Artificial Intelligence
Chapter 7: Logical Agents

Michael Scherger
Department of Computer Science
Kent State University

Contents

- Knowledge Based Agents
- Wumpus World
- Logic in General – models and entailment
- Propositional (Boolean) Logic
- Equivalence, Validity, Satisfiability
- Inference Rules and Theorem Proving
  - Forward Chaining
  - Backward Chaining
  - Resolution

Logical Agents

- Humans can know “things” and “reason”
  - Representation: How are the things stored?
  - Reasoning: How is the knowledge used?
    - To solve a problem…
    - To generate more knowledge…

- Knowledge and reasoning are important to artificial agents because they enable successful behaviors difficult to achieve otherwise
  - Useful in partially observable environments

- Can benefit from knowledge in very general forms, combining and recombining information

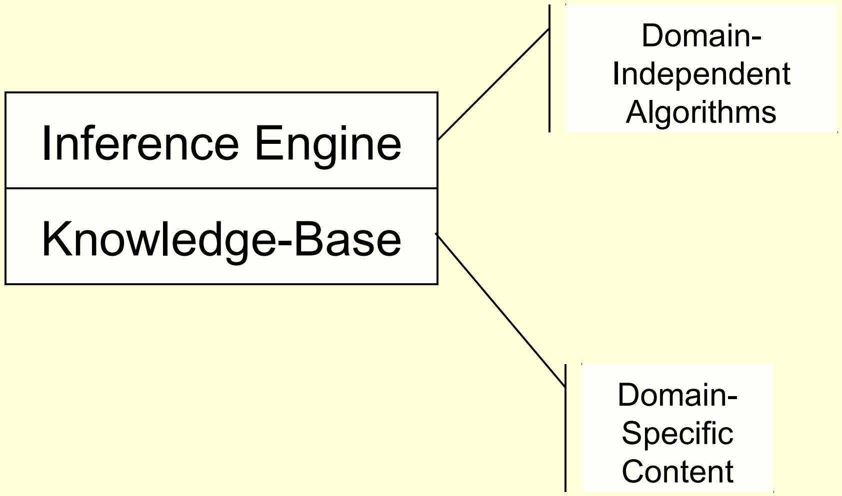
Knowledge-Based Agents

- Central component of a Knowledge-Based Agent is a ***Knowledge-Base***
  - A set of sentences in a formal language
    - Sentences are expressed using a knowledge representation language

- Two generic functions:
  - TELL - add new sentences (facts) to the KB
    - "Tell it what it needs to know"
  - ASK - query what is known from the KB
    - "Ask what to do next"

Knowledge-Based Agents

- The agent must be able to:
  - Represent states and actions
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions



Knowledge-Based Agents

```
function KB-AGENT(percept) returns an action
   static: KB, a knowledge base
           t, a counter, initially 0, indicating time

   TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
   action ← ASK(KB, MAKE-ACTION-QUERY(t))
   TELL(KB, MAKE-ACTION-SENTENCE(action, t))
   t ← t + 1
   return action
```

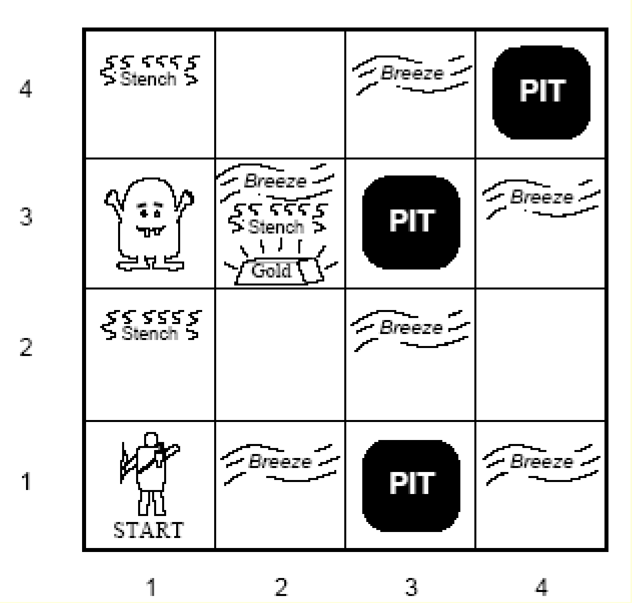
Knowledge-Based Agents

- Declarative
  - You can build a knowledge-based agent simply by "TELLing" it what it needs to know

- Procedural
  - Encode desired behaviors directly as program code
    - Minimizing the role of explicit representation and reasoning can result in a much more efficient system

Wumpus World

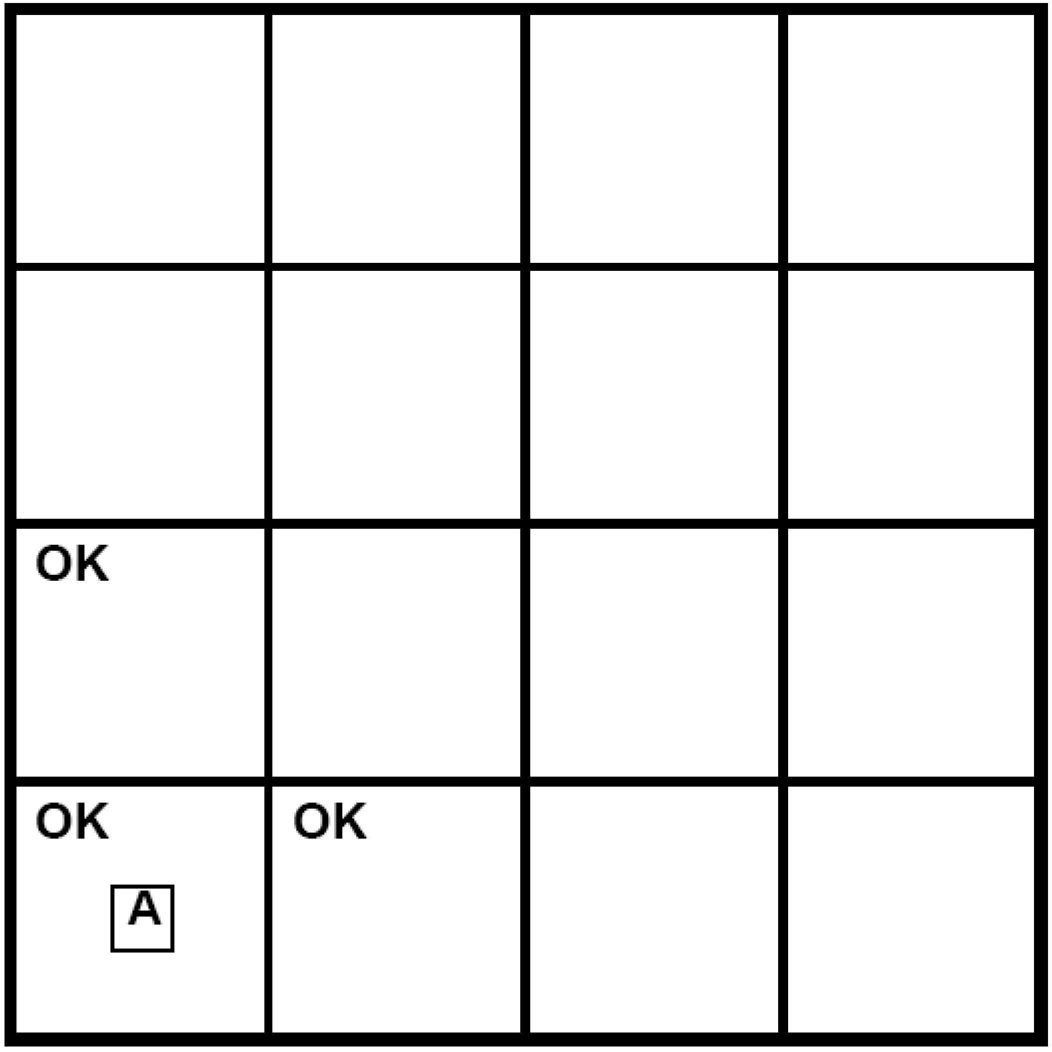
- Performance Measure
  - Gold +1000, Death – 1000
  - Step -1, Use arrow -10
- Environment
  - Square adjacent to the Wumpus are smelly
  - Squares adjacent to the pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills Wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up the gold if in the same square
  - Releasing drops the gold in the same square
- Actuators
  - Left turn, right turn, forward, grab, release, shoot
- Sensors
  - Breeze, glitter, and smell
- See page 197-8 for more details!



