

Logical inference. First-order logic.

Chapters 7 & 8

Last time we saw that

- Logic is a **formal language** to represent knowledge, it has **syntax**, **semantics** and a **proof system** (inference)
- **Propositional logic (PL)** contains sentences made of *true*, *false*, propositional variables, and \neg (**negations**), \wedge (**conjunctions**), \vee (**disjunctions**), \Rightarrow (**conditionals**), \Leftrightarrow (**iff**s)
- A **knowledge base KB** is a set of sentences of a logic; it **entails** a proposition α (written $KB \models \alpha$) iff $(KB \Rightarrow \alpha)$ is a **valid** sentence of the logic (a **tautology**)
- Conclusions (new sentences to be added to the KB) can be inferred by applying a logical proof system
- An inference procedure is **sound** (truth-preserving) if it infers *only* sentences that the KB entails
- An inference procedure is **complete** if it infers *all* the sentences that the KB entails
- **Resolution** is a sound and complete inference procedure

Today

- Revisit resolution and how to use it
- Algorithm for effective propositional inference
 - DPLL
- How to make logical agents
 - KB+inference engine+planner
 - Logic circuits
- Limitations of propositional logic (PL)
- First-Order Logic (FOL)

Revisiting inference

- “Natural deduction” rules

$\frac{\alpha \rightarrow \beta \quad \alpha}{\beta}$	$\frac{\alpha \rightarrow \beta \quad \neg \beta}{\neg \alpha}$	$\frac{\alpha \quad \beta}{\alpha \wedge \beta}$	$\frac{\alpha \wedge \beta}{\alpha}$
modus ponens	modus tollens	and introduction	and elimination

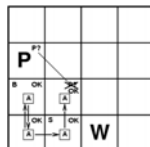
- Sound, but usually incomplete

Resolution: sound and complete

- A single sound and complete proof rule:

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

$$\frac{P_{1,3} \vee P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}$$



- Requires sentences in **conjunctive normal form (CNF)**
 - conjunctions of clauses (clauses = disjunctions)

Conversion to CNF

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

1. Eliminate \Leftrightarrow , replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
 $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate \Rightarrow , replacing $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$.
 $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg (P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. Move \neg inwards using de Morgan's rules and removing double-negation:
 $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4. Apply distributivity law (\wedge over \vee) and flatten:
 $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution: general case

$$\frac{l_1 \vee \dots \vee l_k}{m_1 \vee \dots \vee m_n}$$

$$l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n$$

for all l_i and m_j which are complementary literals

Sept 24, 2008

COMP 131, Lecture 7

7

Resolution soundness

l_i true: $l_i \Rightarrow \neg m_j$, so

$$l_i \Rightarrow m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n \text{ true}$$

or

$$l_i \text{ false: } \neg l_i \Rightarrow l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \text{ true}$$

We conclude that either one or the other is true, which is exactly what the resolution rule concludes. Resolution preserves truth.

Sept 24, 2008

COMP 131, Lecture 7

8

Resolution in use: refutation

- Resolution best applied to proof by refutation
- Take KB, assume $(KB \wedge \neg \alpha)$, apply resolution
- Either of two things can happen:
 - you infer the empty clause $() \Leftrightarrow (KB \wedge \neg \alpha)$ is unsatisfiable \Leftrightarrow no model exists where KB and $\neg \alpha$ are both true \Leftrightarrow KB entails α
 - you cannot apply resolution anymore but you don't have the empty clause $\Leftrightarrow (KB \wedge \neg \alpha)$ is satisfiable \Leftrightarrow KB does not entail α

either way, you can stop: completeness of resolution

Sept 24, 2008

COMP 131, Lecture 7

9

Example of inference by resolution

Given: $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

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

A		

Step 1: convert KB to CNF

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

Step 2: negate α

$$\neg \alpha = P_{1,2}$$

Sept 24, 2008

COMP 131, Lecture 7

10

Example of inference by resolution...

