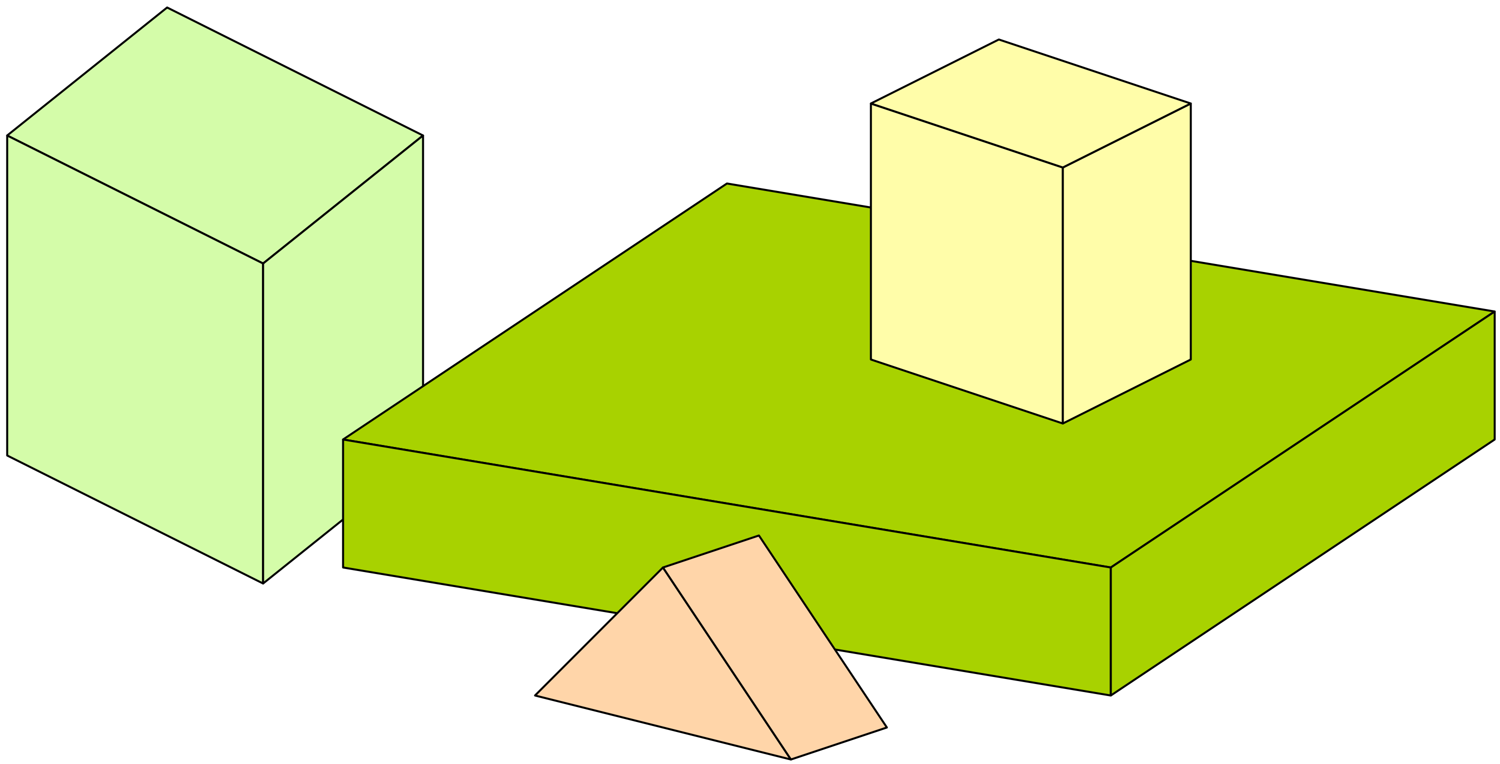
Constraint Satisfaction (edge labelling application)

Summary from last time

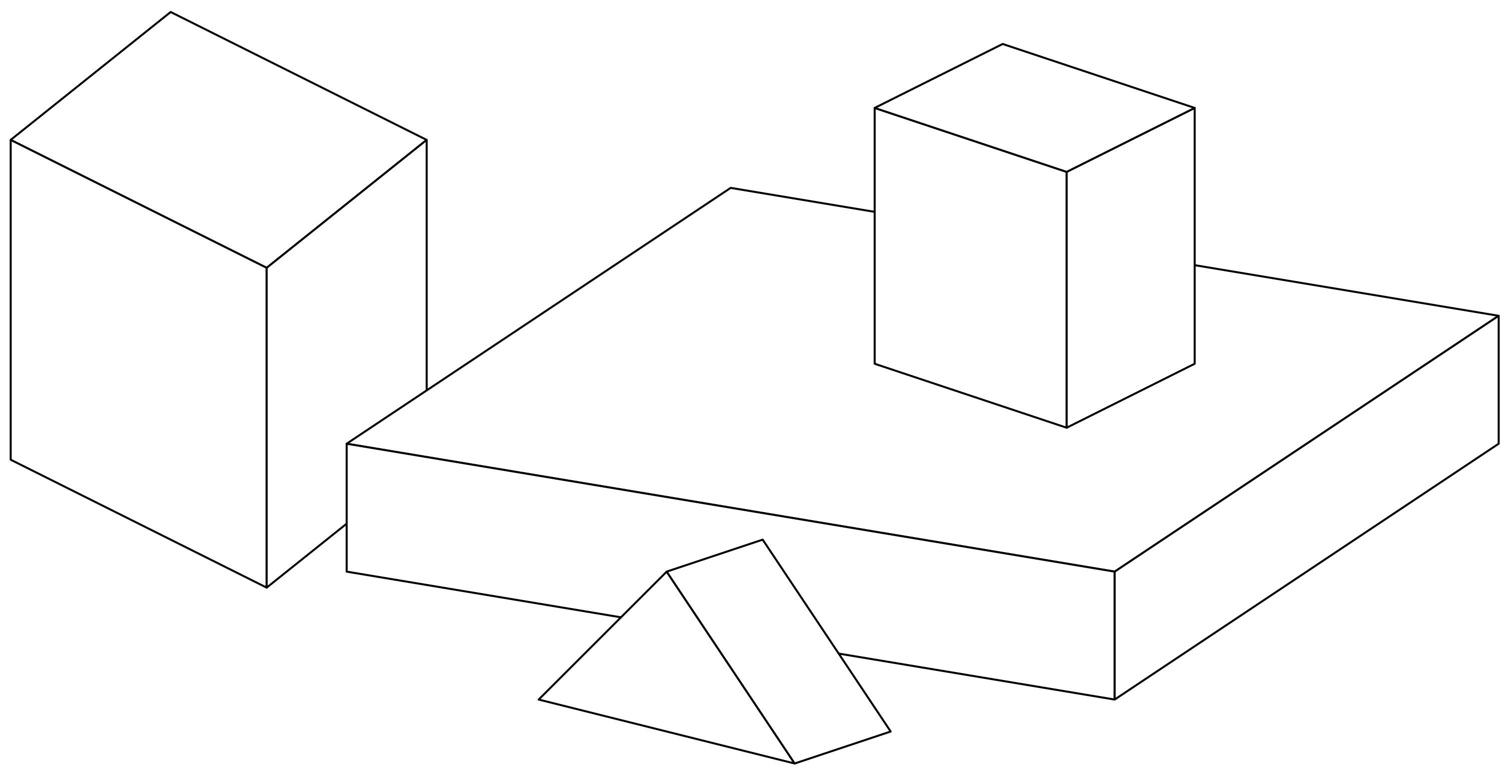
- Constraint Satisfaction Problems (CSP)
  - Variables, domains and constraints
- CSP as a search problem
  - Backtracking algorithm
  - General heuristics
- Forward checking
- Removing Arch Inconsistencies
- Interweaving CP and backtracking

Edge Labeling in Computer Vision - a CSP

Edge Labeling



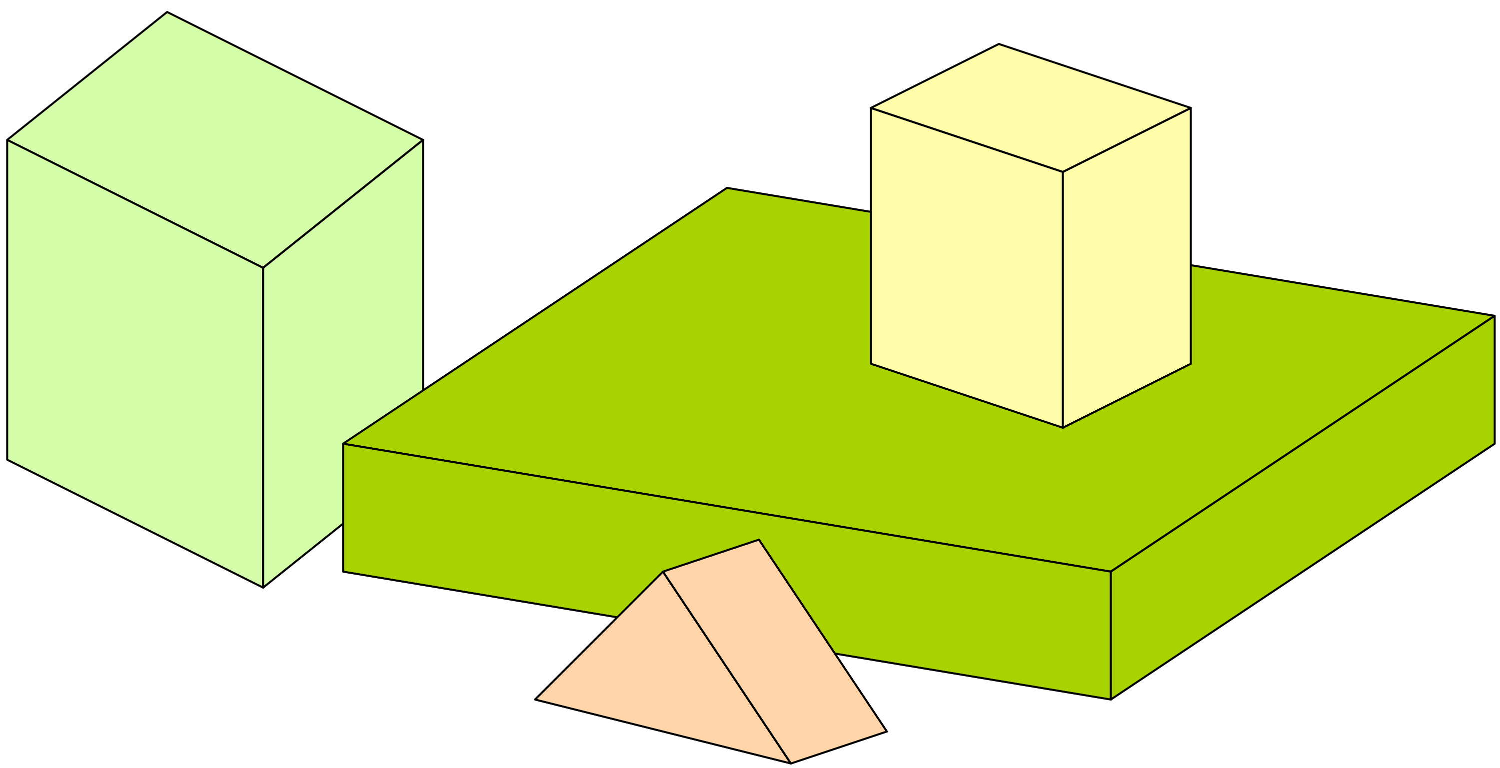
Edge Labeling



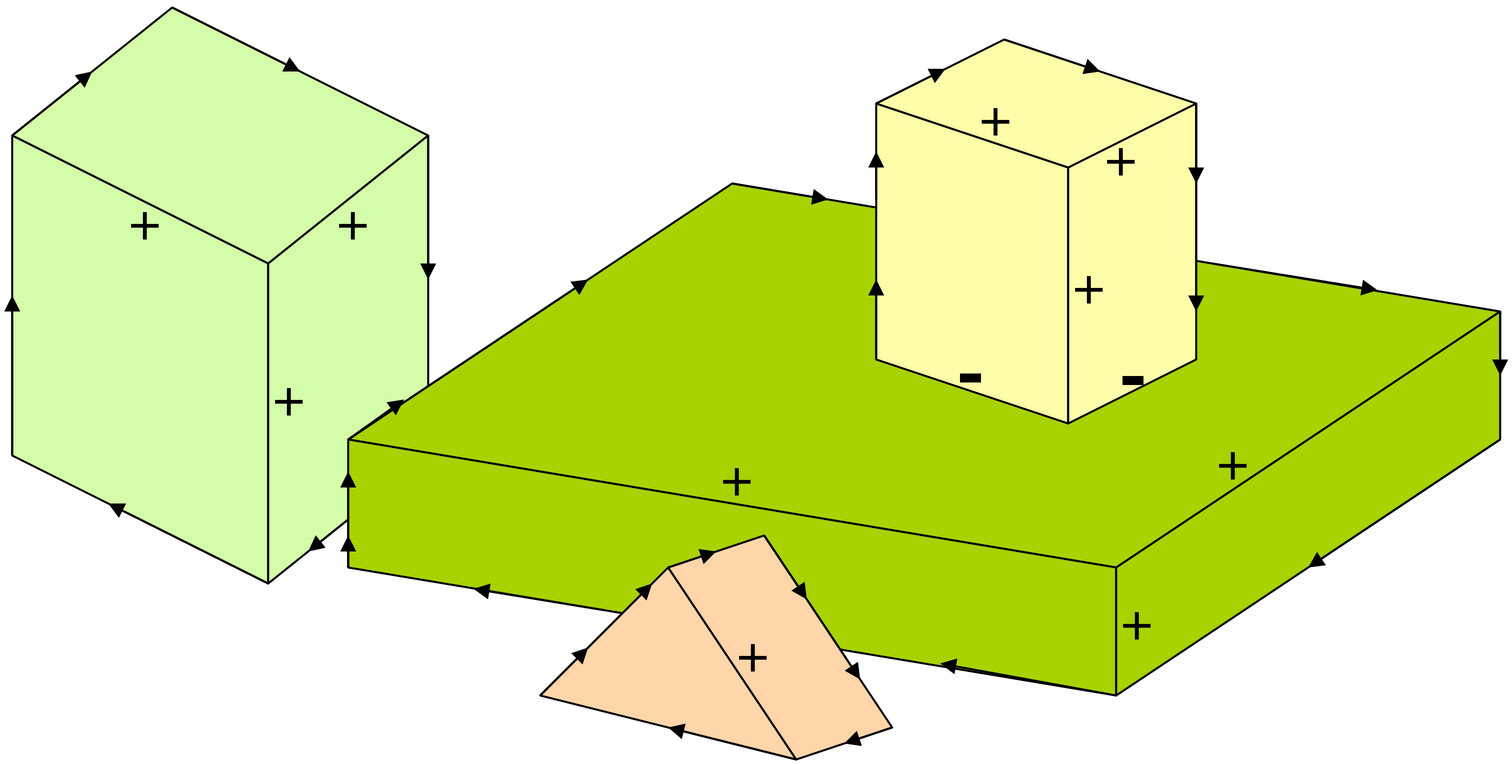
Labels of Edges

- Convex edge:
  - two surfaces intersecting at an angle greater than 180°
- Concave edge
  - two surfaces intersecting at an angle less than 180°
- + convex edge, both surfaces visible
- − concave edge, both surfaces visible
- ← convex edge, only one surface is visible and it is on the right side of ←

