CS 188: Artificial Intelligence

First-Order Logic

Instructor: Stuart Russell

University of California, Berkeley
Spectrum of representations

(a) Atomic
Search, game-playing

(b) Factored
CSPs, planning, propositional logic, Bayes nets, neural nets

(b) Structured
First-order logic, databases, probabilistic programs
Expressive power

- **Rules of chess:**
  - 100,000 pages in propositional logic
  - 1 page in first-order logic

- **Rules of pacman:**
  - $\forall x, y, t \ \text{At}(x, y, t) \Leftrightarrow [\text{At}(x, y, t-1) \land \neg \exists \ u, v \ \text{Reachable}(x, y, u, v, \text{Action}(t-1))] \lor$ 
    $[\exists \ u, v \ \text{At}(u, v, t-1) \land \text{Reachable}(x, y, u, v, \text{Action}(t-1))]$
A possible world for FOL consists of:

- A non-empty set of objects
- For each k-ary predicate in the language, a set of k-tuples of objects (i.e., the set of tuples of objects that satisfy the predicate in this world)
- For each k-ary function in the language, a mapping from k-tuples of objects to objects
- For each constant symbol, a particular object (can think of constants as 0-ary functions)
Possible worlds

- A possible world for FOL consists of:
  - A non-empty set of objects
  - For each k-ary predicate in the language, a set of k-tuples of objects (i.e., the set of tuples of objects that satisfy the predicate in this world)
  - For each k-ary function in the language, a mapping from k-tuples of objects to objects
  - For each constant symbol, a particular object (can think of constants as 0-ary functions)

Knows(A, BFF(B))
A possible world for FOL consists of:
- A non-empty set of objects
- For each k-ary predicate in the language, a set of k-tuples of objects (i.e., the set of tuples of objects that satisfy the predicate in this world)
- For each k-ary function in the language, a mapping from k-tuples of objects to objects
- For each constant symbol, a particular object (can think of constants as 0-ary functions)

How many possible worlds?
Syntax and semantics: Terms

- A term refers to an object; it can be
  - A constant symbol, e.g., A, B, EvilKingJohn
    - The possible world fixes these referents
  - A function symbol with terms as arguments, e.g., BFF(EvilKingJohn)
    - The possible world specifies the value of the function, given the referents of the terms
      - BFF(EvilKingJohn) -> BFF(2) -> 3
  - A logical variable, e.g., x
    - (more later)
An atomic sentence is an elementary proposition (cf symbols in PL)

- A predicate symbol with terms as arguments, e.g., Knows(A,BFF(B))
  - True iff the objects referred to by the terms are in the relation referred to by the predicate
  - Knows(A,BFF(B)) -> Knows(1,BFF(2)) -> Knows(1,3) -> F

- An equality between terms, e.g., BFF(BFF(BFF(B)))=B
  - True iff the terms refer to the same objects
  - BFF(BFF(BFF(B)))=B -> BFF(BFF(BFF(2)))=2 -> BFF(BFF(3))=2 -> BFF(1)=2 -> 2=2 -> T
Syntax and semantics: Complex sentences

- Sentences with logical connectives
  \( \neg \alpha, \alpha \land \beta, \alpha \lor \beta, \alpha \Rightarrow \beta, \alpha \Leftrightarrow \beta \)

- Sentences with universal or existential quantifiers, e.g.,
  \( \forall x \text{ Knows}(x, \text{BFF}(x)) \)
  True in world \( w \) iff true in all extensions of \( w \) where \( x \) refers to an object in \( w \)
    - \( x \rightarrow 1: \text{Knows}(1, \text{BFF}(1)) \rightarrow \text{Knows}(1, 2) \rightarrow \text{T} \)
    - \( x \rightarrow 2: \text{Knows}(2, \text{BFF}(2)) \rightarrow \text{Knows}(2, 3) \rightarrow \text{T} \)
    - \( x \rightarrow 3: \text{Knows}(3, \text{BFF}(3)) \rightarrow \text{Knows}(3, 1) \rightarrow \text{F} \)
Syntax and semantics: Complex sentences

- Sentences with logical connectives
  \( \neg \alpha, \alpha \land \beta, \alpha \lor \beta, \alpha \Rightarrow \beta, \alpha \Leftrightarrow \beta \)

