CS 316: Propositional logic

Stefan D. Bruda

Winter 2023

Agents that reason logically

- Two components: a knowledge base, and an inference engine
- Declarative approach to building an agent
  - We tell it what it needs to know, and
  - It can ask itself what to do; answers follow from the knowledge base
- Two views:
  - Knowledge level or what agents know
  - Implementation level or how the knowledge is actually organized (data structures and algorithms)

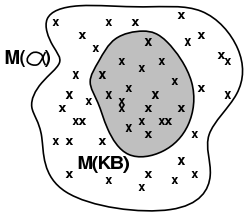
Knowledge-based agents

- A knowledge based agent must be able to
  - Represent states, actions, etc
  - Incorporate new knowledge (percepts)
  - Update internal representation of the world
  - Deduce unknown properties of the world
  - Deduce appropriate actions
- Our agents will use for all of these (formal) logic, i.e., formal languages for representing information such that conclusions can be drawn
  - Syntax defines the sentences in a language
  - Semantics defines the meaning of the sentences, i.e., the meaning of truth

Language	**Ontological Commitment** (what exists)	**Epistemological Commitment** (states of knowledge)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, time	true/false/unknown
Probability theory	facts	degree of belief 0...1
Fuzzy logic	degree of truth	degree of belief 0...1

Entailment and models

- $KB \models \alpha$: Knowledge base KB entails sentence α iff α is true in all the worlds in which KB is true
  - KB containing "Reds won" and "Expos won" entails "Either Reds or Expos won"
- m is a model of sentence α if α is true in m; $M(\alpha)$ is the set of all models of α
  - Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$



Example: $KB =$ "Reds won" and "Expos won"; $\alpha =$ "Expos won"

Inference

- $KB \vdash_i \alpha$: sentence α can be derived (inferred) from KB by procedure i
  - Soundness: i is sound if, whenever $KB \vdash_i \alpha$ it is also true that $KB \models \alpha$
  - Completeness: i is complete if, whenever $KB \models \alpha$ it is also true that $KB \vdash_i \alpha$
- We will eventually see a logic which is
  - Expressive enough (we can say almost anything of interest using it), and
  - For which there exists a complete and sound inference procedure

Propositional logic

- The simplest logic, illustrates basic ideas
- Syntax:
  - An elementary proposition is a symbol (abstract sense)
  - If S is a sentence, $\neg S$ (`\+S`) is a sentence
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ (`S1, S2`) is a sentence
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ (`S1; S2`) is a sentence
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ (`S2 :- S1`) is a sentence
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence
  - Nothing else is a sentence
- Semantics:
  - A model specifies the true/false value for each propositional symbol
  - There are rules to specify truth values of compound propositions with respect to a model m:

$\neg S$	is true iff	S	is false		
$S_1 \wedge S_2$	is true iff both	S_1	and	S_2	are true
$S_1 \vee S_2$	is true iff	S_1	is true or	S_2	is true
$S_1 \Rightarrow S_2$	is true iff	S_1	is false or	S_2	is true
i.e.,	is false iff	S_1	is true and	S_2	is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true and	$S_2 \Rightarrow S_1$	is true

Inference by enumeration

- 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
False	*False*	*False*	*False*	*True*	*False*	*False*
False	*False*	*True*	*True*	*False*	*False*	*False*
False	*True*	*False*	*False*	*True*	*False*	*True*
False	*True*	*True*	*True*	*True*	*True*	*True*
True	*False*	*False*	*True*	*True*	*True*	*True*
True	*False*	*True*	*True*	*False*	*False*	*True*
True	*True*	*False*	*True*	*True*	*True*	*True*
True	*True*	*True*	*True*	*True*	*True*	*True*

- Other inference procedures use syntactic operations on sentences, expressed in standardized forms

Normal forms

- Literal: a propositional symbol, or a propositional symbol negated
- De Morgan rules: For any propositions S_1 and S_2, $\neg(S_1 \wedge S_2)$ is logically equivalent to $\neg S_1 \vee \neg S_2$, and $\neg(S_1 \vee S_2)$ is equivalent to $\neg S_1 \wedge \neg S_2$
  - In addition, $\neg(\neg S_1)$ is equivalent to S_1, and both $\wedge$ and $\vee$ are associative and distributive with respect to each other
- Disjunctive normal form (DNF) (universal): Disjunction of conjunctions of literals
  - E.g., $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$
- Conjunctive normal form (CNF), or clausal form (universal): conjunction of $\underbrace{\text{disjunctions of literals}}_{\text{clauses}}$
  - E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D) \wedge (E \vee \neg D)$
- Horn Form (restricted): conjunction of Horn clauses, i.e., clauses with at most one positive (non-negated) literal. Also written as a conjunction of implications:

