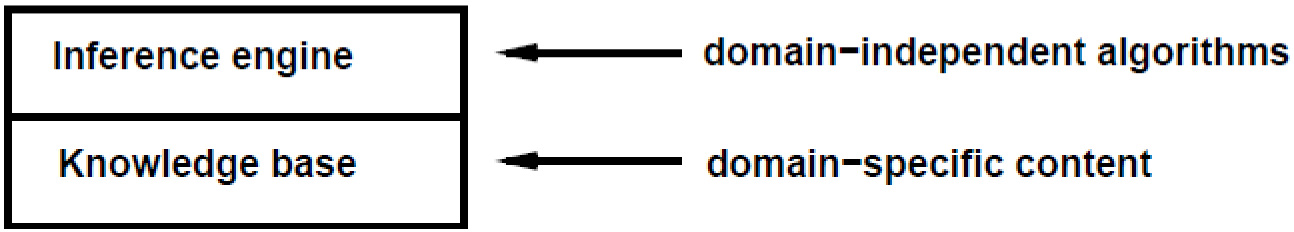
Artificial Intelligence

CS482, CS682, MW 1 – 2:15, SEM 201, MS 227
Prerequisites: 302, 365
Instructor: Sushil Louis, sushil@cse.unr.edu, http://www.cse.unr.edu/~sushil
Logic

Overview



- So we can use the domain independent inference engine to
  - Diagnose disease
  - Configure complex mainframes
  - Tech support
  - Wumpus world navigation
  - …
- The knowledge base is a set of sentences in a formal language that supports sound rules of inference

Logics are formal languages

- Syntax
  - Defines legal sentences in language
- Semantics
  - Defines the meaning of sentences – truth value
- Inference generates new sentences from KB
  - Entailment means that one thing **follows from** another
    - KB
      - Red sox won and
      - Cardinals won
      - Entails
      - Cardinals won
  - Models. m is a model of alpha if alpha is true in m’s world
  - M(alpha) set of all models of alpha

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Consequences of KB are a haystack; α is a needle.
Entailment = needle in haystack; inference = finding it

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 $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

Syntax and Semantics

Rules for evaluating truth with respect to a model m:

$\neg S$	is true iff	S	is false		
$S_1 \wedge S_2$	is true iff	S_1	is true **and**	S_2	is 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

Equivalence

Two sentences are logically equivalent iff true in same models:

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

Validity, Satisfiability

A sentence is valid if it is true in **all** models,

e.g., $True$, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

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

e.g., $A \lor B$, C

A sentence is unsatisfiable if it is true in **no** models

e.g., $A \land \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \land \neg\alpha)$ is unsatisfiable

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

- SAT was first problem to be proven NP-Complete

Proof methods

- Application of inference rules
  - Generate legitimate new sentences from old sentences using sound rules of inference
  - Proof = a sequence of rule applications
    - Search for this sequence using a search algorithm
  - Sentences need to be in **Normal Form** usually
  - If in Horn Clause form then searching is usually linear!!
- Model checking
  - Truth table enumeration
  - Use search with min-conflict h

Horn Form (restricted)

KB = **conjunction of Horn clauses**

Horn clause =

- ◊ proposition symbol; or
- ◊ (conjunction of symbols) $\Rightarrow$ symbol

E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

Modus Ponens

- Use Modus Ponens to prove something
- If there is an sentence of the form E1➔E2, and there is another sentence of the form E1, then E2 logically follows
- If E2 is the theorem you want to prove, you are done, otherwise add E2 to the list of sentences, because E2 will always be true when all the rest of the sentences are true.
  - Monotonicity
- Trivial Example:
  - R1: Feathers(Squigs) ➔ Bird(Squigs)
  - R2: Feathers(Squigs)
  - R3: Feathers(Derks)

  } KB
  - Then
  - Prove Bird(Squigs)
  - Apply Modus Ponens to R1 and R2

Resolution is a sound rule of inference

- Subsumes modus ponens
- If
  - E1 V E2
  - !E2 V E3
- Then
  - E1 V E3 logically follows

- Trivial Example 2
  - Feathers(Squigs)
  - Feathers(Squigs) ➔ Bird(Squigs)
- **Rewrite**
  - Feathers(Squigs)
  - !Feathers(Squigs) V Bird(Squigs)
- Resolve
  - E1 V E2
  - !E2 V E3

- What are E1, E2, E3?

