Introduction to Artificial Intelligence

Lecture 8 – Logical reasoning

CS/CNS/EE 154
Andreas Krause
Logics in general

- **Logics** are formal languages for representing information such that conclusions can be drawn.
- **Syntax** defines the **sentences** in the language.
- **Semantics** defines the “meaning” of sentences, i.e., the **truth** of a sentence in a **world** (environment state).

**Example**: Language of arithmetic

\[
\begin{align*}
3 + x & \neq 4 \quad \text{not well-formed} \\
3 + 4 & = 6 \quad \text{well-formed, but false} \\
3 + x & = 6 \quad \text{true in world } \{(x, 3)\} \\
3 + x & = 6 \quad \text{false in world } \{(x, 2)\} \\
3 & = 3 \quad \text{true in all worlds}
\end{align*}
\]
Models

- Logicians think in terms of models
  - Formally structured worlds w.r.t. which truth can be evaluated

We say $m$ is a model of a sentence $\alpha$ if $\alpha$ is true in $m$

$\alpha \equiv (x + 2 = 5)$ is true in model $m = \{(x, 3)\}$

$M(\alpha)$ is the set of all models of $\alpha$

$\alpha \equiv (x + 2 = y)$  $M(\alpha) = \{\{(x, 0), (y, 2)\}, \{(x, 3), (y, 5)\}, \ldots\}$

Then $KB \models \alpha$ if and only if

$M(KB) \subseteq M(\alpha)$
$KB = \text{wumpus-world rules} + \text{observations}$
$KB = \text{wumpus-world rules + observations}$

$\alpha_1 = "[1,2] \text{ is safe}"$, $KB \models \alpha_1$
Propositional logic: Syntax

Simplest example of a logic; illustrates basic ideas

Propositional symbols are sentences
If $S$ is a sentence, $\neg S$ is a sentence (negation)
If $S_1$ and $S_2$ are sentences, $S_1 \land S_2$ is a sentence (conjunction)
If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ is a sentence (disjunction)
Notation shorthand:
- $S_1 \implies S_2$ for $\neg S_1 \lor S_2$ (implication)
- $S_1 \iff S_2$ for $(S_1 \implies S_2) \land (S_2 \implies S_1)$ (biconditional)
Propositional logic: Semantics

Each **model** specifies *true* or *false* for each proposition symbol

E.g.

\[ P_{1,2} \quad P_{2,2} \quad P_{3,1} \]
\[ \text{false} \quad \text{true} \quad \text{false} \]

Rules for evaluating truth with respect to a model \( m \):

\[ \neg S \quad \text{is true iff} \quad S \text{ is false} \]

\[ S_1 \land S_2 \quad \text{is true iff} \quad S_1 \text{ is true and } S_2 \text{ is true} \]

\[ S_1 \lor S_2 \quad \text{is true iff} \quad S_1 \text{ is true or } S_2 \text{ is true} \]

Simple recursive process evaluates an arbitrary sentence, e.g.,

\[ \neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) \]
Wumpus world in prop. logic

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$$
\mathcal{KB} = \left\{ \neg P_{1,1}, \neg B_{1,1}, B_{2,1} \right\}
$$

$\mathcal{KB} \equiv \neg P_{1,1} \wedge \neg B_{1,1} \wedge B_{2,1}$

"Pits cause breezes in adjacent squares"

$$
B_{1,1} \iff (P_{1,2} \vee P_{2,1})
$$

$$
B_{2,1} \iff (P_{1,1} \vee P_{2,2} \vee P_{3,1})
$$
Two main classes of methods for proving $KB \models \alpha$

**Model checking**
- Truth table enumeration (always exponential in n)
- Better: CSP (e.g., improved backtracking such as DPLL)
  Check whether $(KB \land \neg \alpha)$ is unsatisfiable

**Proof using inference**
- Apply sequence of inference rules (syntactic manipulations)
- Can use inference rules in a standard search algorithm
Logical inference

- **Inference**: procedure $i$ for deducing (proving) sentences from knowledge base.

