CS 2740 Knowledge Representation
Lecture 8

First-order logic

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Limitations of propositional logic

World we want to represent and reason about consists of a number of objects with variety of properties and relations among them

Propositional logic:

- Represents statements about the world without reflecting this structure and without modeling these entities explicitly

Consequence:

- some knowledge is hard or impossible to encode in the propositional logic.
- Two cases that are hard to represent:
  - **Statements about similar objects, relations**
  - **Statements referring to groups of objects**.

First-order logic (FOL)

- More expressive than **propositional logic**

- **Eliminates deficiencies of PL by:**
  - Representing objects, their properties, relations and statements about them;
  - Introducing variables that refer to an arbitrary objects and can be substituted by a specific object
  - Introducing quantifiers allowing us to make statements over groups objects without the need to represent each of them separately

Logic

Logic is defined by**:**

- **A set of sentences**
  - A sentence is constructed from a set of primitives according to syntax rules.
- **A set of interpretations**
  - An interpretation gives a semantic to primitives. It associates primitives with objects, values in the real world.
- **The valuation (meaning) function** V
  - Assigns a truth value to a given sentence under some interpretation

$$V : \text{sentence} \times \text{interpretation} \rightarrow \{True, False\}$$

First-order logic. Syntax.

Term - syntactic entity for representing objects

Terms in FOL:

- **Constant symbols:** represent specific objects
  - E.g. *John, France, car89*
- **Variables:** represent objects of a certain type (type = domain of discourse)
  - E.g. *x,y,z*
- **Functions** applied to one or more terms
  - E.g. *father-of (John)*
    father-of(father-of(John))

First order logic. Syntax.

Sentences in FOL:

- **Atomic sentences:**
  - **A predicate symbol** applied to 0 or more terms
    Examples:
    Red(car12),
    Sister(Amy, Jane);
    Manager(father-of(John));
  - t1 = t2 **equivalence** of terms
    Example:
    John = father-of(Peter)

First order logic. Syntax.

Sentences in FOL:

- **Complex sentences:**
- Assume ϕ, ψ are sentences in FOL. Then:
  - $(\phi \wedge \psi)$ $(\phi \vee \psi)$ $(\phi \Rightarrow \psi)$ $(\phi \Leftrightarrow \psi)$ $\neg \psi$

    and
  - $\forall x\ \phi$ $\exists y\ \phi$

  are sentences

Symbols $\exists, \forall$

- stand for the **existential** and the **universal** quantifier

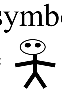
Semantics. Interpretation.

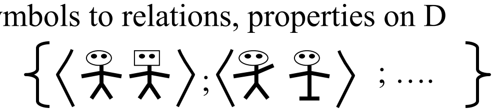
An interpretation ***I*** is defined by a **mapping** to the **domain of discourse D** **or relations on D**

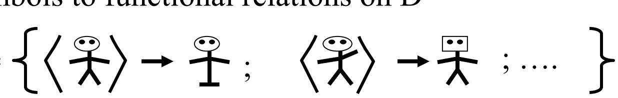
- **domain of discourse:** a set of objects in the world we represent and refer to;

An interpretation *I* maps:

- Constant symbols to objects in D

  I(*John*) = 
- Predicate symbols to relations, properties on D

  I(*brother*) = 
- Function symbols to functional relations on D

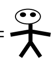
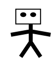
  I(*father-of*) = 

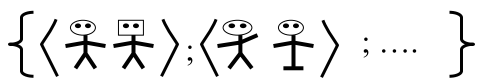
Semantics of sentences.

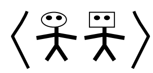
Meaning (evaluation) function:

V : sentence $\times$ interpretation $\rightarrow \{True, False\}$

A **predicate** *predicate(term-1, term-2, term-3, term-n)* is true for the interpretation I, iff the objects referred to by *term-1, term-2, term-3, term-n* are in the relation referred to by *predicate*

I(*John*) =  I(*Paul*) = 

I(*brother*) = 

brother(John, Paul) =  in I(*brother*)

V(*brother(John, Paul), I*) = ***True***

Semantics of sentences.

