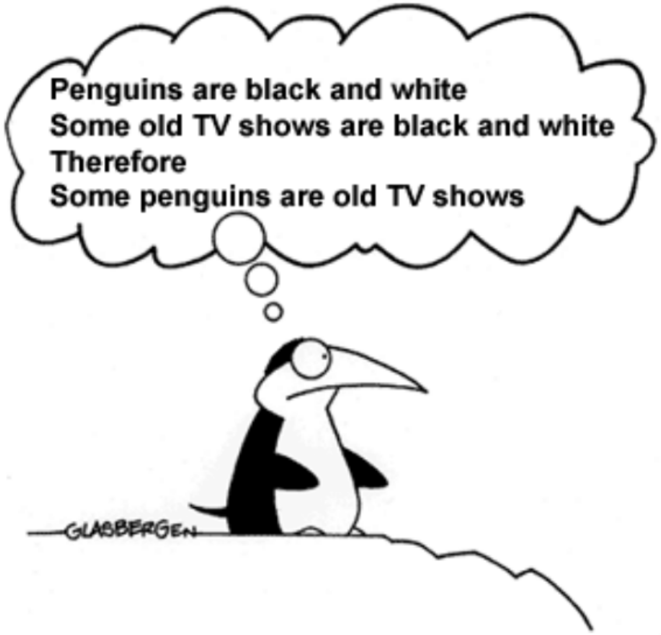
Faculty of Computer Science, Cloud Computing

Vers. 13.12.2021

Goals (formal)

- Students know about logic.
- Students can explain the concept of predicate logic.
- Students understand the difference between propositional logics and predicate logic.
- Students know about first-order logic.

RECAP: Logic


Penguins are black and white
Some old TV shows are black and white
Therefore
Some penguins are old TV shows
GLASBERGEN

RECAP: What is logic?

Logic is a truth-preserving system of inference

Truth-preserving: If the initial statements are true, the inferred statements will be true

System: a set of mechanistic transformations, based on syntax alone

Inference: the process of deriving (inferring) new statements from old statements

RECAP: Propositional Logic

- A *proposition* is a statement that is either *true* or *false*
- Examples:
  - "This class is about logic" --> (true)
  - "Today is Sunday" --> (false)
  - "It is currently snowing in Rosenheim" --> (???)
- Every proposition is *true* or *false*, but its truth value (true or false) may be unknown
- A *propositional statement* is one of:
  - A **simple proposition** denoted by a capital letter, e.g. ‘A’.
  - A negation of a propositional statement, e.g. ¬A : “not A”
  - Two propositional statements joined by a connective, e.g. A ∧ B or A ∨ B
  - If a connective joins complex statements, parenthesis are added, e.g. A ∧ (B∨C)

Rules of Inference

Implication “if ... then ...”

The inference runs as follows:

```
If Alice was in the bar, Alice was with her brother or her son.
Alice was in the bar.

---------------------------------------------------------------
Alice was with her brother or son.
```

- This rule is sometimes known as modus ponens, or “implication elimination,” since it tells us how to use an implication in an argument.

```
A ⇒ B
A

-----
B
```

Rules of Inference

Elimination

- The case of conjunction (“and”) conjunction with each conjunct

For example, informally we might argue:

```
Harry is friends with Ron and Hermione.
_______________________________________
Harry is friends with Hermione
```

In symbols:

```
        A ∧ B            A ∧ B
        _____            _____
        A                B
```

Rules of Inference

Negation

- In logical terms, showing “not A” amounts to showing that A leads to a contradiction.