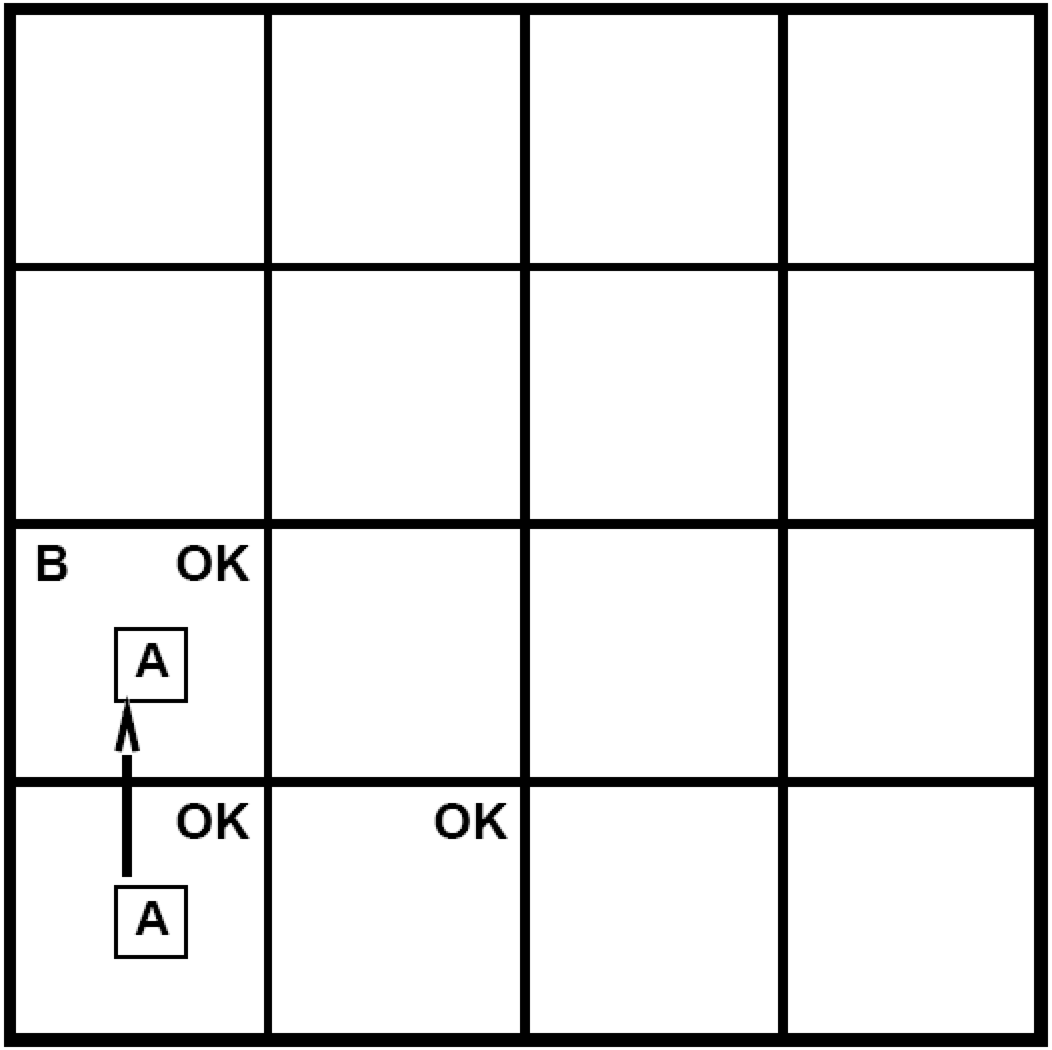
Wumpus World

- Characterization of Wumpus World
  - Observable
    - partial, only local perception
  - Deterministic
    - Yes, outcomes are specified
  - Episodic
    - No, sequential at the level of actions
  - Static
    - Yes, Wumpus and pits do not move
  - Discrete
    - Yes
  - Single Agent
    - Yes

