CMSC 471: Intro to AI

First-Order Logic

Many slides courtesy Tim Finin

FOL Overview

- First Order logic (FOL) is a powerful knowledge representation (KR) system
- Used in AI systems in various ways, e.g., to
  - Directly represent & reason about concepts & objects
  - Formally specify meaning of KR systems (e.g., OWL)
  - Form programming languages (e.g., Prolog) and rule-based systems
  - Make semantic database systems (Datalog) and Knowledge graphs (Wikidata)
  - Provide features useful in neural network deep learning systems

First-order logic

- First-order logic (FOL) models the world in terms of
  - **Objects,** which are things with individual identities
  - **Properties** of objects that distinguish them from others
  - **Relations** that hold among sets of objects
  - **Functions,** a subset of relations where there is only one "value" for any given "input"
- Examples:
  - Objects: students, lectures, companies, cars ...
  - Relations: isa, hasBrother, biggerThan, outside, hasPart, color, occursAfter, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: hasFather, hasSSN, ...

User provides

- **Constant symbols** representing individuals in world
  - BarackObama, Green, John, 3, “John Smith”
- **Predicate symbols** map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)
  - hasBrother(John, Robert)
- **Function symbols** map individuals to individuals
  - hasFather(SashaObama) = BarackObama
  - colorOf(Sky) = Blue

What do these mean?

- User should also indicate what these mean in a way that humans will understand
  - i.e., map to their own internal representations
- May be done via a combination of
  - Choosing good names for formal terms, e.g. calling a concept HumanBeing instead of Q5
  - Comments in the definition `#human being`
  - Descriptions and examples in documentation
  - Reference to other representations , e.g., sameAs /m/0dgw95 in Freebase and Person in schema.org
  - Give examples like *Donald Trump* and *Luke Skywalker* to help distinguish the concepts of a real and fictional person

FOL Provides

- **Variable symbols**
  - e.g., X, Y, ?foo, ?number
- **Connectives**
  - Same as propositional logic: not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), iff ($\leftrightarrow$), equivalence ($\equiv$), …
- **Quantifiers**
  - Universal $\forall$**x** or **(Ax)**
  - Existential $\exists$**x** or **(Ex)**

Sentences: built from terms and atoms

- **term** (denoting an individual): constant or variable symbol, or n-place function of n terms, e.g.:
  - Constants: john, umbc
  - Variables: X, Y, Z
  - Functions: mother_of(john), phone(mother(x))
- **Ground terms** have no variables in them
  - **Ground:** john, father_of(father_of(john))
  - **Not Ground:** father_of(X)
- Syntax may vary: maybe variables must start with a "?" or a capital letter

Sentences: built from terms and atoms

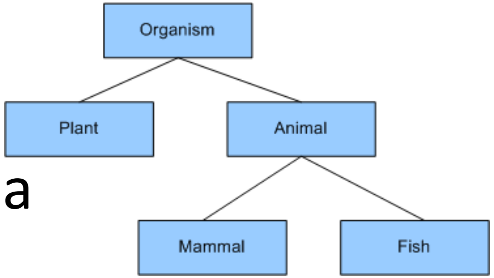
- **atomic sentences** (which are either true or false) are n-place predicates of n terms, e.g.:
  - green(kermit)
  - between(philadelphia, baltimore, dc)
  - loves(X, mother(X))
- **complex sentences** formed from atomic ones connected by the standard logical connectives with quantifiers if there are variables, e.g.:
  - loves(mary, john) $\vee$ loves(mary, bill)
  - $\forall$x loves(mary, x)

What do atomic sentences mean?

- Unary predicates typically encode a **type**
  - muppet(Kermit): kermit is a kind of muppet
  - green(kermit): kermit is a kind of green thing
  - integer(X): x is a kind of integer
- Non-unary predicates typically encode relations or properties
  - Loves(john, mary)
  - Greater_than(2, 1)
  - Between(newYork, philadelphia, baltimore)
  - hasName(john, “John Smith”)

UMBC

Ontology



- Designing a logic representation is like designing a model in an object-oriented language
- **Ontology:** a "formal naming and definition of the types, properties and relations of entities for a domain of discourse"
- E.g.: schema.org ontology used to put semantic data on Web pages to help search engines
  - Here's the semantic markup Google sees about CSEE department of UMBC.