For example:

```
It is not true that Harry did not pass the test.
------------------------------------------------
Harry passed the test
```

In symbols:

```
¬ (¬ A)
-------
A
```

Rules of Inference

Implication Elimination

```
If it is raining, then Harry is inside.
------------------------------------------------
It is not raining or Harry is inside.
```

Formal:

```
  A  ⇒  B
  -------
 ¬A  ∨  B
```

Rules of Inference

Biconditional Elimination

```
It is raining if and only if Harry is inside.
------------------------------------------------
if it is raining, then Harry is inside,
and if Harry is inside, then it is raining.
```

Formal:

```
A ⇔ B
----------------------
( A ⇒ B ) ∧ ( B ⇒ A )
```

Rules of Inference

De Morgan's Law

```
It is not true that both
Harry and Ron passed the test.
------------------------------------------------
Harry did not pass the test
or Ron did not pass the test.
```

Formal:

```
¬(A ∧ B)
--------
¬A ∨ ¬B
```

Rules of Inference

Biconditional Elimination

```
It is not true that
Harry and Ron passed the test.
------------------------------------------------
Harry did not pass the test
or Ron did not pass the test.
```

Formal:

```
¬(A ∨ B)
--------
¬A ∧ ¬B
```

Rules of Inference

Disjunction Elemination

```
(Ron is in the Great Hall) ∨ (Hermione is in the library)
Ron is not in the Great Hall
------------------------------------------------
Hermione is in the library
```

Formal:

```
A ∨ B
¬A
--------
B
```

Rules of Inference

Disjunction Elemination

```
(Ron is in the Great Hall) ∨ (Hermione is in the library)
(Ron is not in the Great Hall) ∨ (Harry is sleeping)
------------------------------------------------
(Hermione is in the library) ∨ (Harry is sleeping)
```

Formal:

```
 A  ∨  B             A
¬A  ∨  C            ¬A
---------           ---
 B  ∨  C            ()
```

Terms

- **clause** : a disjunction of literals, e.g. $P \lor Q \lor R$
- **conjunctive normal form**: logical sentence that is a conjunction of clauses, e.g. $(A \lor B \lor C) \land (D \lor \neg E) \land (F \lor G)$
- **Conversion to CNF**:
    - Eliminate biconditionals: turn $(\alpha \leftrightarrow \beta)$ into $(\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha)$
    - Eliminate implications: turn $(\alpha \rightarrow \beta)$ into $\neg\alpha \lor \beta$
    - Move ¬ inwards using De Morgan's Laws, e.g. turn $\neg(\alpha \land \beta)$ into $\neg\alpha \lor \neg\beta$
    - Use distributive law to distribute ∨ wherever possible

Inference by Resolution

- To determine if KB ⊨ α:
    - Check if (KB ∧ ¬α) is a contradiction?
    - If so, then KB ⊨ α.
    - Otherwise, no entailment.

To determine if KB ⊨ α:

- Convert (KB ∧ ¬α) to Conjunctive Normal Form.
- Keep checking to see if we can use resolution to produce a new clause.
- If ever we produce the empty clause (equivalent to False), we have a contradiction, and KB ⊨ α.
- Otherwise, if we can't add new clauses, no entailment.

Example

Does `(A ∨ B) ∧ (¬B ∨ C) ∧ (¬C) entail A`?

Check `(A ∨ B) ∧ (¬B ∨ C) ∧ (¬C) ∧ (¬A)` is a contradiction!

```
(¬B)(A ∨ B)     (¬B ∨ C)     (¬C)     (¬A) (A) ( )
```

First-Order *Logic*

Propositional Logic

- Propositional logic provides a good start at describing the general principles of logical reasoning.
- Propositional logic has nice properties:
    - Propositional logic is declarative: pieces of syntax correspond to facts
    - Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
    - Propositional logic is compositional: meaning of `B1,1 ∧ P1,2` is derived from meaning of B1,1 and of P1,2
    - Meaning in propositional logic is context-independent (unlike natural language, where meaning depends on context)
- **Limitation**
    - Propositional logic has very limited expressive power, unlike natural language. E.g., we cannot express "pits cause breezes in adjacent squares" except by writing one sentence for each square

Requirements

- Propositional logic does not give us the means to express a general principle that tells us that
  - if Alice is with her son on the beach, then her son is with Alice
  - the general fact that no child is older than his or her parent
  - if someone is alone, they are not with someone else.
- To express principles like these, we need a way to talk about objects and individuals, as well as their properties and the relationships between them.
- These are exactly what is provided by a more expressive logical framework known as *first-order logic*.