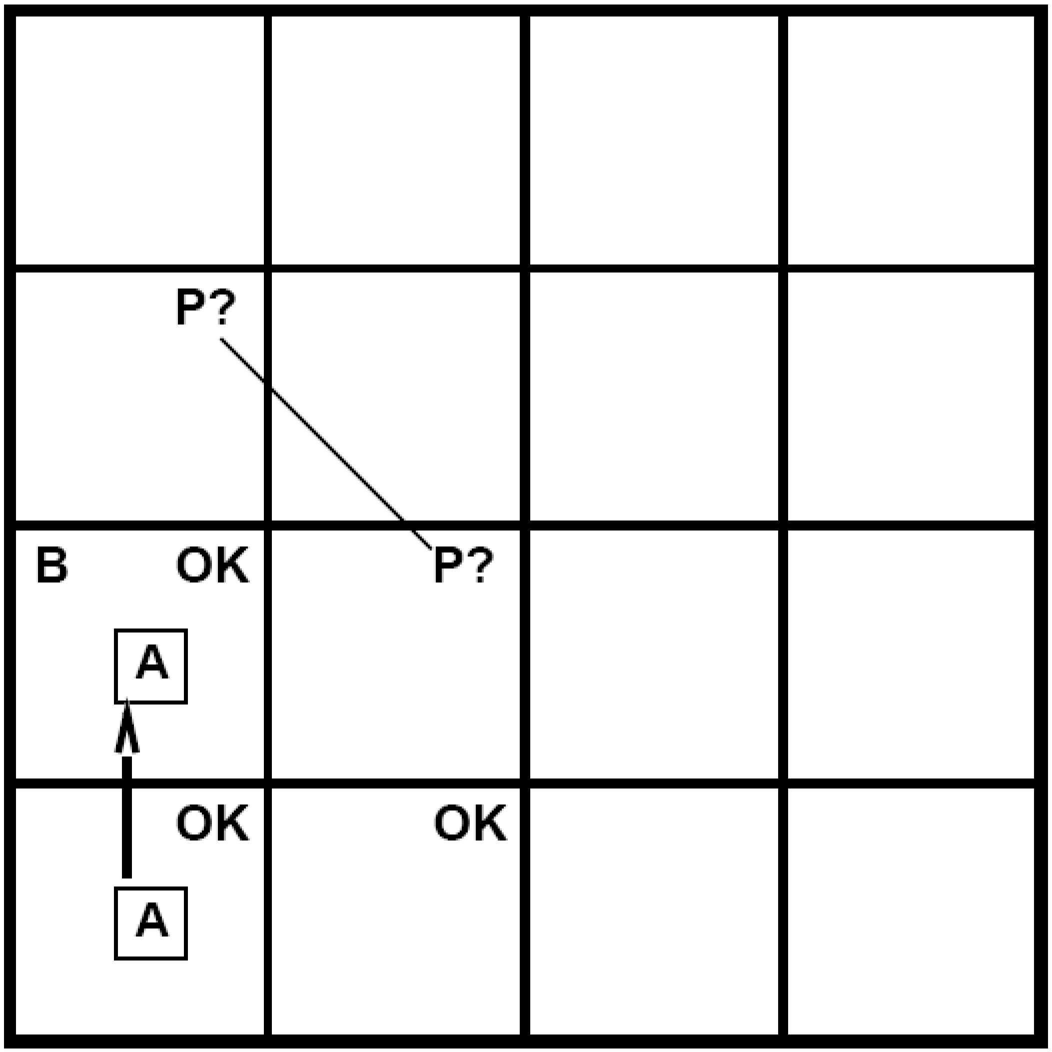
Wumpus World



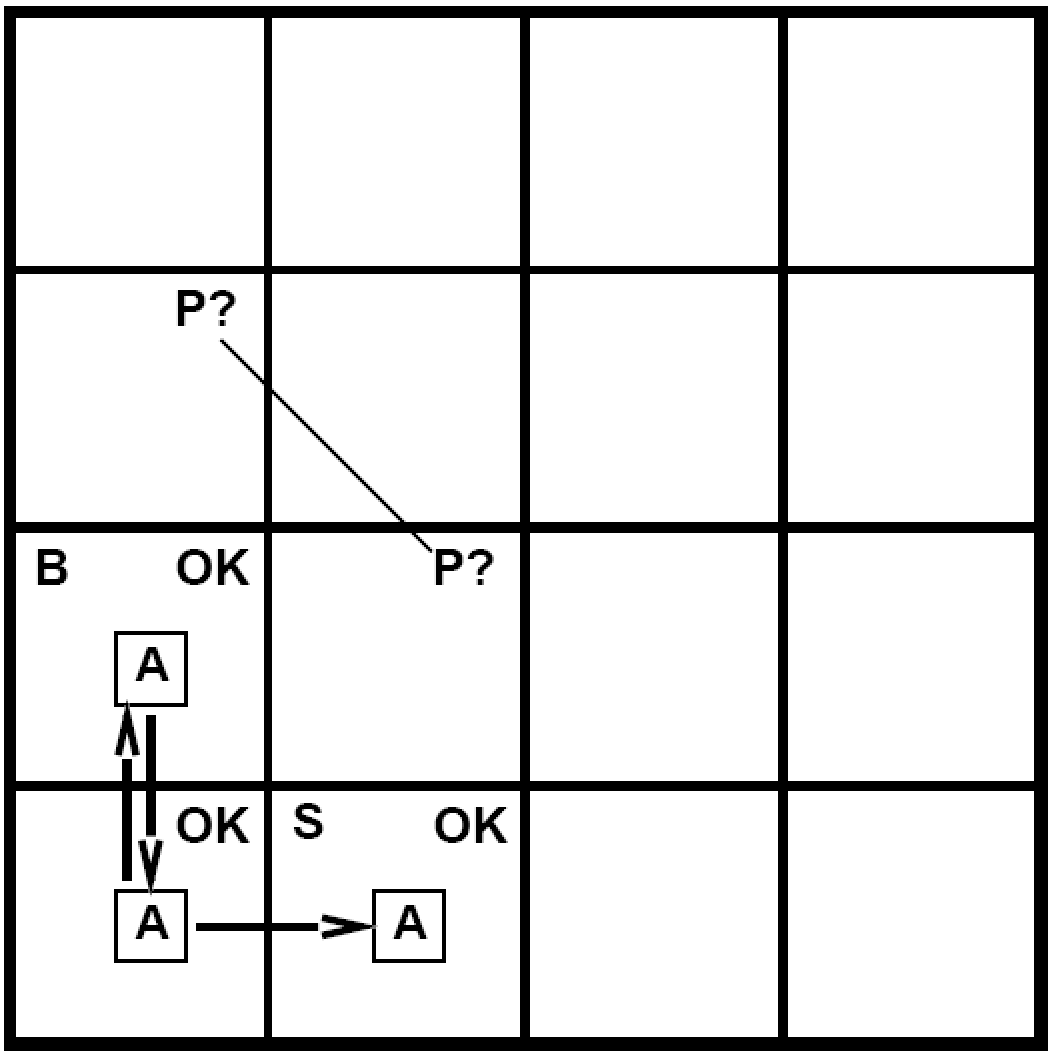
Wumpus World



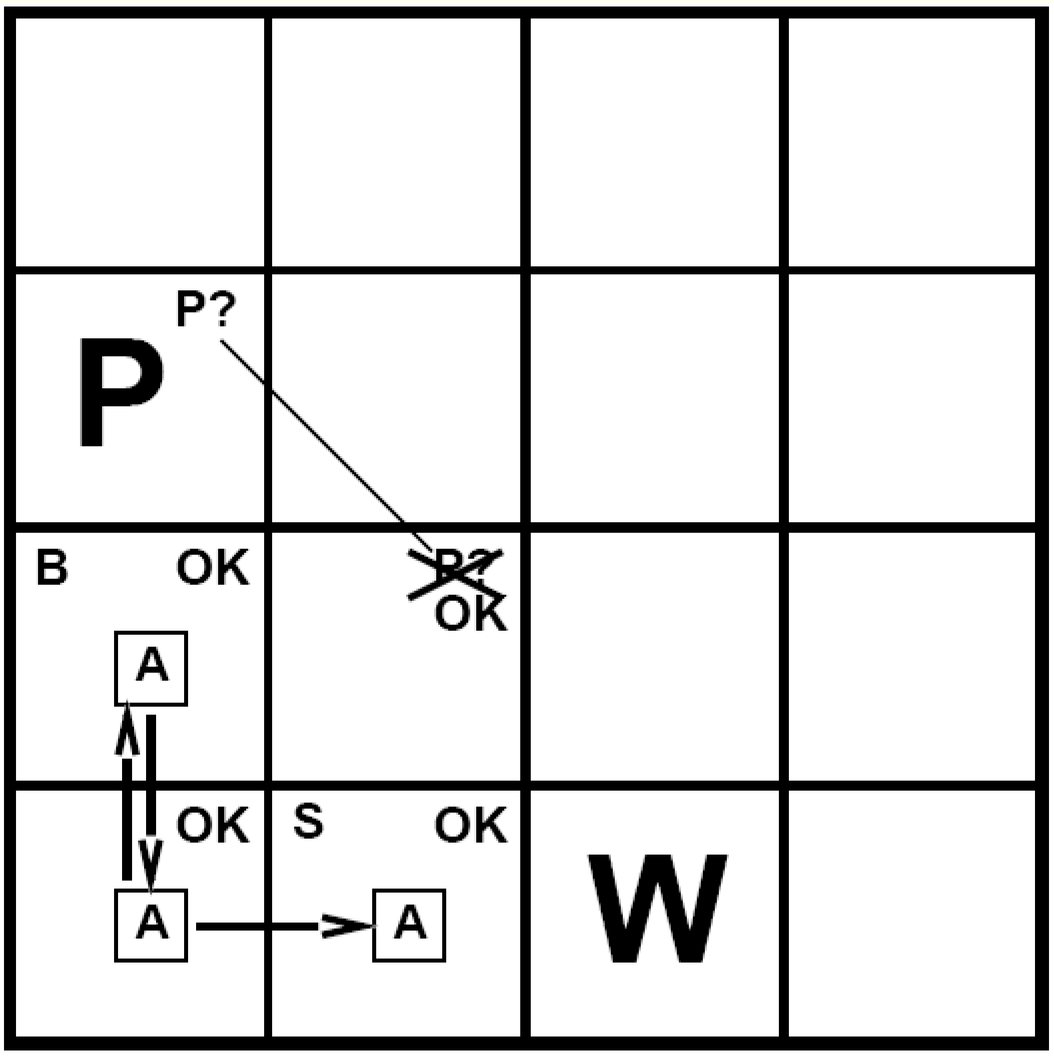
Wumpus World



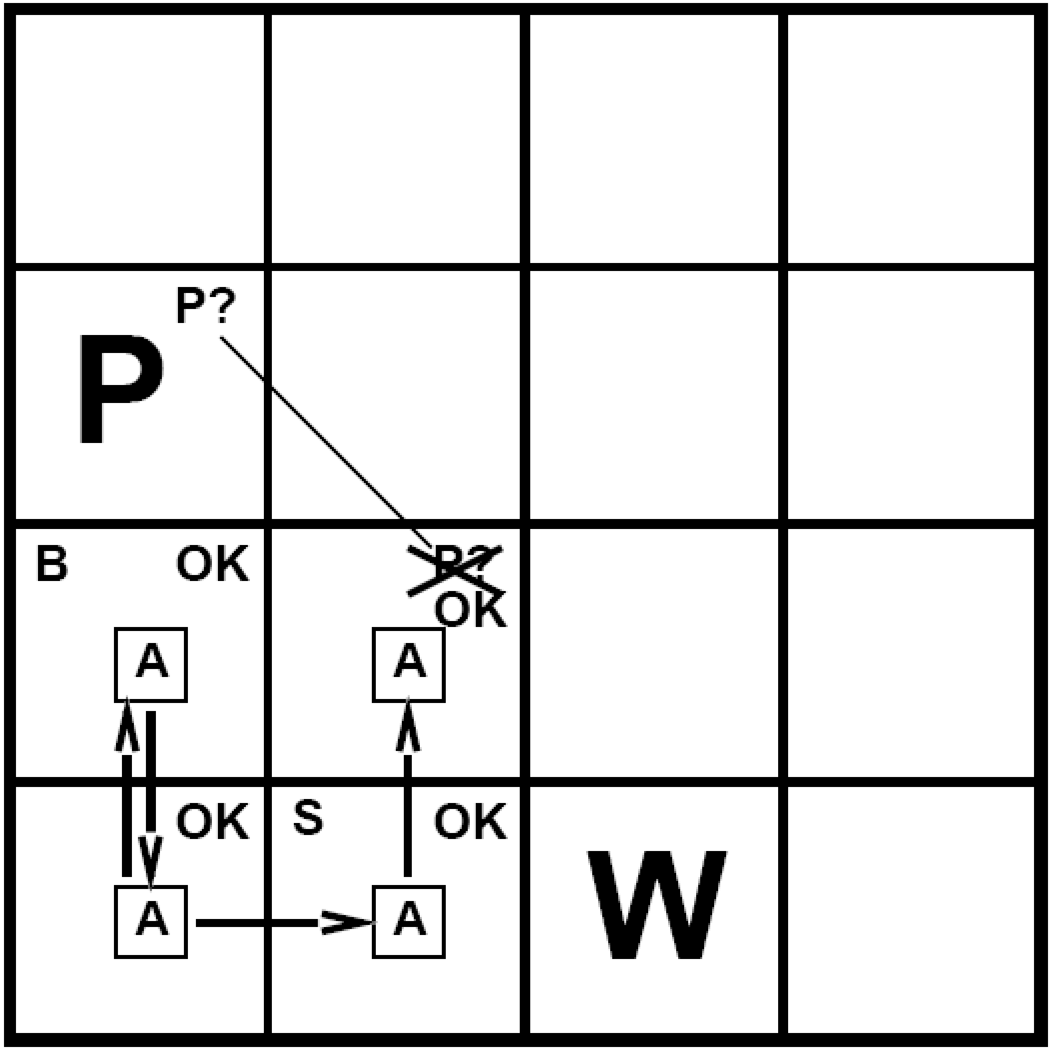
Wumpus World



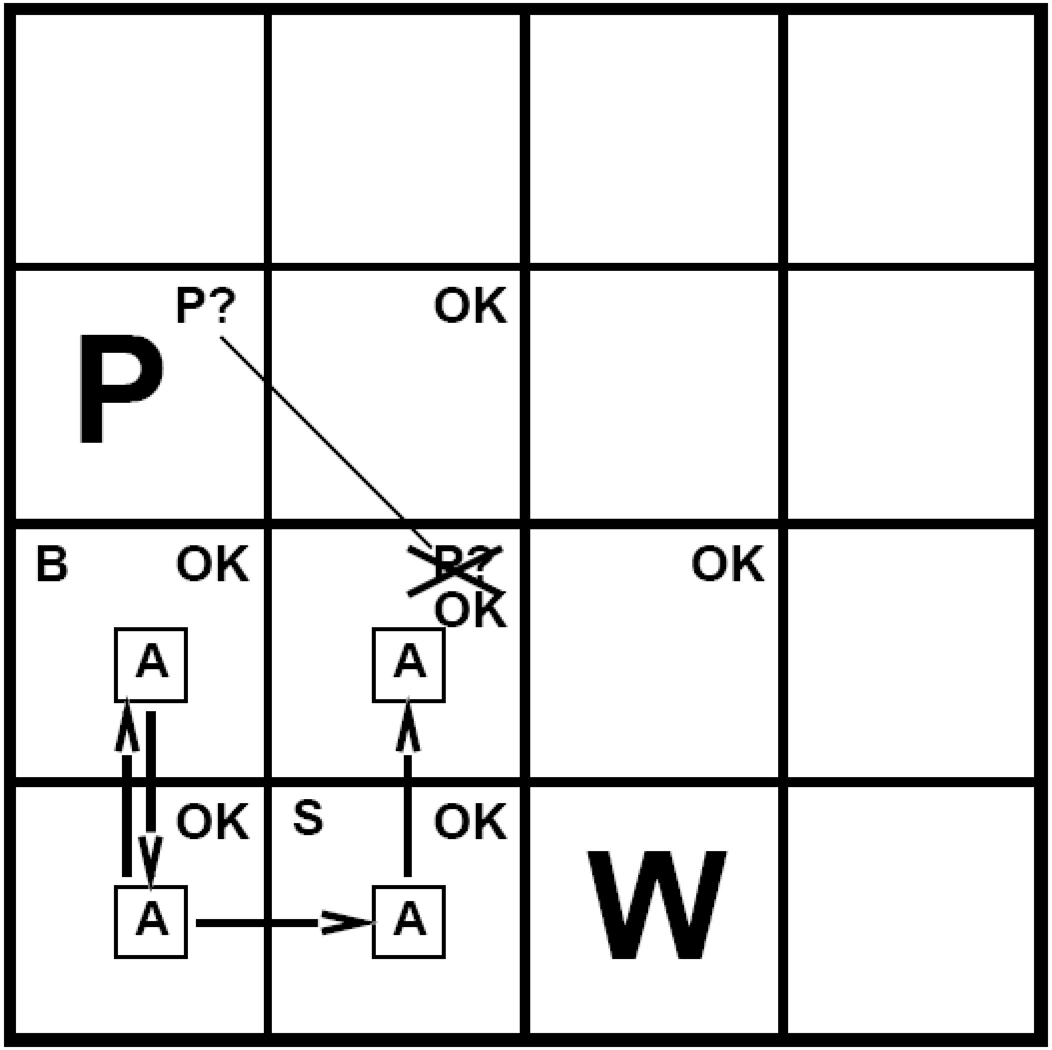
Wumpus World



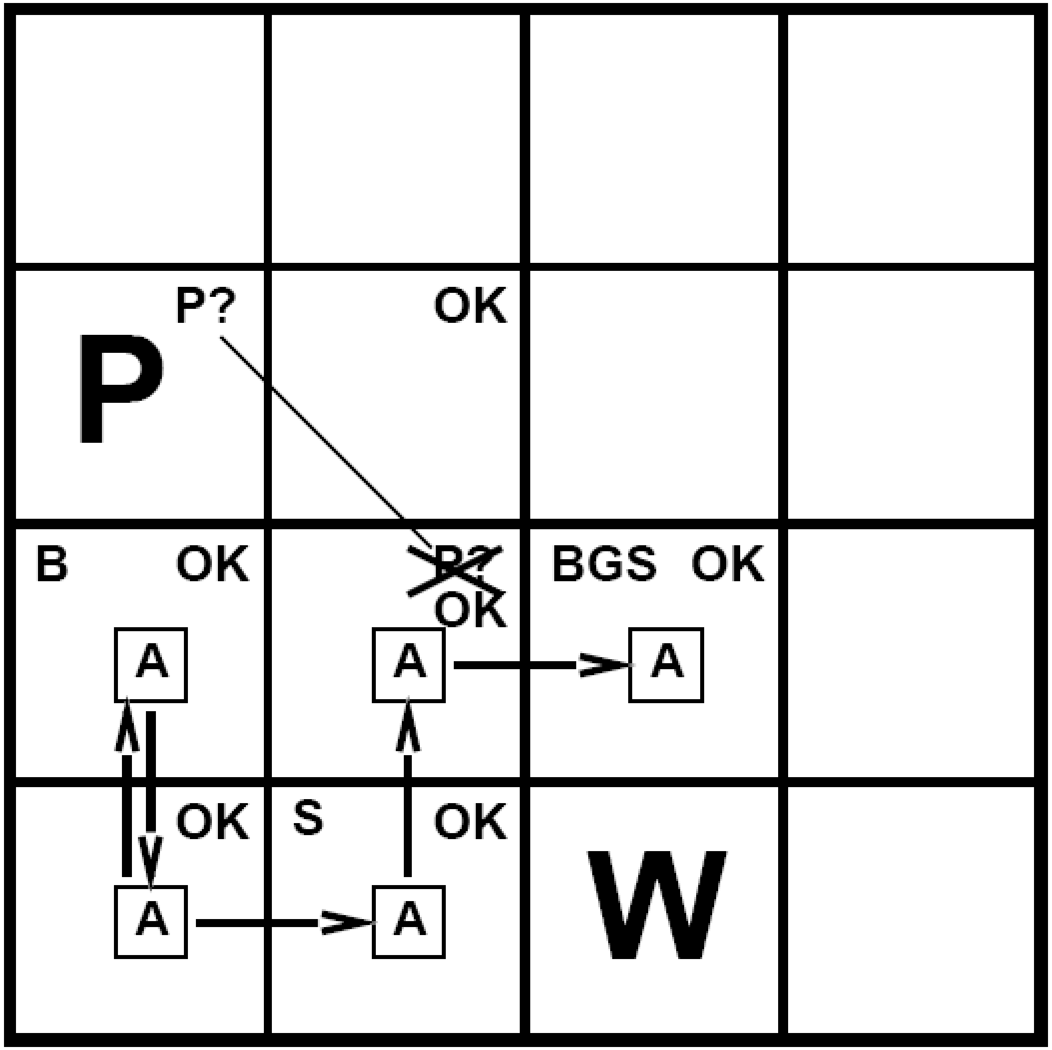
Wumpus World



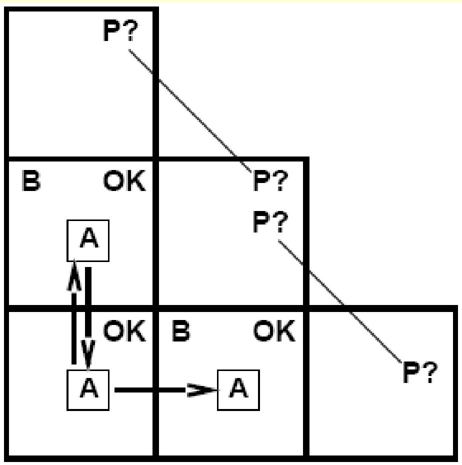
Wumpus World

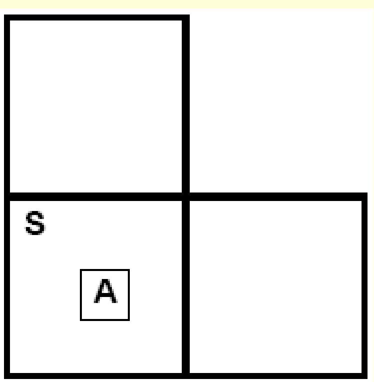


Wumpus World



Other Sticky Situations





- Breeze in (1,2) and (2,1)
  - No safe actions
- Smell in (1,1)
  - Cannot move

Logic

- Knowledge bases consist of sentences in a formal language
  - Syntax
    - Sentences are well formed
  - Semantics
    - The "meaning" of the sentence
    - ***The truth of each sentence with respect to each possible world (model)***

- Example:

  x + 2 >= y is a sentence

  x2 + y > is not a sentence

  x + 2 >= y is true iff x + 2 is no less than y

  x + 2 >= y is true in a world where x = 7, y=1