Sentences: built from terms and atoms

- **quantified sentences** adds quantifiers ∀ and ∃

  ∀x loves(x, mother(x))

  ∃x number(x) ∧ greater(x, 100), prime(x)

- **well-formed formula (wff):** a sentence with no *free* variables or where all variables are *bound* by a universal or existential *quantifier*

  In **(∀x)P(x, y)** x is bound & y is free so it's not a wff

Quantifiers: ∀ and ∃

- **Universal quantification**
  - $(\forall x)P(X)$ means P holds for **all** values of X in the domain associated with variable[1]
  - E.g., $(\forall X)$ dolphin(X) $\rightarrow$ mammal(X)

- **Existential quantification**
  - $(\exists x)P(X)$ means P holds for **some** value of X in domain associated with variable
  - E.g., ($\exists$**X**) mammal(X) $\wedge$ lays_eggs(X)
  - This lets us make statements about an object without identifying it

[1] a variable's domain is often not explicitly stated and is assumed by the context

Universal Quantifier: $\forall$

- **Universal quantifiers typically used with *implies* to form *rules*:**

  Logic: $\forall X\ student(X) \rightarrow smart(X)$

  Means: All students are smart

- **Universal quantification *rarely* used without implies:**

  Logic: $\forall X\ student(X) \wedge smart(X)$

  Means: Everything is a student and is smart

- **What about this, though:**
  - *Logic:* $\forall X\ alive(X) \vee dead(X)$
  - Means: everything is either alive or dead

Universal Quantifier: $\forall$

- **What about this, though:**
  - *Logic:* $\forall X\ alive(X) \lor dead(X)$
  - Means: everything is either alive or dead
- **Can be rewritten using a standard tautology**
  - $A \lor B \equiv \sim A \rightarrow B$
- **Giving both of these** (since $A \lor B \equiv B \lor A$)
  - $\forall X \sim alive(X) \rightarrow dead(X)$
  - $\forall X \sim dead(X) \rightarrow alive(X)$

Existential Quantifier: ∃

- Existential quantifiers usually used with **and** to specify a list of properties about an individual

  Logic: $(\exists X)\ student(X) \wedge smart(X)$

  Meaning: There is a student who is smart

- Common mistake: represent this in FOL as:

  Logic: $(\exists X)\ student(X) \rightarrow smart(X)$

  Meaning: ?

Existential Quantifier: ∃

- Existential quantifiers usually used with **and** to specify a list of properties about an individual

  Logic: $(\exists X)\ student(X) \wedge smart(X)$

  Meaning: There is a student who is smart

- Common mistake: represent this in FOL as:

  Logic: $(\exists X)\ student(X) \rightarrow smart(X)$

  $P \rightarrow Q = \sim P \vee Q$

  $\exists X\ student(X) \rightarrow smart(X) = \exists X \sim student(X) \vee smart(X)$

  Meaning: There's something that is either not a student or is smart

Quantifier Scope

- FOL sentences have structure, like programs
- In particular, variables in a sentence have a **scope**
- Suppose we want to say “everyone who is alive loves someone”

$(\forall X)\ alive(X) \rightarrow (\exists\ Y)\ loves(X, Y)$

- Here’s how we scope the variables

$$(\forall X)\ alive(X) \rightarrow (\exists Y)\ loves(X, Y)$$

Scope of x

Scope of y

Quantifier Scope

- **Switching order of two universal quantifiers *does not* change the meaning**
  - $(\forall X)(\forall Y)P(X,Y) \leftrightarrow (\forall Y)(\forall X)\ P(X,Y)$
  - Dogs hate cats (i.e., all dogs hate all cats)
- **You can switch order of existential quantifiers**
  - $(\exists X)(\exists Y)P(X,Y) \leftrightarrow (\exists Y)(\exists X)\ P(X,Y)$
  - A cat killed a dog
- **Switching order of universal and existential quantifiers *does* change meaning:**
  - Everyone likes someone: $(\forall X)(\exists Y)$ likes(X,Y)
  - Someone is liked by everyone: $(\exists Y)(\forall X)$ likes(X,Y)

```
def verify1():
    # Everyone likes someone: (∀x)(∃y) likes(x,y)
    for p1 in people():
        foundLike = False
        for p2 in people():
            if likes(p1, p2):
                foundLike = True
                break
        if not foundLike:
            print(p1, 'does not like anyone ☹')
            return False
    return True
```

Every person has at least one individual that they like.

