Inference & Logical Agents

Based on slides from AIMA, Marie desJardin, Charles R. Dyer, Richard Lathrop

1

Bookkeeping

- Last time
  - Finished propositional logic
  - Introduction to first-order logic
  - A little about higher-order logics
  - Exercise: English-to-FOL translation
- Today's class
  - More on knowledge-based agents
    - Situations – reasoning over time
  - A little more translation to/from FOL
  - Inference in knowledge bases, 5 ways

2

A Note on Common Sense Reasoning – example adapted from Lenat

- You are told: John drove to the grocery store and bought a pound of noodles, a pound of ground beef, and two pounds of tomatoes.
  - Is John 3 years old?
  - Is John a child?
  - What will John do with the purchases?
  - Did John have any money?
  - Does John have less money after going to the store?
  - Did John buy at least two tomatoes?
  - Were the tomatoes made in the supermarket?
  - Did John buy any meat?
  - Is John a vegetarian?
  - Will the tomatoes fit in John's car?
- Can Propositional Logic support these inferences?

3

A Note on Common Sense Reasoning

- There are a number of inferences and conclusions that we can draw that depend on *background knowledge*
- We refer to this background knowledge as "common sense"
- Can be represented as a set of statements in a knowledge base
  - Only adults can drive
  - Tomatoes weigh 4-8 ounces
  - Purchasing involves spending money
  - When you spend money, you no longer have it [etc.]
- Given these statements, we can **infer** the answers to the questions

4

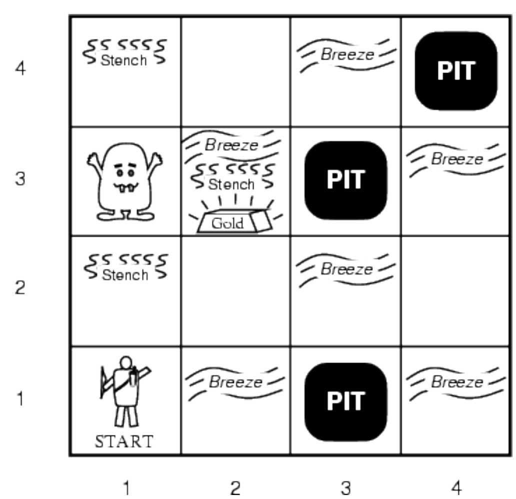
Logical Agents for Wumpus World

- Three (non-exclusive) agent architectures:
  - **Reflex** agents
    - Have rules that classify situations, specifying how to react to each possible situation
  - **Model-based** agents
    - Construct an internal model of their world
  - **Goal-based** agents
    - Form goals and try to achieve them

5

A Typical Wumpus World

- The agent always starts in the field [1,1].
- The task of the agent is to find the gold, return to the field [1,1] and climb out of the cave.



6

A Simple Reflex Agent

- Rules to **map percepts into observations**:
  - $\forall$b,g,u,c,l Percept([Stench, b, g, u, c], l) ⇒ Smelly(l)
  - $\forall$s,g,u,c,l Percept([s, Breeze, g, u, c], l) ⇒ Breezy(l)
  - $\forall$s,b,u,c,l Percept([s, b, Glitter, u, c], l) ⇒ AtGold(l)
- Rules to **select an action given observations**:
  - $\forall$l AtGold(l) ⇒ Action(Grab, l)

7

A Simple Reflex Agent

- Some difficulties:
- Climb?
  - There is no percept that indicates the agent should climb out – **position and holding gold are not part of the percept sequence**
- Loops?
  - The percept will be repeated when you return to a square, which should cause the same response (unless we maintain some **internal model of the world**)

8

KB-Agents Summary

Wumpus percepts:
[Stench, Breeze, Glitter, Bump, Scream]

- Logical agents
  - Reflex: rules map directly from percepts → beliefs or percepts → actions

    $\forall$b,g,u,c,t Percept([Stench, b, g, u, c],) $\Rightarrow$ Smelly(l)

    $\forall$t AtGold(l) $\Rightarrow$ Action(Grab, l)
  - Model-based: construct a *model* (set of t/f beliefs about sentences) as they learn; map from models → actions

    Action(Grab, l) $\Rightarrow$ HaveGold(l)

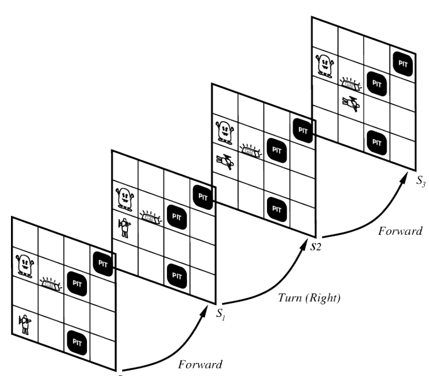
    HaveGold(l) $\Rightarrow$ Action(RetraceSteps, l)
  - Goal-based: form goals, then try to accomplish them
  - Encoded as a rule:

    ($\forall$s) Holding(Gold,s) $\Rightarrow$ GoalLocation([1,1],s)

9

Representing Change

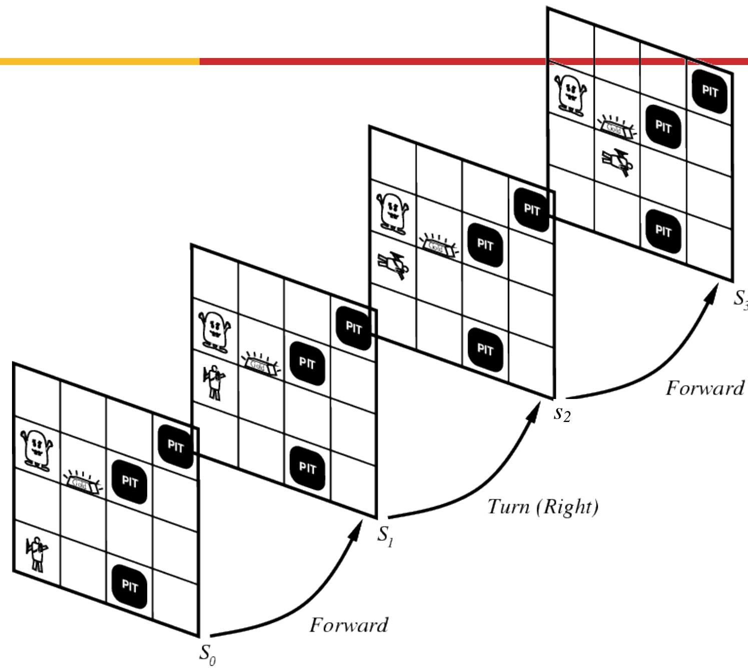
- Representing change in the world in logic can be tricky.
- One way is just to change the KB
  - Add and delete sentences from the KB to reflect changes
  - How do we remember the past, or reason about changes?
- **Situation calculus** is another way
- A **situation** is a snapshot of the world at some instant in time
- When the agent performs an action A in situation S1, the result is a new situation S2.