  x + 2 >= y is false in world where x = 0, y =6

Logic

- ***Entailment*** means that one thing follows logically from another

  $\alpha \models \beta$

- $\alpha \models \beta$ iff in every model in which α is true, β is also true

- if α is true, then β must be true

- the truth of β is “contained” in the truth of α

Logic

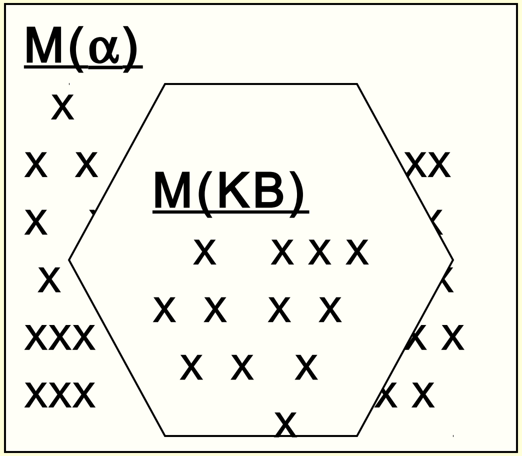
- Example:
  - A KB containing
    - "Cleveland won"
    - "Dallas won"
    - Entails…
      - "Either Cleveland won or Dallas won"

- Example:

  $x + y = 4$ entails $4 = x + y$

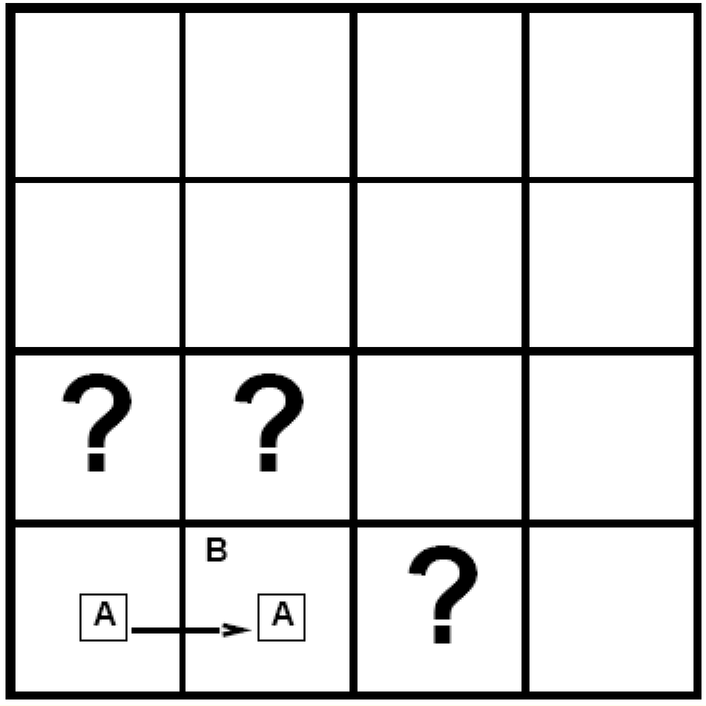
Logic

- A model is a formally structured world with respect to which truth can be evaluated
  - M is a model of sentence α if α is true in m
- Then KB |= α if M(KB) $\subseteq$ M(α)

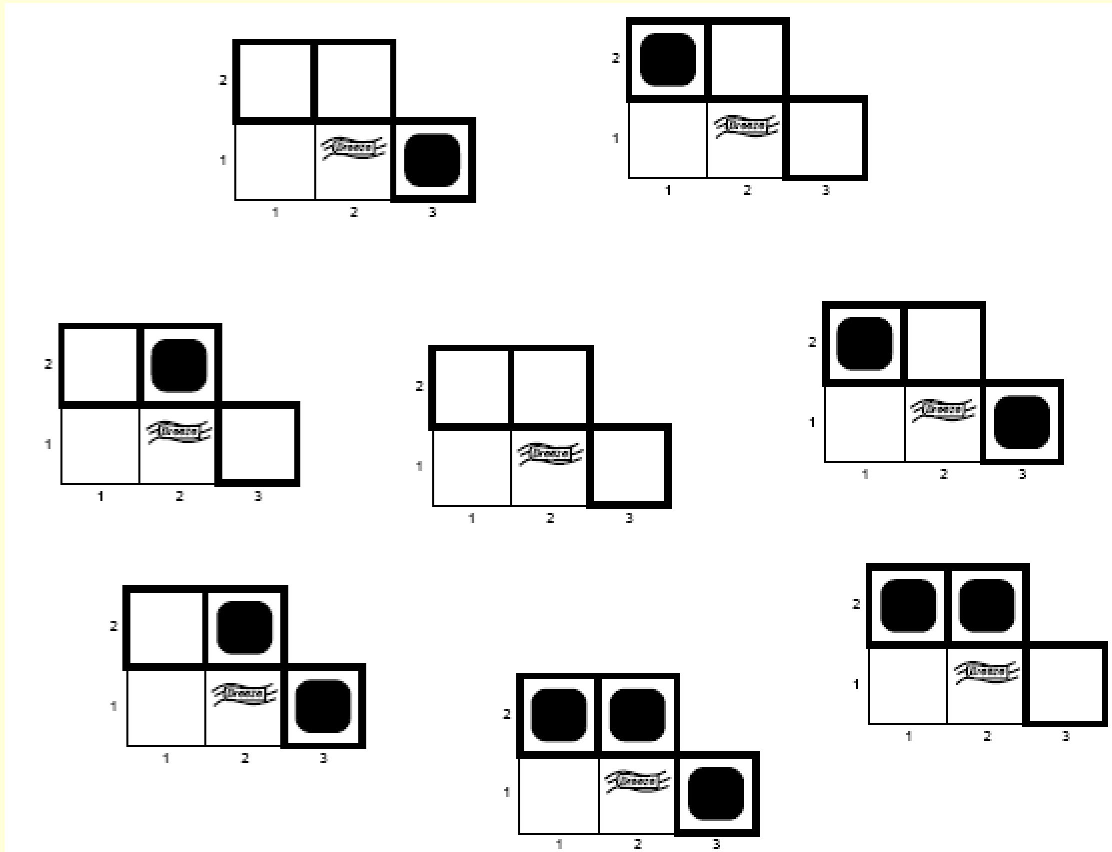


Logic

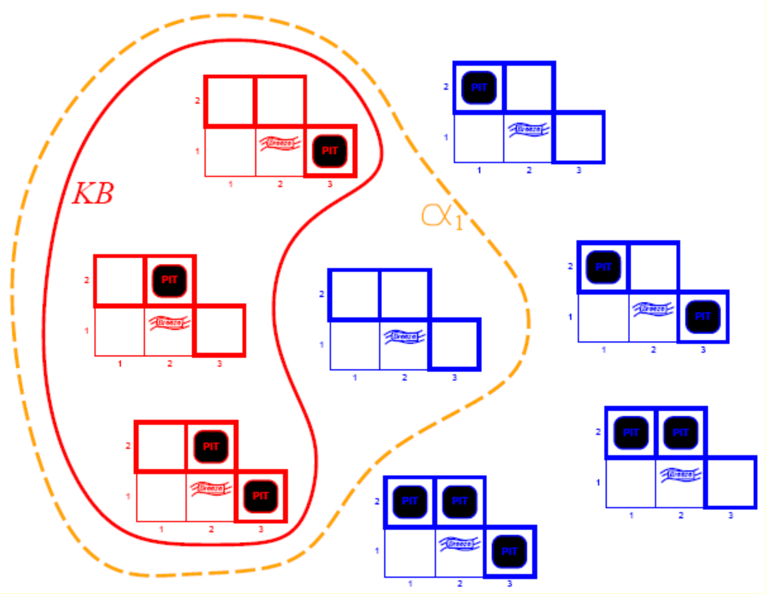
- Entailment in Wumpus World
- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models for ? assuming only pits
- 3 Boolean choices => 8 possible models