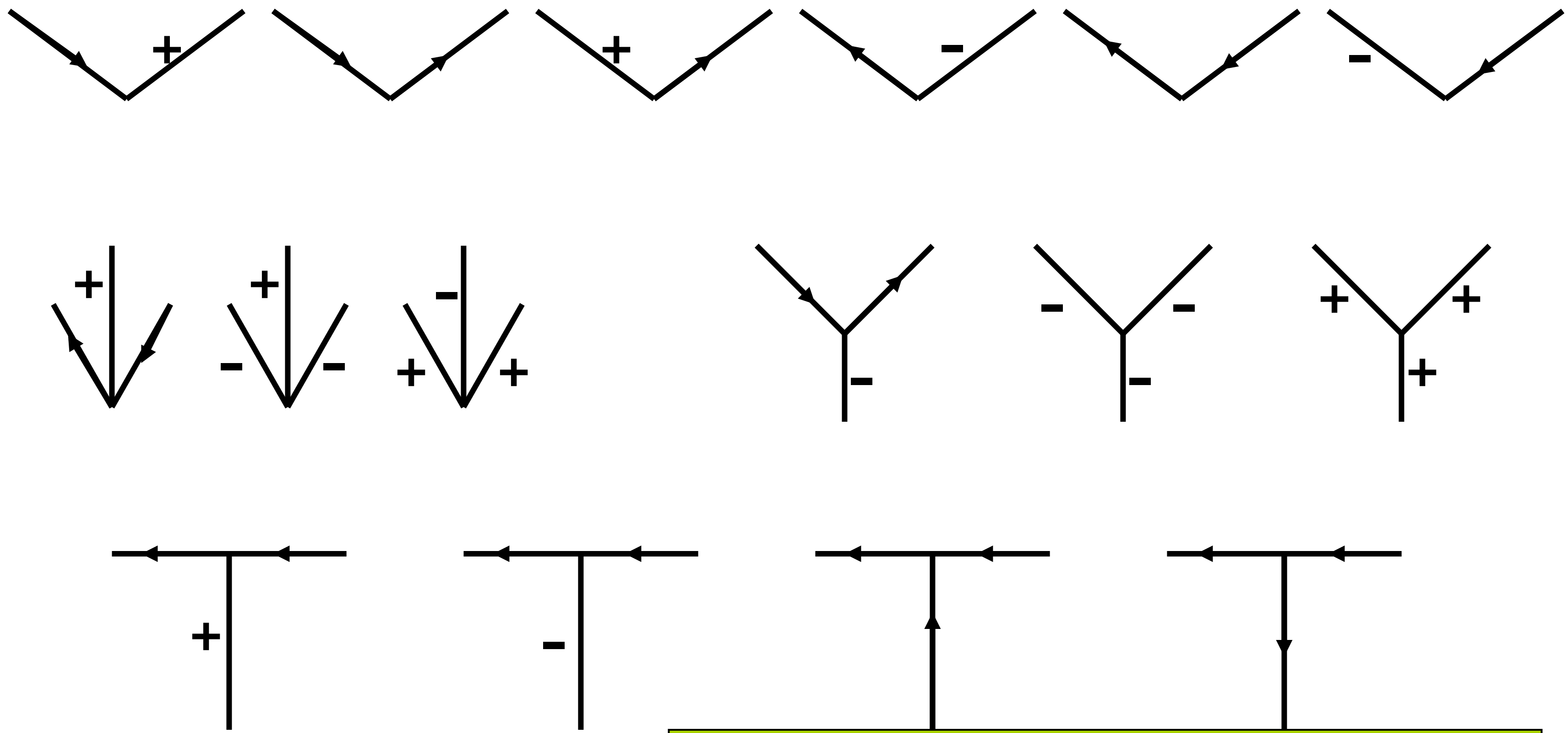
Edge Labeling



Edge Labeling



Junction Label Sets



(Waltz, 1975; Mackworth, 1977)

Edge Labeling as a CSP

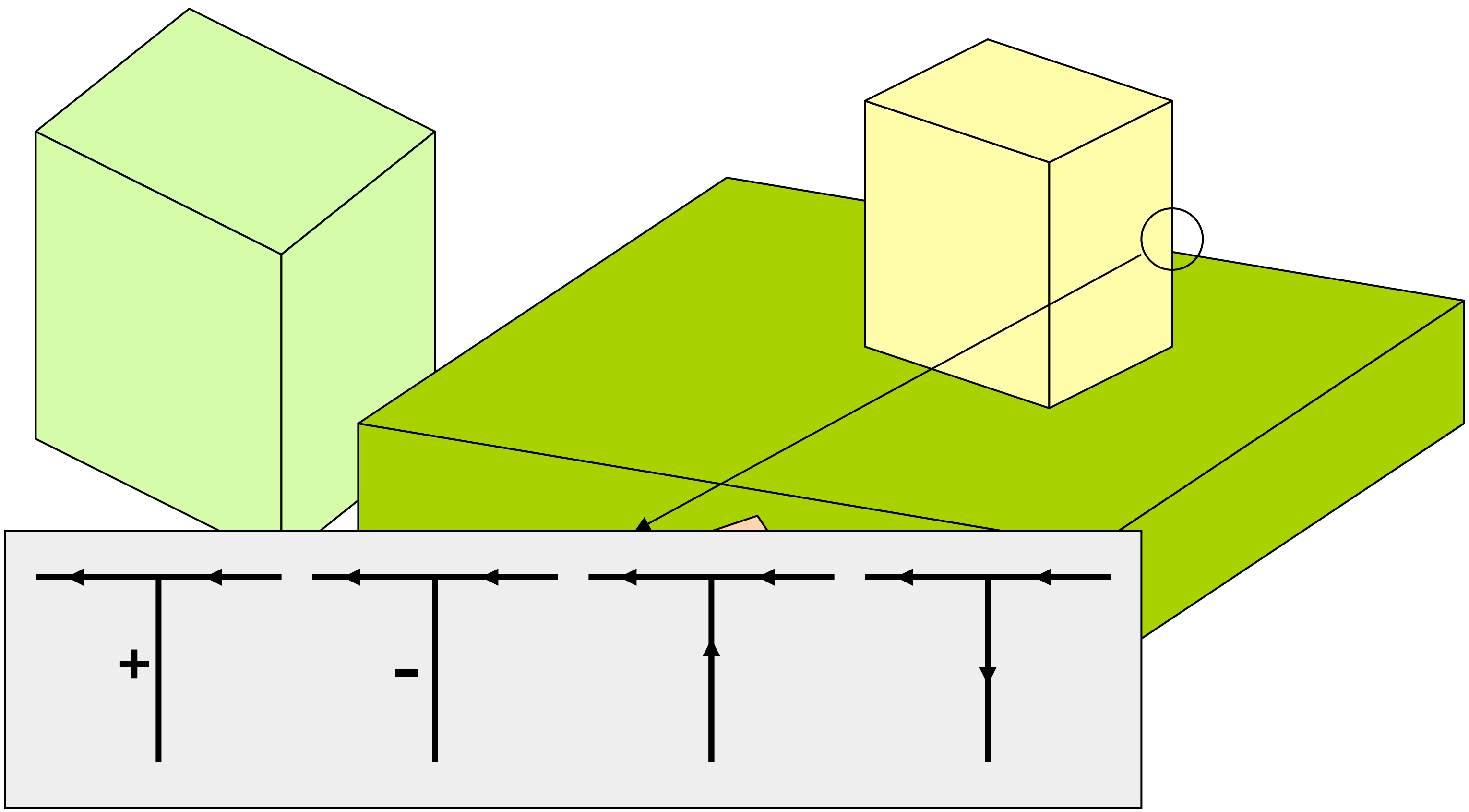
- A variable is associated with each junction
- The domain of a variable is the label set of the corresponding junction
- Each constraint imposes that the values given to two adjacent junctions give the same label to the joining edge

Removal of Arc Inconsistencies

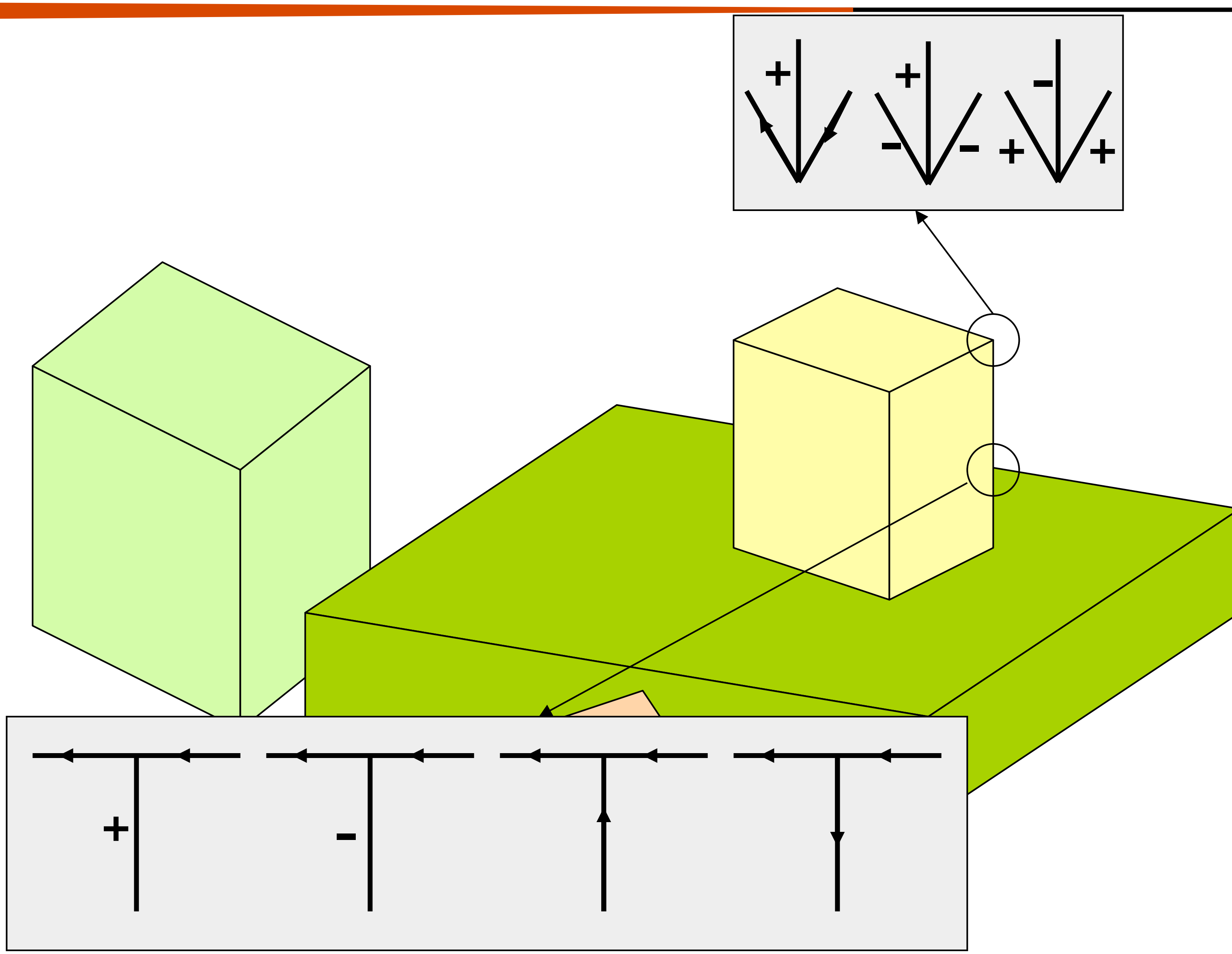
REMOVE-ARC-INCONSISTENCIES(J,K)

- removed ← *false*
- X ← label set of J
- Y ← label set of K
- For every label y in Y do
  - If there exists no label x in X such that the constraint (x,y) is satisfied then
    - Remove y from Y
    - If Y is empty then contradiction ← *true*
    - removed ← *true*
- Label set of K ← Y
- Return removed