Procedural example 1

```
def verify2():
    # Someone is liked by everyone: (∃y)(∀x) likes(x,y)
    for p2 in people():
        foundHater = False
        for p1 in people():
            if not likes(p1, p2):
                foundHater = True
                break
        if not foundHater
            print(p2, 'is liked by everyone ☺')
            return True
    return False
```

There is a person who is liked by every person in the universe.

Procedural example 2

Connections between ∀ and ∃

- We can relate sentences involving ∀ and ∃ using extensions to **De Morgan's laws**:
    1. $(\forall x)\ P(x) \leftrightarrow \neg(\exists x)\ \neg\ P(x)$
    2. $\neg(\forall x)\ P(x) \leftrightarrow (\exists x)\ \neg P(x)$
    3. $(\exists\ x)\ P(x) \leftrightarrow \neg\ (\forall\ x)\ \neg P(x)$
    4. $\neg(\exists x)\ P(x) \leftrightarrow (\forall x)\ \neg P(x)$
- Examples
    1. All dogs sleep ↔ There is no dog that doesn't sleep
    2. Not all dogs bark ↔ There is a dog that doesn't bark
    3. There is a dog that talks ↔ Not all dogs can't talk
    4. No dog likes cats ↔ All dogs don't like cats

Translating English to FOL

Every gardener likes the sun

All purple mushrooms are poisonous

Translating English to FOL

Every gardener likes the sun

$\forall x \text{ gardener}(x) \rightarrow \text{likes}(x,\text{Sun})$

All purple mushrooms are poisonous

Translating English to FOL

Every gardener likes the sun

$\forall x\ gardener(x) \rightarrow likes(x, Sun)$

All purple mushrooms are poisonous

$\forall x\ (mushroom(x) \land purple(x)) \rightarrow poisonous(x)$

Translating English to FOL

Every gardener likes the sun

$\forall x\ gardener(x) \rightarrow likes(x,Sun)$

All purple mushrooms are poisonous

$\forall x\ (mushroom(x) \wedge purple(x)) \rightarrow poisonous(x)$

No purple mushroom is poisonous (two ways)

Translating English to FOL

Every gardener likes the sun

$\forall x \text{ gardener}(x) \rightarrow \text{likes}(x,\text{Sun})$

All purple mushrooms are poisonous

$\forall x \ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$

No purple mushroom is poisonous (two ways)

$\neg\exists x \text{ purple}(x) \wedge \text{mushroom}(x) \wedge \text{poisonous}(x)$

Translating English to FOL

Every gardener likes the sun

$\forall x \; gardener(x) \rightarrow likes(x, Sun)$

All purple mushrooms are poisonous

$\forall x \; (mushroom(x) \land purple(x)) \rightarrow poisonous(x)$

No purple mushroom is poisonous (two ways)

$\neg \exists x \; purple(x) \land mushroom(x) \land poisonous(x)$

$\forall x \; (mushroom(x) \land purple(x)) \rightarrow \neg poisonous(x)$

English to FOL: Counting

Use = predicate to identify different individuals

- There are <u>at least</u> two purple mushrooms

  $\exists x\ \exists y\ mushroom(x) \wedge purple(x) \wedge mushroom(y) \wedge purple(y) \wedge$
  $\boldsymbol{\neg(x=y)}$

- **There are <u>exactly</u> two purple mushrooms**

  $\exists x\ \exists y\ mushroom(x) \wedge purple(x) \wedge mushroom(y) \wedge purple(y) \wedge$
  $\boldsymbol{\neg(x=y)} \wedge$
  $\forall z\ (mushroom(z) \wedge purple(z)) \rightarrow \boldsymbol{((x=z) \vee (y=z))}$

Saying there are 802 different Pokemon is hard!

Direct use of FOL is not for everything!

Translating English to FOL

What do these mean?

- **You can fool *some of* the people *all of* the time**
- **You can fool *all of* the people *some of* the time**

Translating English to FOL

What do these mean?

Both English statements are ambiguous

- **You can fool *some of* the people *all of* the time**

  There is a nonempty subset of people so easily fooled that you can fool that subset every time.

  For any given time, there is a non-empty subset at that time that you can fool

- **You can fool *all of* the people *some of* the time**

  There are one or more times when it's possible to fool everyone.