10

Situations

- Situations over time.
  - (We would not have this level of full knowledge.)



11

Situation Calculus

- A **situation** is:
  - A snapshot of the world
  - At an interval of time
  - During which nothing changes
- Every true or false statement is made wrt. a situation
  - Add **situation variables** to every predicate.
  - *at(Agent,1,1)* becomes *at(Agent,1,1,s0)*:
    at(Agent,1,1) **is true in** situation (i.e., state) s0.

12

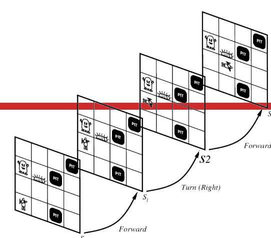
Situation Calculus

- Alternatively, add a special 2nd-order predicate, **holds(f,s),** that means "f is true in situation s." E.g., holds(at(Agent,1,1),s0)
- Or: add a new function, **result(a,s),** that maps a situation s into a new situation as a result of performing action a. For example, result(forward, s) is a function that returns the successor state (situation) to s
- Example: The action *agent-walks-to-location-y* could be represented by

  $(\forall x)(\forall y)(\forall s)\ (at(Agent,x,s) \wedge \neg onbox(s)) \Rightarrow at(Agent,y,result(walk(y),s))$

13

Situations Summary



- Representing a dynamic world
  - Situations ($s_0 \ldots s_n$): the world in situation 0-n
    Teaching(DrM,s_0) — today, 1:00, whenNotSick, …
  - Add 'situation' argument to statements
    AtGold(t,s_0)
  - Or, add a 'holds' predicate that says 'sentence is true in this situation'
    holds(At[2,1], s_1)
  - Or, add a result(action, situation) function that takes an action and situation, and returns a new situation
    results(Action(goNorth), s_0) → s_1

14

Deducing Hidden Properties

- From the perceptual information we obtain in situations, we can **infer properties of locations**

l = location, s = situation

∀l,s at(Agent,l,s) ∧ Breeze(s) ⇒ Breezy(l)

∀l,s at(Agent,l,s) ∧ Stench(s) ⇒ Smelly(l)

- Neither Breezy nor Smelly need situation arguments because pits and Wumpuses do not move around

15

Deducing Hidden Properties II

- We need to write some rules that relate various aspects of a single world state (as opposed to across states)
- There are two main kinds of such rules:

16

Deducing Hidden Properties II

- We need to write some rules that relate various aspects of a single world state (as opposed to across states)
- There are two main kinds of such rules:
  - **Causal rules** reflect assumed direction of causality:
    $(\forall l1,l2,s)$ At(Wumpus,l1,s) $\wedge$ Adjacent(l1,l2) $\Rightarrow$ Smelly(l2)
    $(\forall l1,l2,s)$ At(Pit,l1,s) $\wedge$ Adjacent(l1,l2) $\Rightarrow$ Breezy(l2)
- Systems that reason with causal rules are called **model-based reasoning** systems

17

Deducing Hidden Properties II

- We need to write some rules that relate various aspects of a single world state (as opposed to across states)
- There are two main kinds of such rules:
  - **Diagnostic rules** infer the presence of **hidden properties** directly from the percept-derived information. We have already seen two:
    $(\forall l,s)$ At(Agent,l,s) $\wedge$ Breeze(s) $\Rightarrow$ Breezy(l)
    $(\forall l,s)$ At(Agent,l,s) $\wedge$ Stench(s) $\Rightarrow$ Smelly(l)

18

Frames: A Data Structure

- A frame divides knowledge into substructures by representing "stereotypical situations."
- Situations can be visual scenes, structures of physical objects, …
- Useful for representing commonsense knowledge.

Slots	Fillers
publisher	Thomson
title	Expert Systems
author	Giarratano
edition	Third
year	1998
pages	600

Slot	Fillers
name	computer
specialization_of	a_kind_of machine
types	(desktop, laptop,mainframe,super) if-added: Procedure ADD_COMPUTER
speed	default: faster if-needed: Procedure FIND_SPEED
location	(home,office,mobile)
under_warranty	(yes, no)

intelligence.worldofcomputing.net/knowledge-representation/frames.html#.WCHhCNxBo8A

19

Representing Change: The Frame Problem

- **Frame axioms**: If property x doesn't change as a result of applying action a in state s, then it stays the same.
  - On (x, z, s) $\wedge$ Clear (x, s) $\Rightarrow$
    On (x, table, Result(Move(x, table), s)) $\wedge$
    $\neg$On(x, z, Result (Move (x, table), s))
  - On (y, z, s) $\wedge$ y$\neq$ x $\Rightarrow$ On (y, z, Result (Move (x, table), s))
- The proliferation of frame axioms becomes very cumbersome in complex domains

20

The Frame Problem II

- **Successor-state axiom:** General statement that characterizes **every way** in which a particular predicate can become true:
  - Either it can be **made true**, or it can **already be true and not be changed**:
  - On (x, table, Result(a,s)) $\leftrightarrow$
    [On (x, z, s) $\wedge$ Clear (x, s) $\wedge$ a = Move(x, table)] v
    [On (x, table, s) $\wedge$ a $\neq$ Move (x, z)]
- In complex worlds with longer chains of action, even these are too cumbersome
  - Planning systems use special-purpose inference to reason about the expected state of the world at any point in time during a multi-step plan

21

Qualification Problem

- Qualification problem: How can you possibly characterize every single effect of an action, or every single exception that might occur?
- When I put my bread into the toaster, and push the button, it will become toasted after two minutes, unless…
  - The toaster is broken, or…
  - The power is out, or…
  - I blow a fuse, or…
  - A neutron bomb explodes nearby and fries all electrical components, or…
  - A meteor strikes the earth, and the world we know it ceases to exist, or…

22

Ramification Problem

- How do you describe every effect of every action?
  - When I put my bread into the toaster, and push the button, the bread will become toasted after two minutes, and…
    - The crumbs that fall off the bread onto the bottom of the toaster over tray will also become toasted, and…
    - Some of the aforementioned crumbs will become burnt, and…
    - The outside molecules of the bread will become “toasted,” and…
    - The inside molecules of the bread will remain more “breadlike,” and…
    - The toasting process will release a small amount of humidity into the air because of evaporation, and…
    - The heating elements will become a tiny fraction more likely to burn out the next time I use the toaster, and…
    - The electricity meter in the house will move up slightly, and…

23

Knowledge Engineering!

