CSE 579: Knowledge Representation & Reasoning

Module 2: First Order Logic

Ali Altunkaya

Spring 2023

Module 2 Highlights

We will study 3 different formal languages during the course:

1-) Propositional Logic (PL)

<u>***2-) First Order Logic (FOL)***</u>

3-) Answer Set Theory (AST)

Ultimate goal: Encode <u>natural language</u> in a <u>formal language</u> so we can do … (reasoning)

FOL is more expressive than PL: we can encode more complex sentences from natural language into the formal language.

Outline

1. First Order Logic (FOL)
    1. Limitations of PL (Propositional Logic)
    2. Introduction to FOL
    3. Syntax (alphabet)
    4. Semantics (meaning)
    5. Representing knowledge in FOL
    6. Herbrand Models

Module 2 Highlights

Propositional Logic

Grammar
- ¬, ∧, ∨, →, ↔, ⊥, T
- ( ) ,

Signature
- atoms (p, q, r, ….)

Syntax

- TRUE
- FALSE

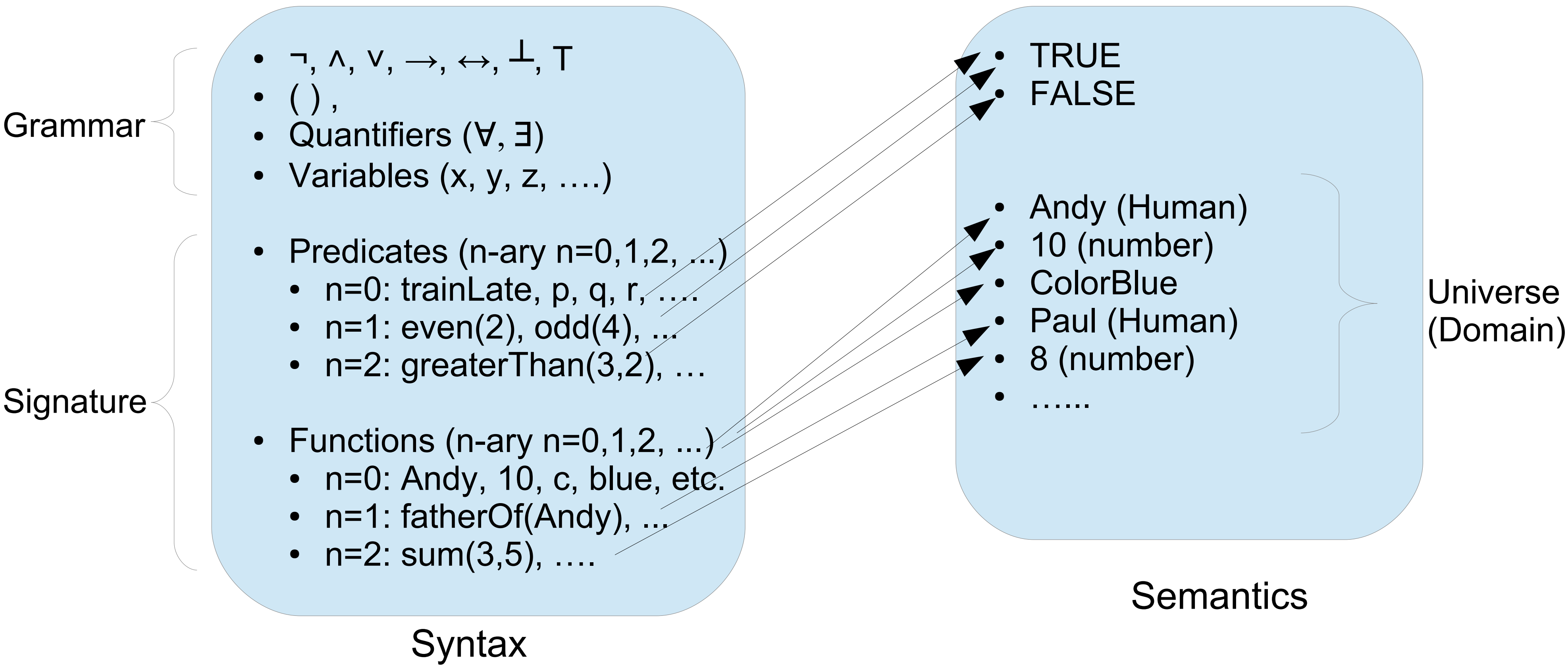
Semantics

Limitations of Propositional Logic:
- No objects (numbers, people, colors etc.)
- No functions(fatherOf(Andy), sum(2,3), etc)
- No relations (lessThan, greaterThan)
- No Quantifiers

- Atoms are only facts which are True or False.
- Atoms are also very coarse-grained.

Module 2 Highlights

First Order Logic



Predicates are atomic formulas, which means we cannot divide it into sub-formulas.
They can only be mapped to True or False in Semantics.

Functions of arity>0 are also called Terms. Functions of arity=0 are also called Object Constants.
We can recursively make complex terms, such as fatherOf(fatherOf(Paul).
They can mapped to any object in Semantics Universe or Domain. They cannot be True/False.

Variables can be either bound to a Quantifier(∀, ∃), or free.

Module 2 Highlights

Propositional Logic (PL):

– Study of propositions (declarative sentences or statements) about the world which can be given a truth value (True/False).

– PL uses components : not, and, or, if then.

– PL is compositional. You can combine propositions (F and G).

Limitations of Propositional Logic (PL):

