Modul - Introduction to AI (AI1)

Bachelor Programme AAI

08 - Propositional Logic and First-Order Logic

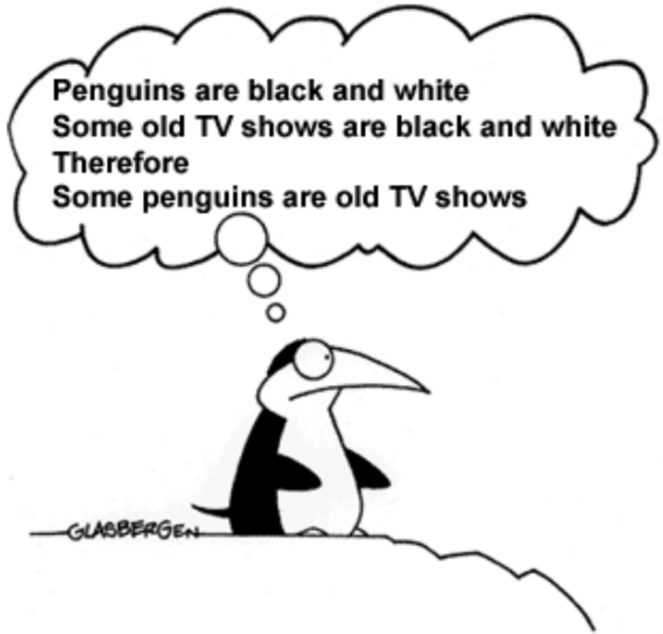
Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

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?

Technische Hochschule Rosenheim

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

Technische Hochschule **Rosenheim**

- 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. $\neg A$: "not A"
  - Two propositional statements joined by a connective, e.g. $A \wedge B$ or $A \vee B$
  - If a connective joins complex statements, parenthesis are added, e.g. $A \wedge (B \vee C)$

Rules of Inference

Technische Hochschule Rosenheim

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

Technische Hochschule Rosenheim

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 ∨ A
```

Rules of Inference

Biconditional Elimination

```
It is raining if and only if Harry is inside.
------------------------------------------------
f 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
--------
Q
```

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 $\models \alpha$:
  - Check if (KB $\land \neg\alpha$) is a contradiction?
  - If so, then KB $\models \alpha$.
  - Otherwise, no entailment.

To determine if KB $\models \alpha$:

- Convert (KB $\land \neg\alpha$) 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 $\models \alpha$.
- 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

Technische Hochschule Rosenheim

- Propositional logic provides a good start at describing the general principles of logical reasoning.
- 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.
- If there are n people and m locations, representing the fact that some person moved from one location to another requires nm^2 separate symbols.
- Predicate logic includes a richer ontology:
  - objects (terms)
  - properties (unary predicates on terms)
  - relations (n-ary predicates on terms)
  - functions (mappings from terms to other terms)
- Allows more flexible and compact representation of knowledge
  - Move(x, y, z) for person x moved from location y to z.

First-Order Logic

Technische Hochschule Rosenheim

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.
  - Allows for the representation of an infinite number of terms.

First-Order Logic

Technische Hochschule Rosenheim

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))
- In a given interpretation, an n-ary predicate can defined as a function from tuples of n terms to {True, False} or equivalently, a set tuples that satisfy the predicate:
    - `{<John, Fred>, <John, Tom>, <Bill, Roger>, ...}`

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.

Quantifiers

- 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

- $\forall x$ Loves(x, FOPC)
- $\forall x$ 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 quantifier: ∃

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

- $\exists x$ Loves(x, FOPC)
- $\exists x(Cat(x) \land Color(x,Black) \land Owns(Mary,x))$
- $\exists x(\forall y\ Dog(y) \Rightarrow Loves(x,y)) \land (\forall z\ Cat(z) \Rightarrow Hates(x,z))$

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.

Nesting Quantifiers

- The order of quantifiers of the same type doesn't matter
  - $\forall x \forall y (Parent(x,y) \land Male(y) \Rightarrow Son(y,x))$
  - $\exists x \exists y (Loves(x,y) \land Loves(y,x))$
- The order of mixed quantifiers does matter:
  - $\forall x \exists y (Loves(x,y))$
  - 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

- Universal and existential quantification are logically related to each other:
  - $\forall x\ \neg Love(x,Saddam) \Leftrightarrow \neg\exists x\ Loves(x,Saddam)$
  - $\forall x\ Love(x,Princess\text{-}Di) \Leftrightarrow \neg\exists x\ \neg Loves(x,Princess\text{-}Di)$
- General Identities
  - $\forall x\ \neg P \Leftrightarrow \neg\exists x\ P$
  - $\neg\forall x\ P \Leftrightarrow \exists x\ \neg P$
  - $\forall x\ P \Leftrightarrow \neg\exists x\ \neg P$
  - $\exists x\ P \Leftrightarrow \neg\forall x\ \neg P$
  - $\forall x\ P(x) \land Q(x) \Leftrightarrow \forall x P(x)\ \land\ \forall x Q(x)$
  - $\exists x\ P(x) \lor Q(x) \Leftrightarrow \exists x P(x)\ \lor\ \exists x Q(x)$

Equality

Technische Hochschule Rosenheim

- 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

- FOPC 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.

Notational Variants

- In Prolog, variables in sentences are assumed to be universally quantified and implications are represented in a particular syntax.
    - son(X, Y) : parent(Y,X) , male(X).
- In Lisp, a slightly different syntax is common.

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

```
- Lisp invented by John McCarthy
```

- Generally argument order follows the convention that P(x,y) in English would read “x is (the) P of y”

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}

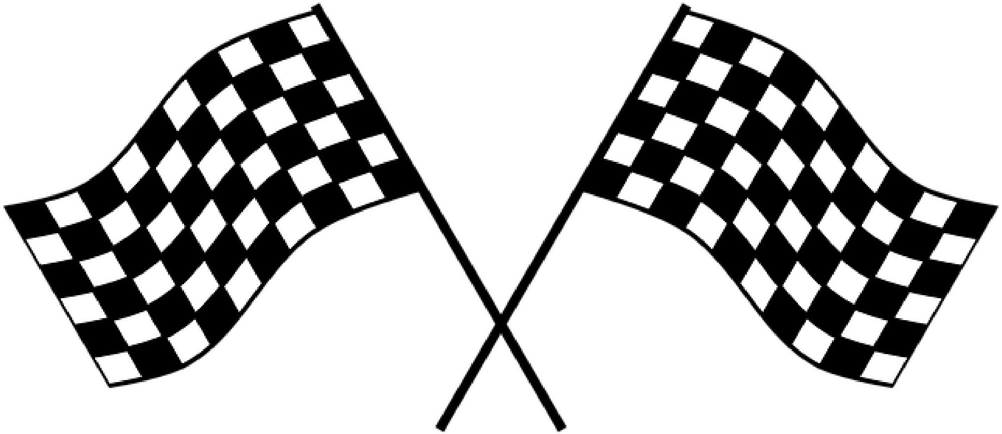
Sample Representations

- 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/