- **Equality** V(*term-1= term-2, I*) = ***True***
  Iff I(*term-1*) =I(*term-2*)
- **Boolean expressions: standard**

  E.g. $V(sentence\text{-}1 \vee sentence\text{-}2, I)$ = ***True***
  Iff V(*sentence-1,I*)= ***True*** or V(*sentence-2,I*)= ***True***

- **Quantifications**

  $V(\forall x\ \phi, I)$ = ***True***
  Iff for all $d \in D$ $V(\phi, \overbrace{I[x/d]}^{\text{substitution of } x \text{ with } d})$= ***True***

  $V(\exists x\ \phi, I)$ = ***True***
  Iff there is a $d \in D$, s.t. $V(\phi, I[x/d])$= ***True***

Representing knowledge in FOL

Example:

Kinship domain

- **Objects:** people

  $John$, $Mary$, $Jane$, …
- **Properties:** gender

  $Male\ (x), Female\ (x)$
- **Relations:** parenthood, brotherhood, marriage

  $Parent\ (x, y), Brother\ (x, y), Spouse\ (x, y)$
- **Functions:** mother-of (one for each person x)

  $MotherOf\ (x)$

Kinship domain in FOL

Relations between predicates and functions: write down what we know about them; how relate to each other.

- Male and female are disjoint categories

  $\forall x\ Male\ (x) \Leftrightarrow \neg Female\ (x)$
- Parent and child relations are inverse

  $\forall x, y\ Parent\ (x, y) \Leftrightarrow Child\ (y, x)$
- A grandparent is a parent of parent

  $\forall g, c\ Grandparent(g, c) \Leftrightarrow \exists p\ Parent(g, p) \wedge Parent(p, c)$
- A sibling is another child of one's parents

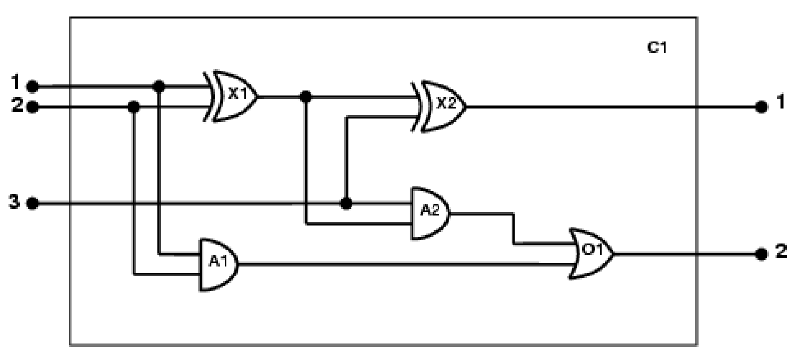
  $\forall x, y\ Sibling\ (x, y) \Leftrightarrow (x \neq y) \wedge \exists p\ Parent\ (p, x) \wedge Parent\ (p, y)$
- And so on ….

Knowledge engineering in FOL

1. Identify the problem/task you want to solve
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

The electronic circuits domain

One-bit full adder



The electronic circuits domain

1. **Identify the task**
   - Does the circuit actually add properly? (circuit verification)
2. **Assemble the relevant knowledge**
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   - Irrelevant attributes: size, shape, color, cost of gates
3. **Decide on a vocabulary**
   - **Alternatives:**

     $Type(X_1) = XOR$

     $Type(X_1, XOR)$

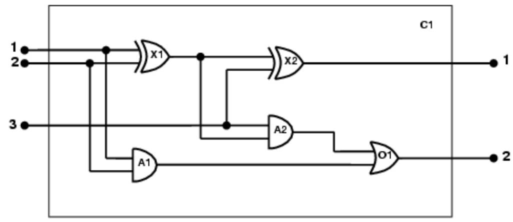
     $XOR(X_1)$

The electronic circuits domain

4. **Encode general knowledge of the domain**
   - $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$
   - $\forall t\ Signal(t) = 1 \vee Signal(t) = 0$
   - $1 \neq 0$
   - $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Connected(t_2, t_1)$
   - $\forall g\ Type(g) = OR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow \exists n\ Signal(In(n,g)) = 1$
   - $\forall g\ Type(g) = AND \Rightarrow Signal(Out(1,g)) = 0 \Leftrightarrow \exists n\ Signal(In(n,g)) = 0$
   - $\forall g\ Type(g) = XOR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow Signal(In(1,g)) \neq Signal(In(2,g))$
   - $\forall g\ Type(g) = NOT \Rightarrow Signal(Out(1,g)) \neq Signal(In(1,g))$

The electronic circuits domain

5. Encode the specific problem instance

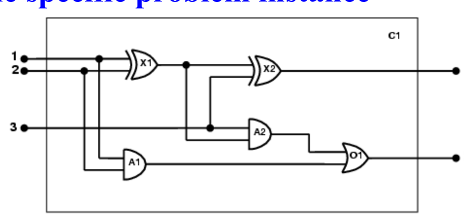


Type(X_1) = XOR
Type(A_1) = AND
Type(O_1) = OR

Type(X_2) = XOR
Type(A_2) = AND

The electronic circuits domain

5. Encode the specific problem instance



Connected(Out(1,X_1),In(1,X_2))
Connected(In(1,C_1),In(1,X_1))
Connected(Out(1,X_1),In(2,A_2))
Connected(In(1,C_1),In(1,A_1))
Connected(Out(1,A_2),In(1,O_1))
Connected(In(2,C_1),In(2,X_1))

• • •

The electronic circuits domain

6. Pose queries to the inference procedure

What are the possible sets of values of all the terminals for the adder circuit?

