CSCI 561 - Foundation for Artificial Intelligence

Discussion Section (Week 9)

PROF WEI-MIN SHEN SHEN@ISI.EDU

Outline

Knowledge-based Agents

Logics, Reasoning, Proving

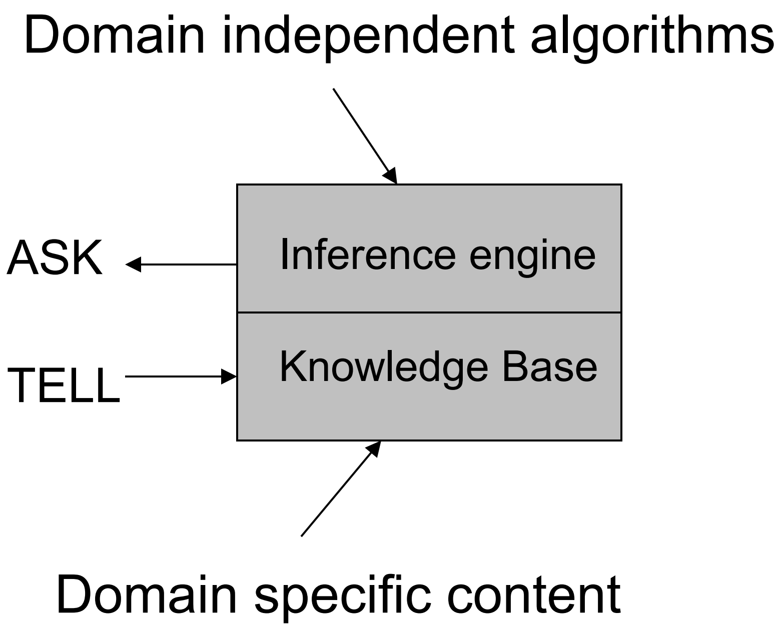
Logic programming

Knowledge Representation

Knowledge Engineering

Planning

Knowledge-Based Agents



Agent that uses prior or acquired knowledge to achieve its goals

- Can make more efficient decisions
- Can make informed decisions

Knowledge Base (KB): contains a set of <u>representations</u> of facts about the Agent's environment

Each representation is called a sentence

Use some knowledge representation language, to TELL it what to know e.g., (temperature 72F)