- Modeling the “right” conditions and the “right” effects at the “right” level of abstraction is very difficult
- Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is a field
- Many researchers hope that automated knowledge acquisition and machine learning tools can fill the gap:
  - Our intelligent systems should be able to learn about the conditions and effects, just like we do.
  - Our intelligent systems should be able to learn when to pay attention to, or reason about, certain aspects of processes, depending on the context.

24

Preferences Among Actions

- A problem with the Wumpus world knowledge base: It's hard to decide which action is best!
  - Ex: to decide between a forward and a grab, axioms describing when it is okay to move would have to mention glitter.
- This is not modular!
- We can solve this problem by **separating facts about actions from facts about goals.**
- This way our **agent can be reprogrammed just by asking it to achieve different goals.**

25

Preferences Among Actions

- The first step is to describe the desirability of actions independent of each other.
- In doing this we will use a simple scale: actions can be Great, Good, Medium, Risky, or Deadly.
- Obviously, the agent should always do the best action it can find:
  - $(\forall a,s)\ Great(a,s) \Rightarrow Action(a,s)$
  - $(\forall a,s)\ Good(a,s) \wedge \neg(\exists b)\ Great(b,s) \Rightarrow Action(a,s)$
  - $(\forall a,s)\ Medium(a,s) \wedge (\neg(\exists b)\ Great(b,s) \vee Good(b,s)) \Rightarrow Action(a,s)$

    ...

26

Preferences Among Actions

- We use this action quality scale in the following way.
- Until it finds the gold, the basic strategy for our agent is:
  - Great actions include picking up the gold when found and climbing out of the cave with the gold.
  - Good actions include moving to a square that’s OK and hasn't been visited yet.
  - Medium actions include moving to a square that is OK and has already been visited.
  - Risky actions include moving to a square that is not known to be deadly or OK.
  - Deadly actions are moving into a square that is known to have a pit or a Wumpus.

27

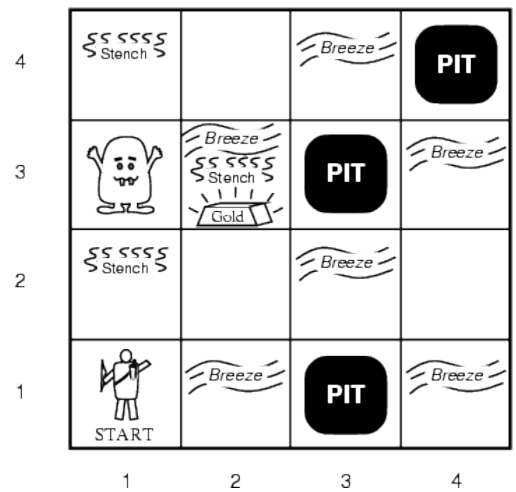
Goal-Based Agents

- Once the gold is found, it is necessary to change strategies. So now we need a new set of action values.
- We could encode this as a rule:
  - $(\forall s)$ Holding(Gold,s) $\Rightarrow$ GoalLocation([1,1]),s)
- We must now decide how the agent will work out a sequence of actions to accomplish the goal.
- Three possible approaches are:
  - **Inference**: good versus wasteful solutions
  - **Search**: make a problem with operators and set of states
  - **Planning**: coming soon!

28

An agent needs to make decisions!

- Where is there a pit?
- Where is there a wumpus?
- Should I fire my arrow?
- Where to explore next?
- Need to draw **conclusions** from knowledge in the knowledge base
- → Inference!



29

Logical Inference

Chapter 9

30

Review: English to FOL using quantifiers

3. There is somebody who is loved by everyone.
   $\exists x\ \forall y\ (loves(y,x))$ — this person also loves themselves
6. Frogs are green.
   $\forall x\ (frog(x) \Rightarrow green(x))$ — this is how we express rules
10. A mechanic likes Bob.
   $\exists x\ (mech.(x) \wedge likes(x, Bob))$ — express existence of something

- Usually:
  - Use AND with ∃ (so we don't have a false antecedent)
  - Use IMPLIES with ∀ (so we don't make too-strong claims)

31

Syntactic Ambiguity

- FOL provides many ways to represent the same thing.
- E.g., "Ball-5 is red."
  - **HasColor(Ball-5, Red):** Ball-5 and Red are objects related by HasColor.
  - **Red(Ball-5):** Red is a unary predicate applied to the Ball-5 object.
  - **HasProperty(Ball-5, Color, Red):** Ball-5, Color, and Red are objects related by HasProperty.
  - **ColorOf(Ball-5) = Red**: Ball-5 and Red are objects, and ColorOf() is a function.
  - **HasColor(Ball-5(), Red()):** Ball-5() and Red() are functions of zero arguments that both return an object, which objects are related by HasColor.
- This can GREATLY confuse a pattern-matching reasoner.

32

More choices to be made

- "For every food, there is a person who eats that food."
- [Use: Food(x), Person(y), Eats(y, x)]
  - $\forall x\ \exists y\ Food(x) \Rightarrow (Person(y) \wedge Eats(y, x))$
  - $\forall x\ Food(x) \Rightarrow \exists y\ (Person(y) \wedge Eats(y, x))$
  - $\forall x\ \exists y\ \neg Food(x) \vee (Person(y) \wedge Eats(y, x))$
  - $\forall x\ \exists y\ (\neg Food(x) \vee Person(y)) \wedge (\neg Food(x) \vee Eats(y, x))$
  - $\forall x\ \exists y\ (Food(x) \Rightarrow Person(y)) \wedge (Food(x) \Rightarrow Eats(y, x))$
- Common Mistakes:
  - $\forall x\ \exists y\ (Food(x) \wedge Person(y)) \Rightarrow Eats(y, x)$
  - $\forall x\ \exists y\ Food(x) \wedge Person(y) \wedge Eats(y, x)$

33

And yet more…

- "Every person eats some food."
- [Use: Person (x), Food (y), Eats(x, y)]
  - $\forall x\ \exists y\ Person(x) \Rightarrow [\ Food(y) \wedge Eats(x, y)\ ]$
  - $\forall x\ Person(x) \Rightarrow \exists y\ [\ Food(y) \wedge Eats(x, y)\ ]$
  - $\forall x\ \exists y\ \neg Person(x) \vee [\ Food(y) \wedge Eats(x, y)\ ]$
  - $\forall x\ \exists y\ [\ \neg Person(x) \vee Food(y)\ ] \wedge [\ \neg Person(x) \vee Eats(x, y)\ ]$
- Common Mistakes:
  - $\forall x\ \exists y\ [\ Person(x) \wedge Food(y)\ ] \Rightarrow Eats(x, y)$
  - $\forall x\ \exists y\ Person(x) \wedge Food(y) \wedge Eats(x, y)$

