Logical agents

Chapter 7

(Some slides adapted from Stuart Russell, Dan Klein, and many others. Thanks guys!)

Outline

- 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

Knowledge-based Agents

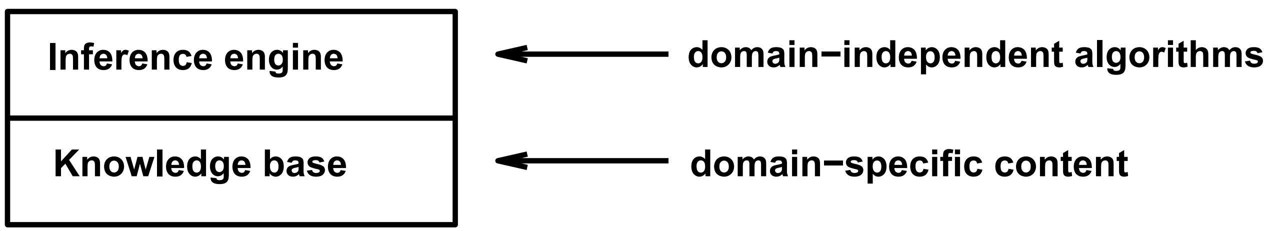
- Previously: Solved problems by *search*
  - Basically brute force. Clever...but is it “intelligent”?
  - “Knowledge” about how world works hidden...embodied in successor fn.

- Knowledge-based agents:
  - Have internal representations of the world...and reason about them.
  - Based on formal symbolic logics: propositional, first-order predicate, etc.

- Advantages:
  - Can combine and recombine base knowledge for *many* purposes
  - Can accept new tasks anytime, explicitly states as *goals*
    - *Q: Could Boggle do any task except...well...boggle search boards?*
  - Can achieve competence quickly
    - Being told new facts about the world
    - Learning new knowledge by deduction + percepts
  - Can adapt to changes in environment by updating knowledge

Knowledge Bases

- Knowledge base is basis for all KB-agent reasoning and action
  - Consists of: set of sentences in a formal language



- Declarative approach to building an agent (or other system):
- Idea:
  - Tell it what it needs to know
  - Then it can Ask itself what to do (autonomous agent) or you Ask it goal.
  - answers should follow from the KB

- KB-Agents can be viewed at the knowledge level
  - i.e., what they know, regardless of how implemented

- Or at the implementation level
  - i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

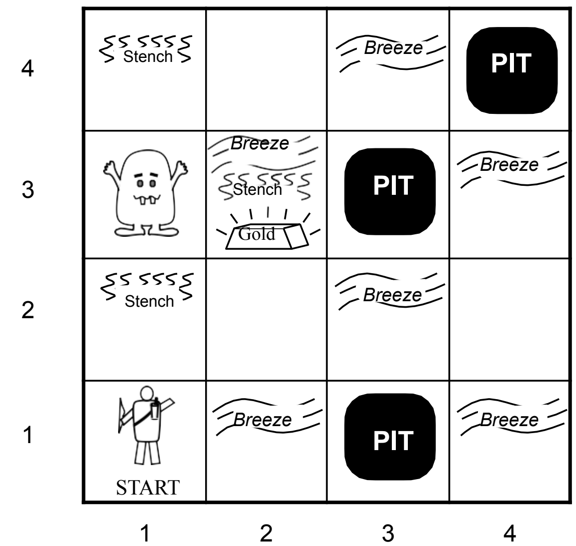
```
function KB-Agent( percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    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
```

- KB-agent function centered around on:
  - Tell: Adding new information to the KB
  - Ask: Posing a query (goal) to be resolved using the KB and universal algorithms

- The agent must be able to:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
    - Things it has not been told explicitly...but arise from evolving facts
  - Deduce appropriate actions (given tacit or explicit goal)

Wumpus World: A classic example

- Simple game of logical deduction
  - Dark cave with deadly pits and voracious wumpus monster
  - Goal: Find hidden pile of gold, avoid dying, return safely



