Title: Logical Agents

AIMA: Chapter 7 (Sections 7.4 and 7.5)

Introduction to Artificial Intelligence

CSCE 476-876, Fall 2022

URL: `www.cse.unl.edu/~choueiry/F22-476-876`

Berthe Y. Choueiry (Shu-we-ri)

(402)472-5444

Outline

- Login in general: models and entailment
- Propositional (Boolean) logic
- Equivalence, validity and satisfiability
- Inference:
  - By model checking
  - Using inference rules
  - Resolution algorithm: Conjunctive Normal form
  - Horn theories: forward and backward chaining

A logic consists of:

1. A formal representation system:
   (a) Syntax: how to make sentences
   (b) Semantics: systematics constraints on how sentences relate to the states of affairs
2. Proof theory: a set of rules for deducing the entailment of a set of sentences

Example:

$\surd$ Propositional logic (or Boolean logic)

$\surd$ First-order logic FOL

Models (I)

A model is a world in which a sentence is true under a particular interpretation. *General definition*

Logicians typically think in terms of <u>models</u>, which are formally <u>structured</u> worlds with respect to which truth can be evaluated

We say m is a <u>model</u> of a sentence α if α is true in m

Typically, a sentence can be true in many models

Models (II)

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

Entailment: A sentence α is entailed by a KB if the models of the KB are all models ofα

KB $\models \alpha$ iff all models of KB are models of α (*i.e.*, $M(KB) \subseteq M(\alpha)$)

Then KB $\models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

Inference

- Example of inference procedure: deduction
- Validity of a sentence: always true (i.e., under all possible interpretations)
  `The Earth is round or not round`
  $\rightarrow$ Tautology
- Satisfiability of a sentence: sometimes true (i.e., $\exists$ some interpretation(s) where it holds)
  `Alex is on campus`
- Insatisfiability of a sentence: never true (i.e., $\nexists$ any interpretation where it holds)
  `The Earth is round and the earth is not round`
  $\rightarrow$ useful for refutation, as we will see later

Beauty of inference:
Formal inference allows the computer to derive **valid** conclusions even when the computer does not know the interpretation you are using

Syntax of Propositional Logic

Propositional logic is the simplest logic—illustrates basic ideas

- Symbols represent whole propositions, sentences
  D says the Wumpus is dead
  The proposition symbols P_1, P_2, etc. are sentences

- Boolean connectives: $\wedge, \vee, \neg, \Rightarrow$ (alternatively, $\rightarrow, \supset$), $\Leftrightarrow$, connect sentences
  If S_1 and S_2 are sentences, the following are sentences too:
  $\neg S_1$, $\neg S_2$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$

Formal grammar of Propositional Logic: Backus-Naur Form, check Figure 7.7 page 218 in AIMA

Terminology

Atomic sentence: single symbol

Complex sentence: contains connectives, parentheses

Literal: atomic sentence or its negation (e.g., P, $\neg Q$)

Sentence $(P \wedge Q) \Rightarrow R$ is an implication, conditional, rule, if-then statement

$(P \wedge Q)$ is a premise, antecedent

R is a conclusion, consequence

Sentence $(P \wedge Q) \Leftrightarrow R$ is an equivalence, biconditional

Precedence order resolves ambiguity (highest to lowest):

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

E.g., $((\neg P) \vee (Q \wedge R)) \Rightarrow S$ Careful: $A \wedge B \wedge C$ and $A \Rightarrow B \Rightarrow C$

Syntax of First-order logic (Chapter 8)

First-Order Logic (FOL) is expressive enough to say almost anything of interest and has a sound and complete inference procedure

- Logical symbols:
  - parentheses
  - connectives ($\neg, \Rightarrow$, the rest can be regenerated)
  - variables
  - equality symbol (optional)
- Parameters:
  - quantifier $\forall$
  - predicate symbols
  - constant symbols
  - function symbols

Semantics of Propositional Logic

Semantics is defined by specifying:
— Interpretation of a proposition symbols and constants (T/F)
— Meaning of logical connectives

Proposition symbol means what ever you want:

D says the Wumpus is dead
Breeze says the agent is feeling a breeze
Stench says the agent is perceiving an unpleasant smell

Connectives are functions: complex sentences meaning derived from the meaning of its parts

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

Note:
$P \Rightarrow Q$: if P is true, Q is true, otherwise I am making no claim

Models in propositional logic *Careful!*

- A model is a mapping from proposition symbols directly to truth or falsehood
- The models of a sentence are the mappings that make the sentence true

Example:

	α:	obj1 $\wedge$ obj2
$\surd$	Model1:	obj1 $=$ 1 and obj2 $=$1
$\times$	Model2:	obj1 $=$ 0 and obj2 $=$1

Wumpus world in Propositional Logic

$P_{i,j}$: there is a pit in $[i,j]$

$B_{i,j}$: there is a breeze in $[i,j]$

- $R_1 : \neg P_{1,1}$
- "Pits cause breezes in adjacent squares"
  $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- Percepts:
  $R_4 : \neg B_{1,1}$
  $R_5 : B_{2,1}$
- KB: $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$
- Questions: KB $\models \neg P_{1,2}$? KB $\not\models P_{2,2}$?