Logic



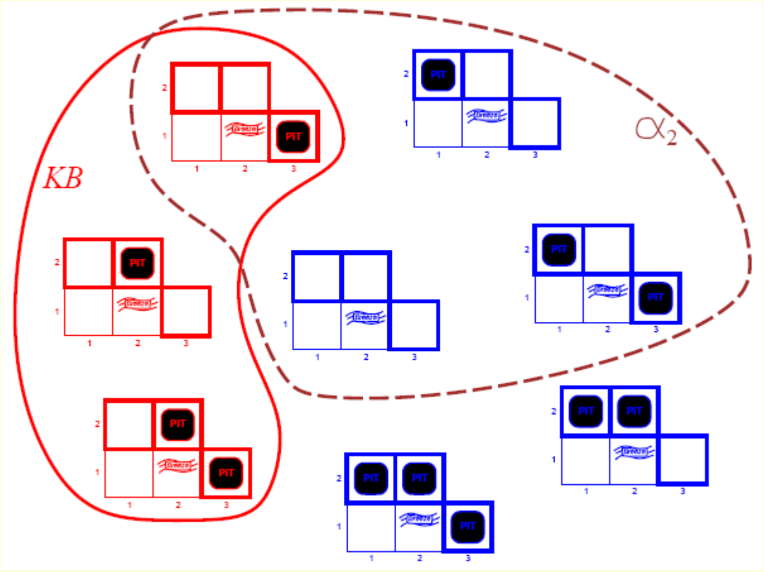
Logic



- KB = wumpus world rules + observations

- α_1 = "[1,2] is safe", KB |= α_1, proved by model checking

Logic



- KB = wumpus world rules + observations
- α_2 = “[2,2] is safe”, KB ¬|= α_2 proved by model checking

Logic

- ***Inference*** is the process of deriving a specific sentence from a KB (where the sentence must be entailed by the KB)
  - KB $|-_i$ α = sentence α can be derived from KB by procedure I
- "KB's are a haystack"
  - Entailment = needle in haystack
  - Inference = finding it

Logic

- Soundness
  - i is sound if…
  - whenever KB $|-_i$ α is true, KB $|=$ α is true

- Completeness
  - i is complete if
  - whenever KB $|=$ α is true, KB $|-_i$ α is true

- 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

Propositional Logic

- AKA Boolean Logic
- False and True
- Proposition symbols P1, P2, etc are sentences

- NOT: If S1 is a sentence, then $\neg$S1 is a sentence (negation)

- AND: If S1, S2 are sentences, then S1 $\wedge$ S2 is a sentence (conjunction)

- OR: If S1, S2 are sentences, then S1 $\vee$ S2 is a sentence (disjunction)

- IMPLIES: If S1, S2 are sentences, then S1 $\Rightarrow$ S2 is a sentence (implication)

- IFF: If S1, S2 are sentences, then S1 $\Leftrightarrow$ S2 is a sentence (biconditional)

Propositional Logic

P	Q	¬P	P∧Q	P∨Q	P⇒Q	P⇔Q
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

Wumpus World Sentences

- 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]

- $\neg P_{1,1}$
- $\neg B_{1,1}$
- $B_{2,1}$

- "Pits cause breezes in adjacent squares"

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

$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,1} \vee P_{3,1})$

- A square is breezy if and only if there is an adjacent pit

A Simple Knowledge Base

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
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*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

A Simple Knowledge Base

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

- R1: $\neg P_{1,1}$
- R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- R3: $B_{2,1}$ $(P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- R4: $\neg B_{1,1}$
- R5: $B_{2,1}$

- KB consists of sentences R_1 thru R_5
- $R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5$

