1 Logic & AI: First-Order Logic

As established before, Propositional Logic does not have any notion of relations or objects. This does of course limit the possible applications of Propositional Logic.

First-Order Logic (FOL) extends on propositional logic to allow for relations and objects.

Some commonly used terms in FOL and scientific papers:

Axioms

Basic facts about the domain, the "initial" knowledge base.

Theorems

Statements that are logically derived from axioms.

1.1 Elements in FOL

- **Objects:**
  - Represent entities in the real world - Can be named accordingly: Person1, John, Earth...
- **Relations:**
  - Represent relations between objects - Can be named accordingly: Has(·, ·), Is[Something](·)...
  - Relations with only one object are called **Unary Relations or Properties:** Has[Something](·), Is[Something](·)...
- **Functions:**
  - Functions refer to objects without a name
  - E.g. Roommate(·) $\rightarrow$ Roommate(Person1) $=$ Person2
  - Can be used to encode integers and data structures:
    - $*$ 0, succ(0) $=$ 1, succ(1) $=$ 2, succ(2) $=$ 3...
    - $*$ tree(value1, tree(value2,...), tree(value3,...))
  - Not specific to specific object: Roommate(Umbrella) is valid, but might be nonsensical

1.2 Quantifiers

Quantifiers can be used to refer to multiple objects at once.

Universal: For All $\forall$

Asserts that a statement is true for all objects.
$\forall x$: Lion(x) $\Rightarrow$ Cat(x): All lions are cats

Existential: Exists $\exists$

Asserts that a statement is true for at least one object.
$\exists x$: Cat(x) $\Rightarrow \neg$(Lion(x)): At least one cat is not a lion

$\forall x : A$ is equivalent to $\neg(\exists x : \neg(A))$.

1.3 Substitution

Substitution: SUBST(·)

The SUBST method replaces one or more variables with something else in a sentence.

Example:

SUBST({x / John}, IsHealthy(x) $\Rightarrow \neg$(HasACold(x)))
becomes
IsHealthy(John) $\Rightarrow \neg$(HasACold(John))

1.3.1 Instantiating Quantifiers with SUBST

Universal $\forall$

Assuming a statement $\forall x : A$ we can obtain a new clause for **any** concrete **ground term g**:
SUBST({x / g}, A)

Example:

$$\forall x : x > 0 \Rightarrow x^2 > 0$$

Substituting any concrete ground term g, in this case $g = 3$, yields:
SUBST({x / 3}, $x > 0 \Rightarrow x^2 > 0$) =
$3 > 0 \Rightarrow 3^2 > 0$

Existential $\exists$

Assuming a statement $\exists x : A$ we can obtain a new clause for **only one Skolem constant k**:
SUBST({x / k}, A)

Example:

$$\exists x : x^2 = 9$$

In this case we do not assign a concrete value, but assign a "placeholder", the **Skolem constant k**, that can later be instantiated:
SUBST({x / k}, $x^2 = 9$) = $k^2 = 9$

Skolem Constant

Important:

- k is a constant, that does not appear elsewhere in the knowledge base
- The result of SUBST({x / k}, A) is a clause that **replaces** the original clause, as they are **equivalent**
-

One must act carefully when Instantiating existentials **after** universals:

Assume the statement $\forall y \exists x : \text{Parent}(x, y)$: "Every person y has a parent x".

Correct Instantiation:

1. Choose a specific person y - Let's say y = John
2. The statement then becomes $\exists x : \text{Parent}(x, \text{John})$: "John has a parent x".
3. We can now substitute x with a skolem constant k_{John} to get a more specific sentence: $\text{Parent}(k_{\text{John}}, \text{John})$: "John has a parent k_{John}".
4. Assigning a concrete value to k_{John} later on does not disturb this logic.

Incorrect Instantiation:

1. Assign x a skolem constant k
2. The statement then becomes $\forall y : \text{Parent}(k, y)$: "Every person y has a parent k". This would mean that every person y has **the same parent** k, which might not be correct.

1.4 Generalized Modus Ponens

1.4.1 Unification

Assume two sentences: $\forall x : \text{Loves}(\text{John}, x)$ "John loves everything" and $\forall y : (\text{Loves}y, \text{Jane} \Rightarrow \text{FeelsAppreciatedBy}(\text{Jane}, y))$ "Jane feels appreciated by everything that loves her".

We can now use substitution:

1. {x / Jane} $\rightarrow$Loves(John, Jane)
2. {y / John} $\rightarrow$Loves(John, Jane) $\Rightarrow$ FeelsAppreciatedBy(Jane, John)
3. As 1 fulfills the condition of 2, we can infer that FeelsAppreciatedBy(Jane, Jane)

1.5 First-Order Conjunctive Normal Form

1. Convert to **Negation Normal Form**: Negation symbols only occur immediately before predicate symbols
2. If variable names are used twice within scopes of different quantifiers, rename one of them such that the name is not used elsewhere
3. **Skolemize statements:**
   (a) Move quantifiers out, so that we got $\forall x_1, x_2 \dots \exists y_1, y_2 \dots : A$. Quantifiers only occur in the prefix, not inside conjuncts.
   (b) Replace existentially quantified variables with skolem constants.
   (c) Discard universal quantifiers
4. **Convert** into clause set

Example:
Assume the sentence: $\forall x, y : \text{eats}(x, y) \wedge \neg(\text{killed}(x)) \Rightarrow \text{food}(y)$ "Anything anyone eats and is not killed is food"

1. Eliminate Implications: $\forall x, y : \neg(\text{eats}(x, y) \wedge \neg(\text{killed}(x))) \vee \text{food}(y)$
2. Move negations inwards (De Morgan): $\forall x, y : \neg(\text{eats}(x, y)) \vee \text{killed}(x) \vee \text{food}(y)$
3. Drop universal quantifiers: $\neg(\text{eats}(x, y)) \vee \text{killed}(x) \vee \text{food}(y)$

Inference in FOL is **not decidable** but **semidicidable**. This means that we can not always conclude that a sentence is not entailed.

1.6 Prolog

Prolog (Programming in Logic) is a FOL based logic programming language.

It is based on three basic components:

- **Facts:** Statements that are unconditionally true. E.g.:
  - `cat(john)` "John is a cat".
  - `parent(john, tom)` "John is the parent of Tom"
- **Rules:** Conditional statements that define relationships. E.g.:
  - `animal(X) :- cat(X)` "X is an animal *if* X is a cat"
- **Queries:** Questions asked to the program to derive answers. If multiple answers are possible, Prolog will return one per iteration in order. E.g.:
  - `?- cat(tom)` "Is Tom a cat?"

1.6.1 Atoms and Variables

Atom

A constant term that represents fixed values. It is denoted by lowercase letters.
Not every constant is an atom: Numbers, empty lists, some constructs (e.g. `parent(john, tom)`) are not atoms but constant.

Variable

A term that represents unknown or general values. It is denoted by uppercase letters or _.

1.6.2 Prolog as a Database

Prolog can be regarded as a standard relational database, that stores facts and rules, which can be accessed and manipulated using Prolog queries. One caveat is that prologs backtracking strategy to retrieve all queries can be inefficient.

1.7 Gödels Incompleteness Theorem

Gödel's Incompleteness Theorem states that in any formal arithmetic system, there exists a sentence that cannot be proven. This is due to the fact, that there is no formal system that is sufficiently powerful so that it is both **complete** (able to prove all statements within the system) and **consistent** (no contradiction exists).