Step 3: apply resolution to $(KB \wedge \neg \alpha)$

rows are clauses

KB is conjunction of rows

Step	Formula	Derivation
1	$\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$	Given
2	$\neg P_{1,2} \vee B_{1,1}$	Given
3	$\neg P_{2,1} \vee B_{1,1}$	Given
4	$\neg B_{1,1}$	Given
5	$P_{1,2}$	Negated conclusion
6	$B_{1,1}$	Resolve 2 and 5
7	false (empty clause)	Resolve 4 and 6

Step 4: therefore, $KB \models \alpha$

Sept 24, 2008

COMP 131, Lecture 7

11

Automated resolution

- Given a KB and conclusion α in CNF
- Make a set of clauses: $Clauses = (KB \wedge \neg \alpha)$
- Make an empty set New for adding new clauses
- For each pair C_i, C_j in $Clauses$
 - $R := \text{Resolve } C_i \text{ and } C_j$
 - If $R = \text{empty}$ return **true** ($KB \models \alpha$)
 - Else add R into New
- If New is contained in $Clauses$ (there's nothing new in New) return **false** ($KB \not\models \alpha$)
- Else add New into $Clauses$
- Go to 4

PL-RESOLUTION p.216

Sept 24, 2008

COMP 131, Lecture 7

12

Resolution and satisfiability

- Resolution finds the empty clause if the set of sentences is unsatisfiable, stops otherwise
- Resolution complete = satisfiability decidable 😊
- Practical problems: e.g., find bugs in software automatically
 - propositional variables represent state of software
 - use logic to describe program and to assert that there is a bug
 - if the sentence is satisfiable, you've found a bug!

Sept 24, 2008

COMP 131, Lecture 7

13

Efficient propositional inference

- Satisfiability is NP-hard 😞
 - 3-SAT is the problem of deciding if a sentence in CNF with 3 literals in each clause is satisfiable or not
- Enumerating all clauses to resolve takes exponential time
- Need to combine resolution with efficient search
- DPLL: a backtracking (DFS-like) search algorithm (complete – will find a model if there is one)
- Based on model-checking (like generating truth tables), but more efficient

Sept 24, 2008

COMP 131, Lecture 7

14

DPLL improvements

- Early termination
 - a clause is *true* if just one literal is *true* (even if others are not assigned yet) $(A \vee B \vee \neg C)$
 - a sentence is *false* if just one clause is *false* $(A \vee B) \wedge \neg C$
- Pure symbol heuristic
 - a pure symbol always appears with the same "sign" $(A \vee B) \wedge (A \vee \neg C) \wedge (\neg B \vee \neg C)$
 - only have to check models s.t. pure symbols have their literals *true*
- Unit clause heuristic
 - a unit clause has just one literal, or just one yet-unassigned literal and others are *false* $(A \vee B \vee \neg C)$
 - assign all symbols in unit clauses first, before branching on others

model:
A = *false*
B = *false*

Sept 24, 2008

COMP 131, Lecture 7

15

DPLL recursive search

```
function DPLL-SATISFIABLE?(s) returns true or false
inputs: s, a sentence in propositional logic
clauses ← the set of clauses in the CNF representation of s
symbols ← a list of the proposition symbols in s
return DPLL(clauses, symbols, [])
```

```
function DPLL(clauses, symbols, model) returns true or false
if every clause in clauses is true in model then return true
if some clause in clauses is false in model then return false
P, value ← FIND-PURE-SYMBOL(symbols, clauses, model)
if P is non-null then return DPLL(clauses, symbols-P, [P = value|model])
P, value ← FIND-UNIT-CLAUSE(clauses, model)
if P is non-null then return DPLL(clauses, symbols-P, [P = value|model])
P ← FIRST(symbols); rest ← REST(symbols)
return DPLL(clauses, rest, [P = true|model]) or
DPLL(clauses, rest, [P = false|model])
```

s is $(KB \wedge \neg \alpha)$
wrapper function checks satisfiability calls DPLL search with unassigned model

early termination

pure-symbol heuristic

unit-clause heuristic

try assigning another symbol

Sept 24, 2008

COMP 131, Lecture 7

16

DPLL on Wumpus World 1

KB: rules + sensory info

Prove: no pit in 1,2

Clauses (related to conclusion):