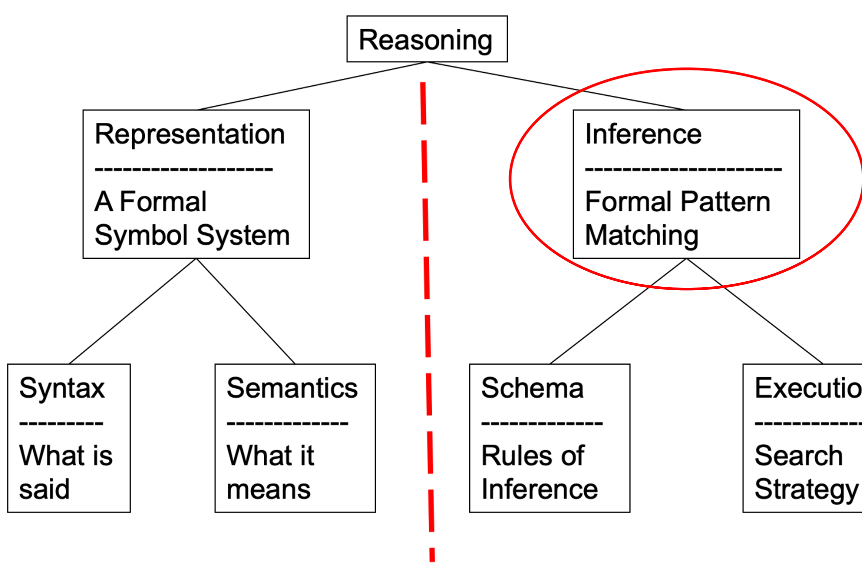
34

Syntactic Ambiguity—Partial Solution

- FOL can be **too** expressive, can offer **too many** choices
- Likely confusion, especially for teams of Knowledge Engineers
- Different team members can make different representation choices
  - E.g., represent "Ball43 is Red." as:
    - a predicate (= verb)? E.g., "Red(Ball43)" ?
    - an object (= noun)? E.g., "Red = Color(Ball43))" ?
    - a property (= adjective)? E.g., "HasProperty(Ball43, Red)" ?
- **Partial solution**
  - An upon-agreed ontology that settles these questions
  - Ontology = what exists in the world & how it is represented
  - The Knowledge Engineering teams agrees upon an ontology BEFORE they begin encoding knowledge

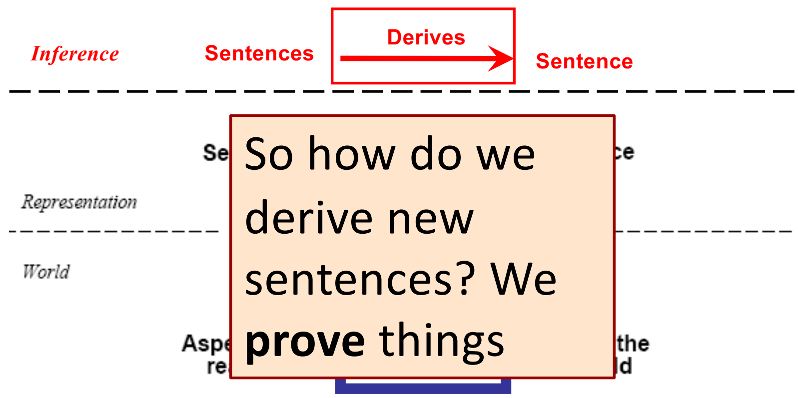
37

FOL (or FOPC) Ontology:
What kind of things exist in the world?
What do we need to describe and reason about?
Objects --- with their relations, functions, predicates, properties, and general rules.



38

Reminder: Schematic perspective



If KB is true in the real world,
then any sentence α ***derived*** *from KB*
by a sound inference procedure
is also true in the real world.

39

Proof methods

- Proof methods divide into (roughly) two kinds:
- Model checking:
  - Searching through truth assignments.
  - Improved backtracking: Davis-Putnam-Logemann-Loveland (DPLL)
  - Heuristic search in model space: Walksat
- Application of inference rules:
  - Legitimate (sound) generation of new sentences from old.
  - Forward & Backward chaining
  - Resolution — KB is in Conjunctive Normal Form (CNF)

40

Model Checking

- Given some knowledge base (KB), does sentence S hold?
- Basically **generate and test**:
  - Generate all the possible models
  - Consider the models M in which KB is TRUE
  - If ∀M(S), then S is **provably true**
  - If ∀M(¬S), then S is **provably false**
  - Otherwise (∃M1 S ∧ ∃M2 ¬S): S is **satisfiable** but neither provably true or provably false

What does model mean?

Model: an interpretation – or assignment of truth values to literals – of a set of sentences such that every sentence is *True*

41

Method 1: Inference by Enumeration

- Also called **Model Checking** or **Truth Table Enumeration**
- LET: KB = A ∨ C, B ∨ ¬C β = A ∨ B
- QUERY: KB ⊨ β ?

A	B	C
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

NOTE: The computer doesn't know the meaning of the proposition symbols

So, *all* logically distinct cases must be checked to prove that a sentence *can* be derived from KB

43

Inference by Enumeration

- LET: KB = A V C, B V ¬C β = A V B
- QUERY: KB ⊨ β ?

A	B	C	AVC	BV ⟵C	KB
false	false	false	false	true	false
false	false	true	true	false	false
false	true	false	false	true	false
false	true	true	true	true	true
true	false	false	true	true	true
true	false	true	true	false	false
true	true	false	true	true	true
true	true	true	true	true	true

Rows where *all* of sentences in KB are true are the models of KB

44

Inference by Enumeration

- LET: KB = A V C, B V ¬C β = A V B
- QUERY: KB ⊨ β ?

A	B	C	AVC	BV ⟵C	KB	AVB	KB⇒β
false	false	false	false	true	false	false	true
false	false	true	true	false	false	false	true
false	true	false	false	true	false	true	true
false	true	true	true	true	true	true	true
true	false	false	true	true	true	true	true
true	false	true	true	false	false	true	true
true	true	false	true	true	true	true	true
true	true	true	true	true	true	true	true

β is entailed by KB if *all* models of KB are models of β, i.e., *all* rows where KB is true, β is also true

In other words: KB ⇒ β is valid

45

Inference by Enumeration

- Using inference by enumeration to build a complete truth table in order to determine if a sentence is entailed by KB is a complete inference algorithm for Propositional Logic
- But very slow: takes exponential time
- Imagine we had 5 literals… or 30, or hundreds

46

Review: Inference Rules for FOL

- Inference rules for **propositional logic** apply to **First Order Logic**
  - Modus Ponens, And-Introduction, And-Elimination, Contraposition, …
- New (sound) inference rules for use with quantifiers:
  - Universal elimination
  - Existential introduction
  - Existential elimination
  - Generalized Modus Ponens (GMP)

47

Method 2: Natural Deduction = Constructing a Proof

- A Proof is a sequence of inference steps that leads from α (i.e., KB) to β (i.e., query)
- This is a search problem!