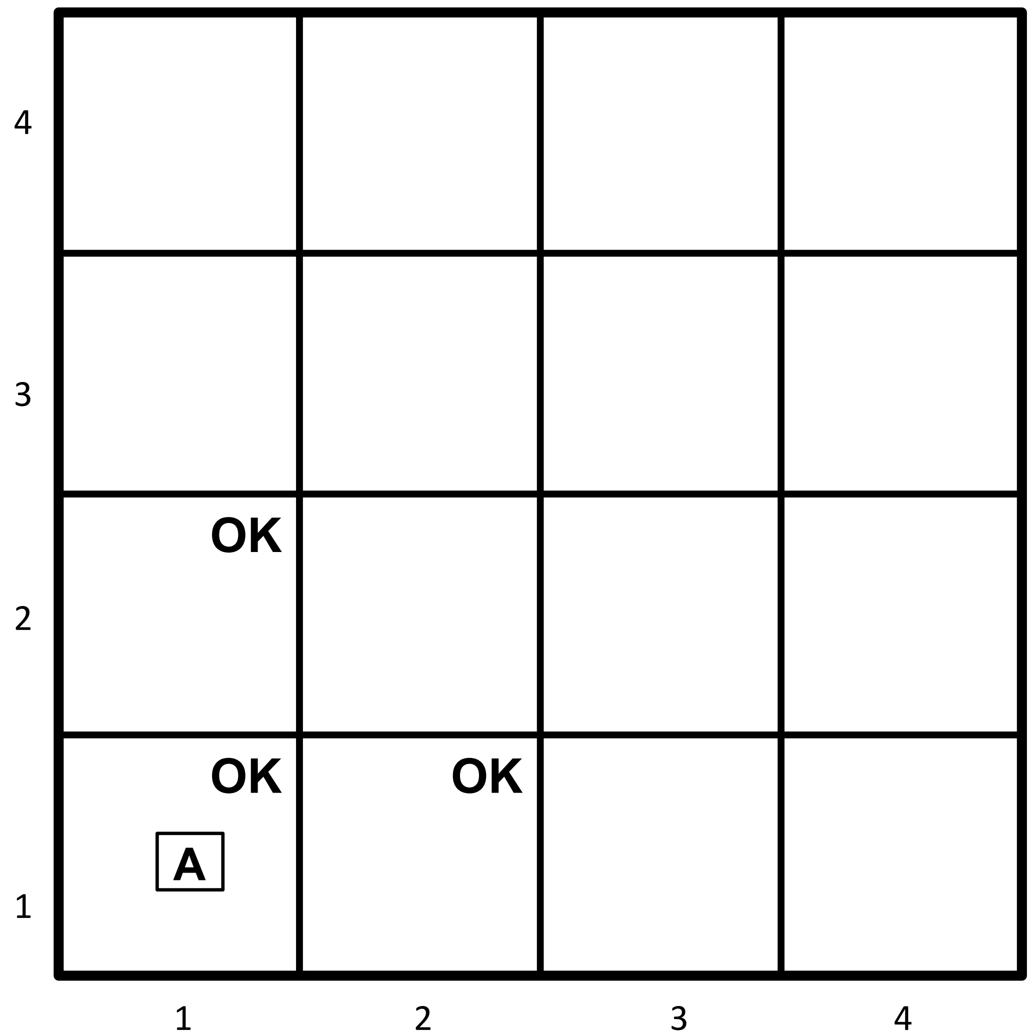
PEAS Description:

- 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
- Actuators:
  - Left turn, Right turn, Forward, Grab, Release, Shoot
- Sensors:
  - Breeze, Glitter, Smell

