Introduction to Artificial Intelligence
COMP 3501 / COMP 4704-4
Lecture 7

Prof. Nathan Sturtevant
JGH 318

Class Overview

- Review from Wednesday
- Inference in propositional logic
- Propositional logic agents
- First-Order Logic (Ch 8)

Entailment

- α entails ß *or* α ⊨ ß
  - ß follows logically from α
  - In every model in which α is true, ß is also true
    - M(α) ⊆ M(ß)

Entailment examples?

Propositional Logic Syntax

- *Sentence* → *AtomicSentence* | *ComplexSentence*
- *AtomicSentence* → *True* | *False* | *P* | *Q* | *R* | …
- *Complex Sentence* → (*Sentence*) | [*Sentence*]

  | ¬ *Sentence* | *Sentence* ∧ *Sentence*

  | *Sentence* ∨ *Sentence* | *Sentence* ⇒ *Sentence*

  | *Sentence* ⇔ *Sentence*
- Operator precedence: ¬, ∧, ∨, ⇒, ⇔

Example statements

- There is no pit in [1, 1]
- A square is breezy iff there is a pit in a neighboring square
- If there is no smell in [1, 1], there can't be a wumpus in [1, 2]

Model checking

- How does it work?
- What is the running time?
- What is the space required?

Theorem proving [7.5]

- No longer consult models
  - Derive inferences (entailment) directly from KB
- In some ways this mimics algebraic theorem proving
  - Start with the known
  - Apply rules/transformations
  - Reach the desired result (if possible)

Logical equivalence

- Two statements are logically equivalent if they are true in the same set of models
  - $\alpha \equiv \beta$
  - $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

Standard logical equivalences

- $(\alpha \land \beta) \equiv (\beta \land \alpha)$
- $(\alpha \lor \beta) \equiv (\beta \lor \alpha)$
- $((\alpha \land \beta) \land \gamma) \equiv (a \land (\beta \land \gamma))$
- $((\alpha \lor \beta) \lor \gamma) \equiv (a \lor (\beta \lor \gamma))$
- $\neg (\neg \alpha) \equiv \alpha$

Standard logical equivalences

- $(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$
- $(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta)$
- $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$
- $\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)$
- $\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)$

Standard logical equivalences

- $(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))$
- $(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$

Validity

- A sentence is valid if it is true in *all* models
  - $P \vee \neg P$
  - $Q \Rightarrow Q$
- Valid sentences are tautologies
- *Deduction* theorem
  - For any sentences α and ß, $\alpha \vDash \beta$ iff $(\alpha \Rightarrow \beta)$ is valid
  - Essence of model checking algorithm

Satisfiability

- A sentence is satisfiable if it is true in *some* model
  - Abbreviated as SAT
  - Can we find a variable assignment that makes some statement true

Validity and Satisfiability

- α is satisfiable iff ¬α is not valid
- $\alpha \vDash \beta$ iff $(\alpha \wedge \neg\beta)$ is unsatisfiable
  - Proof? [Hint: $\alpha \vDash \beta$ iff $(a \Rightarrow \beta)$ is valid]
- This is the logical basis of proof by contradiction

Validity and Satisfiability (Proof)

- α is satisfiable iff ¬α is not valid
  - if α is unsatisfiable, ¬α is valid
  - if ¬α is unsatisfiable, α is valid
- $\alpha \vDash \beta$ iff $(\alpha \wedge \neg\beta)$ is unsatisfiable
  - $\alpha \vDash \beta$ iff $(\alpha \Rightarrow \beta)$ is valid
  - $\alpha \vDash \beta$ iff $\neg(\alpha \Rightarrow \beta)$ is unsatisfiable
  - $\alpha \vDash \beta$ iff $\neg(\neg\alpha \vee \beta)$ is unsatisfiable
  - $\alpha \vDash \beta$ iff $(\alpha \wedge \neg\beta)$ is unsatisfiable

Inference & Proofs

- New notation for inference rules

$$\frac{given1, \quad given2}{conclusion}$$

- We supply the items on the top and conclude what is on the bottom

Modus Ponens

- Latin for *mode that affirms*

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

And-Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Biconditional elimination