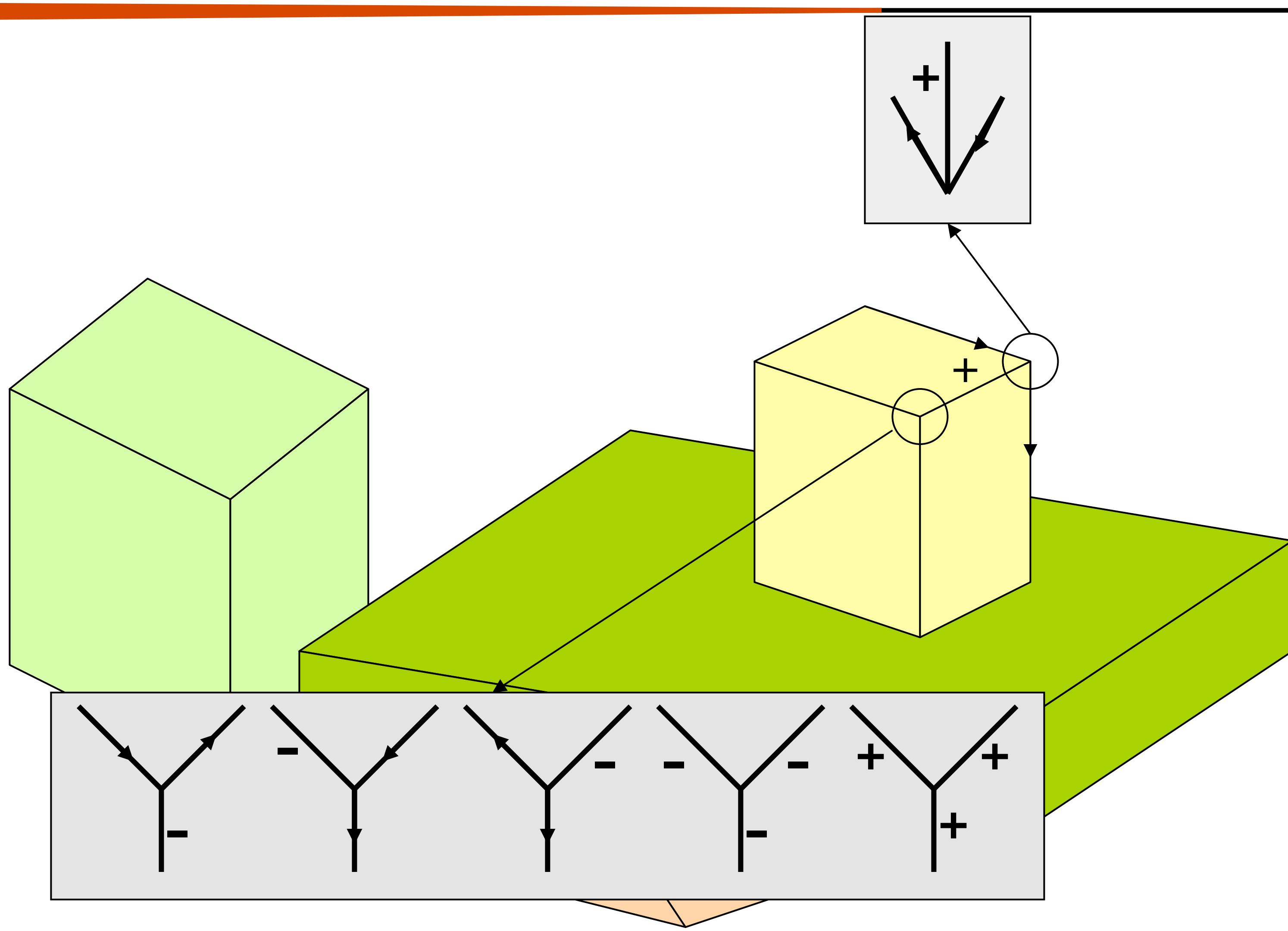
Edge Labeling



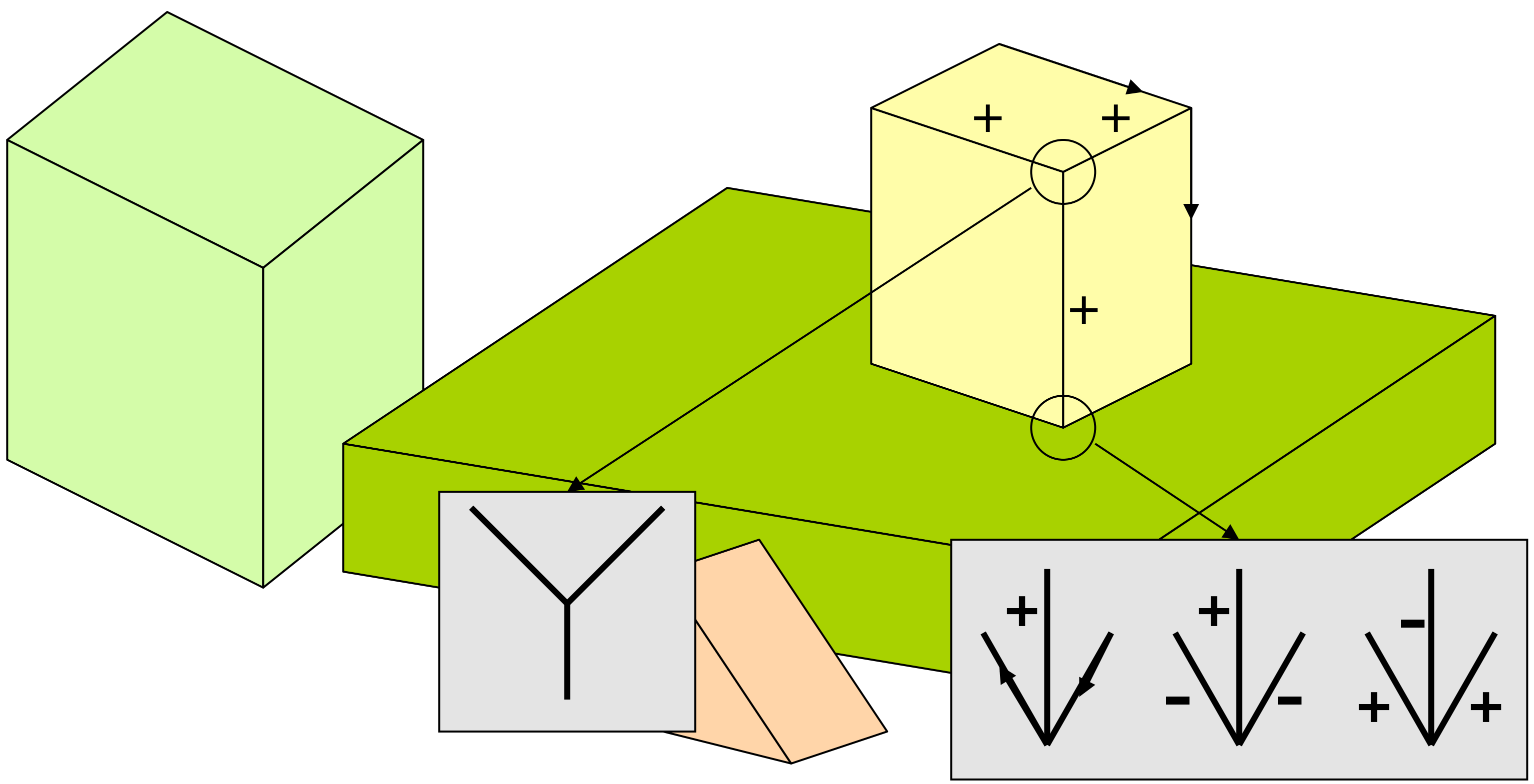
Edge Labeling



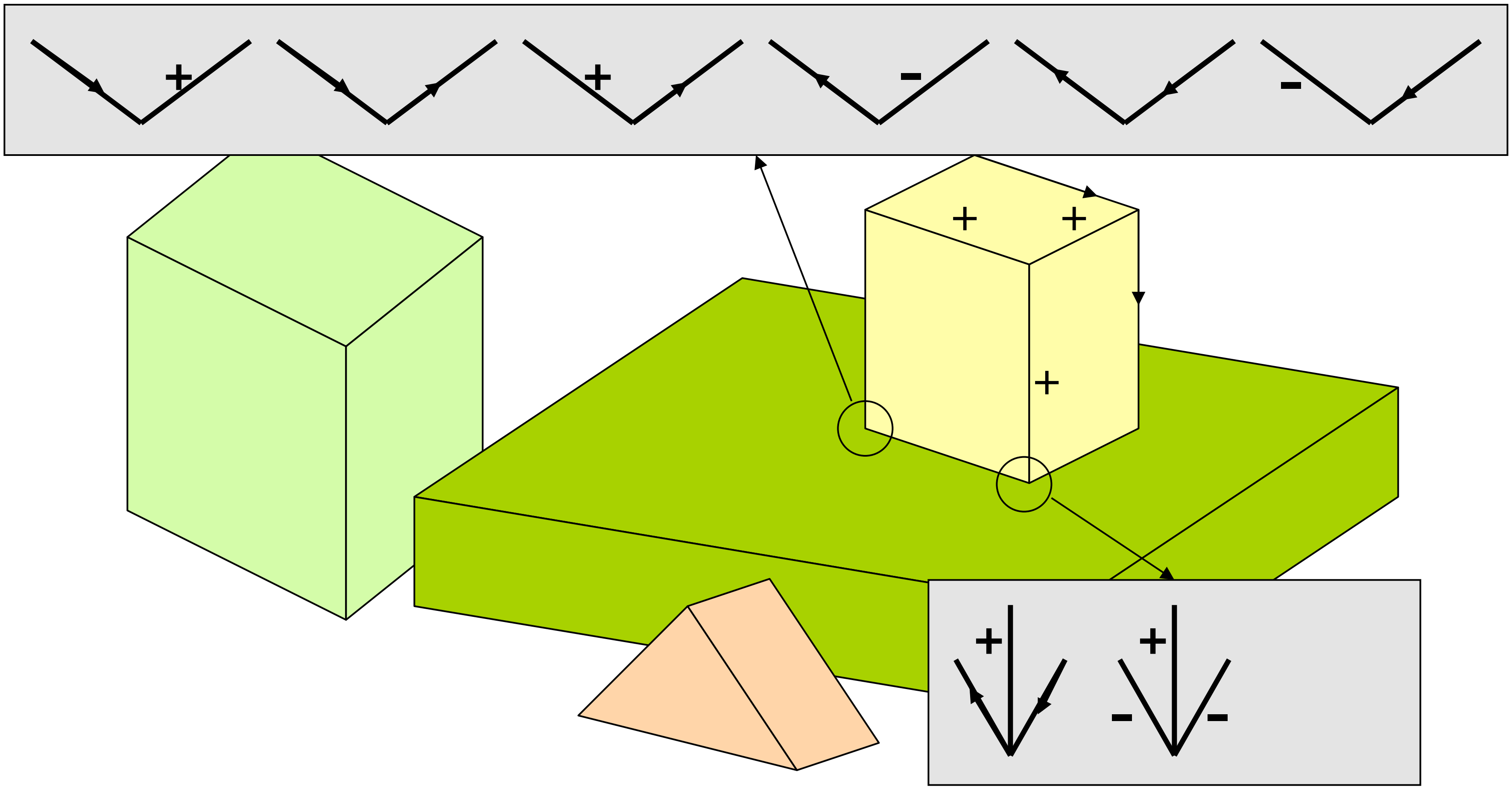
Edge Labeling



Edge Labeling



Edge Labeling



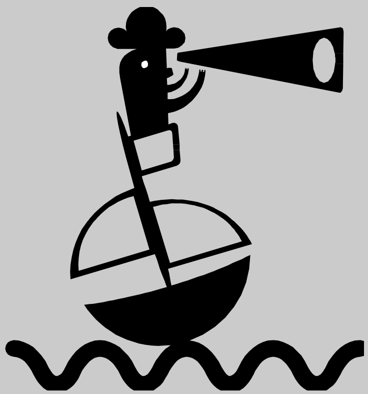
CP Algorithm for Edge Labeling

- Associate with every junction its label set
- Q ← stack of all junctions
- while Q is not empty do
  - J ← UNSTACK(Q)
  - For every junction K adjacent to J do
    - If REMOVE-ARC-INCONSISTENCIES(J,K) then
      - If K's domain is non-empty then STACK(K,Q)
      - Else return false

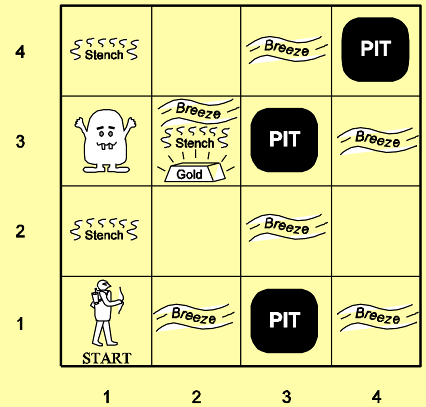
(Waltz, 1975; Mackworth, 1977)

Propositional Logic and AI

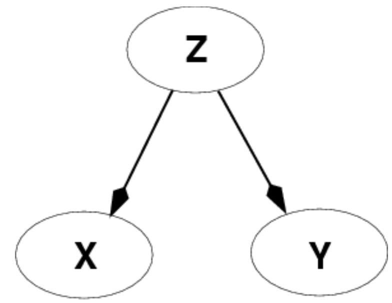
Major Topics in AI



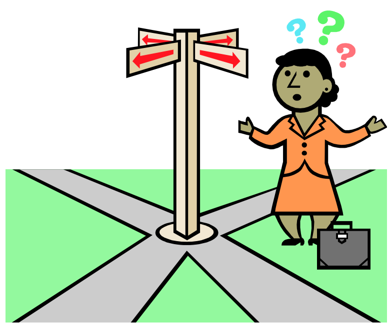
Search



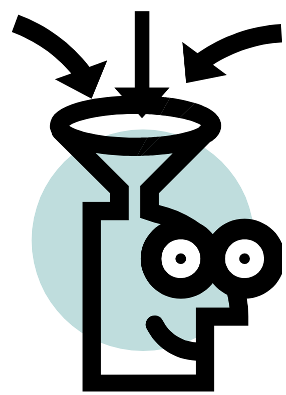
Knowledge-based agents & logic



Bayesian networks

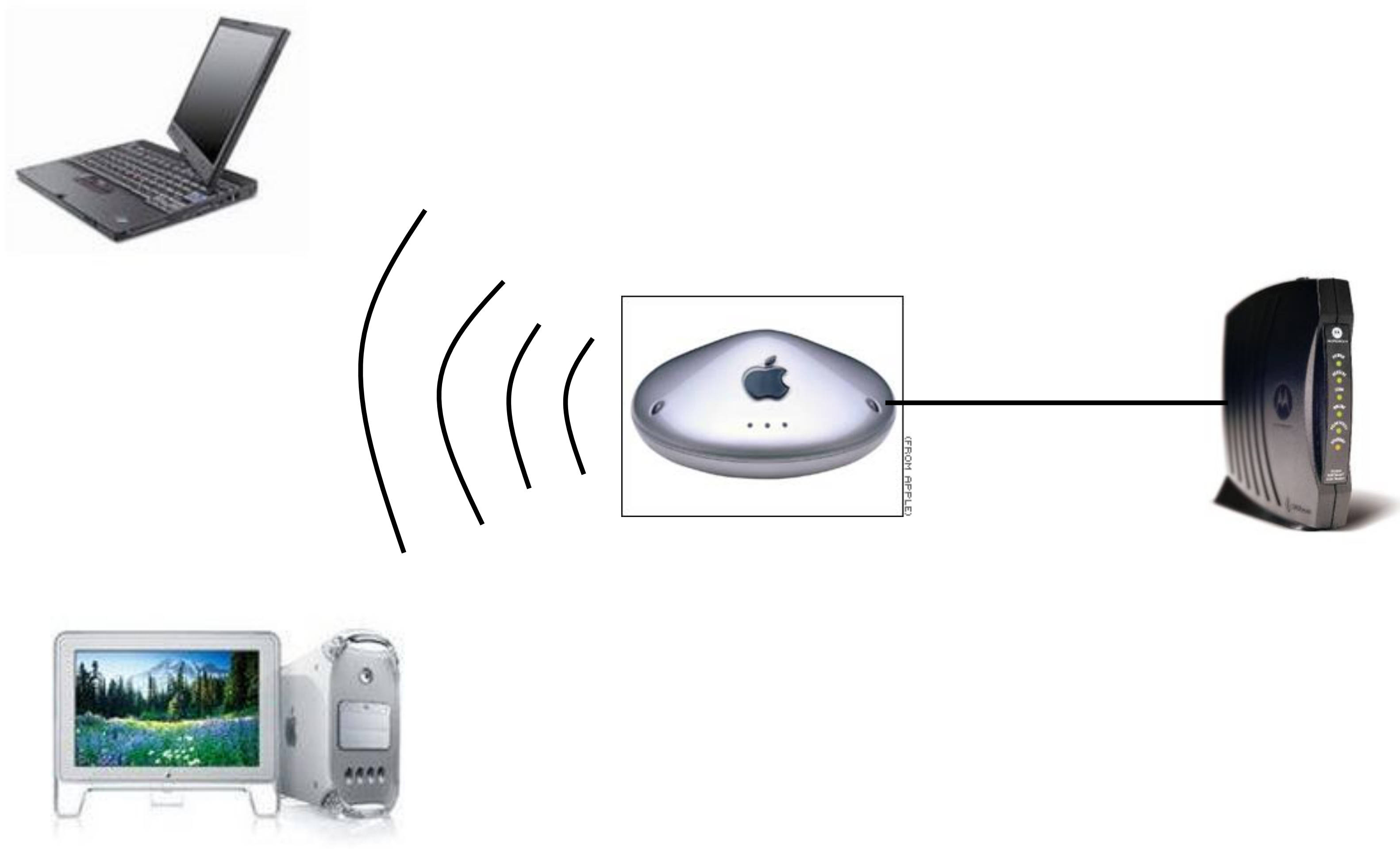


Decision networks (utility)