KB:
$(P \wedge Q) \Rightarrow R$
$(S \wedge T) \Rightarrow Q$
S
T
P

Query:
R

48

Proof by Natural Deduction

KB:
$(P \wedge Q) \Rightarrow R$
$(S \wedge T) \Rightarrow Q$
S
T
P

Query:
R

1.	S	Premise (i.e., given sentence in KB)
2.	T	Premise
3.	$S \wedge T$	Conjunction(1, 2) (And-Introduction)
4.	$(S \wedge T) \Rightarrow Q$	Premise
5.	Q	Modus Ponens(3, 4)
6.	P	Premise
7.	$P \wedge Q$	Conjunction(5, 6)
8.	$(P \wedge Q) \Rightarrow R$	Premise
9.	R	Modus Ponens(7, 8)

This is the expected format of proofs in the homework!

49

Proof by Natural Deduction

- KB:
  - HaveLecture $\Leftrightarrow$ (isTuesday $\vee$ isThursday)
  - $\neg$ HaveLecture
- Query:
  - $\neg$ isTuesday

50

Proof

Contraposition:
$(P \Rightarrow Q) \equiv (\neg Q \Rightarrow \neg P)$

KB:

1. HaveLecture $\Leftrightarrow$ (isTuesday $\vee$ isThursday)
2. $\neg$HaveLecture

3. (HaveLecture $\Rightarrow$ (isTuesday $\vee$ isThursday)) $\wedge$ ((isTuesday $\vee$ isThursday) $\Rightarrow$ HaveLecture)	iff elimination to 1
4. (isTuesday $\vee$ isThursday) $\Rightarrow$ HaveLecture	and-elimination to 3
5. $\neg$HaveLecture $\Rightarrow$ $\neg$(isTuesday $\vee$ isThursday)	contraposition to 4
6. $\neg$(isTuesday $\vee$ isThursday)	Modus Ponens 2,5
7. $\neg$isTuesday $\wedge$ $\neg$isThursday	de Morgan to 6
8. $\neg$isTuesday	and-elimination to 7

51

Automating FOL Inference with Generalized Modus Ponens

52

Automated Inference for FOL

- Automated inference using FOL is harder than PL
  - Variables can take on an infinite number of possible values
    - From their domains, anyway
    - This is a reason to do careful KR!
  - So, potentially infinite ways to apply Universal Elimination
- *Godel's Completeness Theorem* says that FOL entailment is only semidecidable*
  - If a sentence is **true** given a set of axioms, can prove it
  - If the sentence is **false**, then there is no guarantee that a procedure will ever determine this
  - **Inference may never halt**

*The "halting problem"

53

Generalized Modus Ponens (GMP)

- Apply modus ponens reasoning to generalized rules
- Combines And-Introduction, Universal-Elimination, and Modus Ponens
  - From *P(c)* and *Q(c)* and $(\forall x)(P(x) \wedge Q(x)) \Rightarrow R(x)$ derive *R(c)*
- General case: **Given**
  - **atomic sentences** $P_1, P_2, \ldots, P_N$
  - **implication sentence** $(Q_1 \wedge Q_2 \wedge \ldots \wedge Q_N) \Rightarrow R$
    - $Q_1, \ldots, Q_N$ and R are atomic sentences
  - **substitution** $\text{subst}(\theta, P_i) = \text{subst}(\theta, Q_i)$ for $i=1,\ldots,N$
  - **Derive new sentence: subst(θ, R)**

54

Generalized Modus Ponens (GMP)

- **Derive new sentence: subst(θ, R)**
- Substitutions
  - subst(θ, α) denotes the result of applying a **set of substitutions,** defined by θ, to the sentence α
  - A substitution list $\theta = \{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$ means to replace all occurrences of variable symbol v_i by term t_i
  - Substitutions are made in left-to-right order in the list
  - subst({x/IceCream, y/Ziggy}, eats(y,x)) = eats(Ziggy, IceCream)

55

Review: Horn Clauses

- A Horn sentence or Horn clause has the form:

  $P1 \wedge P2 \wedge P3 \ldots \wedge Pn \Rightarrow Q$

- or alternatively

  $\neg P1 \vee \neg P2 \vee \neg P3 \ldots \vee \neg Pn \vee Q$

- where Ps and Q are non-negated atoms
- To get a proof for Horn sentences, apply Modus Ponens repeatedly until nothing can be done
- Horn clauses are a subset of the set of sentences representable in FOL

56

Horn Clauses II

- These are Horn clauses (special cases):
  - $P1 \wedge P2 \wedge \ldots Pn \Rightarrow Q$
  - $P1 \wedge P2 \wedge \ldots Pn \Rightarrow$ false
  - true $\Rightarrow Q$
- These are not Horn clauses:
  - $p \vee q$ (all but one literal must be negated)
  - $(P \wedge Q) \Rightarrow (R \vee S)$ (non-literal after the implication)

57

Inference with Horn KBs

- If everything is Horn clauses, only 1 rule of inference needed
- Generalized Modus Ponens (GMP):

  Given P, Q, and (P ∧ Q) ⇒ R, conclude R
- Written as:

$$\frac{P,\ Q,\ (P \land Q) \Rightarrow R}{R}$$

58

Method 3: Forward Chaining

- Proofs start with the given axioms/premises in KB, deriving new sentences using GMP until the goal/query sentence is derived
- This defines a **forward-chaining** inference procedure because it moves "forward" from the KB to the goal [eventually]
- Forward chaining with Horn clause KB is **complete**
  - A formal system is called complete with respect to a particular property if every formula having the property can be derived using that system, i.e. is one of its theorems;
  - **Intuitively, a system is called complete if it can derive every formula that is true.**

59

Forward Chaining

- "Apply" any rule whose premises are satisfied in the KB
- Add its conclusion to the KB until query is derived

KB:

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

query: Q

60

Forward Chaining

Given KB

1. P ⇒ Q
2. L ∧ M ⇒ P
3. B ∧ L ⇒ M
4. A ∧ P ⇒ L
5. A ∧ B ⇒ L
6. A
7. B
8. L GMP(5,6,7)
9. M GMP(3,7,8)
10. P GMP(2,8,9)
11. Q GMP(1,10)