Wumpus world in Propositional Logic

Given KB: $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

Number of symbols: 7
Number of models: 2^7 =128

<See Figure 7.9, page 221>

KB is true in only 3 models

$P_{1,2}$ is false but $\neg P_{1,2}$ holds in all 3 models of the KB,
thus KB $\models \neg P_{1,2}$

$P_{2,2}$ is true in 2 models, false in third, thus KB $\not\models P_{2,2}$

Enumeration method in Propositional Logic

Let $\alpha = A \vee B$ and $KB = (A \vee C) \wedge (B \vee \neg C)$

Is it the case that KB $\models \alpha$?

Check all possible models—α must be true wherever KB is true

A	B	C	$A \vee C$	$B \vee \neg C$	KB	α
F	F	F				
F	F	T				
F	T	F				
F	T	T				
T	F	F				
T	F	T				
T	T	F				
T	T	T				

Complexity? In propositional logic, inference is exponential in the number of terms in the theory.

Inference by enumeration

- Algorithm: TT-ENTAILS?(KB, α), Figure 7.10 page 221
  – Identifies all the symbols in kb
  – Performs a recursive enumeration of all possible assignments (T/F) to symbols
  – In a depth-first manner
- It terminates: there is only a finite number of models
- It is sound: because it implements definition of entailment
- It is complete, and works for any KB and α
- Time complexity: $O(2^n)$, for a KB with n symbols
- Alert: Entailment in Propositional Logic is co-NP-Complete

Important concepts

- Logical equivalence
- Validity
  Deduction theorem: links validity to entailment
- Satisfiability
  Refutation theorem: links satisfiability to entailment

Logical equivalence

Two sentences are logically equivalent $\alpha \Leftrightarrow \beta$ iff true in same models

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{array}{rcll}
(\alpha \wedge \beta) & \equiv & (\beta \wedge \alpha) & \text{commutativity of } \wedge \\
(\alpha \vee \beta) & \equiv & (\beta \vee \alpha) & \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) & \equiv & (\alpha \wedge (\beta \wedge \gamma)) & \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) & \equiv & (\alpha \vee (\beta \vee \gamma)) & \text{associativity of } \vee \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \implies \beta) & \equiv & (\neg\beta \implies \neg\alpha) & \text{contraposition} \\
(\alpha \implies \beta) & \equiv & (\neg\alpha \vee \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \implies \beta) \wedge (\beta \implies \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) & \equiv & (\neg\alpha \vee \neg\beta) & \text{de Morgan} \\
\neg(\alpha \vee \beta) & \equiv & (\neg\alpha \wedge \neg\beta) & \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) & \equiv & ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) & \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) & \equiv & ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) & \text{distributivity of } \vee \text{ over } \wedge
\end{array}
$$

Validity

A sentence is valid if it is true in all models

e.g., $P \vee \neg P$, $\quad P \Rightarrow P$, $\quad (P \wedge (P \Rightarrow H)) \Rightarrow H$

To establish validity, use truth tables:

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

If every row is true, then he conclusion, P, is entailed by the premises, $((P \vee H) \wedge \neg H)$

Use of validity: Deduction Theorem:

KB $\models \alpha$ iff (KB $\Rightarrow \alpha$) is valid

TT-Entails?(KB, α) checks the validity of (KB $\Rightarrow \alpha$)

Satisfiability

A sentence is satisfiable if it is true in some model
e.g., $A \vee B$, C

Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence (e.g., SAT!)

A sentence is unsatisfiable if it is true in no models
e.g., $A \wedge \neg A$

Satisfiability and validity are connected:
α valid iff $\neg\alpha$ is unsatisfiable and α satisfiable iff $\neg\alpha$ is not valid

Use of satisfiability: refutation

$$\text{KB} \models \alpha \text{ iff } (\text{KB} \wedge \neg\alpha) \text{ is unsatisfiable}$$

i.e., prove α by *reductio ad absurdum*

Proof methods

Proof methods divide into (roughly) two kinds

Model checking

- Truth-table enumeration (sound & complete but exponential)
- Backtrack search in model space (sound & complete)
  e.g., Davis-Putnam Algorithm (DPLL) (Section 7.6)
- Heuristic search in model space (sound but incomplete)
  e.g., the GSAT algorithm, the WalkSat algorithm (Section 7.6)