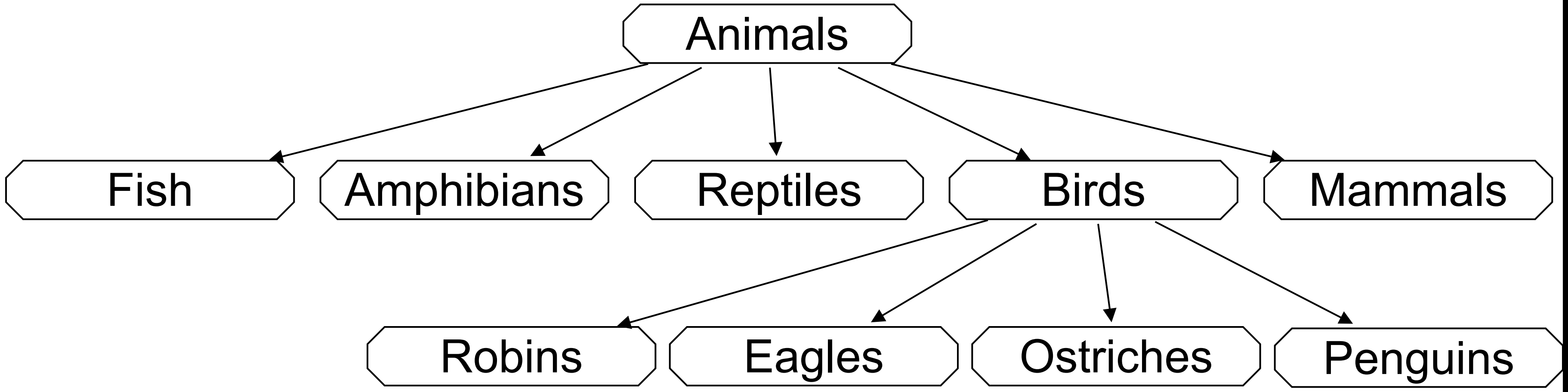
ASK agent to query what to do

Agent can use inference to deduce new facts from TELLed facts

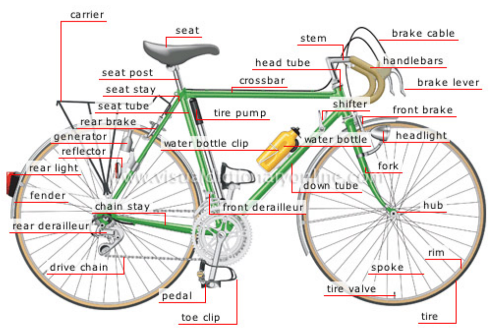
Type Hierarchies

Subclasses implicitly define a type hierarchy

- Also referred to as an ontology, a taxonomy, or a taxonomic hierarchy



PartOf Hierarchies



As with subclasses, the PartOf relation induces hierarchies, but of objects

$$\begin{tikzcd}
 & & \boxed{\text{Bicycle}_1} & \\
\boxed{\text{Seat}_1} \arrow[urr] & \boxed{\text{Frame}_1} \arrow[ur] & \boxed{\text{DriveTrain}_1} \arrow[u] & \boxed{\text{HandleBar}_1} \arrow[ul] \\
\boxed{\text{Gears}_1} \arrow[urr] & \boxed{\text{Chain}_1} \arrow[ur] & \boxed{\text{Wheel}_1} \arrow[u] & \boxed{\text{Wheel}_2} \arrow[ul]
\end{tikzcd}$$

Although can develop a comparable relationship between categories, for example:

- CategoryPartOf(Pistons, Engines) could represent that $\forall x\ [\ x \in \text{Engines} \Rightarrow \exists y\ y \in \text{Pistons} \wedge \text{PartOf}(y,x)\ ]$

Logic Concepts (Exercise)

Keywords:

I	**Syntax**
D	**Semantics**
C	**Model**
E	**Entailment**
B	**Inference**
H	**Soundness**
J	**Completeness**
A	**Equivalence**
F	**Validity**
G	**Satisfiability**

Definitions:

A. sentences are true in the same models
B. determine whether sentence entailed by KB
C. a possible world that defines truth values for all sentences
D. truth of sentences with respect to models
E. necessary truth of one sentence given another
F. sentence is true in all models
G. sentence is true in some model
H. produce only entailed sentences
I. formal structure of sentences
J. can produce all entailed sentences

Converting to CNF

$$\neg[((A \wedge B) \vee (D \wedge E)) \Rightarrow ((Q \Rightarrow R) \vee \neg S)]$$

Converting to CNF

$$\neg[((A \wedge B) \vee (D \wedge E)) \Rightarrow ((Q \Rightarrow R) \vee \neg S)]$$

Using $A \Rightarrow B$ is equivalent to $(\neg A \vee B)$, we get $\neg(\neg A \vee B)$ which is $A \wedge \neg B$

$$((A \wedge B) \vee (D \wedge E)) \wedge \neg((Q \Rightarrow R) \vee \neg S)$$

$$((A \wedge B) \vee D) \wedge ((A \wedge B) \vee E)) \wedge (\neg(Q \Rightarrow R) \wedge S)$$

$$(D \vee A) \wedge (D \vee B) \wedge (E \vee A) \wedge (E \vee B) \wedge \neg(\neg Q \vee R) \wedge S$$

$$(D \vee A) \wedge (D \vee B) \wedge (E \vee A) \wedge (E \vee B) \wedge (Q \wedge \neg R) \wedge S$$

There can also be longer derivation where students will eliminate $\rightarrow$, then move $\neg$ inwards using de Morgan, then eliminate double negative, distribute $\wedge$ over ? and get the correct result.

Inference

Use resolution and a proof by contradiction to prove D(Bob) from the following knowledge base:

~A(x) ∨ B(x)

A(x) ∨ C(x)

~B(x) ∨ D(x)