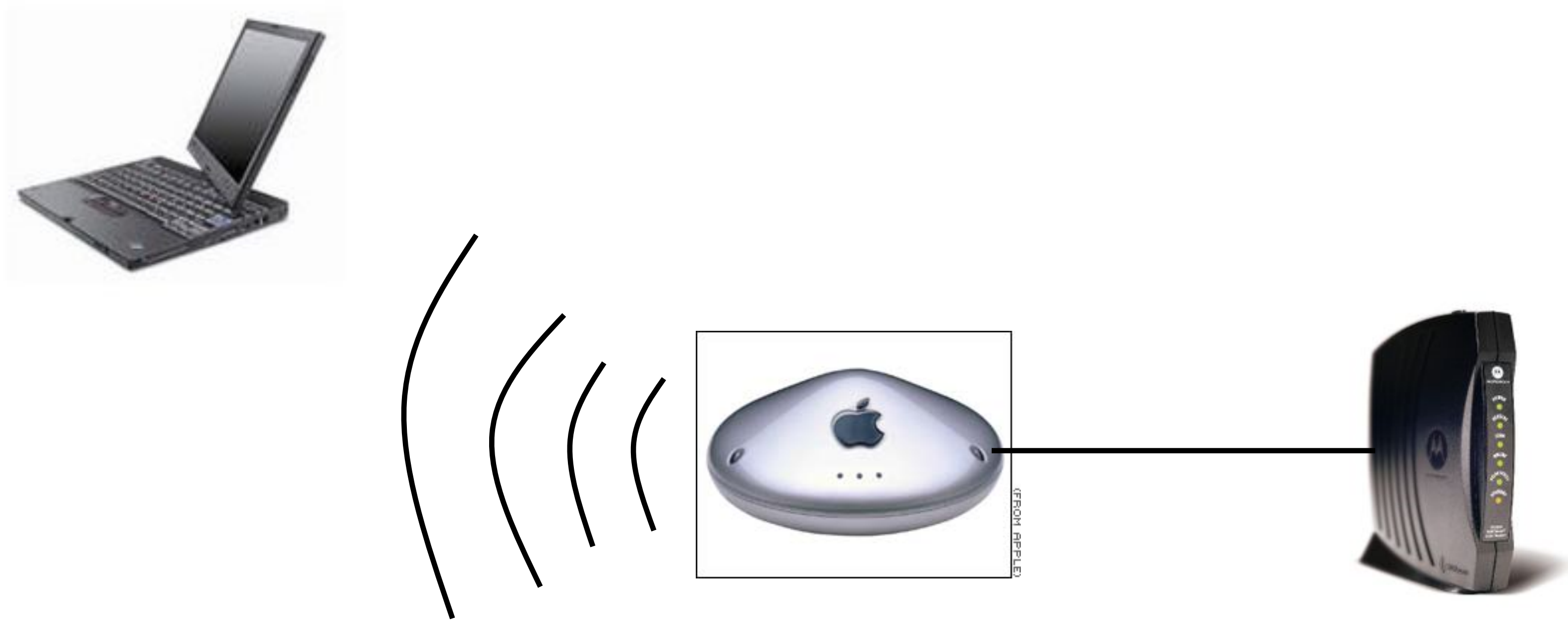


Learning

Example: Connecting to a home network



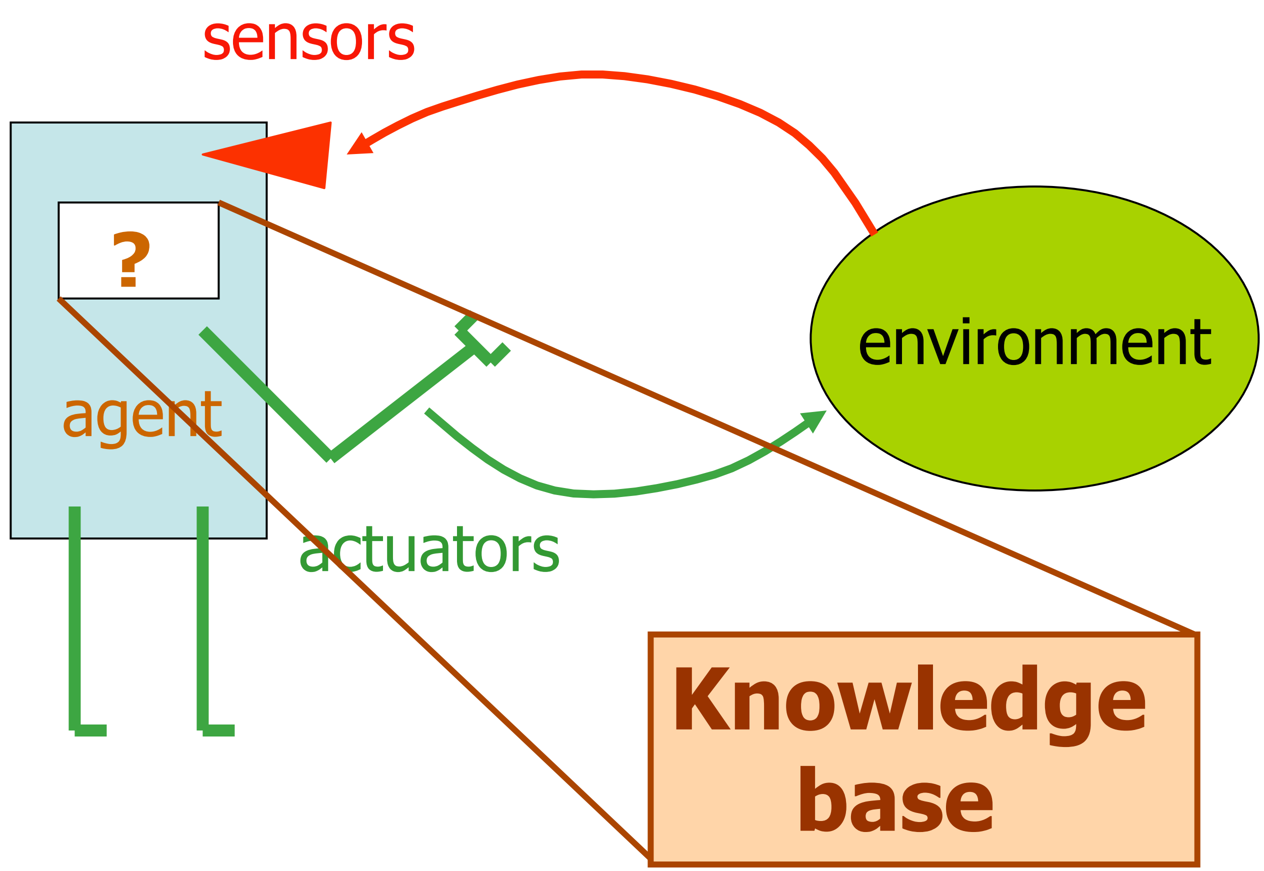
Example: Connecting to a home network



<u>Knowledge Base:</u>

- Tablet computers are flakey
- Flakey computers need to have their network connections reset frequently
- Lights on the router should be flashing
- Lights on the modem should be solid
- If the lights on the modem or the router are off, unplugging it and then reconnecting it often fixes the problem
- Resetting the computer's network connection did not help
- The lights on the modem are off

Knowledge-Based Agent



How do we represent knowledge?

- Procedurally (HOW):
  - Write methods that encode how to handle specific situations in the world
    - **`chooseMoveMancala()`**
    - **`driveOnHighway()`**
- Declaratively (WHAT):
  - Specify facts about the world
    - Two adjacent regions must have different colors
    - If the lights on the modem are off, it is not sending a signal

Logic for Knowledge Representation

Logic is a declarative language to:

- Assert sentences representing facts that hold in a world W (these sentences are given the value true)

- Deduce the true/false values to sentences representing other aspects of W

Logic for Knowledge Representation

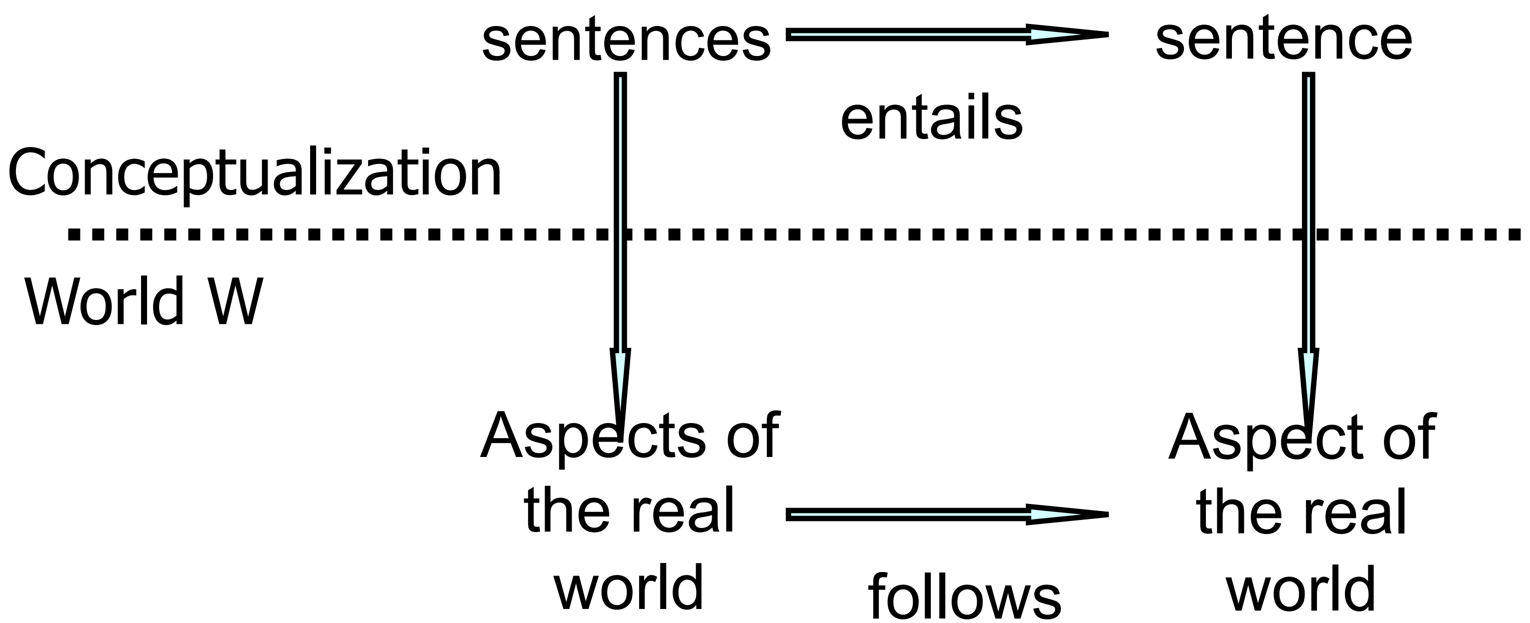
Logic is a declarative language to:

- Assert sentences representing facts that hold in a world W (these sentences are given the value true)

- Deduce the true/false values to sentences representing other aspects of W

We will examine two types of logic:
1. Propositional Logic: Simple, but not very powerful
2. First-Order Logic (aka, First-order predicate calculus): More powerful, but more complicated

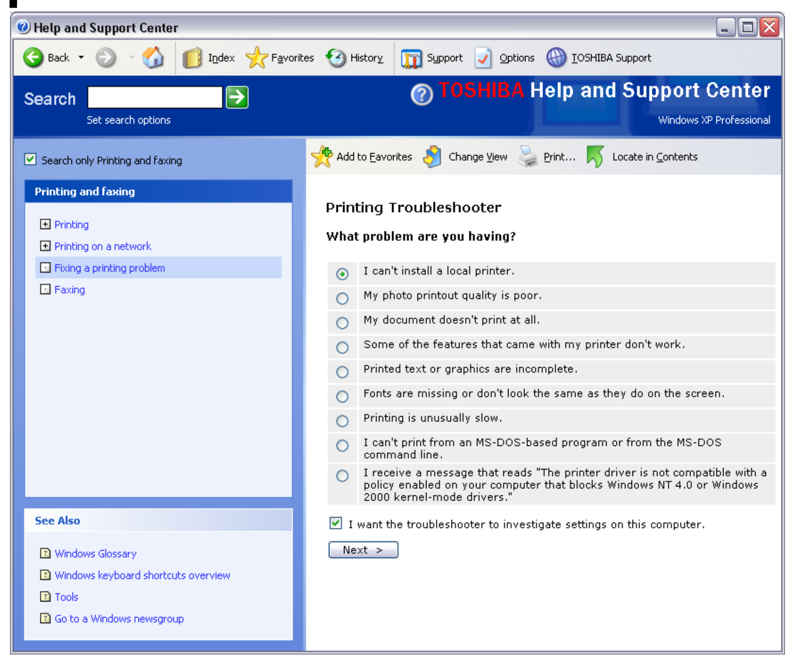
Connection World-Representation



Semantics maps sentences in logic to facts in the world. The property of one fact following from another is mirrored by the property of one sentence being entailed by another.