Propositional vs. Predicate Logic

- In propositional logic, each possible atomic fact requires a separate unique propositional symbol.
- Whereas propositional logic assumes that a world contains facts, predicate logic assumes the world contains of:
  - *Objects* (terms): people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries ...
  - *Properties* (unary predicates on terms): red, round, bogus, prime, ...
  - *Relations* (n-ary predicates on terms): brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . .
  - *Functions* (mappings from terms to other terms): father of, best friend, third inning of, one more than, end of .
- Allows more flexible and compact representation of knowledge
  - Move(x, y, z) for person x moved from location y to z.

First-Order syntax elements

- **Constants**: KingJohn, 2, U CB, . . .
- **Predicates**: Brother, >, . . .
- **Variables**: x, y, a, b, . . .
- **Connectives**: $\wedge$ $\vee$ $\neg$ $\Rightarrow$ $\Leftrightarrow$
- **Equality**: =
- **Quantifiers**: $\forall$ $\exists$
- **Functions**: Sqrt, Lef tLegOf, . .

First-Order Logic

Technische Hochschule Rosenheim

Objects/ Terms

- Objects are represented by **terms**:
  - **Constants**: BuildingA, John
  - **Function symbols**: *father-of*, *successor*, *plus*
  - An *n*-ary function maps a tuple of n terms to another term: `father-of(John)`, `succesor(0)`, `plus(plus(1,1),2)`
- **Terms** are simply names for objects.
- **Logical functions** are not procedural as in programming languages. They do not need to be defined, and do not really return a value.

```
⟨term⟩  →  function( ⟨term⟩ ,. . . )
           | constant
           | variable
```

First-Order Logic

Predicates

- Propositions are represented by a **predicate** applied to a tuple of terms. A predicate represents a property of or relation between terms that can be *true* or *false*:
  - Brother(John, Fred)
  - Left-of(Square1, Square2)
  - GreaterThan(plus(1,1), plus(0,1))

```
⟨atomic sentence⟩  → predicate( ⟨term⟩ ,. . . )
                    |  ⟨term⟩ = ⟨term⟩
```

Sentences in First-Order Logic

- An *atomic sentence* is simply a predicate applied to a set of terms.
  - Owns(John,Car1)
  - Sold(John,Car1,Fred)
  - Semantics is *true* or *false* depending on the interpretation, i.e. is the predicate true of these arguments.
- The standard propositional connectives (∨ ¬ ∧ ⇒ ⇔) can be used to construct complex sentences:
  - `Owns(John,Car1) ∨ Owns(Fred, Car1)`
  - `Sold(John,Car1,Fred) ⇒ ¬Owns(John, Car1)`
  - Semantics same as in propositional logic.

```
⟨complex sentence⟩  → ¬  ⟨sentence⟩
                       | ( ⟨sentence⟩  [∧ | ∨ | ⇒ | ⇔ ]  ⟨sentence⟩
```

Quantifiers

Universal quantification

- Allows statements about entire collections of objects rather than having to enumerate the objects by name.

Universal quantifier: $\forall x$

Asserts that a sentence is true for all values of variable x.

$\forall \langle\text{variables}\rangle \; \langle\text{sentence}\rangle$: $\forall\, x\, P$ is true in a model m iff P is true with x being each possible object in the model.

- $\forall x\ Loves(x, FOL)$
- $\forall x\ Whale(x) \Rightarrow Mammal(x)$
- $\forall x\ Grackles(x) \Rightarrow Black(x)$
- $\forall x\ (\forall y\ Dog(y) \Rightarrow Loves(x,y)) \Rightarrow (\forall z\ Cat(z) \Rightarrow Hates(x,z))$

Quantifiers

Existential quantification

Existential quantifier: ∃

Asserts that a sentence is true for at least one value of a variable x.

∃ ⟨variables⟩ ⟨sentence⟩ :∃ x P is true in a model m iff P is true with x being some possible object in the model.