61

Forward Chaining Example

- KB:
  - allergies(X) ⇒ sneeze(X)
  - cat(Y) ∧ allergic-to-cats(X) ⇒ allergies(X)
  - cat(Felix)
  - allergic-to-cats(Lise)
- Goal:
  - sneeze(Lise)

63

Forward Chaining

Knowledge Base
1. Allergies lead to sneezing.
   allergies(X) ⇒ sneeze(X)
2. Cats cause allergies if allergic to cats.
   cat(Y) ∧ allergic-cats(X) ⇒ allergies(X)
3. Felix is a cat.
   cat(Felix)
4. Lise is allergic to cats.
   allergic-cats(Lise)

sneeze(Lise) ← thing to infer truth of (query)

- Forward Chaining: apply rules

variable binding

cat(Y) ∧ allergic-cats(X) ⇒ allergies(X) ∧ cat(Felix)

⇒

cat(Felix) ∧ allergic-cats(X) ⇒ allergies(X) ∧ allergic-cats(Lise)

add new sentence to KB

⇒

allergies(Lise) ∧ allergies(X) ⇒ sneeze(X)

⇒

sneeze(Lise) ✓

64

Forward Chaining Exercise

- Consider the following KB:
  1. J ⇒ Q
  2. A ∧ I ⇒ J
  3. E ∧ F ⇒ I
  4. B ⇒ F
  5. A ∧ B ⇒ E
  6. A
  7. B
  8. E (GMP 5,6,7)
  9. F (GMP 4,7)
  10. I (GMP 3,8,9)
  11. J (GMP 2,6,10)
  12. Q (GMP 1,11)
- Prove Q. (Remember, you'll just use GMP over and over!)
  - **A, B, (A ∧ B) ⇒ C, ∴ C**

65

Method 4: Backward Chaining

- Forward chaining problem: can generate a lot of irrelevant conclusions
  - Search forward, start state = KB, goal test = state contains query
- Backward chaining
  - Work backwards from goal to premises
  - Find all implications of the form (…) ⇒ query
  - Prove all the premises of one of these implications
  - Avoid loops: check if new subgoal is already on the goal stack
  - Avoid repeated work: check if new subgoal
    - Has already been proved true, or
    - Has already failed

66

Backward Chaining

- Backward-chaining deduction using GMP
  - Complete for KBs containing only Horn clauses.
- Proofs:
  - Start with the goal query
  - Find rules with that conclusion
  - Prove each of the antecedents in the implication
- Keep going until you reach premises!

Avoid loops
- Is new subgoal already on goal stack?

Avoid repeated work: has subgoal
- Already been proved true?
- Already failed?

67

Backward Chaining Example

Given KB

1.	$P \Rightarrow Q$	
2.	$L \wedge M \Rightarrow P$	
3.	$B \wedge L \Rightarrow M$	
4.	$A \wedge P \Rightarrow L$	
5.	$A \wedge B \Rightarrow L$	
6.	A	
7.	B	
8.	Q	Goal
9.	P	Subgoal(1,8)
10.	$L \wedge M$	Subgoal(2,9)
11.	L	Subgoal(10)
12.	$A \wedge B$	Subgoal(5,11)
13.	A	True(6)
14.	B	True(7)
15.	L	True(5,13,14)
16.	M	True(14,15)
17.	P	True(15,16)
18.	Q	True(1,17)

68

Backward Chaining Example

- KB:
  - allergies(X) ⇒ sneeze(X)
  - cat(Y) ∧ allergic-to-cats(X) ⇒ allergies(X)
  - cat(Felix)
  - allergic-to-cats(Lise)
- Goal:
  - sneeze(Lise)

69

Backward Chaining

sneeze(Lise) ← query

- Backward Chaining: apply rules that end with the goal

variable binding

allergies(X) → sneeze(X) + sneeze(Lise)
new query: allergies(Lise)?

cat(Y) ∧ allergic-cats(X) → allergies(X) + allergies(Lise)
new query: cat(Y) ∧ allergic-cats(Lise)?

cat(Felix) + cat(Y) ∧ allergic-cats(Lise)
new sentence: cat(Felix) ∧ allergic-cats(Lise) ✓

Knowledge Base
1. Allergies lead to sneezing.
allergies(X) ⇒ sneeze(X)
2. Cats cause allergies if allergic to cats.
cat(Y) ∧ allergic-cats(X) ⇒ allergies(X)
3. Felix is a cat.
cat(Felix)
4. Lise is allergic to cats.
allergic-cats(Lise)

70

Forward vs. Backward Chaining

- FC is data-driven
  - Automatic, unconscious processing
  - E.g., object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving
  - Where are my keys? How do I get to my next class?
  - Complexity of BC can be much less than linear in the size of the KB

72

Completeness of GMP

- GMP (using forward or backward chaining) is complete for KBs that contain only Horn clauses
- It is ***not* complete** for simple KBs that contain non-Horn clauses
- The following KB entails that S(A) is true:
  - $(\forall x)\ P(x) \Rightarrow Q(x)$
  - $(\forall x)\ \neg P(x) \Rightarrow R(x)$
  - $(\forall x)\ Q(x) \Rightarrow S(x)$
  - $(\forall x)\ R(x) \Rightarrow S(x)$
- If we want to conclude S(A), with GMP we cannot, since the second one is not a Horn clause
- It is equivalent to $P(x) \vee R(x)$

73

Automating FOL Inference with Resolution

74

74

Resolution Rule of Inference

- Resolution Rule of Inference:

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

- Examples

$$\frac{A \vee B, \neg B}{A}$$

$$\frac{A \vee B \vee \neg C \vee D, \neg A \vee \neg E \vee F}{B \vee \neg C \vee D \vee \neg E \vee F}$$

75

Resolution

- Take any two "clauses" where one contains some symbol, and the other contains its complement (negative)

  $P \lor \mathbf{Q} \lor R$ $\neg\mathbf{Q} \lor S \lor T$

- Merge (resolve) them, by throwing away the symbol and its complement, to obtain their **resolvent** clause:

  $P \lor R \lor S \lor T$

- If two clauses resolve and there's no symbol left, you have derived False, aka the empty clause

76

Method 5: Resolution Refutation

- Show KB $\vDash \alpha$ by proving that KB $\land \neg\alpha$ is unsatisifiable, i.e., deducing False from KB $\land \neg\alpha$
- Your algorithm can use all the logical equivalences to derive new sentences, plus:
- **Resolution rule:** a *single* inference rule
  - **Sound:** only derives entailed sentences
  - **Complete:** can derive any entailed sentence
    - Resolution is refutation complete: if KB $\vDash \beta$, then KB $\land \neg \beta \vdash$ False
  - But the sentences need to be preprocessed into CNF
  - But all sentences can be converted into this form

77

Resolution Refutation Algorithm

1. Add negation of query to KB
2. Pick 2 sentences that haven't been used before and can be used with the Resolution Rule of inference
3. If none, halt and answer that the query is NOT entailed by KB
4. Compute resolvent and add it to KB
5. If False in KB
   - Then halt and answer that the query IS entailed by KB
   - Else Goto 2

78

Resolution Examples

$(A \vee B \vee C)$
$(\neg A)$
_ _ _ _ _ _ _ _ _ _ _ _
$\therefore (B \vee C)$

"If A or B or C is true, but not A, then B or C must be true."