- We say $KB \vdash_i \alpha$ if $\alpha$ can be inferred from $KB$ using inference procedure $i$.

- Inference $i$ is called
  - **Sound** if whenever $KB \vdash_i \alpha$ then also $KB \models \alpha$.
  - **Complete** if whenever $KB \models \alpha$ then also $KB \vdash_i \alpha$.

- Thus, a sound and complete inference procedure *correctly* answers *any* question whose answer can be inferred from $KB$. 
Resolution

- Assumes sentences in Conjunctive Normal Form (CNF)
  - This is no restriction (Tseitin transformation)
  - Example: \((P_{i1} \lor \neg B_{i2}) \land (B_{i2} \lor P_{i1} \lor P_{22}) \land \ldots\)

- Resolution inference rule

\[
\begin{array}{c}
\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n \\
\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \ell_k \lor m_1 \lor m_{j-1} \lor m_{j+1} \cdots \lor m_n
\end{array}
\]

- Sound and complete for propositional logic!

- Example:

\[
\begin{array}{c}
\neg R \lor W, \quad \text{Wet Lawn} \Rightarrow \text{Slippery}
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W, \quad \text{Wet Lawn} \Rightarrow \text{Slippery}
\end{array}
\]

\[
\begin{array}{c}
R \Rightarrow \text{Wet Lawn}, \quad \text{Wet Lawn} \Rightarrow \text{Slippery}
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]

\[
\begin{array}{c}
\neg R \lor W
\end{array}
\]
Resolution example

\[ KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \quad \alpha = \neg P_{1,2} \]

Thus, \( KB \vdash \alpha \)
Logical reasoning with resolution

- Resolution is complete
  - Any propositional sentence is entailed if and only if it can be proven by resolution

- BUT: Finding the proof can be difficult!
  - Must search through possible applications of resolution rule
  - Search space exponentially large

- 3CNF SAT is NP complete!
  - Existence of polynomial time algorithm considered unlikely

- Are there special kinds of sentences that are “easy” to prove??
Horn clauses

- Special types of propositional formulae
- A Horn clause is
  - A propositional symbol; or
  - (conjunction of symbols) \( \Rightarrow \) symbol

\[
\begin{align*}
\text{Grades} \land \text{GRE} \land \text{Statement} \land \text{Letter} & \Rightarrow \text{Grad School} \\
\text{Research} & \Rightarrow \text{Letter} \\
\text{SURF} & \Rightarrow \text{Research}
\end{align*}
\]

\[
\begin{align*}
\text{Study} & \Rightarrow \text{GRE} \land \text{Grades} \land \text{Letter} \land \text{NA} \land \text{Horn} \\
& \equiv \neg \text{Study} \lor (\text{GRE} \land \text{Grades}) \\
& \equiv \neg \text{Study} \lor \neg \text{GRE} \lor \neg \text{Grades} \\
& \equiv (\neg \text{Study} \lor \neg \text{GRE}) \land (\neg \text{Study} \lor \neg \text{Grades}) \land \text{Horn Clause}
\end{align*}
\]

\[
\text{Study} \Rightarrow \text{GRE}
\]
Forward and backward chaining

- Inference procedure for special types of KBs, consisting only of Horn clauses

- Modus ponens complete for Horn formulas 😊

\[
\alpha_1, \ldots, \alpha_k, \quad \alpha_1 \land \cdots \land \alpha_k \Rightarrow \beta
\]

- Inference algorithms: forward and backward chaining
**Forward chaining**

- *Idea:* fire any rule whose premises are satisfied in the *KB*,
  - add its conclusion to the *KB*, until query is found

\[
P \Rightarrow Q \\
L \land M \Rightarrow P \\
B \land L \Rightarrow M \\
A \land P \Rightarrow L \\
A \land B \Rightarrow L \\
A \\
B
\]
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Proof of completeness