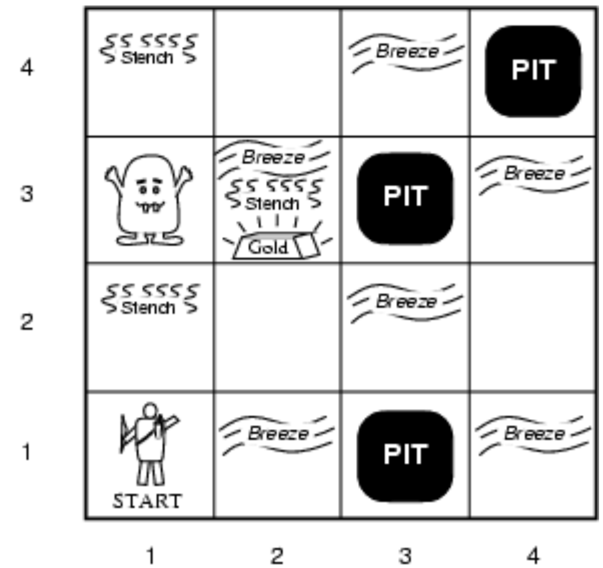
The Importance of Logical Reasoning

- At the heart of AI:
  - Explicit knowledge representation
  - Inference to deduce new knowledge
  - Close ties with Probabilistic Reasoning
- Useful for real world problems



The Wumpus World

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
  - Walking into a wall makes the agent perceive a bump
  - When the wumpus is killed, it emits a scream that is heard throughout the cave

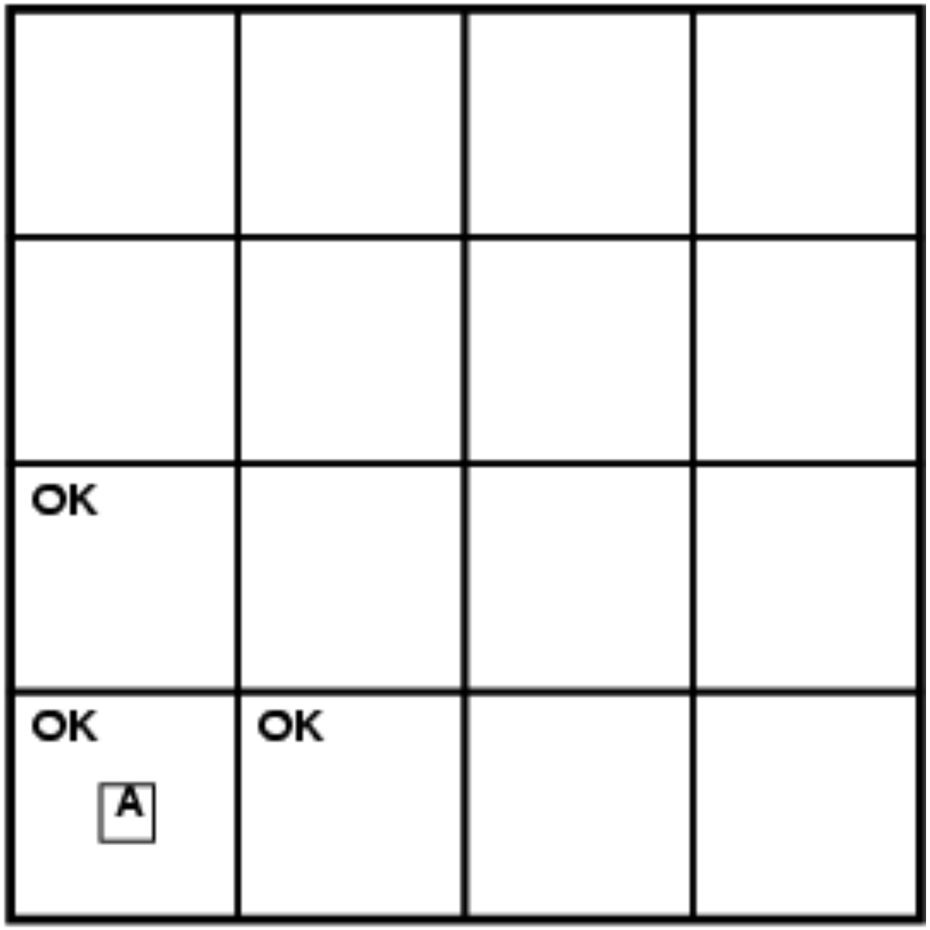


- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: turn left, turn right, move forward, Grab, Release, Shoot

Wumpus world characterization

- Fully Observable?
- Deterministic?
- Static?
- Discrete?

Exploring a wumpus world



Logic for reasoning

- Goal: Deduce new facts (α) using:
  - The rules about the world
  - Information we gather through perception

  $\Big\}$ KB

- Entailment means that one thing follows from another:

$$KB \models \alpha$$

Last night I ate either spaghetti or pizza
I did not eat pizza last night
$\models$ Last night I ate spagett

Model

- Assignment of a truth value – true or false – to every atomic sentence

$S \vee P$

$\neg P$

KB

M_1	S = False P = False
M_2	S = False P = True
M_3	S = True P = False
M_4	S = True P = True

A model m is a model of KB iff it is a model of all sentences in KB, that is, all sentences in KB are true in m

Satisfiability of a KB

A KB is satisfiable iff it admits at least one model; otherwise it is unsatisfiable

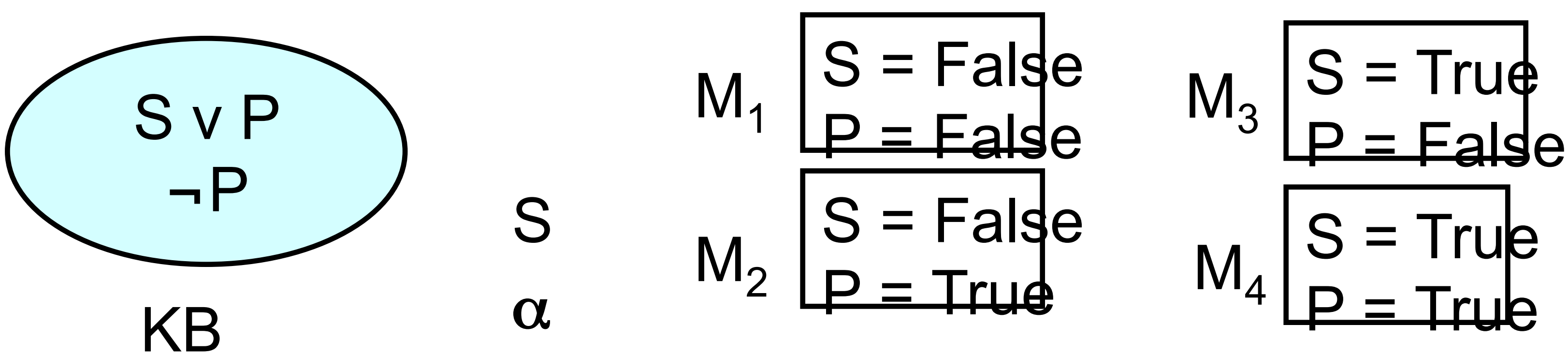
KB1 = $\{P, \neg Q \wedge R\}$ is ____________________

KB2 = $\{\neg P \vee P\}$ is ____________________

KB3 = $\{P, \neg P\}$ is ____________________

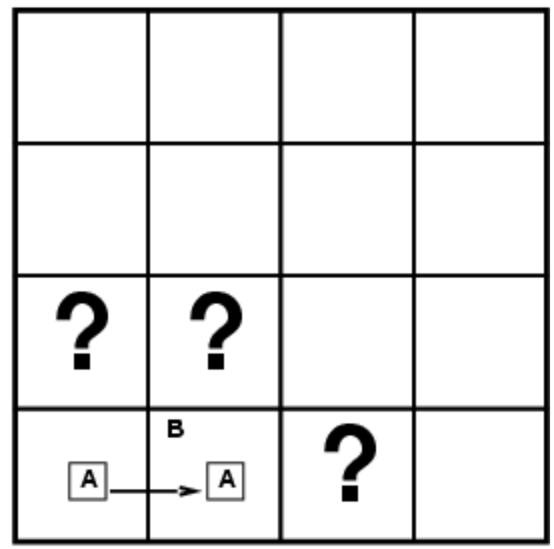
Logical Entailment

- KB : set of sentences
- α : arbitrary sentence
- KB entails α – written KB $\models \alpha$ – iff every model of KB is also a model of α
- Alternatively, KB $\models \alpha$ iff
  - $\{KB, \neg\alpha\}$ is unsatisfiable
  - KB $\Rightarrow \alpha$ is valid



Entailment in the wumpus world

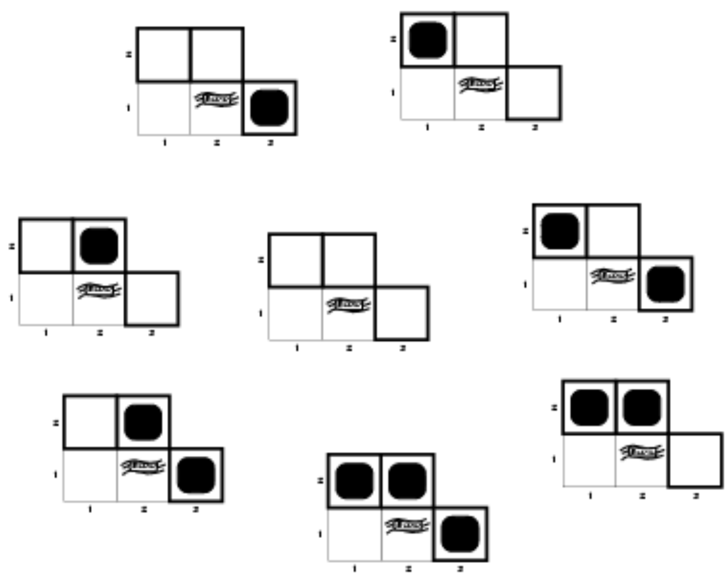
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]



Consider possible models for *KB* assuming only pits

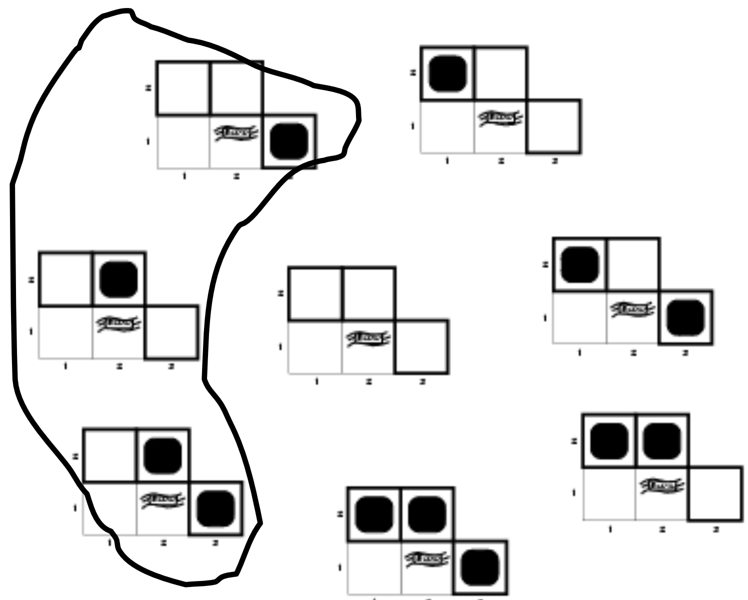
3 Boolean choices $\Rightarrow$ 8 possible models

Wumpus models



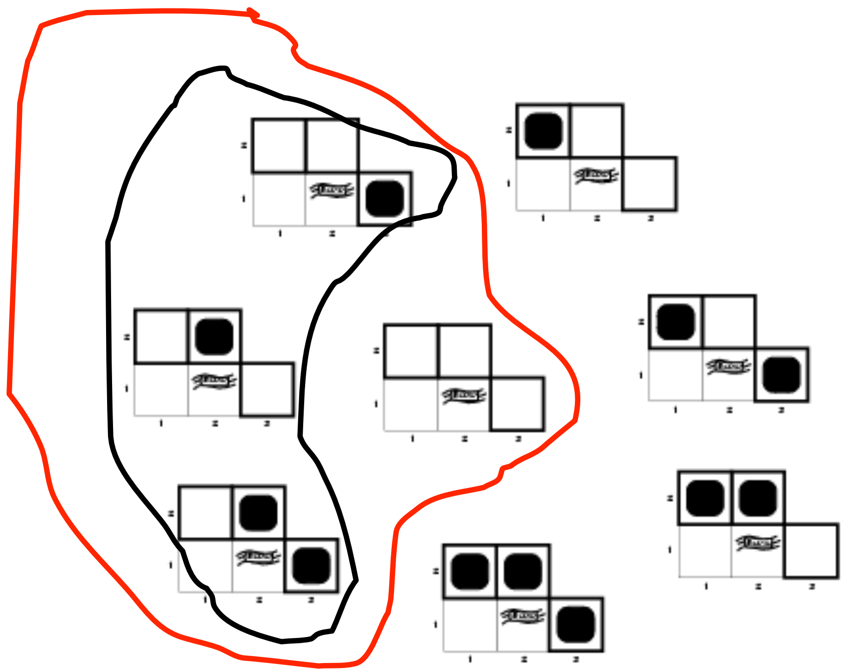
- *KB* = wumpus-world rules +2observations
- α_1 = "[1,2] is safe"