Wumpus World: problem characterization

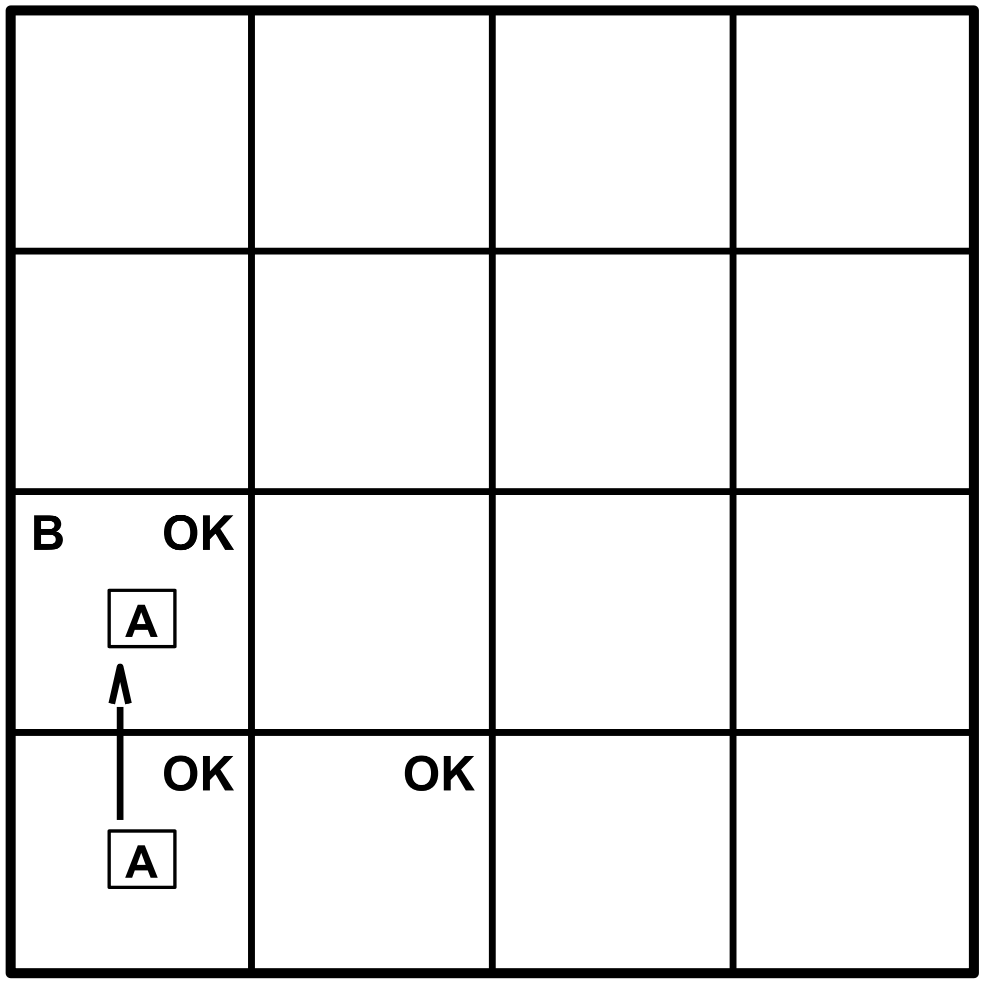
- Observable??
  - No—only local perception
- Deterministic??
  - Yes—outcomes exactly specified
- Episodic??
  - No—sequential at the level of actions
- Static??
  - Yes—Wumpus and Pits do not move
- Discrete??
  - Yes. Actions are discrete and limited. States are definite and finite.
- Single-agent??
  - Yes—Wumpus is essentially a natural feature

Exploring a wumpus world



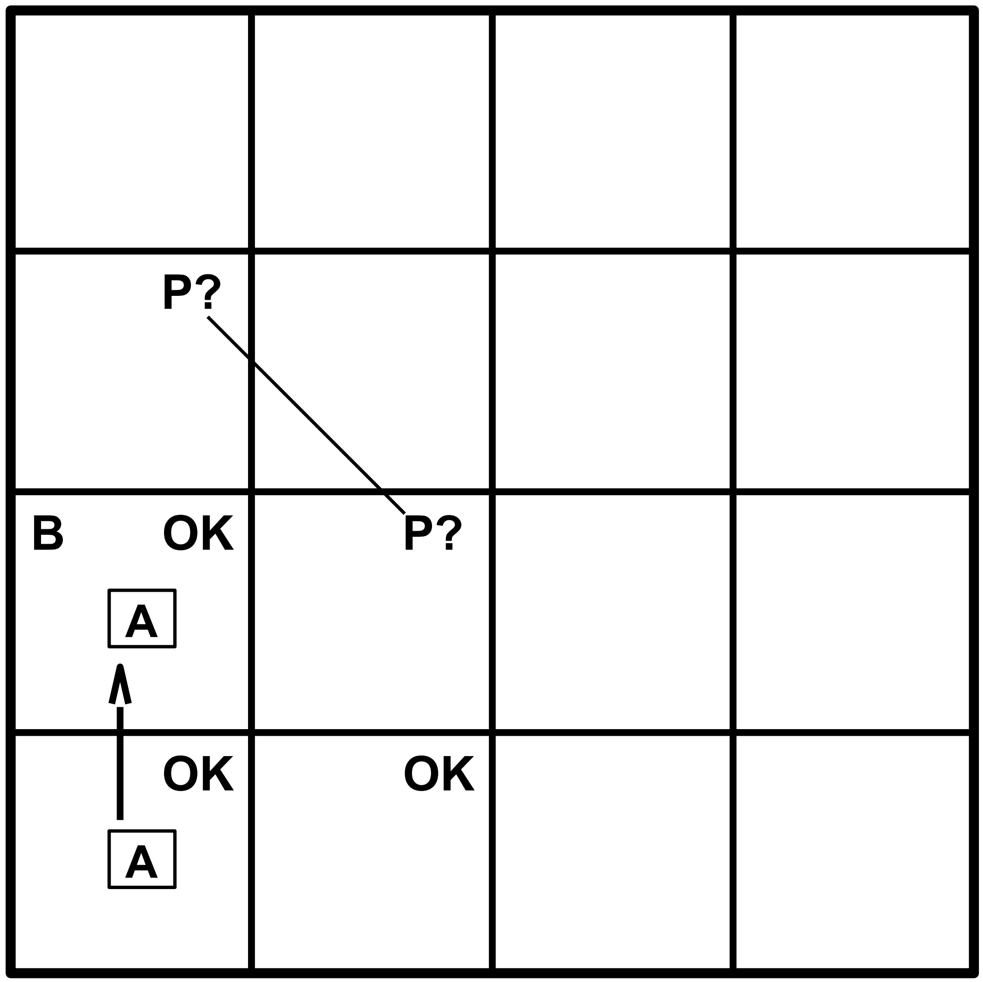
- Start in [1,1]. Cave entry/exit. Guaranteed safe.
- Note: No smells, no breezes → adjacent squares ok.