- ∃x Loves(x, FOL)
- ∃x(Cat(x) ∧ Color(x,Black) ∧ Owns(Mary,x))
- ∃x(∀y Dog(y) ⇒ Loves(x,y)) ∧ (∀z Cat(z) ⇒ Hates(x,z))

Use of Quantifiers

- Universal quantification naturally uses implication:
  - $\forall x\ Whale(x) \land Mammal(x)$
  - Says: *Everything in the universe is both a whale and a mammal.*
- Existential quantification naturally uses conjunction:
  - $\exists x\ Owns(Mary,x) \Rightarrow Cat(x)$
  - Says: *Either there is something in the universe that Mary does not own or there exists a cat in the universe.*
  - $\forall x\ Owns(Mary,x) \Rightarrow Cat(x)$
  - Says: *All Mary owns is cats (i.e. everthing Mary owns is a cat). Also true if Mary owns nothing.*
  - $\forall x\ Cat(x) \Rightarrow Owns(Mary,x)$
  - Says: *Mary owns all the cats in the universe.*
  - Also true if there are no cats in the universe.

Properties of quantifiers

- The order of quantifiers of the same type doesn't matter
  - $\forall x \forall y (\mathrm{Parent}(x,y) \land \mathrm{Male}(y) \Rightarrow \mathrm{Son}(y,x))$
  - $\exists x \exists y (\mathrm{Loves}(x,y) \land \mathrm{Loves}(y,x))$
  - $\forall x \forall y$ is the same as $\forall y \forall x$
  - $\exists x \exists y$ is the same as $\exists y \exists x$
- Quantifier duality: each can be expressed using the other
  - $\forall x\ \mathrm{Likes}(x, \mathrm{IceCream}) \equiv \neg \exists x\ \neg \mathrm{Likes}(x, \mathrm{IceCream})$
  - $\exists x\ \mathrm{Likes}(x, \mathrm{Broccoli}) \equiv \neg \forall x\ \neg \mathrm{Likes}(x, \mathrm{Broccoli})$

Nesting Quantifiers

- The order of mixed quantifiers does matter:
  - $\exists x \forall y$ is not the same as $\forall y \exists x$
  - $\forall x \exists y(Loves(x,y))$
  - Says: *Everybody loves somebody, i.e. everyone has someone whom they love.*
  - $\exists y \forall x(Loves(x,y))$
  - Says: *There is someone who is loved by everyone in the universe.*
  - $\forall y \exists x(Loves(x,y))$
  - Says: *Everyone has someone who loves them.*
  - $\exists x \forall y(Loves(x,y))$
  - Says: *There is someone who loves everyone in the universe.*

Variable Scope

- The scope of a variable is the sentence to which the quantifier syntactically applies.
- As in a block structured programming language, a variable in a logical expression refers to the closest quantifier within whose scope it appears.
  - $\exists x\,(Cat(x) \wedge \forall x(Black\,(x)))$
  - The x in Black(x) is universally quantified
  - Say: *Cats exist and everything is black.*
- In a well-formed formula (wff) all variables should be properly introduced:
  - $\exists x P(y)$ **not** well-formed
- A ground expression contains no variables.

Relation Between Quantifiers

- General Identities
  - $\forall x \neg P \Leftrightarrow \neg \exists x P$
  - $\neg \forall x P \Leftrightarrow \exists x \neg P$
  - $\forall x P \Leftrightarrow \neg \exists x \neg P$
  - $\exists x P \Leftrightarrow \neg \forall x \neg P$
  - $\forall x P(x) \wedge Q(x) \Leftrightarrow \forall x P(x) \wedge \forall x Q(x)$
  - $\exists x P(x) \vee Q(x) \Leftrightarrow \exists x P(x) \vee \exists x Q(x)$

Can you explain?