- $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$
- $\neg P_{1,2} \vee B_{1,1}$
- $\neg P_{2,1} \vee B_{1,1}$
- $\neg B_{1,1}$ (from rules of game)
- $P_{1,2}$ (negated conclusion)

Symbols: $B_{1,1}, P_{1,2}, P_{2,1}$

Model: initially empty []

A			

Sept 24, 2008

COMP 131, Lecture 7

17

Implementation notes

- <http://code.google.com/p/aima-java/>
 - KnowledgeBase, Sentence, dpllSatisfiable(String)
- KB: set of clauses (list)
- Parser of logical clauses (string to data structure)
- To see if clause entailed by KB: ASK (with DPLL)
- To add clause to KB: TELL (append)
- To generate new clauses for ASKing: ??
 - inference doesn't generate conclusions
 - need a **plan**

Sept 24, 2008

COMP 131, Lecture 7

18

Full Wumpus World agent

- Input: percept, output: action
- KB: initially, “physics” of wumpus world, such as
 - $B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$ for every square $[x,y]$
 - $S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$ for every square $[x,y]$
 - $W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4}$ (there’s at least one wumpus)
 - there’s at most one wumpus: $n(n-1)/2$ statements
 - 155 sentences with 64 distinct symbols for a 4x4 world
- Maintains position, orientation, list of visited squares, previous action, **plan** (action sequence)
 - all that is not logic

Sept 24, 2008

COMP 131, Lecture 7

19

Full Wumpus World agent (cont.)

1. Update *position*, *orientation*, *visited* based on previous *action*
2. TELL KB about new sensory inputs
3. If *glitter* then *action* := *grab*
4. Else if *plan* is not empty, then *action* := POP(*plan*)
5. Else if any square $[i,j]$ adjacent to *visited*
 - is provably safe: $\text{ASK}(\text{KB}, (\neg P_{i,j} \wedge \neg W_{i,j}))$ is *true*
 - or if any square $[i,j]$ adjacent to *visited* is possibly safe: $\text{ASK}(\text{KB}, (P_{i,j} \vee W_{i,j}))$ is *false*
 then *plan* := A*-SEARCH(*position*, *orientation*, $[i,j]$, *visited*)
action := POP(*plan*)
6. Else *action* := randomly chosen move
7. Return *action*

Sept 24, 2008

COMP 131, Lecture 7

20

Keeping track of location and orientation

- It’s hard to add to the KB statements that depend on time
- $L_{11} \wedge \text{FacingRight} \wedge \text{MoveForward} \Rightarrow L_{12}$
 - doesn’t work: L_{11} and L_{12} can’t both be true
- $L_{11}^t \wedge \text{FacingRight}^t \wedge \text{MoveForward}^t \Rightarrow L_{12}^{t+1}$
 - superscript is time
- Proliferation of clauses!

Sept 24, 2008

COMP 131, Lecture 7

21

Logic circuit-based agents

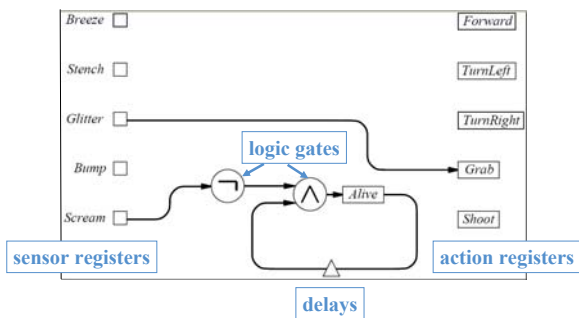
- Instead of the straightforward KB implementation
 - KB+inference engine+planner
- “Wire” agent directly to make the correct inferences and take the correct actions
 - **registers** hold **current values** of propositions (1=true, 0=false)
 - **delay wires** allow access previous values of propositions
 - registers are connected by **logic gates**