A Simple Knowledge Base

```
function TT-ENTAILS?(KB, α) returns true or false
    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols); rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                   TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

- Every known inference algorithm for propositional logic has a worst-case complexity that is exponential in the size of the input. (co-NP complete)

Equivalence, Validity, Satisfiability

Two sentences are logically equivalent iff true in same models:

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

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Equivalence, Validity, Satisfiability

- A sentence if valid if it is true in all models
  - e.g. True, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B) \Rightarrow B$
- Validity is connected to inference via the Deduction Theorem
  - KB |- α iff (KB $\Rightarrow \alpha$) is valid
- A sentence is satisfiable if it is True in some model
  - e.g. $A \vee B$, C
- A sentence is unstatisfiable if it is True in no models
  - e.g. $A \wedge \neg A$
- Satisfiability is connected to inference via the following
  - KB |= α iff (KB $\wedge \neg\alpha$) is unsatisfiable
  - proof by contradiction

Reasoning Patterns

- Inference Rules
  - Patterns of inference that can be applied to derive chains of conclusions that lead to the desired goal.
- Modus Ponens
  - Given: $S1 \Rightarrow S2$ and S1, derive S2
- And-Elimination
  - Given: $S1 \wedge S2$, derive S1
  - Given: $S1 \wedge S2$, derive S2
- DeMorgan's Law
  - Given: $\neg(A \vee B)$ derive $\neg A \wedge \neg B$
  - Given: $\neg(A \wedge B)$ derive $\neg A \vee \neg B$

Reasoning Patterns

- And Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

- From a conjunction, any of the conjuncts can be inferred

- (WumpusAhead $\wedge$ WumpusAlive), WumpusAlive can be inferred

- Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

- Whenever sentences of the form $\alpha \Rightarrow \beta$ and α are given, then sentence β can be inferred

- (WumpusAhead $\wedge$ WumpusAlive) $\Rightarrow$ Shoot and (WumpusAhead $\wedge$ WumpusAlive), Shoot can be inferred

Example Proof By Deduction

- Knowledge

S1: $B_{22} \Leftrightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$ *rule*

S2: $\neg B_{22}$ *observation*

- Inferences

S3: $(B_{22} \Rightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})) \wedge$
$((P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \Rightarrow B_{22})$ *[S1,bi elim]*

S4: $((P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \Rightarrow B_{22})$ *[S3, and elim]*

S5: $(\neg B_{22} \Rightarrow \neg(P_{21} \vee P_{23} \vee P_{12} \vee P_{32}))$ *[contrapos]*

S6: $\neg(P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$ *[S2,S6, MP]*

S7: $\neg P_{21} \wedge \neg P_{23} \wedge \neg P_{12} \wedge \neg P_{32}$ *[S6, DeMorg]*

Evaluation of Deductive Inference

- Sound
  - Yes, because the inference rules themselves are sound. (This can be proven using a truth table argument).
- Complete
  - If we allow all possible inference rules, we're searching in an infinite space, hence not complete
  - If we limit inference rules, we run the risk of leaving out the necessary one…