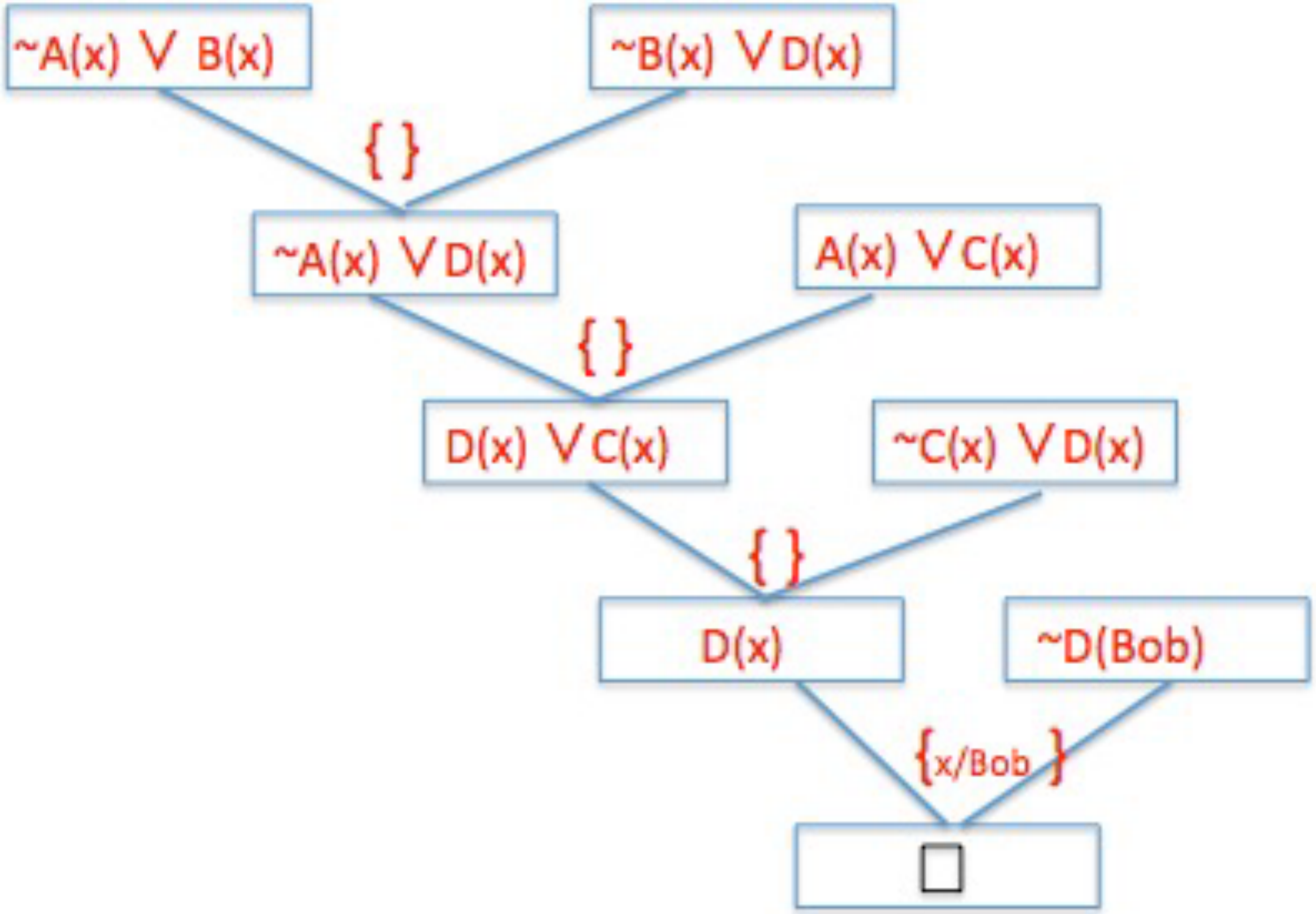
~C(x) ∨ D(x)

Please show the complete resolution proof, including all substitutions used (you will lose points for any missing step or substitution). You will get 0 if you use any other method than resolution and proof by contradiction.

Inference

Use resolution and a proof by contradiction to prove D(Bob) from the following knowledge base:

~A(x) v B(x)
A(x) v C(x)
~B(x) v D(x)
~C(x) v D(x)
~D(Bob)



Unification and Prolog

Answer the questions below given these two rules in Prolog for appending two lists to produce a third:

```
append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).
```

When L=[A], Y=[B, C], X=[[D]] then what are the values for:

- [3%] Z=

  [A, B, C]

- [3%] [X|Z]=

  [[[D]] , A, B, C]

Unification and Prolog

R1: append([],Y,Y).

R2: append([X|L,Y,[X|Z]) :- append(L,Y,Z).

When L=[A], Y=[B, C], X=[[D]]:

append([[[D]]|[A]], [B, C],[[[D]]|Z]) :- append([A], [B, C], Z).