Resolutions proof by refutation

- Assume that the negation of the theorem is T
- Show that the axioms and the assumed negation of the Theorem leads to a contradiction
- Conclude that the assumed negation of the theorem cannot be true because it leads to a contradiction
- Conclude that the Theorem must be true because the assumed negation of the theorem cannot be true
- Trivial Example
  - Feathers(squigs) ➔ Bird(squigs)
  - Feathers(squigs)

Resolution proof by refutation

- Remove → and rewrite
  - !Feathers(squigs) V Bird(squigs)
  - Feathers(squigs)
- Add negation of theorem to be proven
  1. !Bird(squigs)
  2. !Feathers(squigs) V Bird(squigs)
  3. Feathers(squigs)

- RESOLVE
  1. !Bird(squigs)
  2. !Feathers(squigs) V Bird(squigs)
  3. Feathers(squigs)
  4. Bird(squigs)
- Contradiction
  - **!Bird(squigs)**
  - **Bird(squigs)**

- Contradiction! Therefore…Nil,
- Therefore !Bird(squigs) must be false,
- Therefore Bird(squigs) must be true

Limits of PL

- Both proofs were examples of forward chaining in propositional logic
  - Resolution is sound and complete
- There is also backward chaining
- We will look at both in the context of expert systems, later…
- PL is painful. Why?
- Consider
  - We cannot express “pits cause breezes neighboring squares”
  - Instead:
    - B[1,1] ←→ P[1,2] V P[2,1]
    - B[1,2] ←→ ….
    - B[1,3] ←→ …
    - ….
    - ugh

The frame problem

- Effect axioms correspond to the transition model of Wworld
- L[1,1]0 /\ FacingEast0 /\ Forward0 ➔L[2,1]1 /\ !L[1,1]1
- If I am in L[1,1] at time 0 and facing east at time 0 and I act to move Forward at time 0 then
- I will be in L[2,1] at time 1 and I will not be in L[1,1] at time 1
  - *Fluents* refers to aspects of the world that change
  - *Atemporal variables* do not need the superscript 0, 1, …
- Suppose now that I start and I move to L[2,1]
- If I Ask if I am in L[2,1] ➔ can prove it
- If I Ask do I have arrow in L[2,1] I cannot prove or disprove it
  - I need to represent everything that remains unchanged in KB as a result of the action Forward (or any other action sentence)
  - Ugh, I have to represent (have sentences) for **every** thing that changes ➔ this is the frame problem

PL

- Propositional logic is **declarative**: pieces of syntax correspond to facts
- Propositional logic allows partial/disjunctive/negated information
  (unlike most data structures and databases)
- Propositional logic is **compositional**:
  meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- Meaning in propositional logic is **context-independent**
  (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power
  (unlike natural language)
  E.g., cannot say "pits cause breezes in adjacent squares"
  except by writing one sentence for each square

First order logic

Whereas propositional logic assumes world contains **facts**,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: red, round, bogus, prime, multistoried . . .,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- Functions: father of, best friend, third inning of, one more than, end of . . .

Logics:

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

- For each logic (language)
  - What are the sound rules of inference?
  - Are they complete?
  - What is the complexity of finding proofs?