- Sentences with universal or existential quantifiers, e.g.,
  \[ \exists x \text{ Knows}(x, \text{BFF}(x)) \]
  - True in world \( w \) iff true in *some extension* of \( w \) where \( x \) refers to an object in \( w \)
    - \( x \to 1: \text{Knows}(1, \text{BFF}(1)) \to \text{Knows}(1, 2) \to T \)
    - \( x \to 2: \text{Knows}(2, \text{BFF}(2)) \to \text{Knows}(2, 3) \to T \)
    - \( x \to 3: \text{Knows}(3, \text{BFF}(3)) \to \text{Knows}(3, 1) \to F \)
Fun with sentences

- Everyone knows President Obama
- There is someone that everyone knows
- Everyone knows someone
More fun with sentences

- Any two people of the same nationality speak a common language
Entailment is defined exactly as for PL:

- \( \alpha \models \beta \) ("\( \alpha \) entails \( \beta \)" or "\( \beta \) follows from \( \alpha \)"") iff in every world where \( \alpha \) is true, \( \beta \) is also true.
- E.g., \( \forall x \) Knows(x,Obama) entails \( \exists y \forall x \) Knows(x,y)

If asked “Do you know what time it is?”, it’s rude to say “Yes”

Similarly, given an existentially quantified query, it’s polite to provide an answer in the form of a substitution (or binding) for the variable(s):

- KB = \( \forall x \) Knows(x,Obama)
- Query = \( \exists y \forall x \) Knows(x,y)
- Answer = Yes, \( \{y/\text{Obama}\} \)

Applying the substitution should produce a sentence that is entailed by KB.
Inference in FOL: Propositionalization

- Convert $(KB \land \neg \alpha)$ to PL, use a PL SAT solver to check (un)satisfiability
  - Trick: replace variables with ground terms, convert atomic sentences to symbols
    - $\forall x \text{Knows}(x,\text{Obama}) \land \text{Democrat}(\text{Feinstein})$
      - $\text{Knows}(\text{Obama},\text{Obama}) \land \text{Knows}(\text{Feinstein},\text{Obama}) \land \text{Democrat}(\text{Feinstein})$
      - $K_{O,O} \land K_{F,O} \land D_F$
    - and $\forall x \text{Knows}(\text{Mother}(x),x)$
      - $\text{Knows}(\text{Obama},\text{Obama}) \land \text{Knows}(\text{Mother}(\text{Obama}),\text{Obama}) \land \text{Knows}(\text{Mother}(\text{Mother}(\text{Obama})),\text{Obama})$ ……
  - Real trick: for $k = 1$ to infinity, use terms of function nesting depth $k$
    - If entailed, will find a contradiction for some finite $k$; if not, may continue for ever; *semidecidable*
Inference in FOL: Lifted inference

- Apply inference rules directly to first-order sentences, e.g.,
  - KB = Person(Socrates), \( \forall x \text{ Person}(x) \Rightarrow \text{Mortal}(x) \)
  - conclude \( \text{Mortal}(\text{Socrates}) \)
  - The general rule is a version of Modus Ponens:
    - Given \( \alpha[x] \Rightarrow \beta[x] \) and \( \alpha' \), where \( \alpha'\sigma = \alpha[x]\sigma \) for some substitution \( \sigma \) conclude \( \beta[x]\sigma \)
    - \( \sigma \) is \{x/Socrates\}
    - Given Knows(x,Obama) and Knows(y,z) \( \Rightarrow \) Likes(y,z)
      - \( \sigma \) is \{y/x, z/Obama\}, conclude Likes(x,Obama)

- Examples: Prolog (backward chaining), Datalog (forward chaining), production rule systems (forward chaining), resolution theorem provers
FOL is a very expressive formal language

Many domains of common-sense and technical knowledge can be written in FOL (see AIMA Ch. 12)

- circuits, software, planning, law, network and security protocols, product descriptions, ecommerce transactions, geographical information systems, Google Knowledge Graph, Semantic Web, etc.

Inference is semidecidable in general; many problems are efficiently solvable in practice

Inference technology for logic programming is especially efficient (see AIMA Ch. 9)