Wumpus models



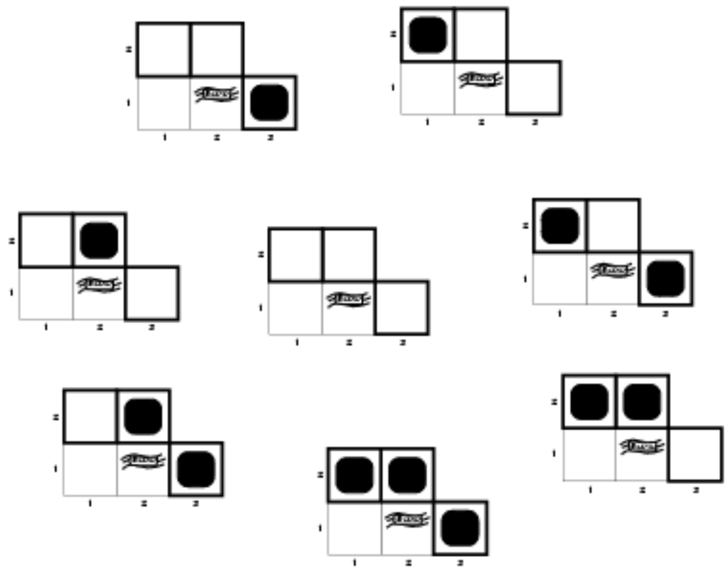
- *KB* = wumpus-world rules + 2observations
- α_1 = "[1,2] is safe"

Wumpus models



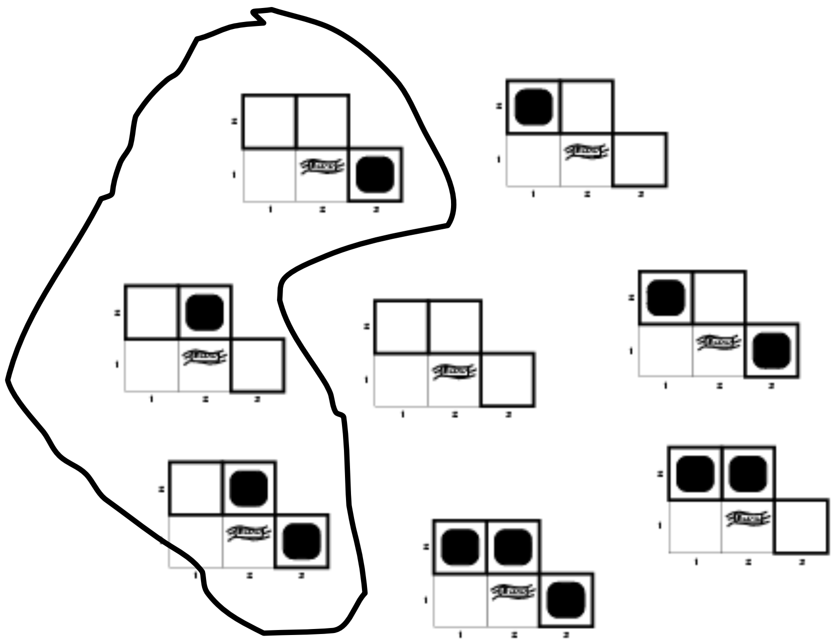
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe"

Wumpus models



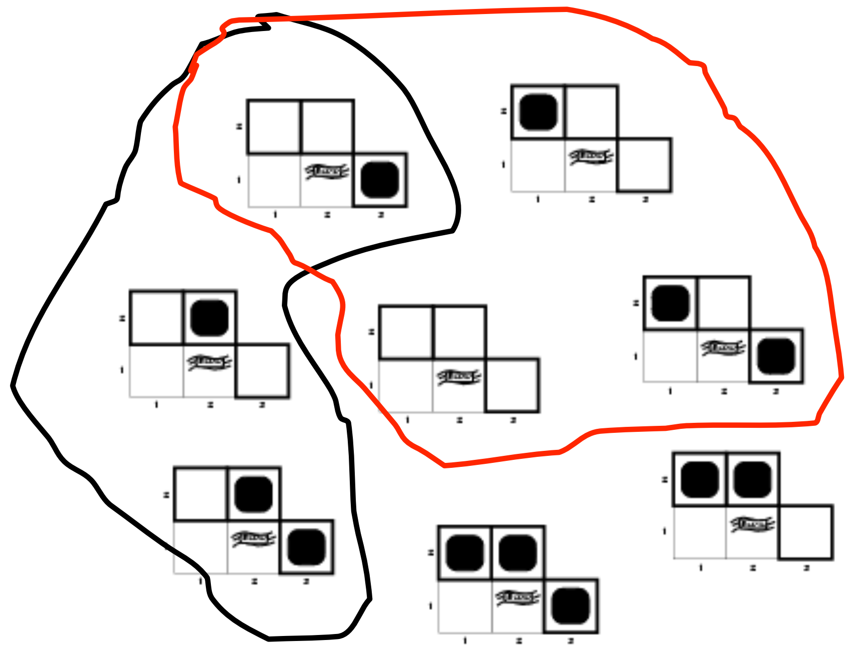
- *KB* = wumpus-world rules + 2observations
- α_2 = "[2,2] is safe"

Wumpus models



- *KB* = wumpus-world rules + 2observations
- α_2 = "[2,2] is safe"

Wumpus models



- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe"

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols P_1, P_2 etc are sentences

  - If S is a sentence, $\neg S$ is a sentence (negation)
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	false	true	false

With these symbols, 8 possible models, can be enumerated automatically.

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
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true and	$S_2 \Rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{true} \vee \mathit{false}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

Inference

- Just because a KB entails a sentence doesn't mean we can find (infer) it
- Inference is the process of generating sentences entailed by the KB

Inference Rule

- An inference rule $\{\xi, \psi\} \vdash \varphi$ consists of 2 sentence patterns ξ and ψ called the **conditions** and one sentence pattern φ called the **conclusion**
- If ξ and ψ match two sentences of KB then the corresponding φ can be inferred according to the rule

Example: Modus Ponens

$$\{\ \alpha \Rightarrow \beta\ ,\ \alpha\ \}\ \vdash\ \beta$$

$$\{\xi,\ \psi\} \vdash \varphi$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work ($\alpha \Rightarrow \beta$)
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK (α)

$\Rightarrow$ Connective symbol (implication)

$\models$ Logical entailment

KB $\models \alpha$ iff KB $\Rightarrow \alpha$ is valid

$\vdash$ Inference

Soundness

- An inference rule is sound if it generates only entailed sentences
- All inference rules previously given are sound, e.g.:
  modus ponens: $\{\alpha \Rightarrow \beta , \alpha\} \vdash \beta$
- Is the following rule sound?

  $\{\alpha \Rightarrow \beta , \beta\} \vdash \alpha$

Completeness

- A set of inference rules is complete if every entailed sentences can be obtained by applying some finite succession of these rules
- Modus ponens *alone* is not complete, e.g.:
  from $A \Rightarrow B$ and $\neg B$, we cannot get $\neg A$

Proof

The proof of a sentence α from a set of sentences KB is the derivation of α by applying a series of sound (legal) inference rules

Proof

The **proof** of a sentence α from a set of sentences KB is the derivation of α by applying a series of sound inference rules

1. Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work
2. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
3. Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. Battery-OK ∧ Starter-OK ← (5+6)
10. Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ← (9+7)
11. Engine-Starts ← (2+10)
12. Engine-Starts ⇒ Flat-Tire ← (3+8)
13. Flat-Tire ← (11+12)

Inference Problem

- Given:
  - KB: a set of sentence
  - α: a sentence
- Answer:
  - KB $\models \alpha$?

We require an automatic, sound and complete inference method

Inference by enumeration

- We can enumerate all possible models and test whether every model of KB is also a model of α

Wumpus world sentences

Percepts:

Let $P_{i,j}$ be true if there is a pit in [i, j]

Let $B_{i,j}$ be true if there is a breeze in [i, j]

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

Rules of the environment:

"Pits cause breezes in adjacent squares"

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
$false$	$false$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$false$	$false$	$false$	$false$	$false$	$true$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$false$	$true$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$true$	$false$	$false$	$false$	$false$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$false$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$true$	$false$	$false$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$false$	$false$

¬ P1,1

¬B1,1

B2,1

B1,1 ⇔ (P1,2 v P2,1)

B2,1 ⇔ (P1,1 v P2,2 v P3,1)

α_1 $\neg P_{1,2}$

Inference by enumeration

- How efficient is this? How much time/space does it take?

Proof methods

- Proof methods divide into (roughly) two kinds:

  – Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
      Can use inference rules as operators in a standard search algorithm
    - Typically require transformation of sentences into a normal form

  – Model checking
    - truth table enumeration (always exponential in *n*)
    - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms

Resolution Inference Rule

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals

clauses

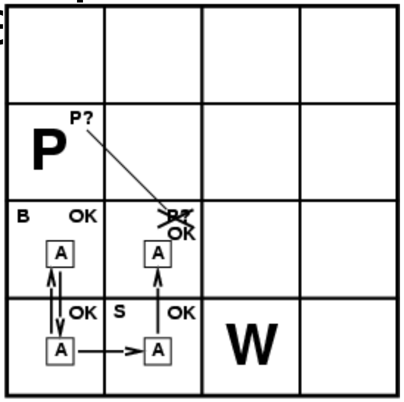
E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Resolution inference rule (for CNF):

$$\frac{l_i \vee \ldots \vee l_k, \qquad m_1 \vee \ldots \vee m_n}{l_i \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary lite

E.g., $\dfrac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$



- Resolution is sound and complete for propositional logic

Resolution Refutation Algorithm

RESOLUTION-REFUTATION(KB,α)

clauses ← set of clauses obtained from KB and $\neg\alpha$

new ← {}

Repeat:

For each C, C' in clauses do

res ← RESOLVE(C,C')

If res contains the empty clause then return yes

new ← new U res

If new ⊆ clauses then return no

clauses ← clauses U new

Example

1. ¬Battery-OK ∨ ¬Bulbs-OK ∨ Headlights-Work
2. ¬Battery-OK ∨ ¬Starter-OK ∨ Empty-Gas-Tank ∨ Engine-S
3. ¬Engine-Starts ∨ Flat-Tire ∨ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. ¬Empty-Gas-Tank
8. ¬Car-OK
9. ¬Flat-Tire

Forward and backward chaining

- 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 (for Horn Form): complete for Horn KBs

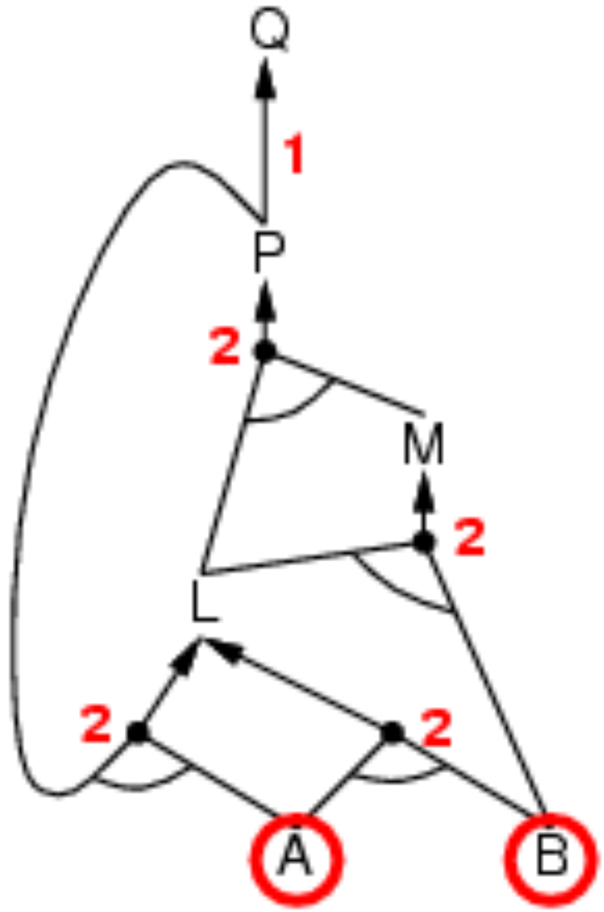
$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

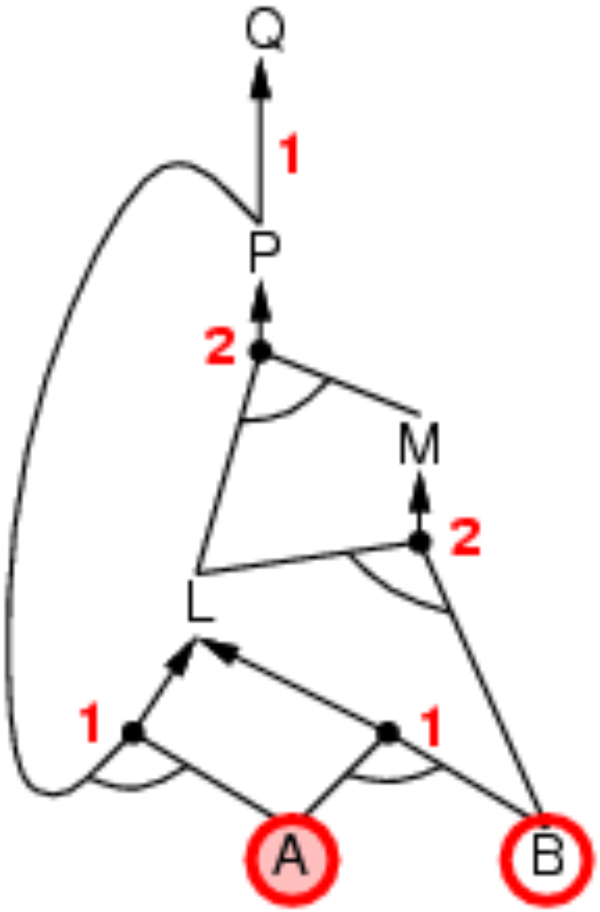
Forward chaining

- Idea: fire any rule whose premises are satisfied in the *KB*,
  - add its conclusion to the *KB*, until query is found