Sept 24, 2008

COMP 131, Lecture 7

22

Wumpus-solving circuit part 1



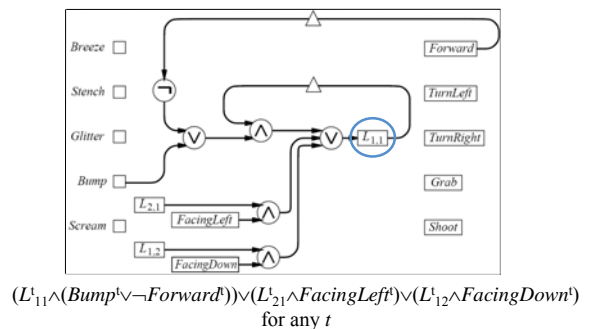
Grabbing gold; determining if wumpus is alive

Sept 24, 2008

COMP 131, Lecture 7

23

Wumpus-solving circuit, part 2



$(L_{11}^t \wedge (\text{Bump}^t \vee \neg \text{Forward}^t)) \vee (L_{21}^t \wedge \text{FacingLeft}^t) \vee (L_{12}^t \wedge \text{FacingDown}^t)$
for any t

Determining if agent is at [1,1]

Sept 24, 2008

COMP 131, Lecture 7

24

Big idea: no proofs required

- When you're attacked by a tiger, you may not have the time to make all the necessary inferences; you'd better start to run fast!
- In animals, a lot is done with (evolutionary) hard wiring or instincts, deliberation is a young add-on
- Limitations in practice:
 - hard to design for complex domains/tasks
 - hard to encode long-range dependencies

Sept 24, 2008

COMP 131, Lecture 7

25

Summary: PL inference

- **Resolution** is a single inference rule that is **sound** and **complete** and can be automated
- **DPLL** is backtracking search that proves satisfiability by **model checking** and uses early termination, and the **pure symbol** and **unit clause** heuristics
- Logic agents can be implemented as **KB+inference+planner** or as a **logic circuit**
- Propositional logic is not very expressive (proliferation of clauses)

Sept 24, 2008

COMP 131, Lecture 7

26

At this point

- You know everything needed for A2
- You've long noticed that propositional logic is very unwieldy

Sept 24, 2008

COMP 131, Lecture 7

27

All men are mortal

- But we can't say that with propositional logic, which only deals with fact statements (no variables)
- More things we can't say in PL
 - “Access to this website is restricted to anyone with a password or anyone from the list of special users.”
 - “When you sterilize a jar, all bacteria are dead.”
 - “You can fool some people some of the time, but you can't fool all people all of the time.”
 - “Any pit generates a sensation of breeze in all adjacent squares.”

Sept 24, 2008

COMP 131, Lecture 7

28

First-Order Logic (FOL)

- a.k.a First-Order Predicate Logic, First-Order Predicate Calculus
- **Variables** can stand for things/objects in the world
- **Quantifiers** allow us to talk about some or all of them
- **Predicates** specify relations between objects

Sept 24, 2008

COMP 131, Lecture 7

29

FOL syntax overview

- **Term**
 - **constant**: Japan, L₁₁, Bacterium23
 - **variable**: x, y, z
 - **function symbol** applied to one or more terms: f(x), g(f(x),y), mother-of(John),
- **Sentence**
 - a **predicate** applied to one or more terms: On(a,b), Sister(mother-of(John),Jane), Has-breeze(L₁₁)
 - term₁=term₂ (equality)
 - if v is a variable and φ is a sentence, then $\forall v.\phi$ and $\exists v.\phi$ are sentences
 - closure under sentence operators: (), ¬, ∧, ∨, ⇒, ⇔

Sept 24, 2008

COMP 131, Lecture 7

30

Example: family relations

- “John is Richard’s brother”

$Brother(Richard, John)$

- “One’s mother is one’s female parent”

$\forall m, c. Mother(c)=m \Leftrightarrow Female(m) \wedge Parent(m, c)$

- “Parent and child are inverse relations”

$\forall p, c. Parent(p, c) \Leftrightarrow Child(c, p)$

- “Everybody has a mother”

$\forall c. \exists m. Mother(c)=m$

Sept 24, 2008

COMP 131, Lecture 7

31

Quantifier tricks

$\forall x. At(x, TuftsUniversity) \Rightarrow Smart(x)$

“Everybody at Tufts is smart”

$\forall x. At(x, TuftsUniversity) \wedge Smart(x)$

“Everybody is at Tufts and everybody is smart”

$\exists x. At(x, TuftsUniversity) \wedge Smart(x)$

“Someone at Tufts is smart”

$\exists x. At(x, TuftsUniversity) \Rightarrow Smart(x)$

True if there’s anyone who’s not at Tufts!