  `A :- B.` [and] `B :- C, D.` [and] `E :- D.`

Clausal form

Any sentence (or KB) can be transformed into a set of clauses (clausal form)

$$\neg((a \Leftrightarrow b) \vee (c \Rightarrow \neg(d \wedge (f \Rightarrow e))))$$

1. Eliminate $\Leftrightarrow$ and $\Rightarrow$: $\alpha \Rightarrow \beta$ is changed to $\neg\alpha \vee \beta$, and $\alpha \Leftrightarrow \beta$ is equivalent to $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

$$\neg(((\neg a \vee b) \wedge (\neg b \vee a)) \vee (\neg c \vee (\neg(d \wedge (\neg f \vee e)))))$$

2. Apply De Morgan rules to move all the negations in, and remove double negations

$$\neg((\neg a \vee b) \wedge (\neg b \vee a)) \wedge \neg(\neg c \vee (\neg(d \wedge (\neg f \vee e))))$$
$$(\neg(\neg a \vee b) \vee \neg(\neg b \vee a)) \wedge (\neg\neg c \wedge (\neg\neg(d \wedge (\neg f \vee e))))$$
$$((a \wedge \neg b) \vee (b \wedge \neg a)) \wedge (c \wedge (d \wedge (\neg f \vee e)))$$

3. Use the distributedness, associativity and commutativity to move the $\wedge$'s out: $\alpha \vee (\beta \wedge \gamma)$ becomes $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

$$((a \vee (b \wedge \neg a)) \wedge (\neg b \vee (b \wedge \neg a))) \wedge c \wedge d \wedge (\neg f \vee e)$$
$$(a \vee b) \wedge (a \vee \neg a) \wedge (\neg b \vee b) \wedge (\neg b \vee \neg a) \wedge c \wedge d \wedge (\neg f \vee e)$$
$$(a \vee b) \wedge (\neg b \vee \neg a) \wedge c \wedge d \wedge (\neg f \vee e)$$

4. Clausal form is more conveniently represented as a set of clauses:

$$\{ (a \vee b), (\neg b \vee \neg a), c, d, (\neg f \vee e)\}$$

Validity and Satisfiability

- A sentence is valid (or a tautology) if it is true in all models, e.g.,

$$A \vee \neg A \qquad A \Rightarrow A \qquad (A \wedge (A \Rightarrow B)) \Rightarrow B$$

- Validity is connected to inference via the Deduction Theorem:

$$KB \vdash \alpha \text{ iff } (KB \Rightarrow \alpha) \text{ is valid}$$

- A sentence is satisfiable if it is true in some model

$$A \vee B \qquad C$$

- A sentence is unsatisfiable if it is true in no models (or false in all models)

$$A \wedge \neg A$$

- Satisfiability is connected to inference via the following: $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable
  - I.e., prove α by reductio ad absurdum

Proof methods

- Model checking
  - Truth table enumeration (sound and complete for propositional)
  - Heuristic search in model space (sound but incomplete)
- Application of inference rules
  - Sound (legitimate) generation of new sentences from old
  - Proof = a sequence of inference rule applications
  - Can use inference rules as operators in a standard search algorithm
  - Inference rules:

Resolution

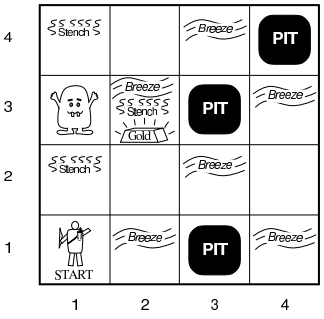
$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Modus ponens

$$\frac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

    - Resolution is complete for propositional logic
    - Modus ponens is only applicable to Horn clauses and is complete for Horn KBs
  - Inference rules can be used with forward chaining or backward chaining

The Wumpus world



- Percepts: Breeze, Glitter, Smell
- Actions: Left turn, Right turn, Forward, Grab, Release, Shoot
- Goals: Get gold back to start without entering pit or wumpus square
- Environment:
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills the wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up the gold if in the same square
  - Releasing drops the gold in the same square

The Wumpus world (cont'd)