– Cannot express individuals and relations between them.

– Cannot deal with modifiers like “there exist”, “all”, “among”, “only”

Need for a Richer Language:

– Represent set of objects (all objects or some objects)

– Represent relationships between objects (affects, younger, etc)

Module 2 Highlights

Introduction to First Order Logic (FOL):

– FOL has a finer logical structure than PL.

PL : every student → s

FOL : every student → ∀s   some student → ∃s

– Propositional Logic assumes world contains only facts(true/false) but First-Order Logic assumes the world contains:

- Objects: people, numbers, colors, cities etc.
- Functions: father of, successor, predecessor, etc.
- Relations/Properties: greater than, less than, part of, owns, etc.

Module 2 Highlights

Introduction to First Order Logic (FOL):

Object/function constants help us to represent individuals

- andy, paul : object constants
- father(andy) : function constant

Predicates help us to represent properties/features:

- S(andy): Andy is a student.
- Y(andy, paul): Andy is younger than Paul.

Variables are placeholders for concrete values:

- S(x): x
- Y(x, y) : x is younger than y.

Quantifiers help us to represent sets of objects:

- every student: ∀s some student: ∃s

Module2 Highlights: Syntax of First-Order Logic

What is the difference between function and predicate?

- A **function** takes one or more arguments and returns a value.
- A **predicate** takes one or more arguments and is either true or false.

Signature contains symbols that we can use to build **formulas**. In FOL, signature consists of two kinds of symbols:

1-) function constants (with arity n, where n= 0,1,2,3,4...)

- arity 0 are called object constants: andy, paul, ….
- arity 1: father(john)
- arity 2: add(3, 5)

2-) predicate constants (with arity n):

- arity 0 are called propositonal constants: TrainLate, Taxi, Stage4, p, q, r, ...
- arity 1: even(2), prime(3),
- arity 2: greaterThan(3,2)

Note: In PL, signature was just a set of atoms such as {p, q, r}.