FC derives every atomic sentence that is entailed by $KB$

1. FC reaches a **fixed point**: no new atomic sentences are derived
2. Consider final state as model $m$, assigning true/false to symbols
3. Every clause in the original $KB$ is true in $m$
   \[ a_1 \land ... \land a_k \Rightarrow b \]
4. Hence $m$ is a model of $KB$
5. If $KB \models q$, $q$ is true in every model of $KB$, including $m$
Backward chaining

*Idea*: work backwards from the query $Q$:

- check if $Q$ is known already, or
- prove by BC all premises of some rule concluding $Q$

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

1. has already been proved true, or
2. has already failed
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Forward vs. backward chaining

- FC is **data-driven**, automatic, unconscious processing,
  - e.g., simple model for object recognition, routine decisions

- May do lots of work that is irrelevant to the goal

- BC is **goal-driven**, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?

- Complexity of BC can be **much less** than linear in size of KB
Summary so far:

- **Logic** = formal language with
  - **Syntax** (what sentences are valid?)
  - **Semantics** (what valid sentences are true?)

- Simple example: **Propositional logic**

- Can infer entailment of sentences using
  - **Model checking** (e.g., Constraint satisfaction)
  - **Logical inference** (should be sound and complete)

- **Inference procedures**
  - **Resolution**: Sound and complete for arbitrary prop. formulas, but exponential search space
  - **Forward-/Backward chaining**: Sound; complete only for *Horn* formulas. Inference in (sub-) linear time!
Issues with propositional Wumpus world

Need “copies” of symbols and sentences for each cell

\[ P_{1,1} \text{ is true if there is a pit in } [1,1] \]
\[ P_{1,2} \text{ is true if there is a pit in } [1,2] \]
\[ \ldots \]
\[ P_{n,n} \text{ is true if there is a pit in } [n,n] \]

\[ B_{1,1} \text{ is true if there is a breeze in } [1,1] \quad \ldots \]
\[ B_{n,n} \text{ is true if there is a breeze in } [n,n] \]
\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}); B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}); \ldots \]
First order logic (FOL)

- Propositional logic is about simple facts
  - “There is a breeze at location [1,2]”

- First order logic is about facts involving
  - *Objects*: Numbers, people, locations, time instants, ...
  - *Relations*: Alive, IsNextTo, Before, ...
  - *Functions*: MotherOf, BestFriend, SquareRoot, OneMoreThan, ...

- Will be able to say:
  - IsBreeze(x); IsPit(x); IsNextTo(x,y)
  - \( \forall x, y : (IsPit(x) \land IsNextTo(x, y)) \Rightarrow IsBreeze(y) \)
Simple example

About King Richard the Lionheart and his evil brother John

Objects:
- Richard
- John
- Crown

Relations
- Richard and John are brothers
- Richard is a king

Function
- Refer to specific properties of Richard and John, e.g., their head, legs, ...
FOL: Basic syntactic elements

- **Constants**: KingJohn, 1, 2, ..., [1,1], [1,2], ..., [n,n], ...
- **Variables**: x, y, z, ...
- **Predicates**: Brother, ≥, =, ...
- **Functions**: LeftLegOf, MotherOf, Sqrt, ...
- **Connectives**: ∧, ∨, ¬
- **Quantifiers**: ∀, ∃

Constant, predicates and functions are mere *symbols* (i.e., have no meaning on their own)
FOL Syntax: Atomic sentences

A (variable-free) term is a
- constant symbol or
- k-ary function symbol: \( \text{function}(\text{term}_1, \text{term}_2, \ldots, \text{term}_k) \)

Example: \( \text{LeftLegOf(KingJohn)}, \text{IsBreeze([1,2])} \)