Application of inference rules

- Legitimate (sound) generation of new sentences from old
- Proof = a sequence of inference rule applications, can use inference rules as operators in a standard search algorithm
- Typically require translation of sentences into a normal form

Inference rules for Propositional Logic (I)

Reasoning patterns

- Modus Ponens (Implication-Elimination)

$$\frac{\alpha \Rightarrow \beta, \;\; \alpha}{\beta}$$

- And-Elimination

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

- We can also use all logical equivalences as inference rules:

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \text{ and } \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Soundness of an inference rule can be verified by building a truth table

Inference rules and equivalences in the wumpus world

Given KB: $R_1 \land R_2 \land R_3 \land R_4 \land R_5$

Prove: $\neg P_{1,2}$

- Biconditional elimination to $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
  $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- And elimination to R_6:
  $R_7 : ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- Logical equivalence of contrapositives:
  $R_8 : (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1}))$
- Modus ponens on R_8 and $R_4 : \neg B_{1,1}$
  $R_9 : \neg(P_{1,2} \lor P_{2,1})$
- De Morgan's rules: $R_{10} : \neg P_{1,2} \land \neg P_{2,1}$

The job of an inference procedure is to construct proofs by
finding appropriate sequences of applications of inference rules

starting with sentences initially in KB

and culminating in the generation of the sentence
whose proof is desired

Complexity of propositional inference

Truth-table: Sound and complete.

2^n rows: exponential, thus impractical

Entailment is co-NP-Complete in Propositional Logic

Inference rules: sound (resolution gives completeness)

NP-complete in general

However, we can focus on the sentences and propositions of interest: The truth values of all propositions need not be considered

Monotonicity

the set of entailed sentences can only increase as information (new sentences) is added to the KB:

if KB $\models \alpha$ then (KB$\wedge\beta$)$\models \alpha$

Monotonicity allows us to apply inference rules whenever suitable premises appear in the KB: the conclusion of the rule follows regardless of what else is in the KB

PL, FOL are monotonic, probability theory is not

Monotonicity essential for soundness of inference

Resolution for completeness of inference rules

- Unit resolution:

$$\frac{l_1 \vee l_2, \quad \neg l_2}{l_1}$$

  More generally:

$$\frac{l_1 \vee \cdots \vee l_k, \quad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

  where l_i and m are complementary literals

- Resolution:

$$\frac{l_1 \vee l_2, \quad \neg l_2 \vee l_3}{l_1 \vee l_3}$$

  More generally:

$$\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  where l_i and m_j are complementary literals

Resolution in the wumpus world

Agent goes [1, 1], [2, 1], [1, 1], [1, 2]

Agent perceives a stench but no breeze: $R_{11} : \neg B_{1,2}$
But $R_{12} : B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
Applying biconditional elimination to R_{12}, followed by and-elimination, contraposition, and finally modus ponens with R_{11}, we get:
$R_{13} : \neg P_{2,2}$ and $R_{14} : \neg P_{1,3}$

$R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
$R_5 : B_{2,1}$
Now, applying biconditional elimination too R_3 and modus ponens with R_5, we get:
$R_{15} : P_{1,1} \vee P_{2,2} \vee P_{3,1}$
Resolving R_{13} and R_{15}: $R_{16} : P_{1,1} \vee P_{3,1}$
Resolving R_{16} and R_1: $R_{17} : P_{3,1}$

Soundness of the resolution rule

Resolution rule: $\frac{\alpha \lor \beta, \quad \neg\beta \lor \gamma}{\alpha \lor \gamma}$ or equivalently $\frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$

α	β	γ	$\alpha \lor \beta$	$\neg\beta \lor \gamma$	$\alpha \lor \gamma$
False	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*True*	*True*
False	*True*	*False*	*True*	*False*	*False*
False	*True*	*True*	*True*	*True*	*True*
True	*False*	*False*	*True*	*True*	*True*
True	*False*	*True*	*True*	*True*	*True*
True	*True*	*False*	*True*	*False*	*True*
True	*True*	*True*	*True*	*True*	*True*

Resolution for completeness of inference rules

- Inference rules can be used as successor functions in a search-based agent
- Any complete search algorithm, applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic.
- Refutation completeness:
  Resolution can always be used to either prove or refute a sentence
  $\longrightarrow$ Resolution algorithm on CNF
- Caveat:
  Resolution cannot be used to enumerate true sentences.
  Given A is true, resolution cannot generate $A \vee B$

Conjunctive Normal Form

- Resolution applies only to disjunctions of literals
- We can transform any sentence in PL in CNF
- A k-CNF has exactly k literals per clause:
  $(l_{1,1} \vee \cdots \vee l_{1,k}) \wedge \cdots \wedge (l_{n,1} \vee \cdots \vee l_{n,k})$

Conversion procedure:

- Eliminate $\Leftrightarrow$ using biconditional elimination
- Eliminate $\Rightarrow$ using implication elimination
- Move $\neg$ inwards using (repeatedly) double-negation elimination and de Morgan rules
- Apply distributivity law, distributing $\vee$ over $\wedge$ whenever possible

Finally, the KB can be used as input to a resolution procedure

Example of conversion to CNF

$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

- Eliminate $\Rightarrow$ using biconditional elimination
  $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$ using implication elimination
  $(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
- Move $\neg$ inwards using (repeatedly) double-negation elimination and de Morgan rules
  $(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
- Apply distributivity law, distributing $\vee$ over $\wedge$ whenever possible $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution algorithm

- KB $\models \alpha$ iff (KB $\wedge \neg\alpha$) is unsatisfiable

- (KB $\wedge \neg\alpha$) is converted to CNF
  then we apply resolution rule repeatedly, until:
  - no clause can be added (i.e., KB $\not\models \alpha$)
  - we derive the empty clause (i.e., KB $\models \alpha$)

<PL-Resolution(KB, α), Fig 7.13, page 228>

- Ground resolution theorem:
  If a set of clauses is unsatisfiable, then the resolution closure of those clauses contains the empty clause.

Example of applying resolution algorithm to wumpus world

With KB=$R_2 \wedge R_4$, prove that $\neg P_{1,2}$

$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$R_4 : \neg B_{1,1}$

See Figure 7.14 on page 229

Note that many steps are pointless and could be avoided, example:

$B_{1,1} \vee P_{2,1} \vee \neg B_{1,1} = True \vee P_{2,1} = True$

Restriction to Horn clauses: a subset of PL

- Disjunctive form: disjunction of literals of which at most one is positive
  Example: $\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n \vee Q$ where P_i and Q are non-negated atoms
- Implicative form: an implication whose premise is a conjunction of literals and whose conclusion is a single positive literal
  Example: $P_1 \wedge P_2 \wedge \ldots P_n \Rightarrow Q$, where P_i and Q are non-negated atoms

Significance of Horn clauses:

- Real-world KB's are easy to write in implicative form
- Inference can be done with Forward and Backward chaining (easy to trace and understand) PL-FC-ENTAILS?(KB, q)
- Deciding entailment with Horn clauses is linear time

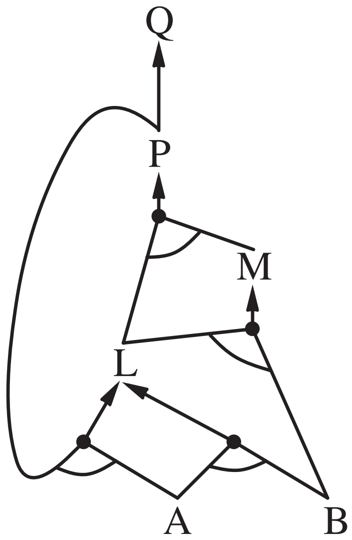
Forward chaining PL-FC-ENTAILS?(KB, q)

- determines whether p is entailed by a KB of Horn clauses
- begins with known facts
- asserts the conclusion (head) of an implication whose premises hold
- continues until the query q is added to KB (success) or no further inferences can be made (failure)

Forward chaining terminates (reaches a fixed point), is sound and complete

Execution example of Forward chaining

KB: $(P \Rightarrow Q) \wedge (L \wedge M \Rightarrow P) \wedge (B \wedge L \Rightarrow M) \wedge (A \wedge P \Rightarrow L) \wedge (A \wedge B \Rightarrow L) \wedge A \wedge B$



Inference in Horn theories

Runs in linear time

- Forward chaining:
  – data-driven
  – infers every possible conclusion
  – may do lots of work that is irrelevant to the goal
- Backward chaining:
  – goal-directed reasoning
  – works back from q, tries to find known facts that support the query
  – touches only relevant facts, often cost much less than linear in the size of KB

Problems with Propositional Logic

- good vehicle for introducing what logic / inference is
- too weak to even handle the Wumpus world
  'Don't go forward when Wumpus is in front of you' requires 64 rules (16 squares x 4 orientations for agent)
- Problem: generating rules, handling truth tables (2^n rows for n symbols)!
- Important problem: How to represent change when agent moves from [1,1] to [1,2]?
  Solution: Time-stamp symbols :-(
  But (1) length of game is not known in advance, (2) rewrite a time-dependent version of each rule..

Problem: PL only allows propositions, no relations, no objects

Summary

Logical agents apply inference to a knowledge base 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 can produce all entailed sentences

– Truth-table method is sound and complete for propositional logic
– Forward and backward chaining are linear-time, sound and complete for Horn clauses
– Resolution is complete for propositional logic

$\longrightarrow$ Propositional logic suffices for some of these tasks, not all