$(A \vee B \vee C)$
$(\neg A \vee D \vee E)$
_ _ _ _ _ _ _ _ _ _ _
$\therefore (B \vee C \vee D \vee E)$

"If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true."

$(A \vee B)$
$(\neg A \vee B)$
_ _ _ _ _ _ _ _
$\therefore (B \vee B) \equiv B$

"If A or B is true, and not A or B is true, then B must be true."

Simplification is done always.

79

Review: Converting to CNF

- Replace all ⇔ using biconditional elimination
  - $\alpha \Leftrightarrow \beta \;\equiv\; (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
- Replace all ⇒ using implication elimination
  - $\alpha \Rightarrow \beta \;\equiv\; \neg\alpha \vee \beta$
- Move all negations inward using
  - double-negation elimination
    - $\neg(\neg\alpha) \;\equiv\; \alpha$
  - de Morgan's rule
    - $\neg(\alpha \vee \beta) \;\equiv\; \neg\alpha \wedge \neg\beta$
    - $\neg(\alpha \wedge \beta) \;\equiv\; \neg\alpha \vee \neg\beta$
- Apply distributivity of ∨ over ∧
  - $\alpha \wedge (\beta \vee \gamma) \;\equiv\; (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$

Result: something with clauses made up of ORs, separated by ANDs:

$\underline{(\neg A \vee B \vee C)} \wedge \underline{(\neg B \vee A)} \wedge \underline{(\neg C \vee A)}$

80

Resolution Refutation Steps

- Given KB and β (query)
- Add ¬β to KB, and convert all sentences to CNF
- Show this leads to False (aka “empty clause”). Proof by contradiction
- Example KB:
  - $A \Leftrightarrow (B \vee C)$
  - $\neg A$
- Example query: $\neg B$

81

Review: Example Conversion to CNF

- Example: $A \Leftrightarrow (B \vee C)$
- Eliminate $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
  - $= (A \Rightarrow (B \vee C)) \wedge ((B \vee C) \Rightarrow A)$
- 2. Eliminate $\Rightarrow$ by replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$ and simplify.
  - $= (\neg A \vee B \vee C) \wedge (\neg(B \vee C) \vee A)$
- 3. Move $\neg$ inwards using de Morgan's rules and simplify.
  - $= (\neg A \vee B \vee C) \wedge ((\neg B \wedge \neg C) \vee A)$
- 4. Apply distributive law ($\wedge$ over $\vee$) and simplify.
  - $= (\neg A \vee B \vee C) \wedge (\neg B \vee A) \wedge (\neg C \vee A)$

82

Resolution Refutation Preprocessing

Example KB:
$A \Leftrightarrow (B \vee C)$
$\neg A$
Example query:
$\neg B$

- Add $\neg\beta$ to KB, and convert to CNF:
  - 1: $\neg A \vee B \vee C$
  - 2: $\neg B \vee A$
  - 3: $\neg C \vee A$
  - 4: $\neg A$
  - 5: B

CNF conversion of $A \Leftrightarrow (B \vee C)$

- Want to reach goal: False (empty clause)

83

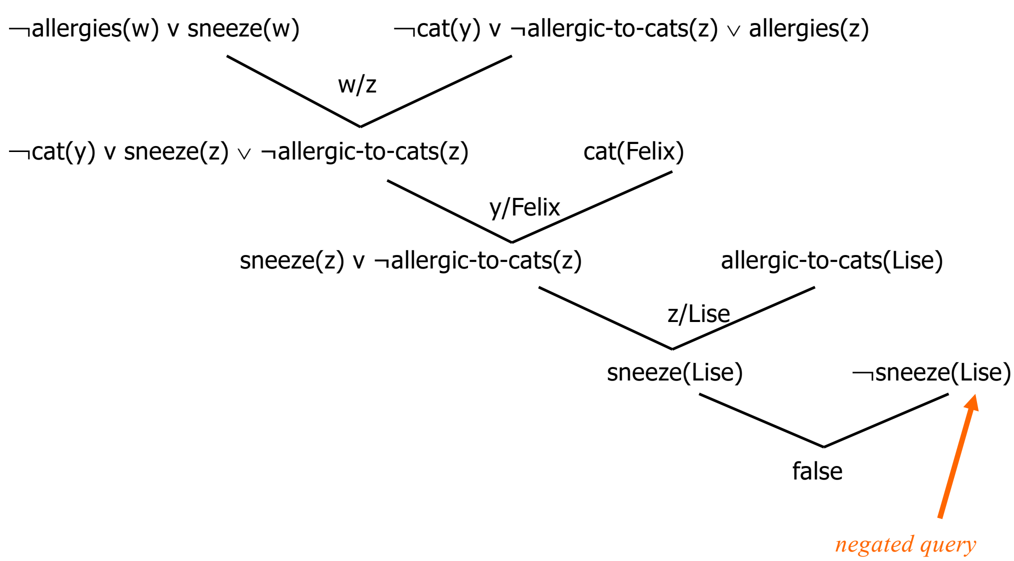
Resolution Refutation Example

- 1: ¬A V B V C
- 2: ¬B V A
- 3: ¬C V A
- 4: ¬A
- 5: B

- 6: A Resolve 2, 5
- 7: false/empty clause Resolve 6, 4

84

Refutation Resolution Proof Tree



85

Exercise: Did Curiosity Kill the Cat?