Equality

- Can include equality as a primitive predicate in the logic, or require it to be introduced and axiomitized as the identity relation.
- Useful in representing certain types of knowledge:

```
∃x∃y(Owns(Mary, x) ∧ Cat(x)  ∧ Owns(Mary,y) ∧ Cat(y)
∧ ¬(x=y))

Mary owns two cats.  Inequality needed to insure x and y
are distinct.
```

```
∀x ∃y married(x, y) ∧ ∀z(married(x,z) ⇒ y=z)

Everyone is married to exactly one person.
Second conjunct is needed to guarantee there is only one unique spouse.
```

Higher-Order Logic

- FOL is called *first-order* because it allows quantifiers to range over objects (terms) but not properties, relations, or functions applied to those objects.
- Second-order logic allows quantifiers to range over predicates and functions as well:
    - $\forall x \forall y [(x=y) \Leftrightarrow (\forall p\ p(x) \Leftrightarrow p(y))]$
    - Says: *Two objects are equal if and only if they have exactly the same properties.*
    - $\forall f \forall g [(f=g) \Leftrightarrow (\forall x\ f(x) = g(x))]$
    - Says: *Two functions are equal if and only if they have the same value for all possible arguments.*
- Third-order would allow quantifying over predicates of predicates, etc.
    - For example, a second-order predicate would be Symetric(p) stating that a binary predicate p represents a symmetric relation.

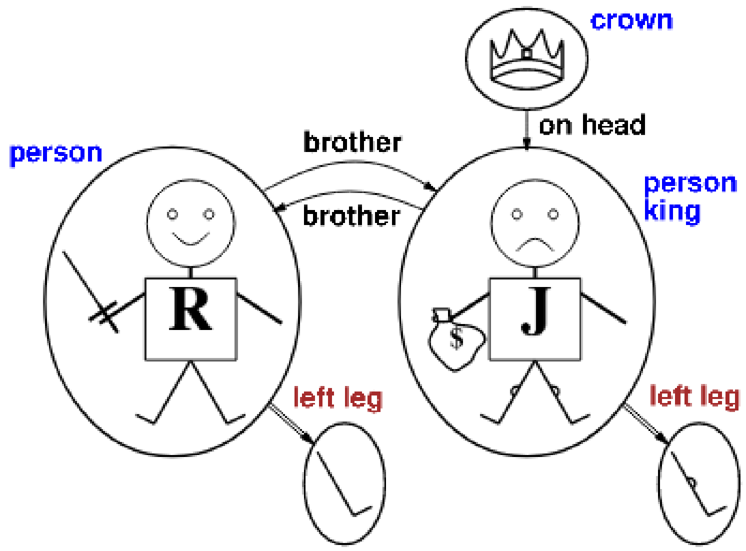
Logical KB

- KB contains general axioms describing the relations between predicates and definitions of predicates using ⇔.
  - ∀x,y Bachelor(x) ⇔ Male(x) ∧ Adult(x) ∧ ¬∃yMarried(x,y).
  - ∀x Adult(x) ⇔ Person(x) ∧ Age(x) >=18.
- May also contain specific ground facts.
  - Male(Bob), Age(Bob)=21, Married(Bob, Mary)
- Can provide queries or goals as questions to the KB:
  - Adult(Bob)?
  - Bachelor(Bob)?
- If query is existentially quantified, would like to return substitutions or binding lists specifying values for the existential variables that satisfy the query.
  - ∃x Adult(x) ? --> {x/Bob}
  - ∃x Married(Bob,x) ? --> {x/Mary}
  - ∃x,y Married(x,y) ? --> {x/Bob, y/Mary}

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- A model contains ≥ 1 objects and relations among them
- An interpretation specifies referents for
  - constant symbols $\rightarrow$ objects
  - predicate symbols $\rightarrow$ relations
  - function symbols $\rightarrow$ functional relations
- An atomic sentence predicate(term1, . . . , termn) is true iff the objects referred to by term1, ... , termn are in the relation referred to by predicate

Models for FOL: Example



How does th FOL look like?