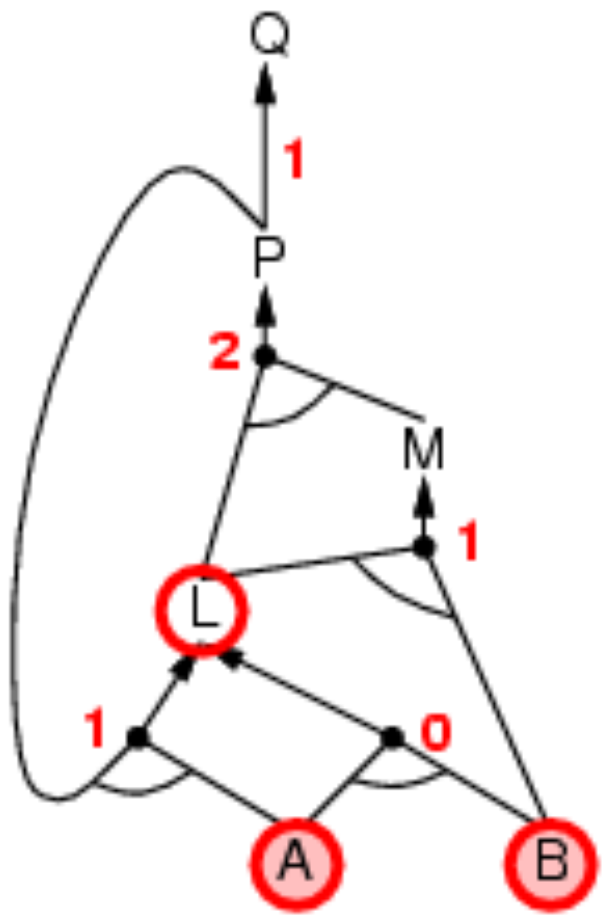
Forward chaining example



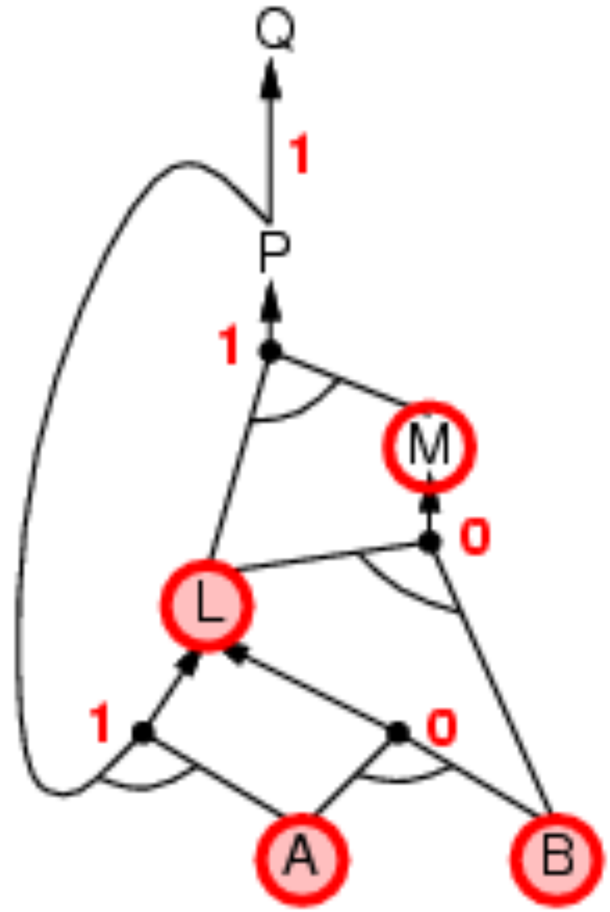
Forward chaining example



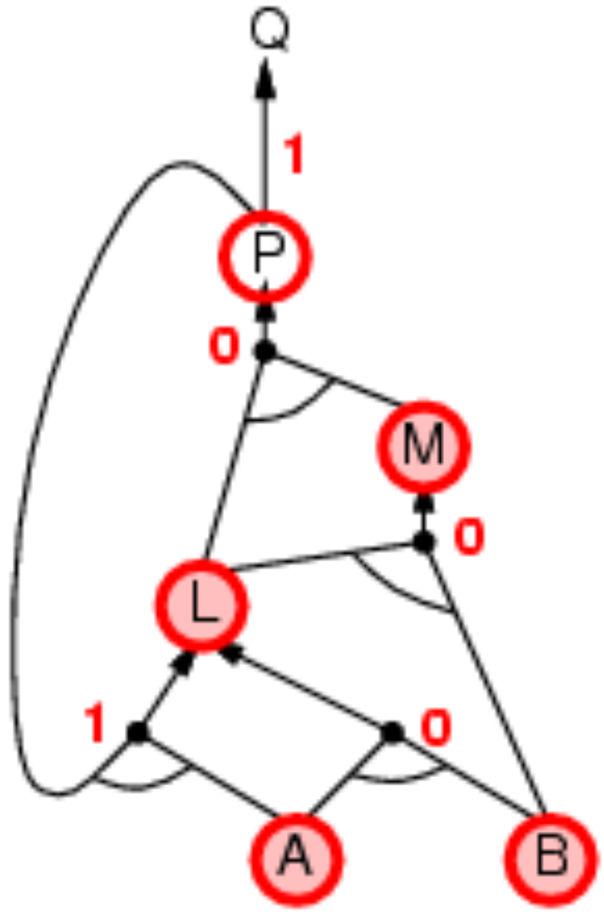
Forward chaining example



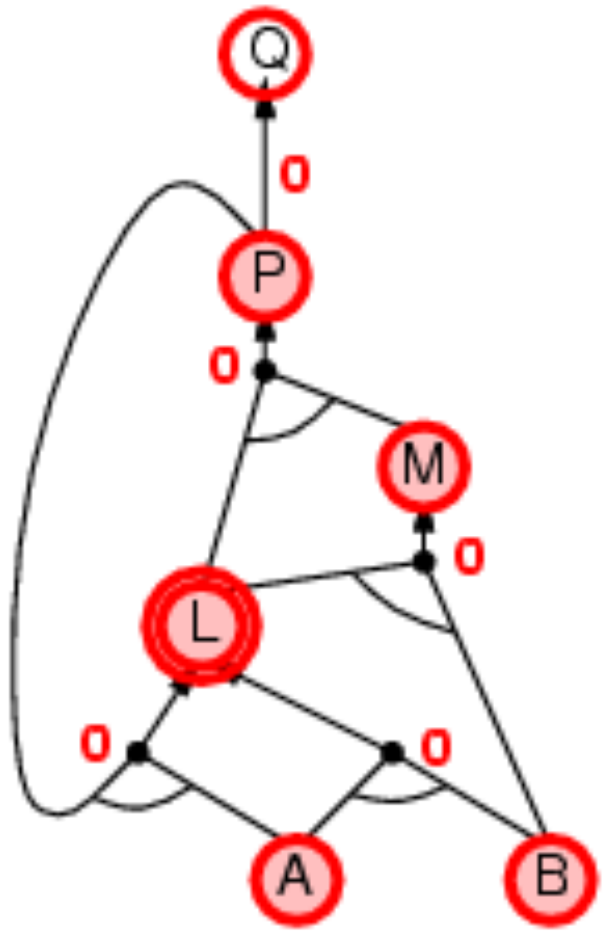
Forward chaining example



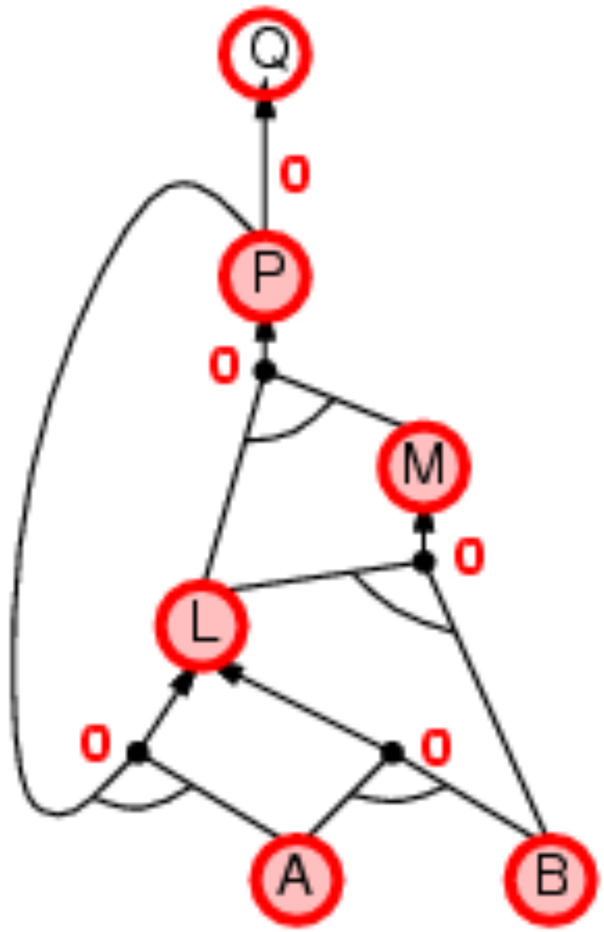
Forward chaining example



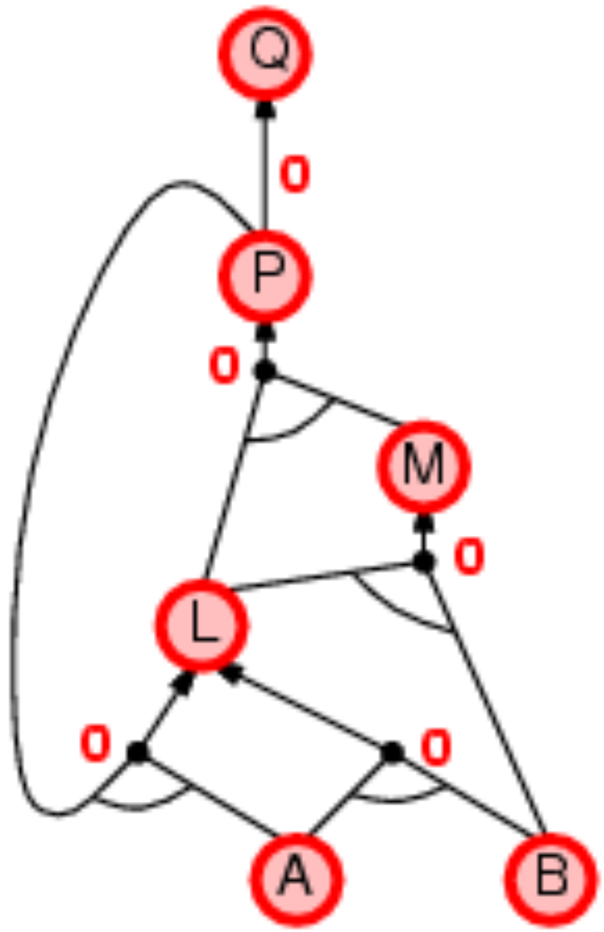
Forward chaining example



Forward chaining example



Forward chaining example



Backward chaining

Idea: work backwards from the query *q*:

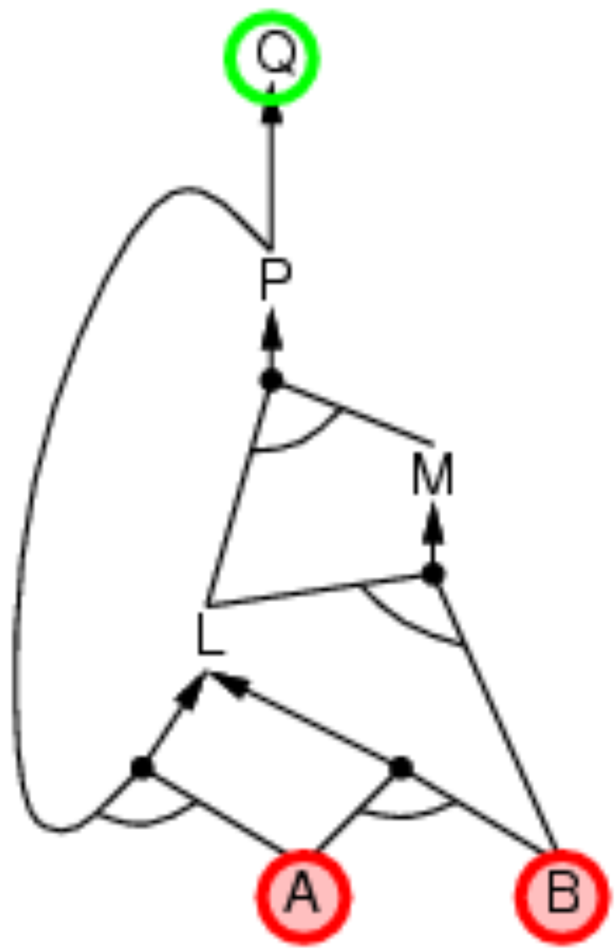
- to prove *q* by BC,
  - check if *q* is known already, or
  - prove by BC all premises of some rule concluding *q*

Avoid loops: check if new subgoal is already on the goal stack

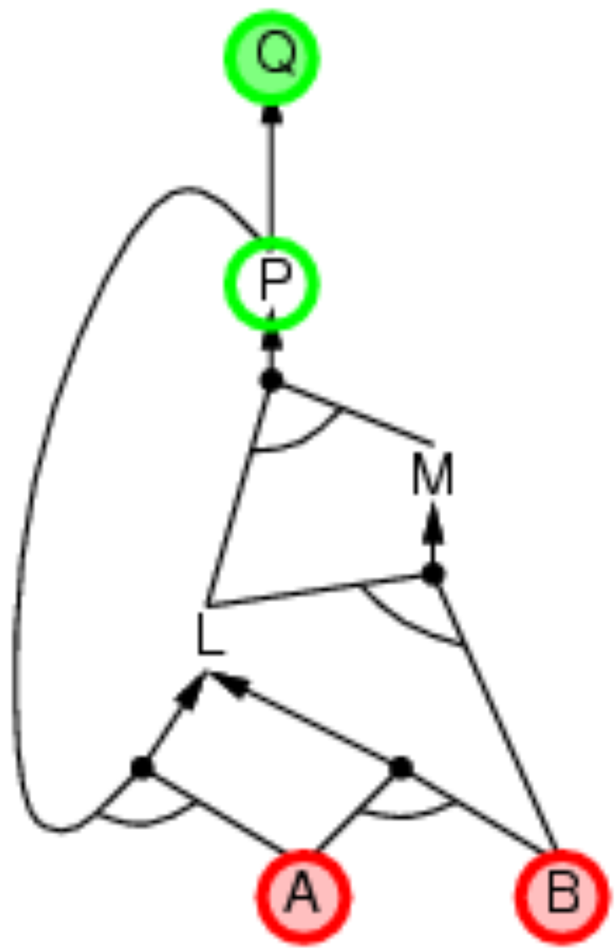
Avoid repeated work: check if new subgoal

