First-Order Logic

CS 4804 Fall 2020
Virginia Tech
Today’s Topics

• First-Order Logic (FOL)

(a) Atomic
Search Game

(b) Factored
CSP
Propositional Logic

(c) Structured
First-order Logic
Recap: Propositional Logic

• Knowledge Based Agent:
  – Maintains a **knowledge base**, which is a collection of logical sentences
  – Perform **logical inference** in order to make actions

• TELL

• ASK
Recap: Propositional Logic

- Logic has **syntax** and **semantics**
- **Entailment**: $\alpha \models \beta$ iff in every model where $\alpha$ is true, $\beta$ is also true
- **Logical Inference**
  - Naive approach: Model-checking
  - Better approach: Theorem proving
- **Inference Rules**: Modus Ponens, And-Elimination, and Monotonicity
- **Forward-chaining and Backward-chaining Algorithm**
Inference Rules

• Modus Ponens: \[ \frac{\alpha \Rightarrow \beta, \alpha}{\beta} \]
  - Whenever any sentences of the form \( \alpha \Rightarrow \beta \) and \( \alpha \) are given, sentence \( \beta \) can be inferred
  - \( \alpha \Rightarrow \beta \) and \( \alpha \) are in KB, new sentence \( \beta \) can be derived
  - \( \text{Rain} \Rightarrow \text{Wet}, \text{Rain} \Rightarrow \text{Wet} \), Wet can be inferred

• And-Elimination: \[ \frac{\alpha \land \beta}{\alpha} \]
  - From a conjunction, any of the conjuncts can be inferred
  - \( \text{Rain} \land \text{Wet} \Rightarrow \text{Rain} \), Wet can be inferred

• Monotonicity: if \( KB \models \alpha \) then \( KB \land \beta \models \alpha \)
Standard Logical Equivalences

\[(\alpha \land \beta) \equiv (\beta \land \alpha) \quad \text{commutativity of } \land\]
\[(\alpha \lor \beta) \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor\]
\[((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land\]
\[((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor\]
\[\neg(\neg \alpha) \equiv \alpha \quad \text{double-negation elimination}\]
\[(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition}\]
\[(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination}\]
\[(\alpha \iff \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{bidirectional elimination}\]
\[\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \quad \text{De Morgan}\]
\[\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \quad \text{De Morgan}\]
\[(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor\]
\[(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land\]
Example: CNF

\[ KB \]

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]

1. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \((\alpha \implies \beta) \land (\beta \implies \alpha)\).

\[ (B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1}) \]

2. Eliminate \( \implies \), replacing \( \alpha \implies \beta \) with \( \neg \alpha \lor \beta \):

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

\[ \neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \quad \text{(De Morgan)} \]

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1}) \]

\[ (\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land \]

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]
Recap: Horn, Definite, and Goal clause

- Definite clause: a disjunction of literals of which exactly one literal is positive
- Horn clause: a disjunction of literals of which at most one literal is positive
- Goal clause: a disjunction of literals of which no literal is positive
Horn clause

• Horn clause can be re-written as implications
  – E.g. (¬ A ∨ ¬ B ∨ C) ≡ (A ∧ B ⇒ C)
    \[¬(α ∧ β) ≡ (¬α ∨ ¬β) \quad \text{De Morgan} \quad (α ⇒ β) ≡ (¬α ∨ β) \quad \text{implication elimination}\]

• Forward chaining applies Modus Ponens to generate new facts:
  – Given \(X_1 \land X_2 \land \ldots X_n ⇒ Y\) and \(X_1, X_2, \ldots, X_n\)
  – Infer Y

• Forward chaining keeps applying this rule, adding new facts, until nothing more can be added

• Requires KB to contain only definite clauses:
  – (Conjunction of symbols) ⇒ symbol; or
  – A single symbol (note that X is equivalent to True ⇒ X)
Forward Chaining and Backward-chaining

\[
P \Rightarrow Q
\]
\[
L \land M \Rightarrow P
\]
\[
B \land L \Rightarrow M
\]
\[
A \land P \Rightarrow L
\]
\[
A \land B \Rightarrow L
\]
\[
A
\]
\[
B
\]
Possible Worlds