An atomic sentence is a predicate symbol applied to terms

Example:
- \( \text{Brother(KingJohn, RichardLionheart)} \)
- \( \text{IsNextTo([1,1],[1,2])} \)
- \( > (\text{Length(LeftLegOf(KingJohn))}, \text{Length(LeftLegOf(RichardLionheart))}) \)
FOL Syntax: Composite sentences

- Composite sentences are
  - Atomic sentences or
  - Composite sentences joined by connectives

Example:

\[ \text{BrotherOf(KingJohn, RichardLionheart)} \Rightarrow \text{BrotherOf(RichardLionheart, KingJohn)} \]
Models in FOL

- Much more complicated than in Propositional Logic
- Models contain
  - Set of objects (finite or countable)
  - Set of relations between objects (map obj’s to truth values)
  - Set of functions (map objects to other objects)

and their interpretations:
- Mapping from constant symbols to model objects
- Mapping from predicate symbols to model relations
- Mapping from function symbols to model functions

An atomic sentence \( \text{predicate}(\text{term}_1, \text{term}_2, \ldots, \text{term}_k) \) is true if the objects referred to by \( \text{term}_1, \text{term}_2, \ldots, \text{term}_k \) are in the relation referred to by \( \text{predicate} \)
Models in FOL: Example
Objects: R, J, C, LegR, LegJ, N

Functions: LLO
  - LLO(R) = LegR; LLO(J) = LegJ; LLO(C) = N; LLO(LegR) = N; ...

Relations:
  - B = \{(R, J)\}; OH = \{(C, J)\};
  - K = \{J\}; P = \{R, J\}

Mappings:
  - Richard: R; John: J
  - LeftLegOf: LLO;
  - Brother: B; OnHead(OH)
Specifying known facts is tedious

E.g., need

- \( \neg \text{OnHead}(R, J) \)
- \( \neg \text{OnHead}(\text{LeftLeg1}, J) \)
- \( \neg (R = J) \)
- \( \neg \text{OnHead}(\text{LeftLeg1}, \text{LeftLeg2}) \)
- \( \neg (R = \text{LeftLeg1}) \)
- ...

![Diagram showing relationships between R and J, including on head, brother, and left leg connections.](image-url)
Indeterminate number of objects

Let’s look at all possible models for a language with
- Two constants: R, J
- One binary relation: B
"Database" semantics

- Typically conventions
  - Closed-world: Atomic sentences not in KB are false
  - Unique names: Different constants refer to different objects
  - Domain closure: Only allow model objects that are associated with constant symbols
Quantifiers

- Allow variables in addition to constants
  \[ \text{Homework}(x, 154) \]

- Sentences with free variables: \( S(x) \)

- Quantifiers bind free variables
  \[ \forall x : S(x) \] is true if \( S(x) \) is true for all instantiations of \( x \)
  (i.e., for each possible object in the model)

  \[ \exists x : S(x) \] is true if \( S(x) \) is true for at least one
  instantiation of \( x \) (i.e., for some object)

- Example:

  - All homeworks in 154 are hard
    \[ \forall x : (\text{Homework}(x, 154) \Rightarrow \text{Hard}(x)) \]
  
  - At least one of the 154 homeworks is hard
    \[ \exists x : \text{Homework}(x, 154) \land \text{Hard}(x) \]
Properties of quantifiers

- Is $\forall x \; \forall y \; S(x, y)$ the same as $\forall y \; \forall x \; S(x, y)$?

- Is $\exists x \; \exists y \; S(x, y)$ the same as $\exists y \; \exists x \; S(x, y)$?

- Is $\exists x \; \forall y \; S(x, y)$ the same as $\forall y \; \exists x \; S(x, y)$?

  - $\exists x \; \exists y \; \text{Love}(x, y)$ \quad There is someone who loves everyone

  - $\forall y \; \exists x \; \text{Love}(x, y)$ \quad Everybody is loved by someone
De Morgan’s law for quantifiers

Each quantifier can be expressed by the other (they are dual to each other)

\[ \forall x \in S(k) \equiv \exists x \in S(k) \]