  Each individual can be fooled at some point in time

Translating English to FOL

You can fool *some of* the people *all of* the time

There is a nonempty group of people so easily fooled that you can fool that group every time

≡ There's (at least) one person you can fool every time

$\exists x\ \forall t\ \ person(x) \wedge time(t) \rightarrow canFool(x, t)$

For any given time, there is a non-empty group at that time that you can fool

≡ For every time, there's a person at that time that you can fool

$\forall t\ \exists x\ \ person(x) \wedge time(t) \rightarrow canFool(x, t)$

Translating English to FOL

You can fool *all of* the people *some of* the time

There's at least one time when you can fool everyone

$\exists t\ \forall x\ \text{time}(t) \land \text{person}(x) \rightarrow \text{canFool}(x, t)$

Everybody can be fooled at some point in time

$\forall x\ \exists t\ \text{person}(x) \land \text{time}(t) \rightarrow \text{canFool}(x, t)$

Representation Design

- Many options for representing even a simple fact, e.g., something’s color as red, green or blue, e.g.:
  - green(kermit)
  - color(kermit, green)
  - hasProperty(kermit, color, green)
- Choice can influence how easy it is to use
- Last option of representing properties & relations as triples used by modern knowledge graphs
  - Easy to ask: What color is Kermit? What are Kermit’s properties?, What green things are there? Tell me everything you know, …

Simple genealogy KB in FOL

Design a knowledge base using FOL that

- Has facts of immediate family relations, e.g., spouses, parents, etc.
- Defines more complex relations (ancestors, relatives)
- Detect conflicts, e.g., you are your own parent
- Infers relations, e.g., grandparent from parent
- Answers queries about relationships between people

Extend with relations and constraints

- Simple two argument relations, e.g.
  - spouse, has_child, has_parent
- Add general constraints to relations, e.g.
  - spouse(X,Y) => ~ (X = Y)
  - spouse(X,Y) => person(X) ∧ person(Y)
- Add FOL sentences for inference, e.g.
  - spouse(X,Y) ⇔ spouse(Y,X)
- Add instance data
  - e.g., spouse(Djt, Mt)

Example: A simple genealogy KB in FOL

Predicates:

- parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
- spouse(x, y), husband(x, y), wife(x,y)
- ancestor(x, y), descendant(x, y)
- male(x), female(y)
- relative(x, y)

Facts (ground terms):

- husband(Joe, Mary), son(Fred, Joe)
- spouse(John, Nancy), male(John), son(Mark, Nancy)
- father(Jack, Nancy), daughter(Linda, Jack)
- daughter(Liz, Linda)
- etc.

Example Axioms

$(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)

$(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x)

$(\forall x,y)$ mother(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ female(x)

$(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x)

$(\forall x,y)$ son(x, y) $\leftrightarrow$ child(x, y) $\wedge$ male(x)

$(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x)

$(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x)

…

Axioms, definitions and theorems

- **Axioms**: facts and rules that capture (important) facts & concepts in a domain; used to prove **theorems**
  - Dependent axioms can make reasoning faster, however
  - Choosing a good set of axioms is a design problem

- A **definition** of a predicate is of the form "$p(X) \leftrightarrow \ldots$" and can be decomposed into two parts
  - **Necessary** description: "$p(x) \rightarrow \ldots$"
  - **Sufficient** description "$p(x) \leftarrow \ldots$"
  - Some concepts have definitions (e.g., triangle) and some don't (e.g., person)

More on definitions

Example: define father(x, y) by parent(x, y) & male(x)

- **parent(x, y)** is a necessary (but not sufficient) description of father(x, y)

  father(x, y) $\rightarrow$ parent(x, y)

- **parent(x, y) ^ male(x) ^ age(x, 35)** is a sufficient (but not necessary) description of father(x, y):

  father(x, y) $\leftarrow$ parent(x, y) ^ male(x) ^ age(x, 35)

- **parent(x, y) ^ male(x)** is a necessary and sufficient description of father(x, y)

  parent(x, y) ^ male(x) $\leftrightarrow$ father(x, y)

Higher-order logic

- FOL only lets us quantify over variables, and **variables can only range over objects**
- HOL allows us to quantify over relations, e.g.