Subquery: append([A], [B, C], Z)

Z= [A, B, C]

Unify with R2: append([A|[]], [B, C],[A|Z]):-append([], [B, C], Z)

Subquery: append([], [B, C], Z)

Z= [B, C]

Unify with R1: append([], [B, C], [B, C])

Z= [A, B, C] [X|Z]= [[[D]] , A, B, C]

Knowledge Representation

a. Circle the sentence that is the CNF form of this sentence $A \Leftrightarrow (B \vee E)$

1) $(\neg A \vee B \vee E) \wedge (B \vee A) \wedge (E \vee A)$
[✓ 2)] $(\neg A \vee B \vee E) \wedge (\neg B \vee A) \wedge (\neg E \vee A)$
3) $(\neg A \wedge B \wedge E) \vee (\neg B \wedge A) \vee (\neg E \wedge A)$
4) $(B \vee A) \wedge (E \vee A)$
5) $(B \wedge A) \wedge (E \wedge A)$

b. To unify α: <(x, +(y, x)), and β: <(10, +(a, b)), circle the most general unifier θ that makes α and β identical:

1) $\theta = \{x/a, y/b\}$
[✓ 2)] $\theta = \{x/10, y/a, b/10\}$
3) $\theta = \{x/10, y/b, a/10\}$
4) $\theta = \{x/a, y/b, b/10\}$
5) $\theta = \{x/10, y/a, a/b\}$

Knowledge Engineering

Circle the letter that corresponds to the best answer for the question:

[4%] What substitutions result from unifying P(x, John, y) with P(Sarah, F(y), z)

[✓ a.] These literals fail to unify.
b. {x/Sarah, John/F(y), y/z}
c. {x/John, y/Sarah, F/z}
d. The empty set: { }

[4%]] Knowledge Engineering is expensive because:

a. Encoding knowledge in a formal system is hard
b. It is an iterative modeling process
c. Domain experts don’t know what they know
[✓ d.] All of the above
e. None of the above

Planning

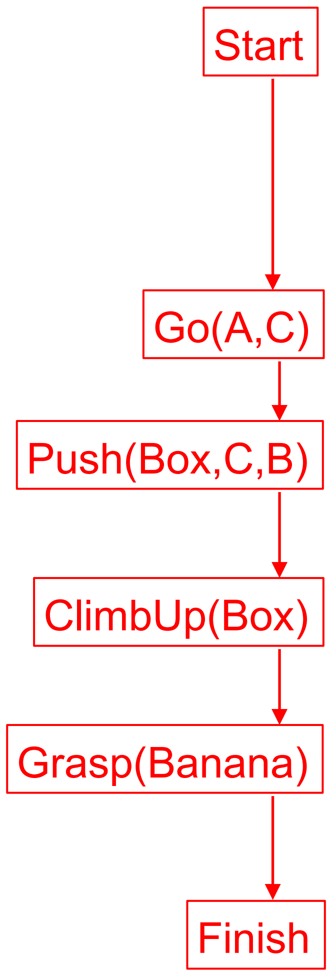
What is the plan to achieve the goal Have(Monkey, Banana)?

Action(ACTION:**Go(x,y)**,
PRECOND: At(Monkey,x),
EFFECT: At(Monkey,y)
∧¬(At(Monkey,x)))

Action(ACTION:**Push(b,x,y)**,
PRECOND: At(Monkey,x) ∧
Pushable(b),
EFFECT: At(b,y) ∧ At(Monkey,y)
∧¬At(b,x) ∧¬At(Monkey,x))

Action(ACTION:**ClimbUp(b)**,
PRECOND:At(Monkey,x) ∧At(b,x) ∧
Climbable(b),
EFFECT: On(Monkey,b)
∧¬Height(Monkey,Low) ∧
Height(Monkey,High))

Action(ACTION:**Grasp(o)**,
PRECOND:Height(Monkey,h)
∧Height(o,h) ∧At(Monkey,x) ∧
At(b,x), EFFECT:Have(Monkey,o))



At(Monkey,A) ∧ At(Bananas,B) ∧
At(Box,C) ∧ Height(Monkey,Low) ∧
Height(Box,Low) ∧
Height(Bananas,High) ∧ Pushable(Box)
∧ Climbable(Box)

At(Monkey,A)

At(Monkey,C) ∧ Pushable(Box)

At(Monkey,B) ∧At(Box,B) ∧
Climbable(Box)