- Monotonic
  - If we have a proof, adding information to the DB will not invalidate the proof

Resolution

- Resolution allows a complete inference mechanism (search-based) using only one rule of inference
- Resolution rule:
  - Given: $P_1 \vee P_2 \vee P_3 \ldots \vee P_n$ and $\neg P_1 \vee Q_1 \ldots \vee Q_m$
  - Conclude: $P_2 \vee P_3 \ldots \vee P_n \vee Q_1 \ldots \vee Q_m$

    Complementary literals P_1 and $\neg P_1$ "cancel out"
- Why it works:
  - Consider 2 cases: P_1 is true, and P_1 is false

Resolution in Wumpus World

- There is a pit at 2,1 or 2,3 or 1,2 or 3,2
  - $P_{21} \vee P_{23} \vee P_{12} \vee P_{32}$
- There is no pit at 2,1
  - $\neg P_{21}$
- Therefore (by resolution) the pit must be at 2,3 or 1,2 or 3,2
  - $P_{23} \vee P_{12} \vee P_{32}$

Proof using Resolution

- To prove a fact P, repeatedly apply resolution until either:
  - No new clauses can be added, (KB does not entail P)
  - The empty clause is derived (KB does entail P)

- This is proof by contradiction: if we prove that KB $\wedge$ $\neg$P derives a contradiction (empty clause) and we know KB is true, then $\neg$P must be false, so P must be true!

- To apply resolution mechanically, facts need to be in ***Conjunctive Normal Form (CNF)***

- To carry out the proof, need a search mechanism that will enumerate all possible resolutions.

CNF Example

1. $B_{22} \Leftrightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$

2. Eliminate $\Leftrightarrow$, replacing with two implications

   $(B_{22} \Rightarrow (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})) \wedge ((P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \Rightarrow B_{22})$

1. Replace implication $(A \Rightarrow B)$ by $\neg A \vee B$

   $(\neg B_{22} \vee (P_{21} \vee P_{23} \vee P_{12} \vee P_{32})) \wedge (\neg(P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \vee B_{22})$

1. Move $\neg$ "inwards" (unnecessary parens removed)

   $(\neg B_{22} \vee P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \wedge ((\neg P_{21} \wedge \neg P_{23} \wedge \neg P_{12} \wedge \neg P_{32}) \vee B_{22})$

4. Distributive Law

   $(\neg B_{22} \vee P_{21} \vee P_{23} \vee P_{12} \vee P_{32}) \wedge (\neg P_{21} \vee B_{22}) \wedge (\neg P_{23} \vee B_{22}) \wedge (\neg P_{12} \vee B_{22}) \wedge (\neg P_{32} \vee B_{22})$

(Final result has 5 clauses)

Resolution Example

- Given B_{22} and $\neg P_{21}$ and $\neg P_{23}$ and $\neg P_{32}$, prove P_{12}
- $(\neg B_{22} \vee P_{21} \vee P_{23} \vee P_{12} \vee P_{32})$; $\neg P_{12}$
- $(\neg B_{22} \vee P_{21} \vee P_{23} \vee P_{32})$; $\neg P_{21}$
- $(\neg B_{22} \vee P_{23} \vee P_{32})$; $\neg P_{23}$
- $(\neg B_{22} \vee P_{32})$; $\neg P_{32}$
- $(\neg B_{22})$; B_{22}
- [empty clause]

Evaluation of Resolution

- Resolution is sound
  - Because the resolution rule is true in all cases
- Resolution is complete
  - Provided a complete search method is used to find the proof, if a proof can be found it will
  - Note: you must know what you're trying to prove in order to prove it!
- Resolution is exponential
  - The number of clauses that we must search grows exponentially…