• Models for first-order logic:
  – A **nonempty** set of **objects**
  – The **domain** of a model is the set of objects (**domain elements**)
  – The domain is required to be **nonempty**
  – A **relation** is a set of tuples of objects that are related
  – **Unary** relation: properties e.g. “person”
  – **Binary** relations: relate pairs of objects
  – **Functions**: A given object must be related to exactly one object in this way
Example: Models in FOL

- Properties
  - Unary relation
- Unary function
- Binary relations
- Properties
  - Unary relation

Diagram:
- Objects labeled R and J
- Relations: brother, crown, on head
- Unary functions on objects
- Left leg connections among objects
- Objects labeled as Person and King
FOL Syntax

Sentence → AtomicSentence | ComplexSentence

AtomicSentence → Predicate | Predicate(Term, ...) | Term = Term

ComplexSentence → ( Sentence )
| ¬ Sentence
| Sentence ∧ Sentence
| Sentence ∨ Sentence
| Sentence → Sentence
| Sentence ↔ Sentence
| Quantifier Variable, ... Sentence

Term → Function(Term, ...)
| Constant
| Variable

Quantifier → ∀ | ∃
Constant → A | X₁ | John | ...
Variable → a | x | s | ...
Predicate → True | False | After | Loves | Raining | ...
Function → Mother | LeftLeg | ...

Operator Precedence:
\[ \neg, =, \land, \lor, \Rightarrow, \Leftrightarrow \]
Syntax and semantics: Symbols

- Stand for objects, relations, and functions
- **Constant** symbols stand for **objects**
- **Predicate** symbols stand for **relations**
- **Function** symbols stand for **functions**
- Predicate and function can have any **arity**: unary, binary, 3-ary, ..., n-ary
Syntax and semantics: Term

- A logical expression that refers to an object
- Constant: John
- Function: LeftLeg(John)
- Variable: The anonymous variables $a$, $x$, $y$ are standing for objects, and can be substituted for actual objects
  - If $x$ is a student, then $x$ is a person
Syntax and semantics: Sentences

• Atomic sentence: a predicate optionally followed by a parenthesized list of terms
  – Brother(Ryan, John)
  – Married(Father(Ryan), Mother(John))

• Complex sentence: Sentences with logical connectives
  – Brother(Ryan, John) ∧ Brother(John, Ryan)
  – Student(Ryan) ∨ Student(John)

• A sentence is true under a model if the relations described by the sentence are true under the mapping
Syntax and semantics: Quantifiers

• Quantifiers: Express properties of entire collections of objects

• Universal quantification ∀, has the meaning "for all"
  – ∀xP(x) is true in model m if P is true in all extension of m where x refers to an object in m (like P(A) ∧ P(B) ∧ P(C) ∧ …)
  – ∀xStudent(x) => Knows(x, programming)

• Existential quantification ∃, has the meaning "there exists"
  – ∃xP(x) is true in model m if P is true in some extension of m where x refers to an object in m (like P(A) ∨ P(B) ∨ P(C) ∨ …)
  – ∃xStudent(x) ∧ Knows(x, programming)

• Nested quantifiers: ∀x ∀y Brother(x, y) => Sibling(x, y)
Examples

- $\exists y \text{Course}(y) \land [\forall x \text{Student}(x) \Rightarrow \text{Takes}(x, y)]$
  - There is some course that every student has taken

- $\forall x \text{EvenInteger}(x) \land \text{Greater}(x, 2) \Rightarrow \exists y \exists z \text{Equals}(x, \text{Sum}(y, z)) \land \text{Prime}(y) \land \text{Prime}(z)$
  - Every even integer greater than 2 is the sum of two primes

- $\forall x \exists y \text{Love}(x, y)$ – Everybody loves somebody

- $\exists y \forall x \text{Love}(x, y)$ – There is someone who is loved by everyone

- $\forall x \forall y \forall z (\text{Student}(x) \land \text{Takes}(x, y) \land \text{Course}(y) \land \text{Covers}(y, z)) \Rightarrow \text{Knows}(x, z)$
  - If a student takes a course and the course covers a concept, the student knows that concept
De Morgan Rules