Sept 24, 2008

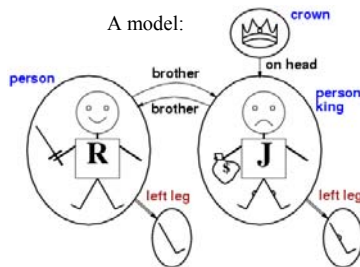
COMP 131, Lecture 7

32

FOL semantics: models

FOL sentences have truth values with respect to **models** and **interpretations**

Model =
= set of **objects** that
are connected by
relations
= **domain**
= **universe**



Sept 24, 2008

COMP 131, Lecture 7

33

FOL semantics: interpretations

Interpretations map symbols in logic to objects and relations in a model:

- map constants to objects that are elements of the model
- map predicates to relations in the model
- map functions to functions in the model
 - a **function** is a binary relation defined for every element of the model, which returns an object that is always the same for the same element (like a mathematical function)
 - e.g., $Mother(x)$ assuming no adoptions, or $Sqrt(x)$

Sept 24, 2008

COMP 131, Lecture 7

34

FOL semantics: truth

- A sentence $predicate(term_1, \dots, term_n)$ is true in a model m and under an interpretation i iff
 - $term_1, \dots, term_n$ refer to objects of m under i
 - $predicate()$ refers to a relation on m under i
 - the objects referred to by $term_1, \dots, term_n$ are in fact in the relation referred to by $predicate()$ in m

Sept 24, 2008

COMP 131, Lecture 7

35

Quantifier truth

- $\exists x. Predicate(x)$ is true in m under i iff
 - there is a relation on m that $Predicate()$ refers to under i , and
 - there is at least one object in m for which that relation is true
- $\forall x. Predicate(x)$ is true in m under i iff
 - there is a relation on m that $Predicate()$ refers to under i , and
 - for all objects in m that relation is true

Sept 24, 2008

COMP 131, Lecture 7

36

Interpretation example

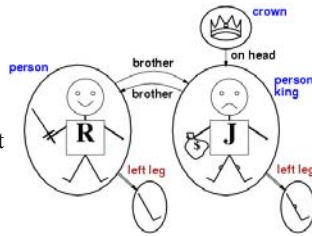
Richard refers to R

John refers to J

Brother refers to
brotherhood

Crown refers to the object
that looks like a crown

OnHead refers to the
relation between
something and someone,
where something is on
someone's head,...



Intended interpretation
But there are many other possible ones

Sept 24, 2008

COMP 131, Lecture 7

37

Model-checking impossible

- Entailment in PL is computed by enumeration and checking of possible models
- We *could* do that in FOL for a given KB vocabulary

for each number of model elements n from 1 to infinity
for each k -ary predicate P_k in the vocabulary
for each possible k -ary relation on n objects
for each constant term C in the vocabulary
for each possible choice of C referent among n objects...

Will take a long time to check truth in all combinations

Sept 24, 2008

COMP 131, Lecture 7

38

Interpretation of equality

- $term_1 = term_2$ is true in an interpretation iff both $term_1$ and $term_2$ denote (refer to) the same object

- definition of *Sibling* in terms of *Parent*:

$$\forall x, y . \text{Sibling}(x, y) \leftrightarrow$$

$$[\neg(x = y) \wedge \exists m, f . \neg(m = f) \wedge \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$$

Sept 24, 2008

COMP 131, Lecture 7

39

Summary: FOL

- Syntax includes **terms** (constants, variables, function symbols) and **sentences** (predicates, maybe with connectives and maybe **quantified**)
- Semantics (truth) depends on both the **model** and the **interpretation**
- We cannot use anything like **model-checking**

Sept 24, 2008

COMP 131, Lecture 7

40

Next time

- Inference (proof) in FOL systems
- Knowledge representation using FOL
 - Ontologies
- A2 due on Monday Sept 29th
- Graded A1 and answers will be available

Sept 24, 2008

COMP 131, Lecture 7

41