- Jack owns a dog. Every dog owner is an animal lover. No animal lover kills an animal. Either Jack or Curiosity killed the cat, who is named Tuna. **Did Curiosity kill the cat?**
- These can be represented as follows:
  - A. $(\exists x)$ Dog(x) $\wedge$ Owns(Jack,x)
  - B. $(\forall x)$ $((\exists y)$ Dog(y) $\wedge$ Owns(x, y)) $\rightarrow$ AnimalLover(x)
  - C. $(\forall x)$ AnimalLover(x) $\rightarrow$ $((\forall y)$ Animal(y) $\rightarrow$ $\neg$Kills(x,y))
  - D. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
  - E. Cat(Tuna)
  - F. $(\forall x)$ Cat(x) $\rightarrow$ Animal(x)
  - G. Kills(Curiosity, Tuna) ← Query

86

Resolution Proof

CLAUSAL FORM CONVERSION:
1. Eliminate all ↔ connectives
2. Eliminate all → connectives
3. Reduce the scope of each negation symbol to a single predicate
4. Standardize variables
5. Eliminate existential quantification with Skolem constants/functions
6. Remove universal quantifiers
7. Put into conjunctive normal form (conjunction of disjunctions)
8. Split conjuncts into separate clauses
9. Standardize variables again

- **Did Curiosity kill the cat?**
  - A. $(\exists x)$ Dog(x) $\wedge$ Owns(Jack,x)
  - B. $(\forall x)$ $((\exists y)$ Dog(y) $\wedge$ Owns(x, y)) $\rightarrow$ AnimalLover(x)
  - C. $(\forall x)$ AnimalLover(x) $\rightarrow$ $((\forall y)$ Animal(y) $\rightarrow$ $\neg$Kills(x,y))
  - D. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
  - E. Cat(Tuna)
  - F. $(\forall x)$ Cat(x) $\rightarrow$ Animal(x)
  - **G. Kills(Curiosity, Tuna) – goal: must be negated in your KB!**

RESOLUTION STEP: Given sentences

$P_1 \vee \ldots \vee P_n \qquad Q_1 \vee \ldots \vee Q_m$

- if P_j and $\neg Q_k$ **unify** with substitution list θ, then derive the resolvent sentence:

$\text{subst}(\theta, P_1 \vee \ldots \vee P_{j-1} \vee P_{j+1} \ldots P_n \vee Q_1 \vee \ldots Q_{k-1} \vee Q_{k+1} \vee \ldots \vee Q_m)$

87

Steps

- Convert to clause form
  - A1. (Dog(D)) ← D is a skolem constant
  - A2. (Owns(Jack,D))
  - B. (¬Dog(y), ¬Owns(x, y), AnimalLover(x))
  - C. (¬AnimalLover(a), ¬Animal(b), ¬Kills(a,b))
  - D. (Kills(Jack,Tuna), Kills(Curiosity,Tuna))
  - E. Cat(Tuna)
  - F. (¬Cat(z), Animal(z))
- Add the negation of query:
  - ¬G: (¬Kills(Curiosity, Tuna))

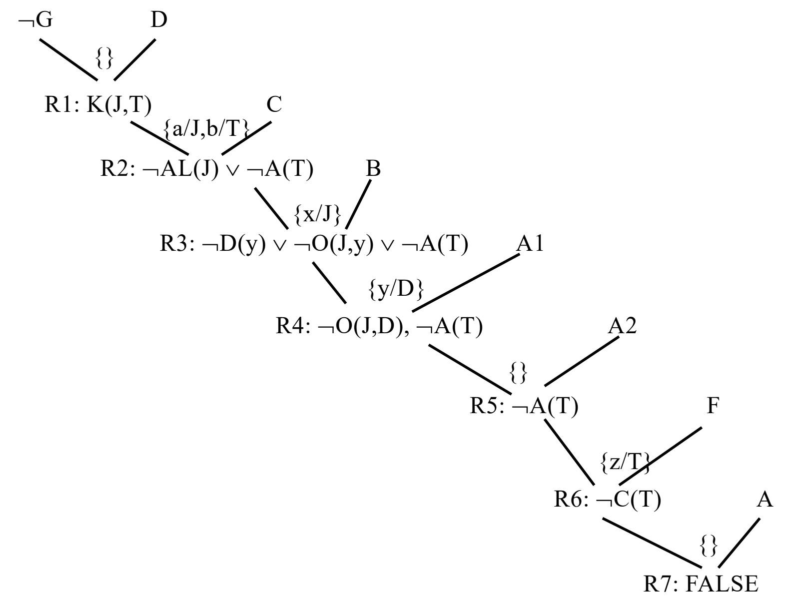
88

The Resolution Refutation Proof

• R1: ¬G, D, {}	(Kills(Jack, Tuna))
• R2: R1, C, {a/Jack, b/Tuna}	(~AnimalLover(Jack), ~Animal(Tuna))
• R3: R2, B, {x/Jack}	(~Dog(y), ~Owns(Jack, y), ~Animal(Tuna))
• R4: R3, A1, {y/D}	(~Owns(Jack, D), ~Animal(Tuna))
• R5: R4, A2, {}	(~Animal(Tuna))
• R6: R5, F, {z/Tuna}	(~Cat(Tuna))
• R7: R6, E, {}	FALSE

89

The proof tree



90

Resolution Refutation

- Given a consistent set of axioms KB and goal sentence Q, show that KB ⊨ Q
- **Proof by contradiction:** Add ¬Q to KB and try to prove false.
  - i.e., (KB ⊢ Q) ⟷ (KB ∧ ¬Q ⊢ False)
- Resolution is **refutation complete:** it can establish that a given sentence Q is entailed by KB, but can't (in general) be used to generate all logical consequences of a set of sentences
  - Also, it cannot be used to prove that Q is not entailed by KB.

91

Efficiency of Resolution Refutation

- Run time can be exponential in the worst case
  - Often much faster
- Factoring: if a new clause contains duplicates of the same symbol, delete the duplicates
  - $P \lor R \lor P \lor T \equiv P \lor R \lor T$
- If a clause contains a symbol and its complement, the clause is a tautology and is useless; it can be thrown away
  - a1: $(\neg A \lor B \lor C)$
  - a2: $(\neg B \lor A)$
  - Resolvent of a1 and a2 is: $B \lor C \lor \neg B$
  - Which is valid, so throw it away

92

Resolution Theorem Proving as Search

- Resolution can be thought of as the **bottom-up construction of a search tree**, where the leaves are the clauses produced by KB and the negation of the goal
- When a pair of clauses generates a new resolvent clause, add a new node to the tree with arcs directed from the resolvent to the two parent clauses
- **Resolution succeeds** when a node containing the **False** clause is produced, becoming the **root node** of the tree
- A strategy is **complete** if its use guarantees that the empty clause (i.e., false) can be derived whenever it is entailed

93

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
- Resolution is complete for propositional logic.
  Forward and backward chaining are linear-time, complete for Horn clauses
- Propositional logic lacks expressive power

94

Next Time

- Final bits of reasoning via inference
- Knowledge Representation
- Beginning of Planning?
- Project work – **bring computers**

95