1. has already been proved true, or
2. has already failed

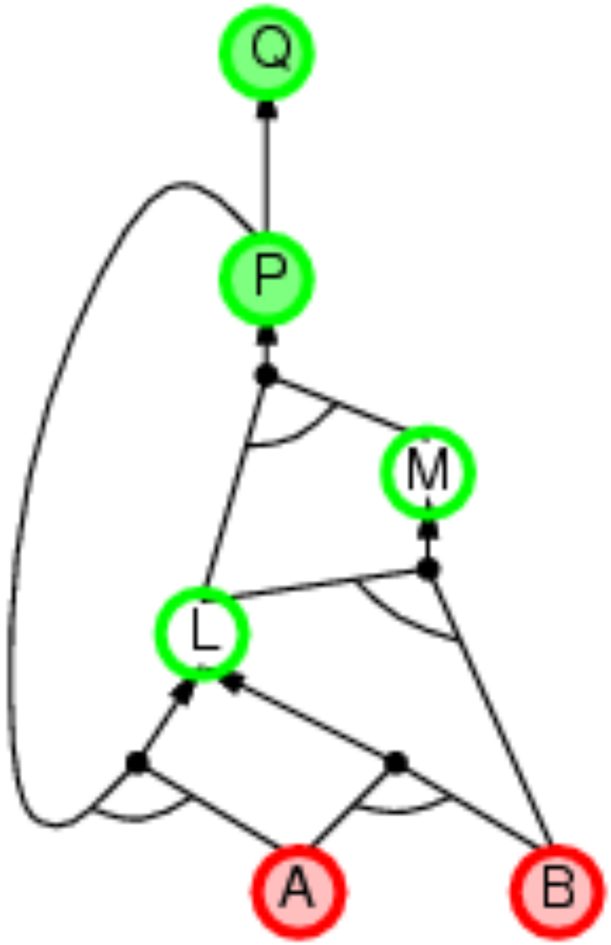
Backward chaining example



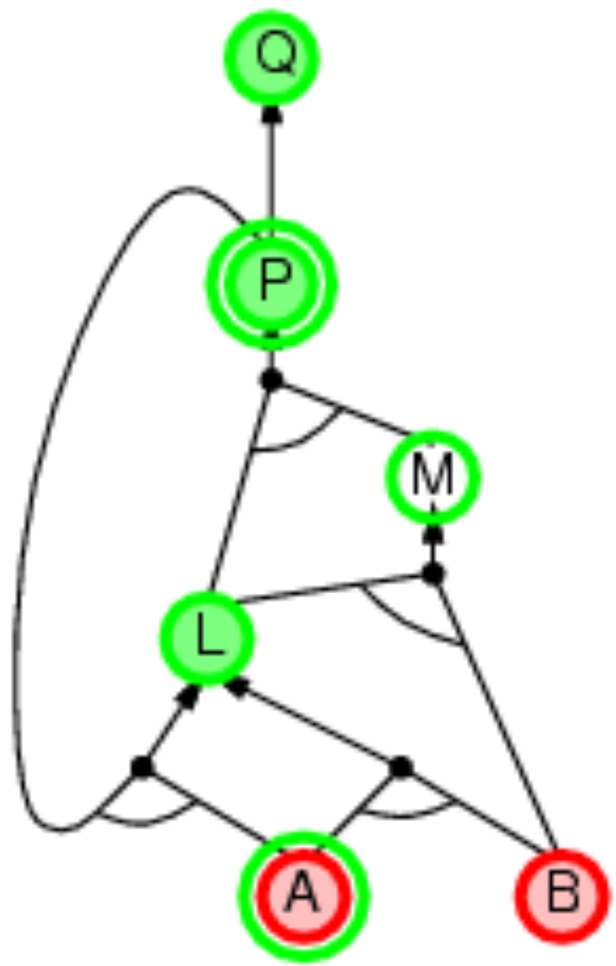
Backward chaining example



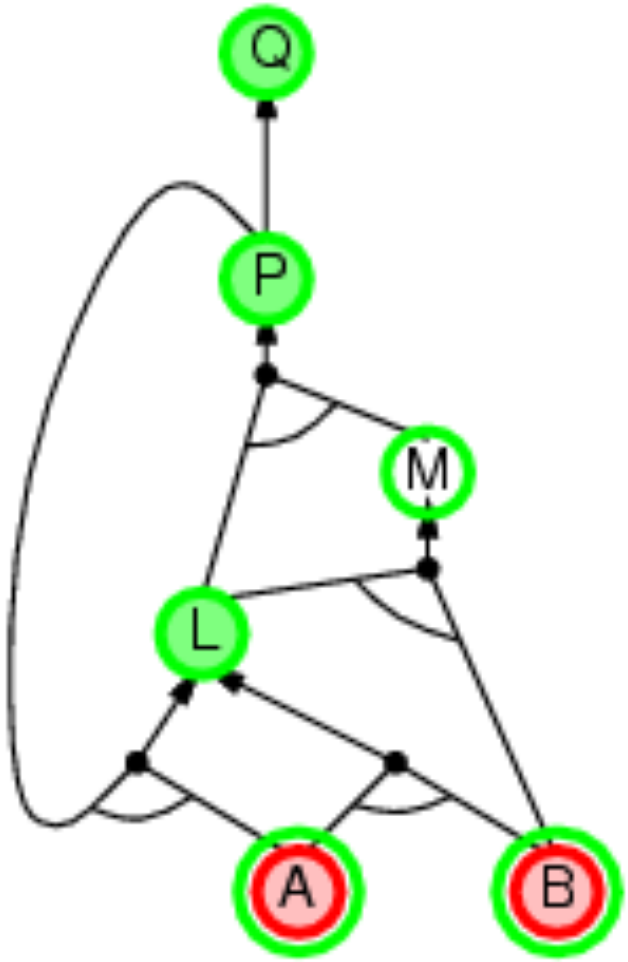
Backward chaining example



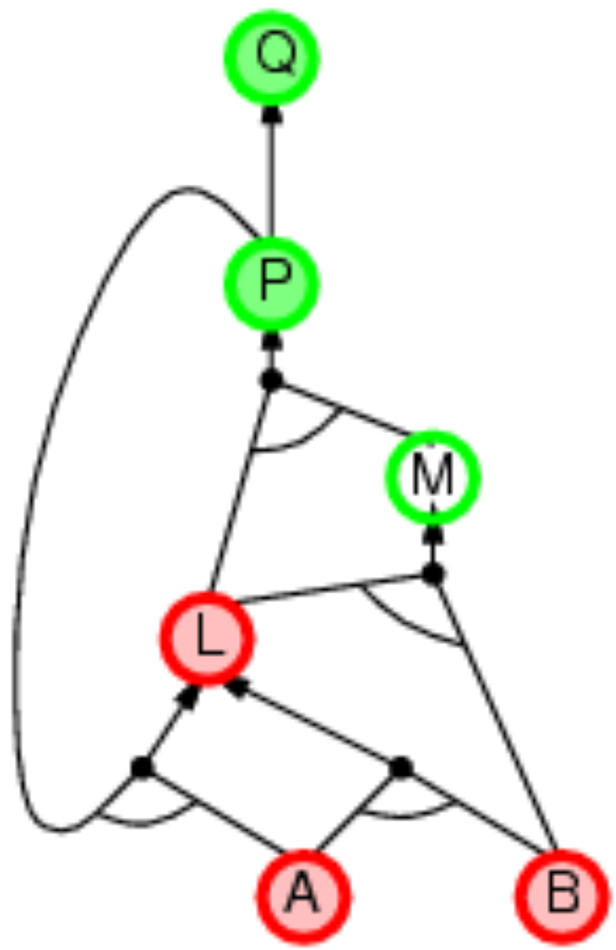
Backward chaining example



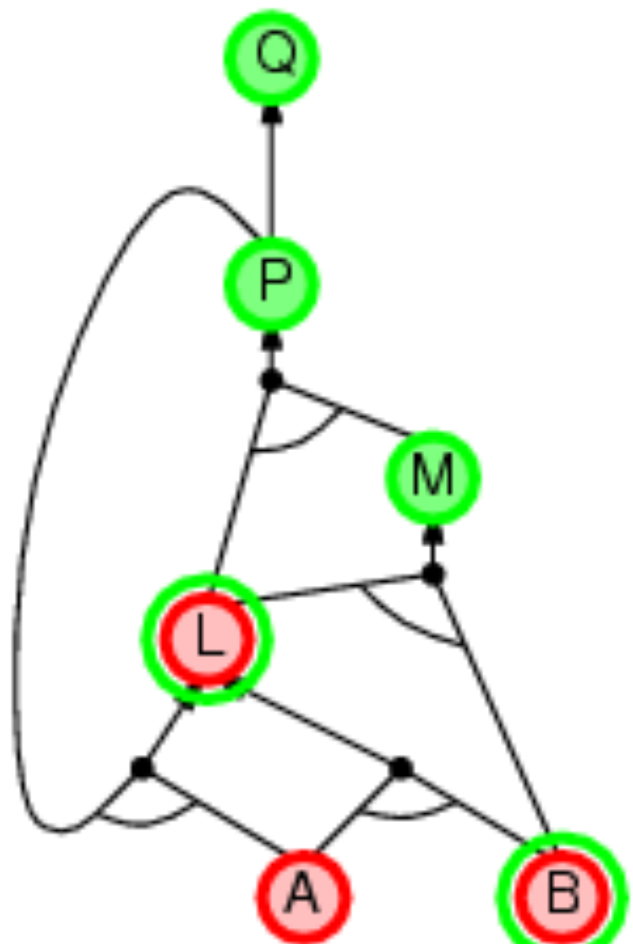
Backward chaining example



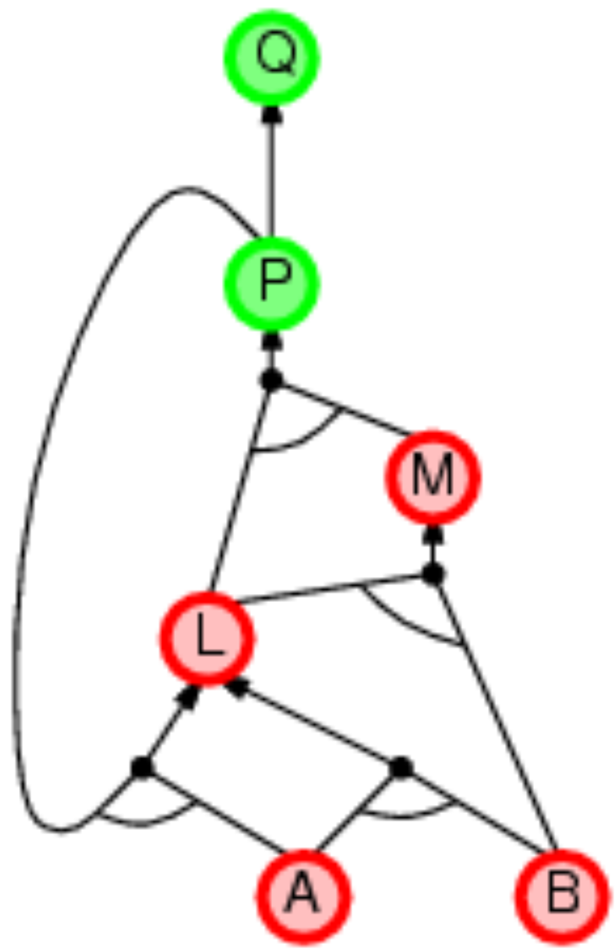
Backward chaining example



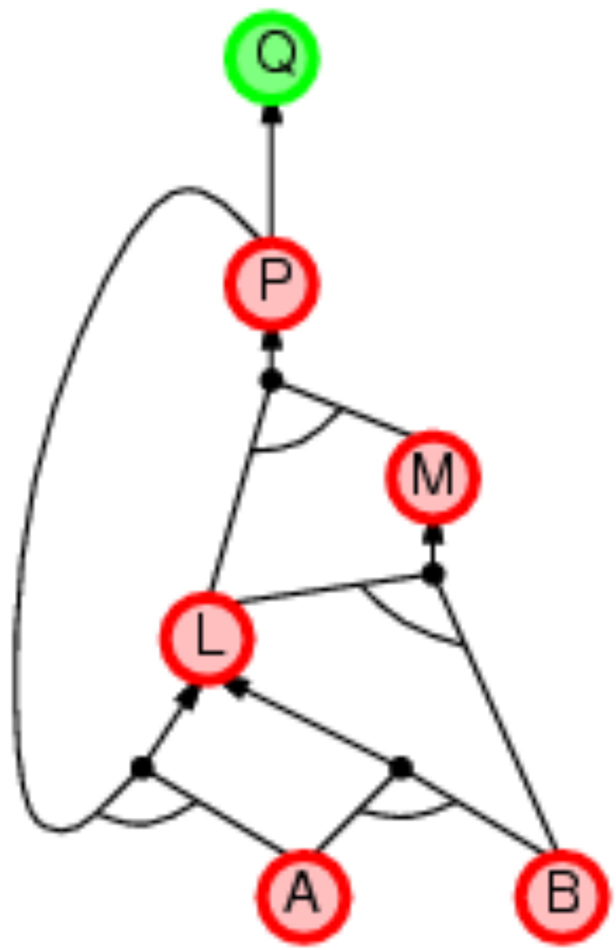
Backward chaining example



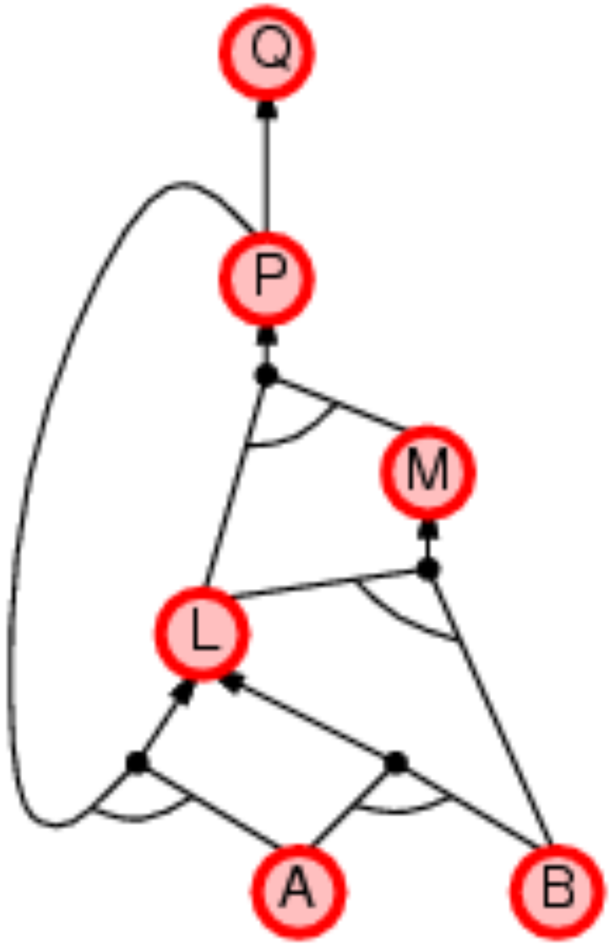
Backward chaining example



Backward chaining example



Backward chaining example



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,
  - e.g., Where are my keys? How do I get into a PhD program?

- Complexity of BC can be much less than linear in size of KB