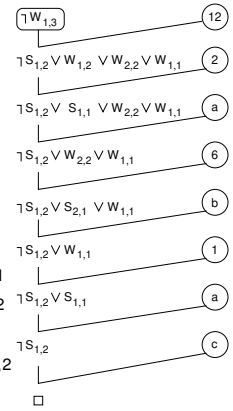
- Propositions: $S_{x,y}$, $B_{x,y}$, $W_{x,y}$, with $1 \leq x, y \leq 4$
- Knowledge base:
  - Facts:
    (a) $\neg S_{1,1}$ (b) $\neg S_{2,1}$ (c) $S_{1,2}$
    (d) $\neg B_{1,1}$ (e) $\neg B_{1,2}$ (f) $B_{2,1}$
  - Rules:
    $\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
    $\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$
    $\neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$
    $S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
  - Clausal form:
    (1) $S_{1,1} \vee \neg W_{1,1}$ (2) $S_{1,1} \vee \neg W_{1,2}$ (3) $S_{1,1} \vee \neg W_{2,1}$
    (4) $S_{2,1} \vee \neg W_{1,1}$ (5) $S_{2,1} \vee \neg W_{2,1}$ (6) $S_{2,1} \vee \neg W_{2,2}$
    (7) $S_{2,1} \vee \neg W_{3,1}$
    (8) $S_{1,2} \vee \neg W_{1,1}$ (9) $S_{1,2} \vee \neg W_{1,2}$ (10) $S_{1,2} \vee \neg W_{2,2}$
    (11) $S_{1,2} \vee \neg W_{1,3}$
    (12) $\neg S_{1,2} \vee W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
    (13) $\neg W_{1,3}$

Resolution:

$$\frac{\alpha \vee \beta, \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

The Wumpus world (cont'd)

- Propositions: $S_{x,y}$, $B_{x,y}$, $W_{x,y}$, with $1 \leq x, y \leq 4$
- Knowledge base:
  - Facts:
    (a) $\neg S_{1,1}$ (b) $\neg S_{2,1}$ (c) $S_{1,2}$
    (d) $\neg B_{1,1}$ (e) $\neg B_{1,2}$ (f) $B_{2,1}$
  - Rules:
    $\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
    $\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$
    $\neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$
    $S_{1,2} \Rightarrow W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
  - Clausal form:
    (1) $S_{1,1} \vee \neg W_{1,1}$ (2) $S_{1,1} \vee \neg W_{1,2}$ (3) $S_{1,1} \vee \neg W_{2,1}$
    (4) $S_{2,1} \vee \neg W_{1,1}$ (5) $S_{2,1} \vee \neg W_{2,1}$ (6) $S_{2,1} \vee \neg W_{2,2}$
    (7) $S_{2,1} \vee \neg W_{3,1}$
    (8) $S_{1,2} \vee \neg W_{1,1}$ (9) $S_{1,2} \vee \neg W_{1,2}$ (10) $S_{1,2} \vee \neg W_{2,2}$
    (11) $S_{1,2} \vee \neg W_{1,3}$
    (12) $\neg S_{1,2} \vee W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
    (13) $\neg W_{1,3}$



Control strategies for resolution

- Resolution itself is sound and complete (in fact, refutation-complete)
- Typically at each step there are many pairs of parent clauses that could be resolved. A control strategy is a policy for prioritizing which resolutions to perform next
- A control strategy is complete if its use preserves (refutation-)completeness
  I.e., if a contradiction exists then it can be found while respecting the strategy
  - Input resolution: at least one parent of each resolution step must be in the original KB, or part of the negated goal
    - Very efficient, but also incomplete (but complete for Horn clauses)
  - Unit resolution: at least one parent of each resolution step must be a unit clause, i.e., a single literal
    - The conclusion is always shorter than its parent, hence it is guaranteed to finish in bounded time
    - It is however incomplete (e.g., on $\{ P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q \}$)
  - Heuristic: For example, unit preference, the heuristic of prioritizing resolutions where one parent is a unit clause

Inference with Horn clauses

- In practice the full power of resolution is not needed
- Real-world knowledge bases usually contains only Horn clauses
  - Convenient because a Horn clause has the form

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow C$$

    which illustrates logical implication (if all the body is true then the head C becomes true)
- Modus ponens ("mode that affirms by affirming") is then used as inference rule
- Can be used with various control algorithms:
  - Forward chaining or data driven
  - Backward chaining or goal driven

AND/OR GRAPHS

```
a.
b.
l :- a, b.
l :- a, p.
m :- b, l.
p :- l, m.
q :- p.
```