- constants, predicates, functions

Model Checking for FOL

- Entailment in propositional logic can be computed by enumerating models
- We can also enumerate the FOL models for a given KB:

```
For each number of domain elements n from 1 to ∞
    For each k-ary predicate Pk in the vocabulary
       For each possible k-ary relation on n objects
           For each constant symbol C in the vocabulary
               For each choice of referent for C from n objects . . .
```

- LISP:

```
(forall ?x (forall ?y (implies (and (parent ?y ?x) (male ?x))(son ?x ?y)))
```

- Enumerating FOL models is very inefficient

FOL Inference

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens and forward & backward chaining
- Resolution

Universal instantiation (UI)

- Whenever a KB contains a universally quantified sentence, we may add to the KB any instantiation of that sentence, where the logic variable v is replaced by a concrete ground term g:

```
∀ v w
----------------
Subst({v/g}, w)
```

- E.g., ∀ x King(x) ∧ Greedy(x) ⇒ Evil(x) yields
  - King(John) ∧ Greedy(John) ⇒ Evil(John)
  - King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard)
  - King(Father(John)) ∧ Greedy(Father(John)) ⇒ Evil(Father(John))

Existential instantiation (EI)

- Whenever a KB contains a existentially quantified sentence ∃ v α, we may add a single instantiation of that sentence to the KB, where the logic variable v is replaced by a Skolem constant symbol k which must not appear elsewhere in the knowledge base:

```
∃ v w
---------------
Subst({v/k}, w)
```

- E.g., ∃ x Crown(x) ∧ OnHead(x, John) yields
  - Crown(C1) ∧ OnHead(C1, John)

provided C1 is a new constant symbol, called a Skolem constant

Instantiations

- UI can be applied several times to add new sentences; the new KB is logically equivalent to the old.
- EI can be applied once to replace the existential sentence; the new KB is not equivalent to the old, but is satisfiable iff the old KB was satisfiable.

Reduction to propositional inference

Instantiating all quantified sentences allows us to ground the KB, that is, to make the KB propositional

Example

```
  ∀ x King(x) ∧ Greedy(x) ⇒ Evil(x)
  King(John)
  Greedy(John)
  Brother(Richard, John)
```

Instantiating the universal sentence in all possible ways, we have

```
King(John) ∧ Greedy(John) ⇒ Evil(John)
King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard)
King(John)
Greedy(John)
Brother(Richard, John)
```

The new KB is propositionalized: proposition symbols are King(John), Greedy(John), Evil(John), King(Richard) etc.

Inefficiency of naive propositionalization

Propositionalization generates lots of irrelevant sentences.

Example:

```
∀ x King(x) ∧ Greedy(x) ⇒ Evil(x)
King(John)
∀ y Greedy(y)
Brother(Richard, John)
```

- Propositionalization produces not only Greedy(John), but also Greedy(Richard) which is irrelevant for a query Evil(John)
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations
- With function symbols, it gets much much worse!

Unification

- Instead of instantiating quantified sentences in all possible ways, we can compute specific substitutions "that make sense". These are substitutions that unify abstract sentences so that rules (Horn clauses, GMP, see next slide) can be applied.
- In the previous example, the "Evil-rule" can be applied if we can find a substitution θ such that King(x) and Greedy(x) match King(John) and Greedy(y). Namely, θ = {x/John, y/John} is such a substitutions.
- We write θ unifies(α, β) iff $\alpha\theta = \beta\theta$

p	**q**	**θ**
Knows(John, x)	Knows(John, Jane)	{x/Jane}
Knows(John, x)	Knows(y, OJ)	{x/OJ, y/John}
Knows(John, x)	Knows(y, Mother(y))	{y/John, x/Mother(John)}
Knows(John, x)	Knows(x, OJ)	fail

- Standardizing apart the names of logic variables eliminates the overlap of variables, e.g., Knows(z17, OJ)

Generalized Modus Ponens (GMP)

- For every substitution θ such that ∀i : θ unifies(p'i, pi) we can apply:

```
p1', p2', . . . , pn', (p1 ∧ p2 ∧ . . . ∧ pn ⇒ q)
---------------------------------------------------
                       qθ
```

Example:

```
    p1' is King(John)         p1 is King(x)
    p2' is Greedy(y)          p2 is Greedy(x)
    θ is {x/John, y/John}     q is Evil(x)
    qθ is Evil(John)
```

- This GMP assumes a KB of definite clauses (exactly one positive literal). By default, all variables are assumed universally quantified.