- $\forall x \neg P \equiv \neg \exists x P$
- $\neg \forall x P \equiv \exists x \neg P$
- $\forall x P \equiv \neg \exists x \neg P$
- $\exists x P \equiv \neg \forall x \neg P$
- $\neg (P \lor Q) \equiv \neg P \land \neg Q$
- $\neg (P \land Q) \equiv \neg P \lor \neg Q$
- $P \land Q \equiv \neg (\neg P \lor \neg Q)$
- $P \lor Q \equiv \neg (\neg P \land \neg Q)$

- $\forall x \text{Likes}(x, \text{IceCream})$ is equivalent to $\neg \exists x \neg \text{Like}(x, \text{IceCream})$
- $\forall x \neg \text{Likes}(x, \text{Parsnips})$ is equivalent to $\neg \exists x \text{Like}(x, \text{Parsnips})$
Syntax and semantics: Equality

• Signify that two terms refer to the same object
  – Father(John) = Henry
  – \( \neg(x = y) \) is \( x \neq y \)
TELL / ASK

- **TELL**: Add sentences (assertions) to a knowledge base (KB)
  - TELL(KB, King(John))
  - TELL(KB, $\forall x$ King(x) $\Rightarrow$ Person(x))

- **ASK(KB, King(John))**: return true
- **ASK(KB, Person(John))**: return true
- **ASK(KB, $\exists x$ Person(x))**: return true ???!
  - Query: Can you tell me the time? Answer: Yes

- **ASKVARS(KB, Person(x))**: return {x/John}
Substitution

- A substitution $\theta$ is a mapping from variables to terms
- $\text{SUBST}(\theta, \alpha)$ denote the result of applying the substitution $\theta$ to the sentence $\alpha$
- $\text{SUBST}(\{x/\text{ryan}\}, P(x)) = P(\text{ryan})$
- $\text{SUBST}(\{x/\text{ryan}, y/z\}, P(x) \land Q(x,y)) = P(\text{ryan}) \land Q(\text{ryan}, z)$
First-Order Logical Inference

- Propositionalization
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution
Propositionalization

- Convert the first-order logic KB to propositional logic and use propositional inference
- Eliminate quantifiers
- Each universal (existential) quantifier sentence can be converted to a conjunction (disjunction) of sentences with a clause for each possible object that could be substituted in for the variable

**KB in first-order logic**
- Student(ryan) ∧ Student(john)
- ∀x Student(x) ⇒ Person(x)
- ∃x Student(x) ∧ Creative(x)

**KB in propositional logic**
- Student(ryan) ∧ Student(john)
- (Student(ryan) ⇒ Person(ryan)) ∧ (Student(john) ⇒ Person(john))
- (Student(ryan) ∧ Creative(ryan)) ∨ (Student(john) ∧ Creative(john))
Unification

• Takes two sentences and returns a substitution $\theta$ which is the most general unifier, if one exist

• $\text{UNIFY}(p, q) = \theta$ where $\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$ or “fail” if no such $\theta$ exists

• $\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(\text{John}, \text{Jane})) = \{x/\text{Jane}\}$

• $\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(y, \text{Bill})) = \{x/\text{Bill}, y/\text{John}\}$

• $\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(y, \text{Mother}(y))) = \{y/\text{John}, x/\text{Mother}(\text{John})\}$

• $\text{UNIFY}(\text{Knows}(\text{John}, x), \text{Knows}(x, \text{Ryan})) = \text{failure}$
Generalized Modus Ponens

\[ p_1', \ p_2', \ \ldots, \ p_n', \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q) \]
\[ \text{SUBST}(\theta, q) \]

- Premises:
  - King(John), Greedy(y), ...
  - King(x) \wedge \text{Greed}(x) \wedge \ldots \Rightarrow \text{Evil}(x)
- Conclusion:
  - \( \theta = \{x/\text{John}, \ y/\text{John}, \ \ldots\} \)
  - Derive \( \text{SUBST}(\theta, q) = \text{Evil}(\text{John}) \)
- Generalized Modus Ponens is a lifted version of Modus Ponens
  - It raises Modus Ponens from ground propositional logic to first-order logic
Semidecidable