Syntax

Constants	$KingJohn,\ 2,\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Atomic sentence = $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term = $function(term_1, \ldots, term_n)$
or $constant$ or $variable$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$$

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$

$>(1,2) \vee \leq(1,2)$

$>(1,2) \wedge \neg>(1,2)$

Here's a(nother) vocabulary

- Objects + Variables == Terms
- Terms + Predicates == Atomic Formulas
- Atomic formulas + negation == Literals
- Literals + Connectives + quantifiers == wffs
- Well formed formulas (wffs)
- Sentences (all variables bound)
- A(x)[Feathers(x) V !Feathers(y)]
  - Y is not bound

Interpretation

- Objects in a world correspond to object symbols in logic
- Relations in a world correspond to predicates in logic

- Interpretation: Full accounting of the correspondence between objects and object symbols and between relations and predicates

Quantification

- Universal
  - A(x)[UNRStudent(x) ➔ Smart(x)]
  - If the above expression is true it implies that you get a true expression when you substitute any object for x inside the square brackets
  - Common Issue:
    - Typically ➔ is the main connective with A
    - A(x) [UNRStudent(x) /\ Smart(x)]
    - Everyone is at UNR and Everyone is Smart

Existential Quantification

- Existential
  - E(x) [UNLVStudent(x) /\ Smart(x)]
  - There exists at least one object substitutable for x inside the square brackets that makes the sentence true
  - Common issue
  - /\ is the main connective with E
  - Typically not ➔
  - E(x) [UNLVStudent(x) ➔ Smart(x)]
  - Is true if there is anyone not at UNLV

Quantifiers

Quantifier duality: each can be expressed using the other

$\forall x \;\; Likes(x, IceCream) \qquad \neg\exists x \;\; \neg Likes(x, IceCream)$

$\exists x \;\; Likes(x, Broccoli) \qquad \neg\forall x \;\; \neg Likes(x, Broccoli)$

$\exists x \;\; \forall y$ is **not** the same as $\forall y \;\; \exists x$

$\exists x \;\; \forall y \;\; Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y \;\; \exists x \;\; Loves(x, y)$
"Everyone in the world is loved by at least one person"

Marcus intuition for informal proofs

- Man(marcus)
- Pompein(marcus)
- Born(marcus, 40)
- A(x) [man(x) ➔mortal(x)]
- Erupted(Volcano, 79)
- A(x) [Pompein(x) ➔ Died(x, 79)]
- A(x) A(t1) A(t2) [mortal(x) & born(x, t1) & gt(t2 – t1, 150) ➔ Dead(x, t2)]
- Now = 2013

- Is Marcus alive?
  - That is, what is the truth of: !Alive(Marcus, Now) or
  - That is, what is the truth of: Dead(Marcus, Now)

Need a couple more assertions

1. Man(marcus)
2. Pompein(marcus)
3. Born(marcus, 40)
4. A(x) [man(x) → mortal(x)]
5. Erupted(Volcano, 79)
6. A(x) [Pompein(x) → died(x, 79)]
7. A(x) A(t1) A(t2) [mortal(x) & born(x, t1) & gt(t2 – t1, 150) → dead(x, t2)]
8. Now = 2013
9. A(x) A(t) [!dead(x, t) → alive(x, t)]
10. A(x)A(t) [alive(x, t) → !dead(x, t)]
11. A(x)A(t1) A(2)[died(x, t1) & gt(t2, t1) → dead(x, t2)]

Not a resolution proof

- We deduced that Marcus was not alive
- We used a variety of rules and bound variables to literals
- Search for rules and bindings
  - Guided by what we were trying to prove
  - Looking for sentences that involved Alive
- Ensure you understand the proof for Wumpus world that proves that there is no pit in [1,2] and no pit in [2,1]

- It would be far simpler for search to find proofs if we had a smaller branching factor for our search procedure
  - Use the single resolution rule in searching for proof

Resolutions proof by refutation

- Assume that the negation of the theorem (sentence you are trying to prove) is T
- Show that the sentences and the assumed negation of the Theorem leads to a contradiction
- Conclude that the assumed negation of the theorem cannot be true because it leads to a contradiction
- Conclude that the Theorem must be true because the assumed negation of the theorem cannot be true
- NOTE
  - Sentences must be in a specific form: “Clause form”
  - Once you put all your sentences in clause form, you cleverly keep applying the resolution rule until you get a contradiction (nil)