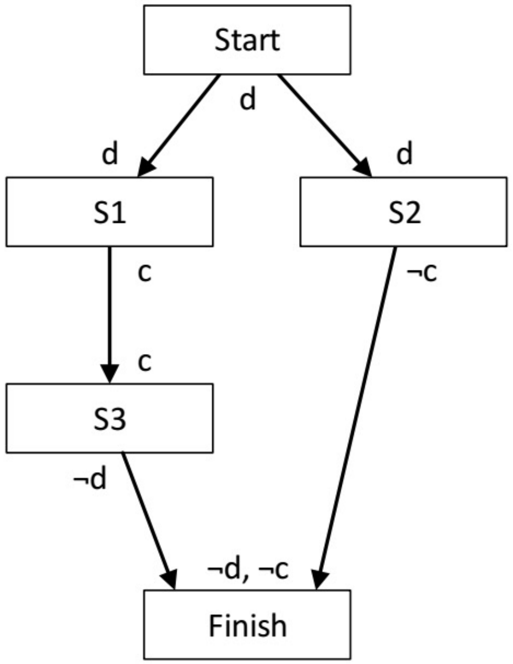
Height(Monkey,High) ∧
Height(Bananas,High) ∧
At(Monkey,B) ∧ At(Bananas,B)

Have(Monkey, Bananas)

Planning

6. [5%] Partial Order Planning

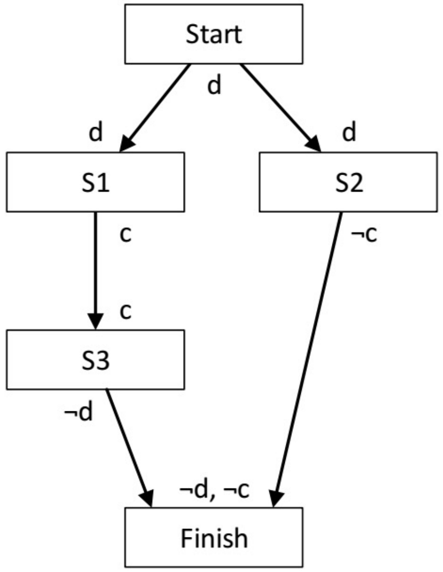
Consider the following partial order planning diagram. It currently contains no ordering constraints other than those implied by the partial order of the partial plan. S_1, S_2, and S_3 represent the actions taken, and c, ¬c, d and ¬d are the effects/preconditions.



Planning

6. [5%] Partial Order Planning

Consider the following partial order planning diagram. It currently contains no ordering constraints other than those implied by the partial order of the partial plan. S_1, S_2, and S_3 represent the actions taken, and c, ¬c, d and ¬d are the effects/preconditions.



Describe the flaw in this partial plan. Show how you would resolve it.

The problem with the plan is that S2's effect negates S3's precondition and S3's effect negates S2's precondition. So without any ordering constraints, the plan goes wrong. [4%]

Because the effects of both S2 and S3 are needed for the finish state, and we have no other states providing them, we cannot resolve the problem with current states, even if we add ordering constraints. We need more states. [1%]"

Neats and scruffies

- Based on page: http://en.wikipedia.org/wiki/Neats_vs._scruffies
- What are the strengths and weaknesses of each approach?
- Are these distinctions still relevant today?
- How can considering these approaches inform how we approach building AI applications?

What you should know

- What is Unification? How is it different than pattern matching? When is it used? What is the algorithm? Why is it important for Generalized Modus Ponens?
- What is the Lifted Resolution Rule?
- How does Proof by Contradiction compare to Forward Chaining? Backward Chaining? What is the differences between them?
- What is planning? Situation calculus? What is the difference between fluents and atemporals?
- What is the difference between total order planning and partial order planning or planning graphs?
- **Complete vs. Efficient?** How does this relate to the discussion on Neat vs Scruffy?

What you should know (CH 10, 12)

- What is an ontology? Why is it useful? What is inheritance? Why is it important?
- Know how to create an ontology. How do you define a category?
- What does it mean that logic is monotonic?
- What is knowledge sharing? What are the advantages and disadvantages?
- What is planning? Situation calculus? **What is the difference between fluents and atemporals?**
- What is the difference between total order planning and partial order planning?

Want More?

- Check out some of these exercises in the book:

8.1-3, 8.6, 8.9-10, 8.14, 8.17, 8.28
9.3, 10, 15, 20, 23