- Generalized Modus Ponens is complete for definite clauses
- The question of entailment for first-order logic is semidecidable
  - Algorithms exist that find every entailed sentence ($KB \models f$)
  - No algorithm exists that find every nonentailed sentence (Says no to $KB \models f$)
Forward Chaining

• FOL definite clauses: a disjunction of literals of which exactly one is positive
  – Existential quantification $\exists$ is not allowed
  – Universal quantification $\forall$ is left implicit
  – $\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

• Problem statement: The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

• Can we prove if West is a criminal?
Forward Chaining – Datalog KB

“… it is a crime for an American to sell weapons to hostile nations”:

\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x).
\]  
(9.3)

“Nono … has some missiles.” The sentence \( \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \) is transformed into two definite clauses by Existential Instantiation, introducing a new constant \( M_1 \):

\[
\begin{align*}
\text{Owns} & (\text{Nono}, M_1) \quad (9.4) \\
\text{Missile} & (M_1) \quad (9.5)
\end{align*}
\]

“All of its missiles were sold to it by Colonel West”:

\[
\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}).
\]  
(9.6)

We will also need to know that missiles are weapons:

\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x)
\]  
(9.7)

and we must know that an enemy of America counts as “hostile”:

\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x).
\]  
(9.8)

“West, who is American …”:

\[
\text{American}(\text{West}).
\]  
(9.9)

“The country Nono, an enemy of America …”:

\[
\text{Enemy}(\text{Nono}, \text{America}).
\]  
(9.10)
American\( (x) \land Weapon\( (y) \land Sells\( (x, y, z) \land Hostile\( (z) \Rightarrow Criminal\( (x) \)
Forward Chaining – Second Iteration

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
Backward Chaining

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

- **Criminal**(West)
- **American**(West)
- **Weapon**(y)
- **Sells**(West, M₁, z)
- **Hostile**(Nono)
- **Missile**(y)
- **Missile**(M₁)
- **Owns**(Nono, M₁)
- **Enemy**(Nono, America)

KB

\{\}

\{z/Nono\}

\{y/M₁\}

\{\}

\{\}

\{\}

\{\}
Resolution

- First-order logic includes non-Horn clauses
- High-level strategy
  - Convert all formulas to CNF
  - Repeatedly apply resolution rule
- Input: \( \forall x \ American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow \text{Criminal}(x) \)
- Output: \( \neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x, y, z) \lor \neg Hostile(z) \lor \text{Criminal}(x) \)

\[
\text{SUBST}(\theta, \ell_1 \lor \cdots \lor \ell_k, \ m_1 \lor \cdots \lor m_n)
\]

\[
[\text{Animal}(F(x)) \lor \text{Loves}(G(x), x)] \quad \text{and} \quad [\neg \text{Loves}(u, v) \lor \neg \text{Kills}(u, v)]
\]

\[
\theta = \{u/G(x), v/x\}
\]

[\text{Animal}(F(x)) \lor \neg \text{Kills}(G(x), x)]
Summary

• In **propositional logic**, we model our world as a set of **attributes**, that are **true** or **false**. This **binary view** of the world is what is known as a **factored representation**, which we used when we were solving CSP’s.

• In **first-order logic**, our world consists of **objects** that relate to one another. This **object-oriented view** of the world is known as a **structured representation**, is in many ways more expressive and is more closely aligned with the language we naturally use to speak about the world.

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

• Inference technology for logic programming is especially efficient.
## Summary

<table>
<thead>
<tr>
<th>Propositional Logic</th>
<th>First-order Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Checking</td>
<td>N/A</td>
</tr>
<tr>
<td>Modus pones</td>
<td>Generalized Modus pones*</td>
</tr>
<tr>
<td>Forward chaining</td>
<td>Forward chaining*</td>
</tr>
<tr>
<td>Backward chaining</td>
<td>Backward chaining*</td>
</tr>
<tr>
<td>Resolution</td>
<td>Resolution*</td>
</tr>
</tbody>
</table>

* Unification and substitution
Reading and Next Class

- First-order logic: AIMA 8, 9
- Next: MDP: AIMA 17.1-17.3