  "two functions are equal iff they produce the same value for all arguments"

  $\forall f\ \forall g\ (f = g) \leftrightarrow (\forall x\ f(x) = g(x))$
- E.g.: (quantify over predicates)

  $\forall r\ \text{transitive}(\ r\ ) \rightarrow (\forall xyz)\ r(x,y) \wedge r(y,z) \rightarrow r(x,z))$
- More expressive, but reasoning is undecide-able, in general

Examples of FOL in use

- Semantics of W3C's Semantic Web stack (RDF, RDFS, OWL) is defined in FOL
- OWL Full is equivalent to FOL
- Other OWL profiles support a subset of FOL and are more efficient
- FOL oriented knowledge representation systems have many user friendly tools
- E.g.: Protégé for creating, editing and exploring OWL ontologies

Examples of FOL in use

Many practical approaches embrace the approach that "some data is better than none"

- The semantics of schema.org is only defined in natural language text
- Wikidata's knowledge graph has a rich schema
  - Many constraint/logical violations are flagged with warnings
  - However, not all, see this Wikidata query that finds people who are their own mother or father

Automated inference for FOL

- Automated inference for FOL is harder than PL
  - Variables can take on an infinite number of possible values from their domains
  - Hence there are potentially an infinite number of ways to apply the Universal Elimination rule
- Godel's Completeness Theorem says that FOL entailment is only semi-decidable
  - If a sentence is true given a set of axioms, there is a procedure that will determine this
  - If a sentence is false, there's no guarantee a procedure will ever discover this — it may never halt

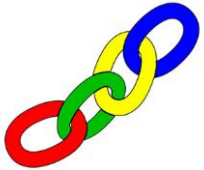
Generalized Modus Ponens (GMP)

- Modus Ponens: **P, P=>Q |= Q**
- Generalized Modus Ponens extends this to rules in FOL
- Combines And-Introduction, Universal-Elimination, and Modus Ponens, e.g.
  - given P(c) , Q(c) , $\forall x\ P(x) \wedge Q(x) \rightarrow R(x)$
  - derive R(c)
- Must deal with
  - more than one condition on rule's left side
  - variables

Forward & Backward Reasoning

- We often talk about two reasoning strategies:
  - Forward chaining and
  - Backward chaining
- Both are equally powerful, but optimized for different use cases
- You can also have a mixed strategy

Forward chaining



- Proofs start with given axioms/premises in KB, deriving new sentences using GMP until the goal/query sentence is derived
  - The process follows a chain of rules and facts going from the KB to the conclusion
- This defines a forward-chaining inference procedure because it moves "forward" from the KB to the goal [eventually]
- Inference using GMP is sound and complete for KBs containing only Horn clauses

Forward chaining example

- KB:
  - allergies(X) $\rightarrow$ sneeze(X)
  - cat(Y) $\wedge$ allergicToCats(X) $\rightarrow$ allergies(X)
  - cat(felix)
  - allergicToCats(mary)
- Goal:
  - sneeze(mary)

Backward chaining

- Backward-chaining deduction using GMP is also complete for KBs containing only Horn clauses
- Proofs start with the goal query, find rules with that conclusion, and then tries to prove each of the antecedents in the rule
- Keep going until you reach premises
- Avoid loops by checking if new subgoal is already on the goal stack
- Avoid repeated work: use a cache to check if new subgoal already proved true or failed

Backward chaining example

- KB:
  - allergies(X) $\rightarrow$ sneeze(X)
  - cat(Y) $\wedge$ allergicToCats(X) $\rightarrow$ allergies(X)
  - cat(felix)
  - allergicToCats(mary)
- Goal:
  - sneeze(mary)

Forward vs. backward chaining

- Forward chaining is data-driven
  - Automatic, unconscious processing, e.g., object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal
  - Efficient when you want to compute all conclusions

- Backward chaining is goal-driven, better for problem-solving and query answering
  - Where are my keys? How do I get to my next class?
  - Complexity can be much less than linear w.r.t KB size
  - Efficient when you want one or a few decisions
  - Good where the underlying facts are changing

Mixed strategy

- Many practical reasoning systems do both forward and backward chaining
- The way you encode a rule determines how it is used, as in

  % this is a forward chaining rule

  spouse(X,Y) => spouse(Y,X).

  % this is a backward chaining rule

  wife(X,Y) <= spouse(X,Y), female(X).
- Given a model of the rules you have and the kind of reason you need to do, it's possible to decide which to encode as FC and which as BC rules.

Summary

- Logical agents apply inference to KB to derive new information and make decisions
- Basic concepts of logic:
  – Syntax: formal structure of sentences
  – Semantics: truth of sentences wrt models
  – Entailment: necessary truth of one sentence given another
  – Inference: deriving sentences from other sentences
  – Soundness: derivations produce only entailed sentences
  – Completeness: derivations produce all entailed sentences
- FC and BC linear time, complete for Horn clauses
- Resolution is sound and complete for propositional and first-order logic