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

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

Wumpus World Example

- Previous definitions

Search

- We can formulate theorem proving as a search problem
  - Initial state: KB
  - Actions: all inference rules that apply (top of rule)
  - Result: inference in bottom of rule added to KB
  - Goal: sentence we want to prove

Monotonicity

- The set of entailed sentences can only *increase* as information is added to the KB
  - if KB $\models \alpha$ then KB $\land \beta \models \alpha$
  - Adding ß to our KB will not decrease what we can entail from the KB

Inference: sound & complete

- The previous inference rules were all sound
  - Derive entailed sentences
- Are they complete? No
  - There are some things they can't derive
  - (Example?)

Unit Resolution

$$\frac{\ell_1 \vee \ell_2, \qquad \neg \ell_2}{\ell_1}$$

- Can be generalized to more clauses (see book)

Resolution

- Generalized resolution can handle more clauses

$$\frac{\ell_1 \vee \ell_2, \qquad \neg \ell_2 \vee \ell_3}{\ell_1 \vee \ell_3}$$

- Completely general form in the book

Examples

Conjunctive Normal Form (CNF)

- Resolution only applies to clauses with disjunction ($\vee$)
  - All propositional logic can be reduce to clauses or conjunctive normal form (CNF)

Example

- $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- $B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1}) \wedge (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
- $(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
- $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge \underline{(\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})}$
  - $(\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}$
  - $((B_{1,1} \vee \neg P_{1,2}) \wedge (B_{1,1} \vee \neg P_{2,1}))$
- $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (B_{1,1} \vee \neg P_{1,2}) \wedge (B_{1,1} \vee \neg P_{2,1})$

Using resolution

- Proofs using resolution are proofs by contradiction
  - $\alpha \models \beta$ iff $(\alpha \wedge \neg\beta)$ is unsatisfiable
- Assume we want to prove $\alpha \models \beta$
  - Add $\neg \beta$ to KB
  - If we can infer *false*, we have a contradiction
  - If we can't, then $\alpha \not\models \beta$

Example

- R1: $dog_{fred} \Rightarrow likesbones_{fred}$; R2: dog_{fred}
- R3: $\neg dog_{fred} \vee likesbones_{fred}$
- Prove: $likesbones_{fred}$
- Add R4: $\neg likesbones_{fred}$ to KB
- Resolve R4 and R3: R5: $\neg dog_{fred}$
- Resolve R5 and R2: (null)
  - Contradiction!

Special Case: Horn & definite Clauses

- A Horn clause is a disjunction of literals of which *at most one* is positive
  - $\neg A \vee \neg B \vee C$
  - In Definite clause *exactly one* is positive
- Definite clauses correspond to implications
  - $A \wedge B \Rightarrow C$
- Modus Ponens is sound and complete with Horn clauses

Building Logic Agents

- Can we now build propositional logic agents?
  - There are a few important details!
- All percepts depend on the current time/location of the agent
  - Frame problem: need to reason about what does/does not change as time goes forward
  - This tremendously complicates writing proper logical descriptions of the world

Building Logic Agents

- Can now build an agent
  - Use A* to plan movement
  - Use logical inference to decide where to go
- Caveat: planning gets more expensive as more time passes, even if the agent just moves around the know part of the state space
- Harder to build an agent that generates a full plan

First-Order Logic: Motivation

- Returning to fred likes bones:
  - Expensive to have to specify if everyone likes bones
  - Works in wumpus world, but can be computationally infeasable
  - Cannot make statements like:
    - “All dogs like bones”

First-Order Logic

- Propositional logic only has variables
  - These are *true* or *false*
- First-order logic adds objects, functions and relations
- Also adds quantifiers:
  - $\exists$: There exists
  - $\forall$: For all

First-Order Logic Examples

- *Occupation(p, o); Boss(p1, p2); Customer(p1, p2)*
- *Emily; Doctor, Surgeon, Lawyer*

- Emily is either a surgeon or a lawyer.
- All surgeons are doctors.
- Emily has a boss who is a lawyer.
- Every surgeon has a lawyer.

Homework: 7.14

-or-

Show a problem that resolution can solve and other rules cannot. (2 HW's -- do not use the internet.)