Module2 Highlights: Syntax of First-Order Logic

Signature contains symbols that we can use to build **formulas**. In FOL, signature consists of two kinds of symbols:

1-) function constants

2-) predicate constants

These symbols are not in the Signature but can be used to build formulas, in addition to the symbols in Signature:

- (object) variables: x, y, z, …
- The propositional connectives: $\bot$ $\top$ $\neg$ $\wedge$ $\vee$ $\rightarrow$ $\leftrightarrow$
- The universal quantifier $\forall$ and the existential quantifier $\exists$
- The parentheses and the comma

Module2 Highlights: Syntax of First-Order Logic

A **term** denotes an individual. It can be defined recursively:

– An object constant (function constant arity=0) is a term

- john, andy, mary, ...

– An object variable is a term

- x, y, z, …

– For every function constant f or arity n>0, if $t_1, \ldots, t_n$ are terms then so is $f(t_1, \ldots, t_n)$

- father(john), +(3, 5)
- father(father(father(john))), +(3, +(5, +(1,1)))

A **term** is not TRUE/FALSE, it just represent individuals or values.

Module2 Highlights: Syntax of First-Order Logic

An **atomic formula** denotes a base fact that is either true or false.

- It is similar to atoms in PL(smallest unit that can be assigned true or false.), but has more complicated internal structure.

An **atomic formula** is a predicate constant or equality of two terms (or function constants). Atomic formulas are either:

- Propositional constants R (Predicate constant of arity n=0) or
  - -arity 0: TrainLate
- $R(t_1, \ldots, t_n)$ where R is a predicate constant t_i are terms
  - arity 1: even(2)
  - arity 2: greaterThan(3,2)
  - arity 3: ...
- $t_1 = t_2$ where t_i are terms
  - father(john) = james,   1+2 = 3

Module2 Highlights: Syntax of First-Order Logic

A (first-order) **formula** of signature σ is defined recursively.

- Every atom is a formula.
- Both 0-place connectives ($\bot$ T) are formulas.
- If F is a formula then is $\neg F$ a formula too.
- For any binary connective ◌ ($\wedge$, $\vee$, $\rightarrow$), if F and G are formulas then (F◌G) is a formula too.
- If F is a formula then $\forall x$ F and the existential quantifier $\exists x$ F are formulas too.

Note: The rules are the same with PL. FOL has an additional rule of the last one.

See examples...

Module2 Highlights: Syntax of First-Order Logic

– **Bound and Free Variables**:

- an **occurrence of a variable v** in a formula F is **bound** if it belongs to a subformula of F that has the form ∀v G or ∃v G, (in other words, in the parse tree, one of its ancestors is ∀v or ∃v)
- otherwise it is **free**.

– v is a **free variable** of F if v has at least one free occurrence in F.

– Bound variables can be renamed without changing meaning:

- ∀x P(x) means the same as ∀y P(y)

– A **sentence** is a formula without free variables:

- Also known as “closed formula”
- In other words, all variables are bounded with either ∀ or ∃
- We will define meaning for formulas only without free variables (sentences).

– The **universal closure** of F means, we will assume ∀ to any formula to make it a sentence. Otherwise cannot do semantics.

Module2 Highlights: Syntax of First-Order Logic

– **Examples**

Let $\sigma = \{a, P, Q\}$, where a is an object constant, P is a unary and Q is a binary predicate constant

Q: Are these formulas?

1. a No
2. P(a) yes
3. Q(a) No
4. $\forall x\, P(a)$ yes
5. $\neg P(a) \vee \exists x(P(x) \wedge Q(x, y))$ yes

Module2 Highlights: Syntax of First-Order Logic

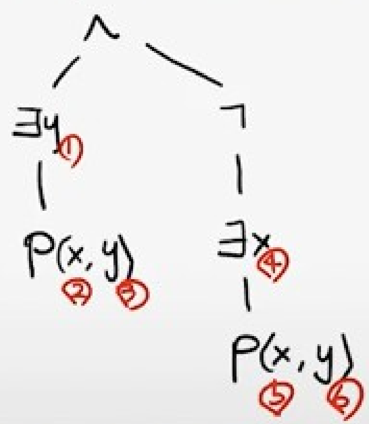
– **Examples**

An occurrence of a variable v in a formula F is **bound** if it belongs to a subformula of F that has the form Qv G; otherwise it is **free**.