$\exists i_1,i_2,i_3,o_1,o_2$ Signal(In(1, C_1)) = i_1 $\wedge$ Signal(In(2,C_1)) = i_2 $\wedge$ Signal(In(3,C_1)) = i_3 $\wedge$ Signal(Out(1,C_1)) = o_1 $\wedge$ Signal(Out(2,C_1)) = o_2

7. Debug the knowledge base

May have omitted assertions like $1 \neq 0$

Inference in First order logic

Logical inference in FOL

Logical inference problem:

- Given a knowledge base KB (a set of sentences) and a sentence α , does the KB semantically entail α?

$$KB \models \alpha \quad ?$$

In other words: In all interpretations in which sentences in the KB are true, is also α true?

Logical inference problem in the first-order logic is undecidable !!!. No procedure that can decide the entailment for all possible input sentences in a finite number of steps.

Logical inference problem in the Propositional logic

Computational procedures that answer:

$$KB \models \alpha \ ?$$

Three approaches:

- **Truth-table approach**
- **Inference rules**
- **Conversion to the inverse SAT problem**
  - **Resolution-refutation**

Inference in FOL: Truth table

- Is the Truth-table approach a viable approach for the FOL?

  ?

Inference in FOL: Truth table approach

- Is the Truth-table approach a viable approach for the FOL?

  ?
- **NO!**
- Why?
- It would require us to enumerate and list all possible interpretations I
- I = (assignments of symbols to objects, predicates to relations and functions to relational mappings)
- Simply there are too many interpretations

Inference in FOL: Inference rules

- Is the Inference rule approach a viable approach for the FOL?
  ?

Inference in FOL: Inference rules

- Is the Inference rule approach a viable approach for the FOL?
  ?
- **Yes.**
- The inference rules represent sound inference patterns one can apply to sentences in the KB
- What is derived follows from the KB
- **Caveat:**
  – we need to add rules for handling quantifiers

Inference rules

- **Inference rules from the propositional logic:**
  - Modus ponens
    $$\frac{A \Rightarrow B, \quad A}{B}$$
  - Resolution
    $$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$
  - and others: And-introduction, And-elimination, Or-introduction, Negation elimination
- **Additional inference rules** are needed for sentences with quantifiers and variables
  - Must involve variable substitutions

Variable substitutions

- Variables in the sentences can be substituted with terms.
  (terms = constants, variables, functions)
- **Substitution:**
  - Is a mapping from **variables to terms**
    $$\{x_1 / t_1, x_2 / t_2, \ldots\}$$
  - Application of the substitution to sentences

$$SUBST(\{x / Sam, y / Pam\}, Likes(x, y)) = Likes(Sam, Pam)$$

$$SUBST(\{x / z, y / fatherof(John)\}, Likes(x, y)) =$$
$$Likes(z, fatherof(John))$$

Inference rules for quantifiers

- **Universal elimination**

$$\frac{\forall x\ \phi(x)}{\phi(a)} \qquad a \text{ - is a constant symbol}$$

  – substitutes a variable with **a constant symbol**
- **Example:**

$$\forall x\ Likes(x, IceCream)$$

$$\downarrow$$

$$Likes(Ben, IceCream)$$

Inference rules for quantifiers

- **Existential elimination**

$$\frac{\exists x\ \phi(x)}{\phi(a)}$$

  – Substitutes a variable with **a constant symbol** that does not appear elsewhere in the KB
- **Examples:**
- $\exists x\ Kill(x, Victim) \longrightarrow Kill(Murderer, Victim)$

  Special constant called **Skolem** constant
- $\exists x\ Crown(x) \wedge OnHead(x, John)$

  $Crown(C_1) \wedge OnHead(C_1, John)$

Reduction to propositional inference

Suppose the KB contains just the following:

$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
Greedy(John)
Brother(Richard,John)

- Instantiating the universal sentence in **all possible** ways, we have:
  King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
  King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ 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.

Reduction contd.

- Every FOL KB can be **propositionalized** so as to preserve entailment
- (A ground sentence is entailed by new KB iff entailed by original KB)
- **Idea of the inference:**
  - propositionalize KB and query, apply resolution, return result
- **Problem:** with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*)))

Reduction contd.

Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

Idea: For $n = 0$ to ∞ do

create a propositional KB by instantiating with depth-\$n\$ terms

see if α is entailed by this KB

Problem: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is **semidecidable** (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)