Conversion to CNF

- Everyone who loves all animals is loved by someone:

```
∀ x [∀ y Animal(y) ⇒ Loves(x, y)] ⇒ [∃ y Loves(y, x)]
```

- Eliminate biconditionals and implications

```
∀ x [¬∀ y ¬Animal(y) ∨ Loves(x, y)] ∨ [∃ y Loves(y, x)]
```

- Move ¬ inwards: ¬∀ x, p ≡ ∃ x ¬p, ¬∃ x, p ≡ ∀ x ¬p:

```
∀ x [∃ y ¬(¬Animal(y) ∨ Loves(x, y))] ∨ [∃ y Loves(y, x)]
∀ x [∃ y ¬¬Animal(y) ∧ ¬Loves(x, y)] ∨ [∃ y Loves(y, x)]
∀ x [∃ y Animal(y) ∧ ¬Loves(x, y)] ∨ [∃ y Loves(y, x)]
```

Conversion to CNF contd.

- Standardize variables: each quantifier should use a different one

```
∀ x [∃ y Animal(y) ∧ ¬Loves(x, y)] ∨ [∃ z Loves(z, x)]
```

- Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

```
∀ x [Animal(F (x)) ∧ ¬Loves(x, F (x))] ∨ Loves(G(x), x)
```

- Drop universal quantifiers:

```
[Animal(F (x)) ∧ ¬Loves(x, F (x))] ∨ Loves(G(x), x)
```

- Distribute ∧ over ∨:

```
[Animal(F (x)) ∨ Loves(G(x), x)] ∧ [¬Loves(x, F (x)) ∨ Loves(G(x), x)]
```

Resolution: Modus Ponens

Example:

```
¬Rich(x)  ∨  Unhappy(x)
 Rich(Ken)
---------------------------------
        Unhappy(Ken)

with θ = {x/Ken}
```

- Apply resolution steps to CN F (KB ∧ ¬α); complete for FOL

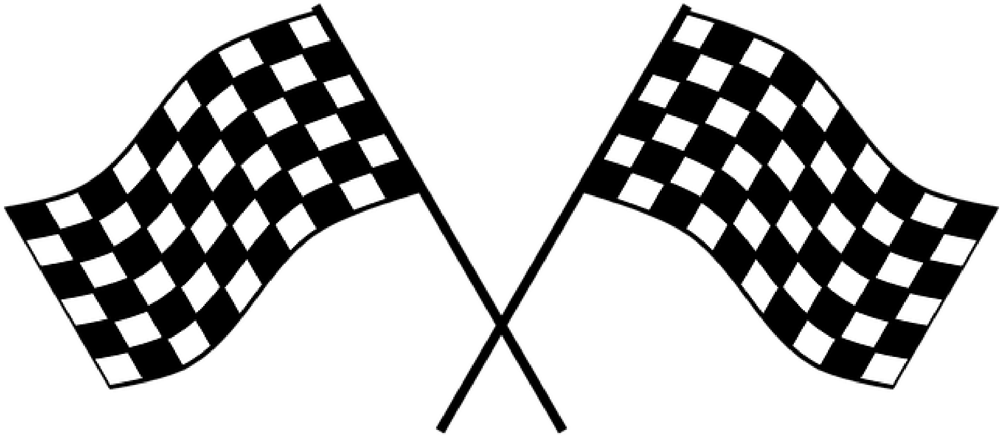
Sample Representations

- There is a barber in town who shaves all men in town who do not shave themselves.
  - $\exists x\,(Barber(x) \land InTown(x) \land \forall y\,(Man(y) \land InTown(y) \land \neg Shave(y,y) \Rightarrow Shave(x,y)))$
- There is a barber in town who shaves only and all men in town who do not shave themselves.
  - $\exists x\,(Barber(x) \land InTown(x) \land \forall y\,(Man(y) \land InTown(y) \land \neg Shave(y,y) \Leftrightarrow Shave(x,y)))$

> Classic example of Bertrand Russell used to illustrate a paradox in set theory: Does the set of all sets contain itself?

Summary

- Propositional Logic
- Rules of Inference
- Predicate Logic
- First-Order Logic



More Links

- CS50’s Introduction to Artificial Intelligence with Python: https://cs50.harvard.edu/ai/2020/weeks/1/