- Informally speaking, the occurrence is bound if, in the parse tree, one of its ancestors is Qv.

$$\exists y \, P(x, y) \land \neg \exists x \, P(x, y)$$

1 2 3 4 5 6



Q: Which occurrences are free? 2, 6

v is a **free variable** of F if v has a free occurrence in F.

x, y.

Module2 Highlights: Syntax of First-Order Logic

– **Examples**

Formula: $\left(\forall x \left(P(x) \land Q(x)\right)\right) \to \left(\neg P(x) \lor Q(y)\right)$

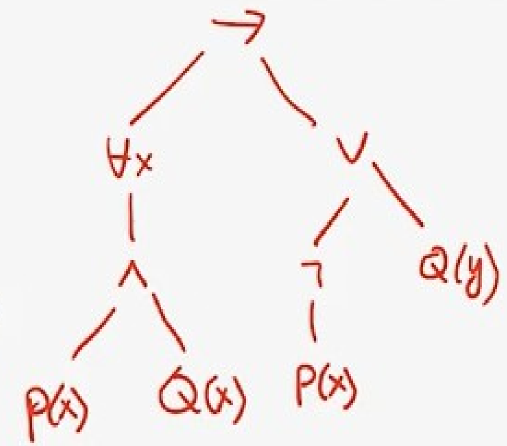
1 2 3 4 5

Q: What are the free occurrences of a variable?

4, 5

Q: What are the free variables of the formula?

x, y.



Module2 Highlights: Syntax of First-Order Logic

– **Examples**

Assume that the signature consists of the object constant *Me*, the unary predicate constant *Male*, and the binary predicate constant *Parent*, and nothing else. Express each of the given English sentences in first-order logic.

1. I have no daughters

$$\neg \exists x \, (Parent(Me, x) \wedge \neg Male(x))$$

$$\forall x \, (Parent(Me, x) \rightarrow Male(x))$$

2. I have a granddaughter

$$\exists y \exists z \, (Parent(Me, y) \wedge Parent(y, z) \wedge \neg Male(z))$$

3. I have a brother.

$$\exists y \exists z \, (Parent(y, Me) \wedge Parent(y, z) \wedge Male(z) \wedge \neg (Me = z))$$

Module2 Highlights: Syntax of First-Order Logic

– **Examples**

Let the underlying signature be

{a, P, Q} **(1)**

where *a* is an object constant, *P* is a unary predicate constant, and *Q* is a binary predicate constant.

We will think of object variables as ranging over the set N of nonnegative integers, and interpret the signature as follows:

- a represents the number 10,
- P(x) represents the condition "x is a prime number,"
- Q(x, y) represents the condition "x is less than y."

As an example, the sentence All prime numbers are greater than x can be represented by the formula

$$\forall y(P(y) \rightarrow Q(x, y)).$$ **(2)**

In the following two problems, represent the given English sentences by predicate formulas.

Problem 1

a) There is a prime number that is less than 10. $\exists x(P(x) \land Q(x, a))$

b) x equals 0. $\neg \exists y\, Q(y, x)$

c) x equals 9. $Q(x, a) \land \neg \exists y(Q(x, y) \land Q(y, a))$ $\boxed{x < y < 10}$

11 11 11

Problem 2

There are infinitely many prime numbers

$\forall x\, \exists y\, (Q(x, y) \land P(y))$

Module 2 Highlights: Semantics of FOL

Semantics: Check whether the Natural Language (e.g. English) meaning also matches with Formal Language meaning.

"Colorless green ideas sleep furiously."   syntax: ok   semantics: x

Proposition Logic:

- It was easy to do interpretations using the truth table (T/F),
- because the signature only contains atoms,
- There are finite (2^n) possible interpretations for a signature of n elements.