Horn Clauses

- A Horn Clause is a CNF clause with exactly one positive literal
  - The positive literal is called the head
  - The negative literals are called the body
  - Prolog: head:- body1, body2, body3 …
  - English: "To prove the head, prove body1, …"
  - Implication: If (body1, body2 …) then head
- Horn Clauses form the basis of forward and backward chaining
- The Prolog language is based on Horn Clauses
- Deciding entailment with Horn Clauses is *linear in the size of the knowledge base*

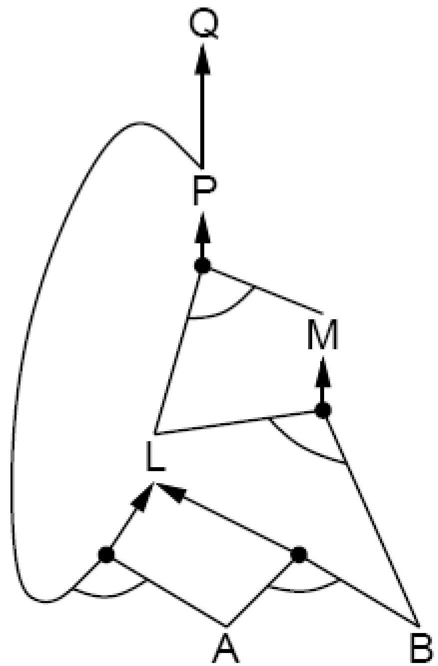
Reasoning with Horn Clauses

- Forward Chaining
  - For each new piece of data, generate all new facts, until the desired fact is generated
  - Data-directed reasoning
- Backward Chaining
  - To prove the goal, find a clause that contains the goal as its head, and prove the body recursively
  - (Backtrack when you chose the wrong clause)
  - Goal-directed reasoning

Forward Chaining

- Fire any rule whose premises are satisfied in the KB
- Add its conclusion to the KB until the query is found

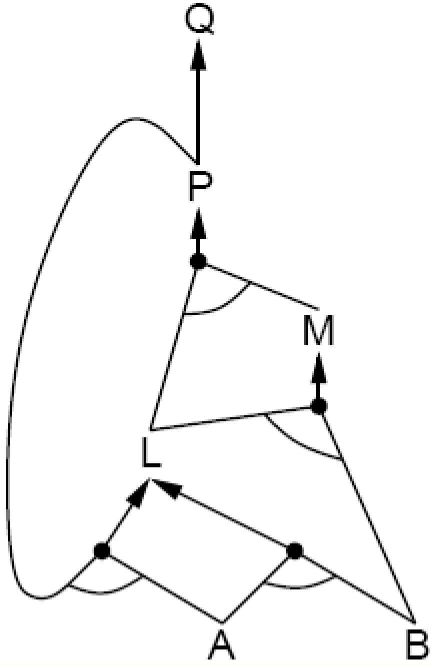
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



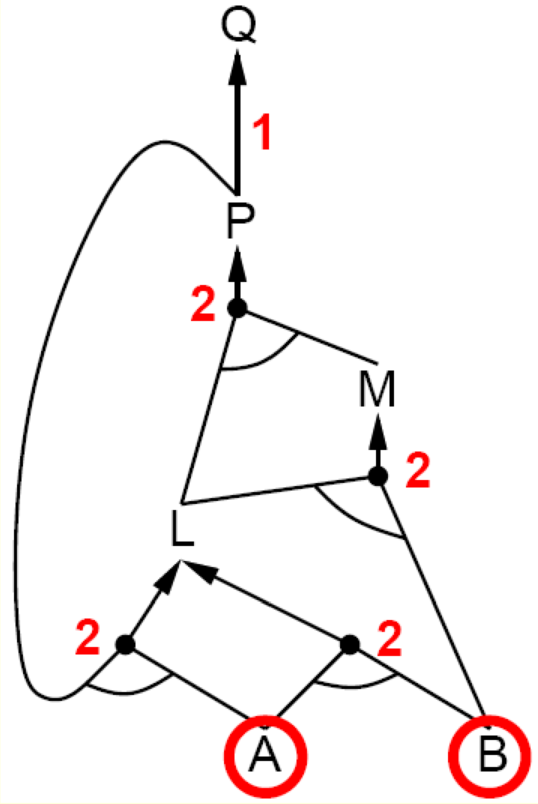
Forward Chaining

- AND-OR Graph
  - multiple links joined by an arc indicate conjunction – every link must be proved
  - multiple links without an arc indicate disjunction – any link can be proved

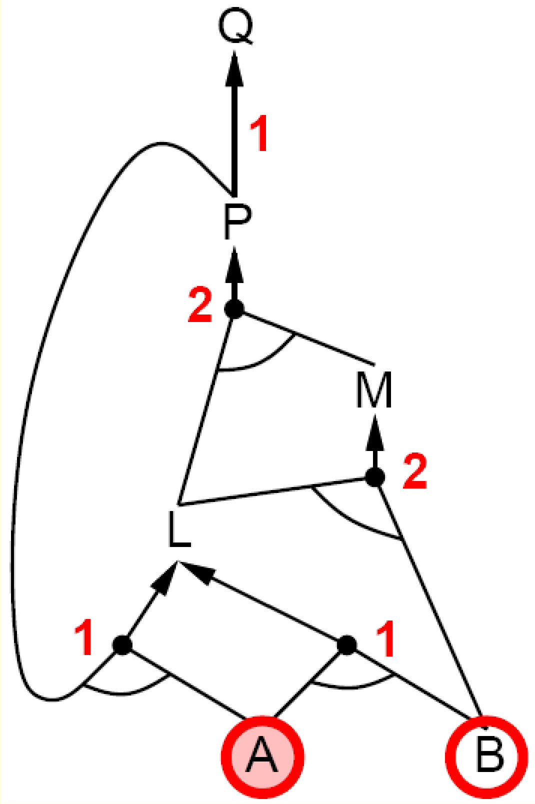
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



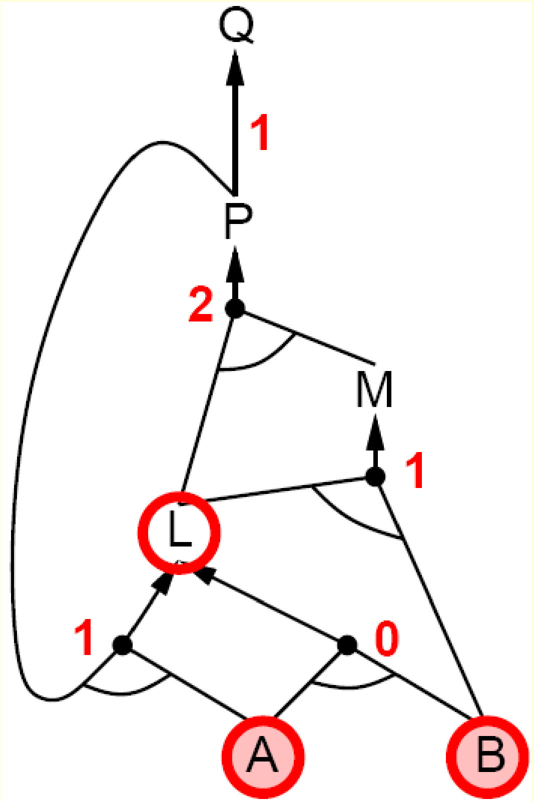
Forward Chaining



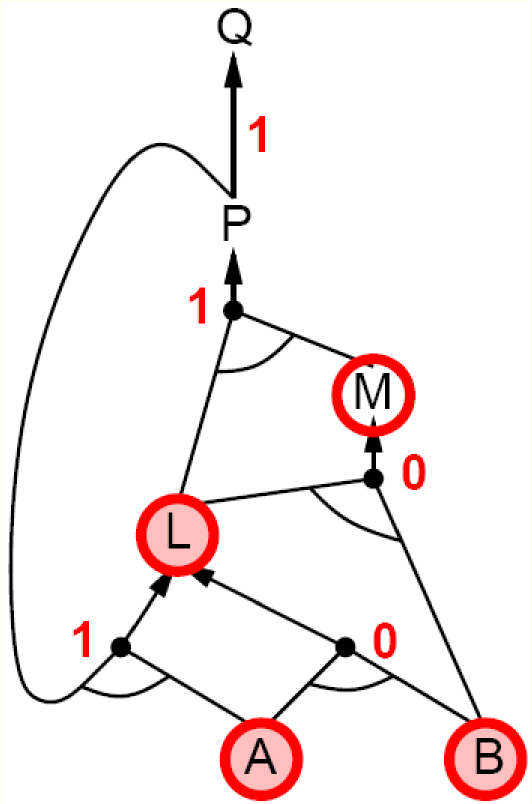
Forward Chaining



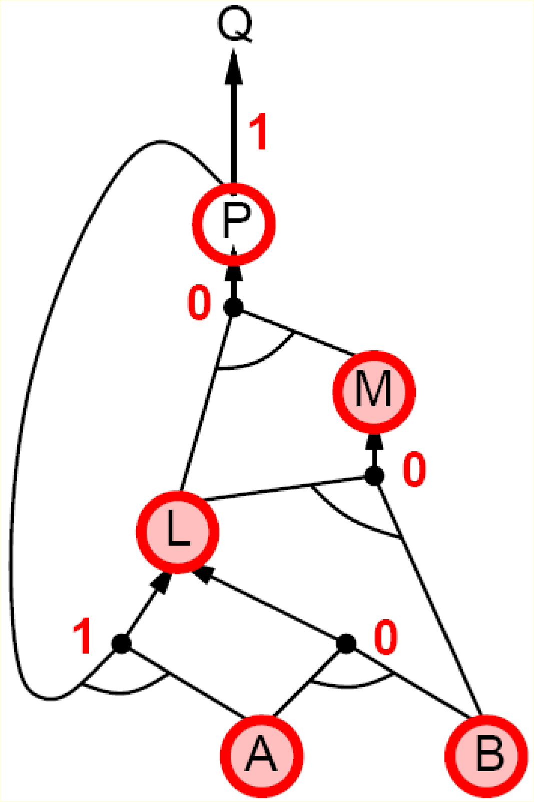
Forward Chaining



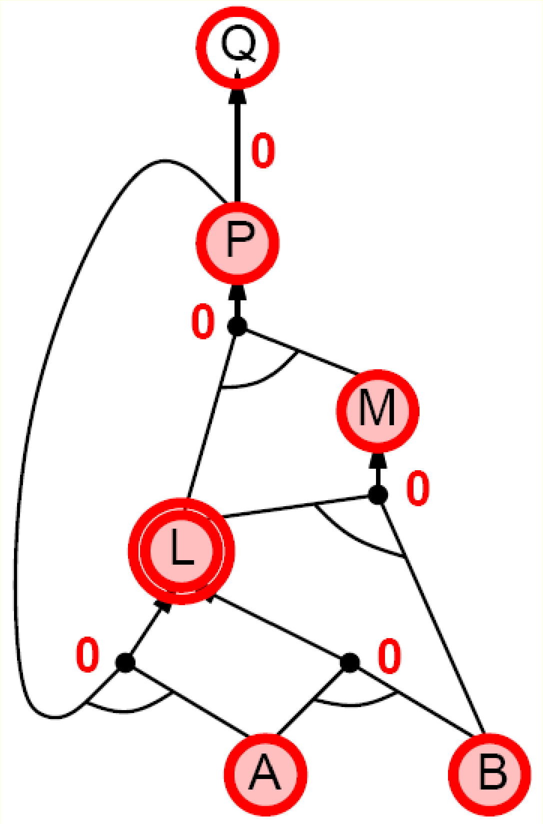
Forward Chaining



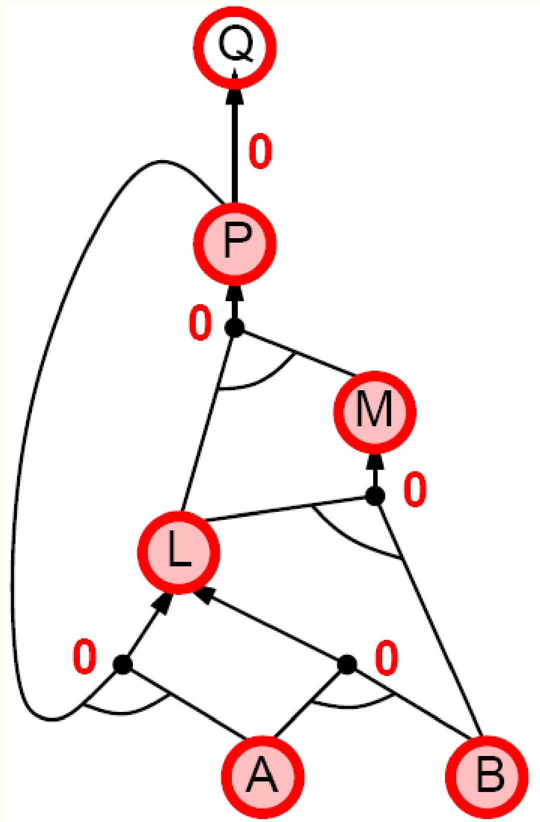
Forward Chaining



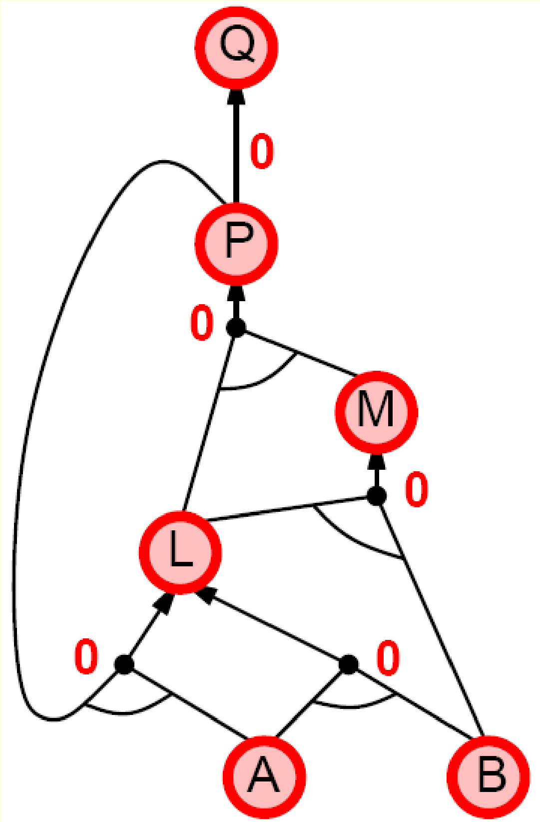
Forward Chaining



Forward Chaining



Forward Chaining



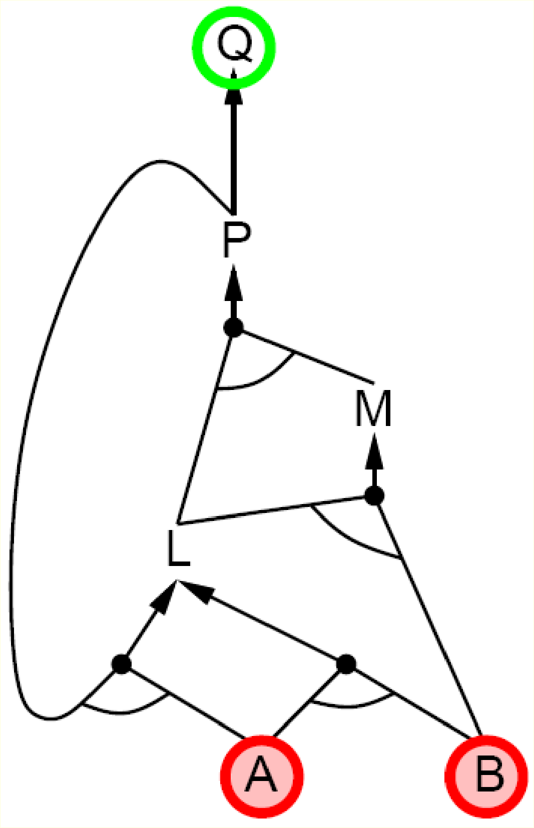
Backward Chaining

- 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

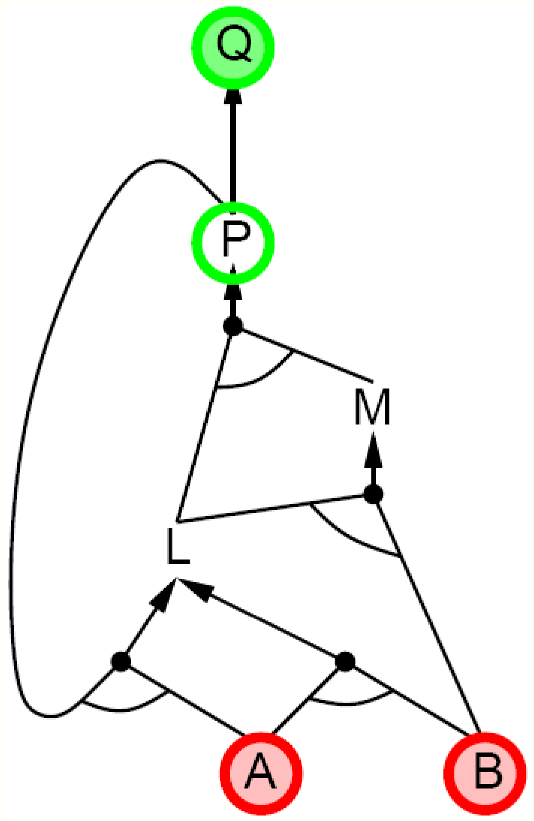
- Avoid loops
  - Check if new subgoal is already on the goal stack

- Avoid repeated work: check if new subgoal
  - Has already been proved true, or
  - Has already failed

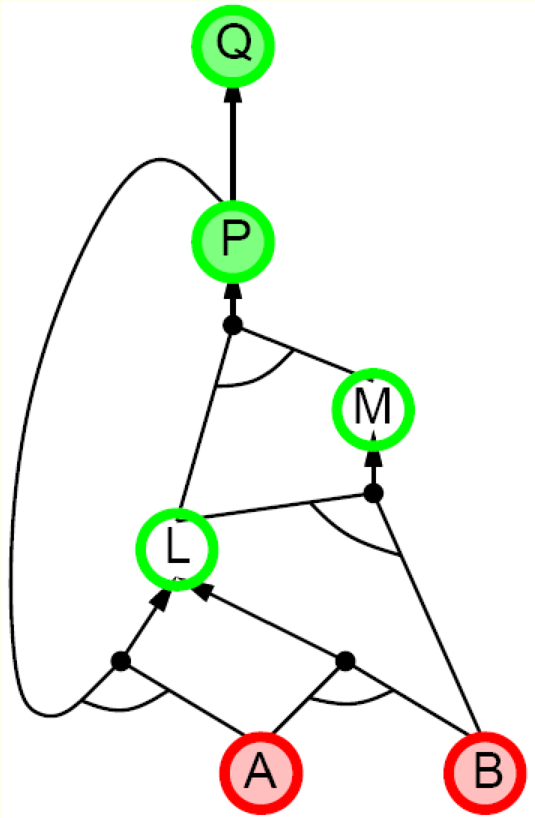
Backward Chaining



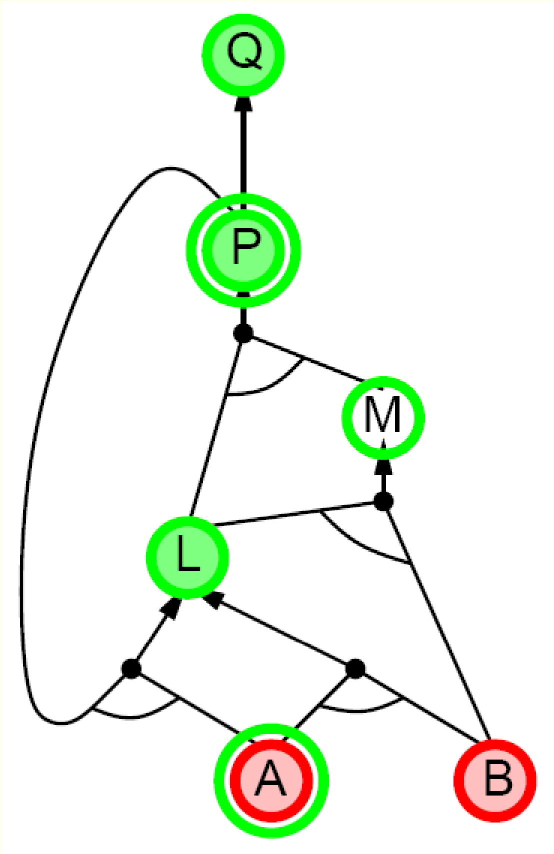
Backward Chaining



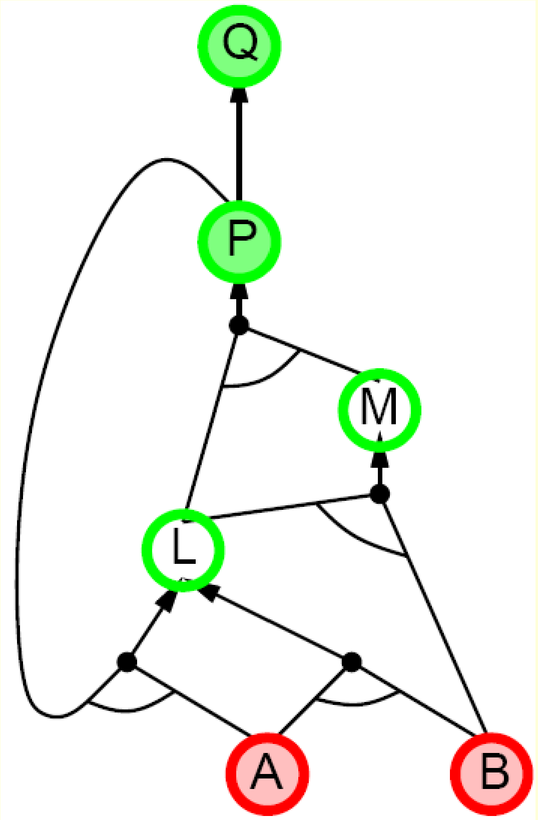
Backward Chaining



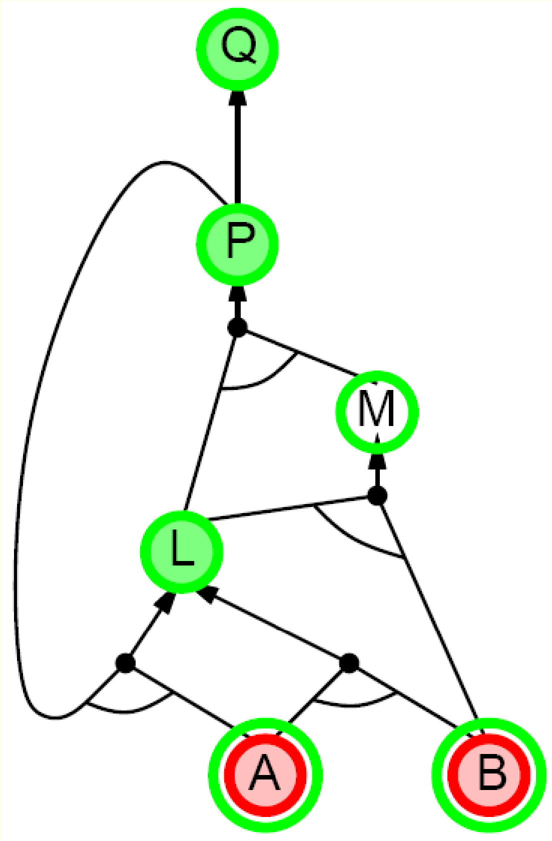
Backward Chaining



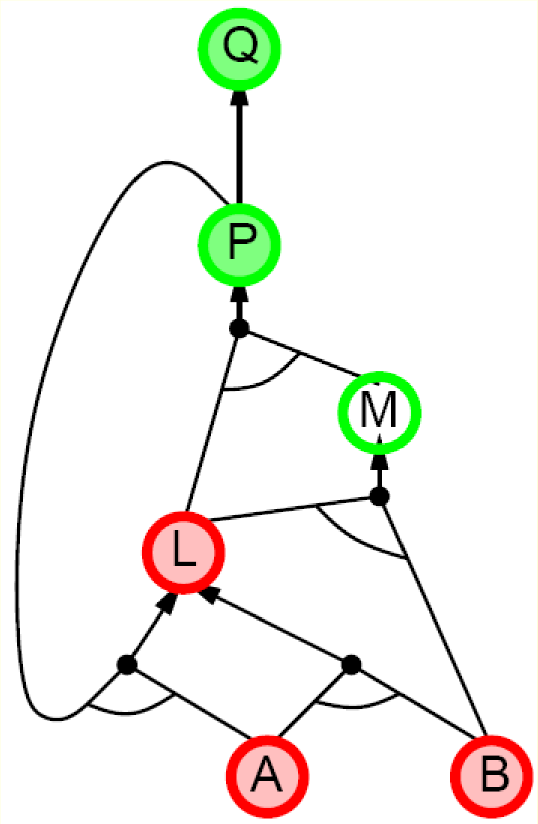
Backward Chaining



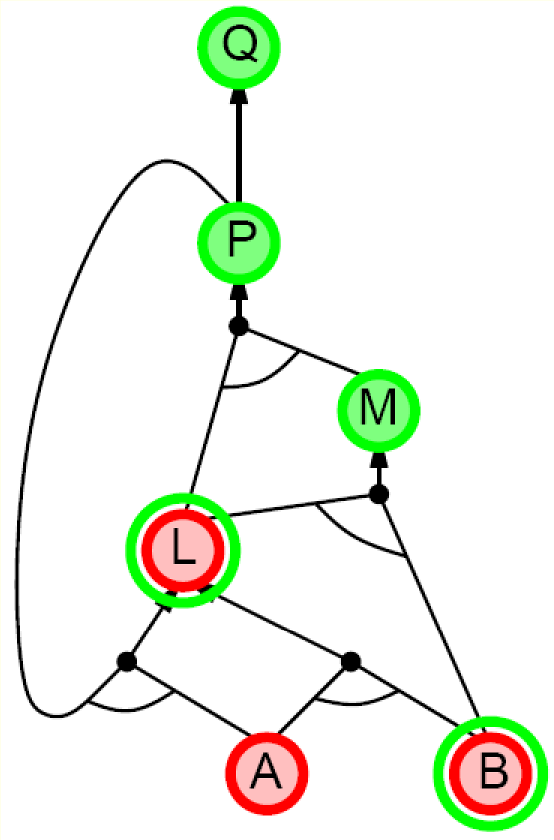
Backward Chaining



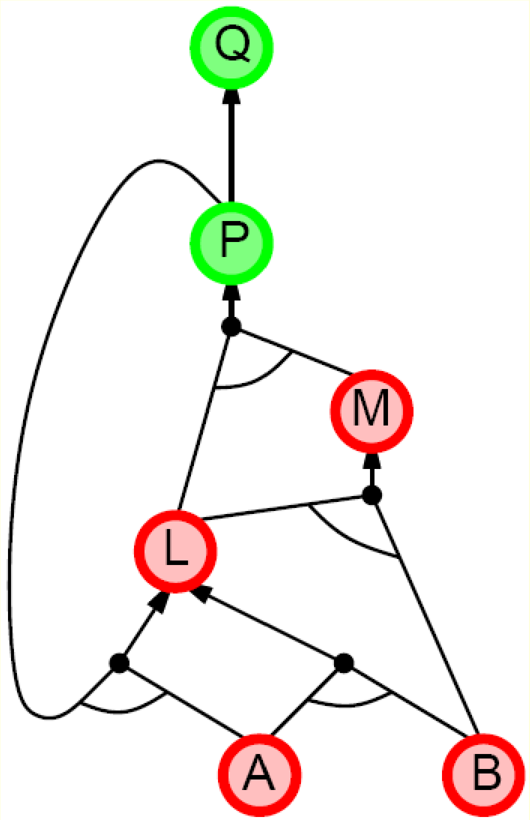
Backward Chaining



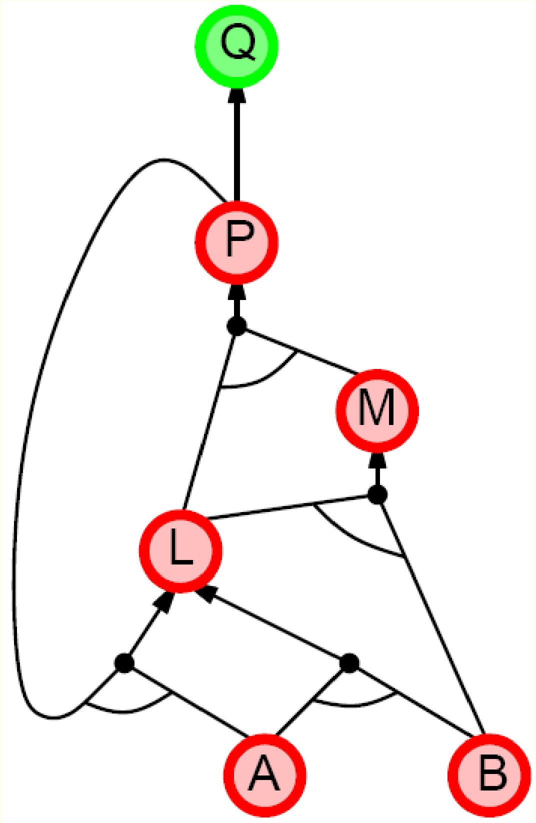
Backward Chaining



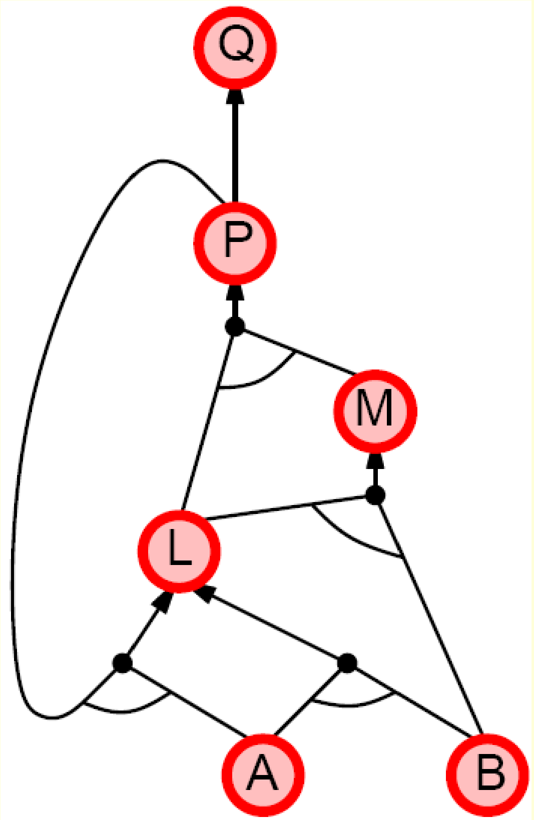
Backward Chaining



Backward Chaining



Backward Chaining



Forward Chaining vs. Backward Chaining

- Forward Chaining is data driven
  - Automatic, unconscious processing
  - E.g. object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal
- Backward Chaining is goal driven
  - Appropriate for problem solving
  - E.g. “Where are my keys?”, “How do I start the car?”
- The complexity of BC can be much less than linear in size of the KB