First Order Logic:

- We have more fine-level detail in the signature (functions and predicates)
- Give different meaning to different categories of symbols
- Signature contains a non-empty set callled the universe (or domain). Universe contains elements/objects we refer to.
- Quantifiers forAll ∀ thereExist ∃
- There are infinite number of interpretations for a given signature.

Module 2 Highlights: Semantics of FOL

Semantics – Terms:

- Term denotes an individual in the universe |I|.
- We can interprete function constants: map a value of a function constant to a value in the universe

Semantics – Formulas:

- We can use terms in the formulas.
- We can interprete predicate constants: map a value of a predicate constant to TRUE or FALSE

Note: There is a mapping function for each symbol in the signature.

Interpretation is to give meaning to the symbols in the signature.

Module 2 Highlights: Semantics of FOL

Why we need Extended Signature: There is a technical problem in the interpretation:

- We only have limited function constants in the signature
- We don't have name for all individuals in the universe.
- Because of that, quantifications (∀, ∃), it does not work properly. When we use quantifications, we shouldn't be limited to named individuals.

Extended Signature: For any element ξ of universe $|I|$, select a new symbol ξ^*, called the "name" of ξ.

- Named individuals, ξ^*, are not available to knowledge engineer: which means you can not use them in formulas or terms.
- You can use it just for semantic interpretation of Quantifiers.

ξ individuals are part of the semantics.

ξ^*, named individuals are part of the syntax.

Module 2 Highlights: Semantics of FOL

The notions are carried over from Propositional Logic:

1) Satisfaction
2) Tautology (Logical Validity)
3) Equivalence
4) Entailment

See slides …

Module 2 Highlights: Semantics of FOL

Undecidability of FOL: FOL is more expressive, but there is no general algorithm to check the basic properties such as satisfiability.

- Interpretation is more complex in FOL.
- We cannot even enumerate all interpretations,
- because there are infinite elements in the universe.
- Therefore, for a given sentence, tautology(logical validity) is undecidable.
-
- Since all notions are related and can be reduced to each other:
- Satisfiability is undecidable too.
-
- We need to restrict FOL in a meaningful way, see Herbrand models.

Undecidable: There is no algorithm, which terminates and produces a result (T/F).

Module 2: Representing Knowledge in FOL

Starting from a textual description, we can identify objects, relationships and functions. Benefits example:

i. We couldn't encode the example disease knowledge in PL (Propositional Logic), as we have seen in the limitations of PL.

ii. But we will be able to encode the same knowledge in FOL.

In FOL,

- **Data**: we also need data to encode more complex information. (in PL, there was no data)
- **Terminological axioms**: independent of any concrete (individual) data, explains general properties of data: such as “all childs are not adults”, not “Mary is not an adult”.
    - Kinds of axioms: sub-type statements, full definitions, disjointment statements, covering statements, type restrictions, and others.

See slides …

Module 2: Representing Knowledge in FOL

Data vs Terminological Knowledge: What is the difference between them? What is ontology? What is knowledge base?

The Role of Reasoning: Why are reasoning problems (satisfiability, entailment) useful and important?

Expressivity vs Complexity tradeoff: FOL is more expressive but also its reasoning is undecidable.

Limitations of FOL: Transitive closure, defaults, exceptions, probabilities, vague knowledge, etc.

see slides

Module 2: Herbrand Models

FOL is undecidable. We will restrict FOL so it will become decidable. Herbrand interpretations are special case of FOL.

A Herbrand interpretation of signature is such that:

- its universe (Herbrand universe) is the set of all ground terms.
- every ground term is interpreted as itself.

Actually it is very similar to PL in terms of interpretations. But:

- Without functions, Herbrand models are finitely enumerable (decidable).
- With functions, not true, since nested functions will make the universe infinite
- Its semantics contains only True/False for the predicates.

What is the difference between “interpretation” and “model”?

- Models are the interpretations that satisfies a given formula.

Thanks

&

Questions