Exploring a wumpus world



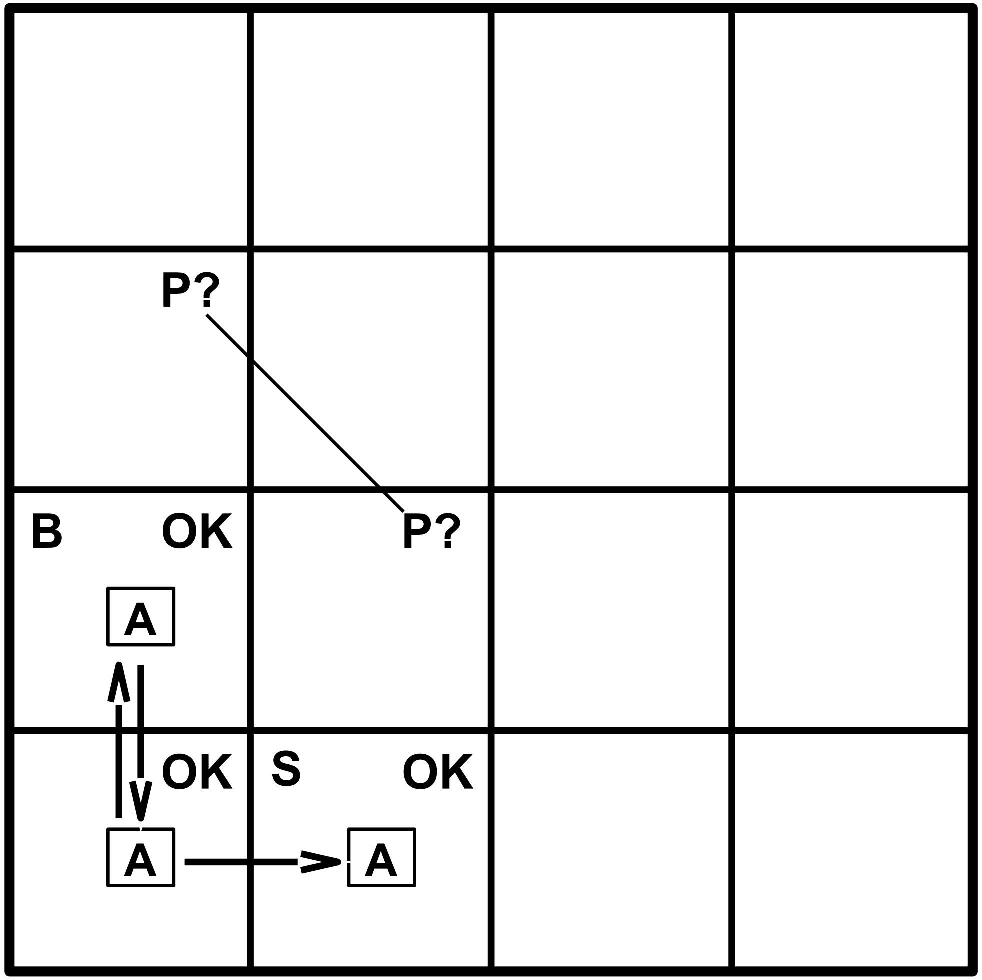
- Move to [2,1]. Sensor detects breeze (B)

Exploring a wumpus world



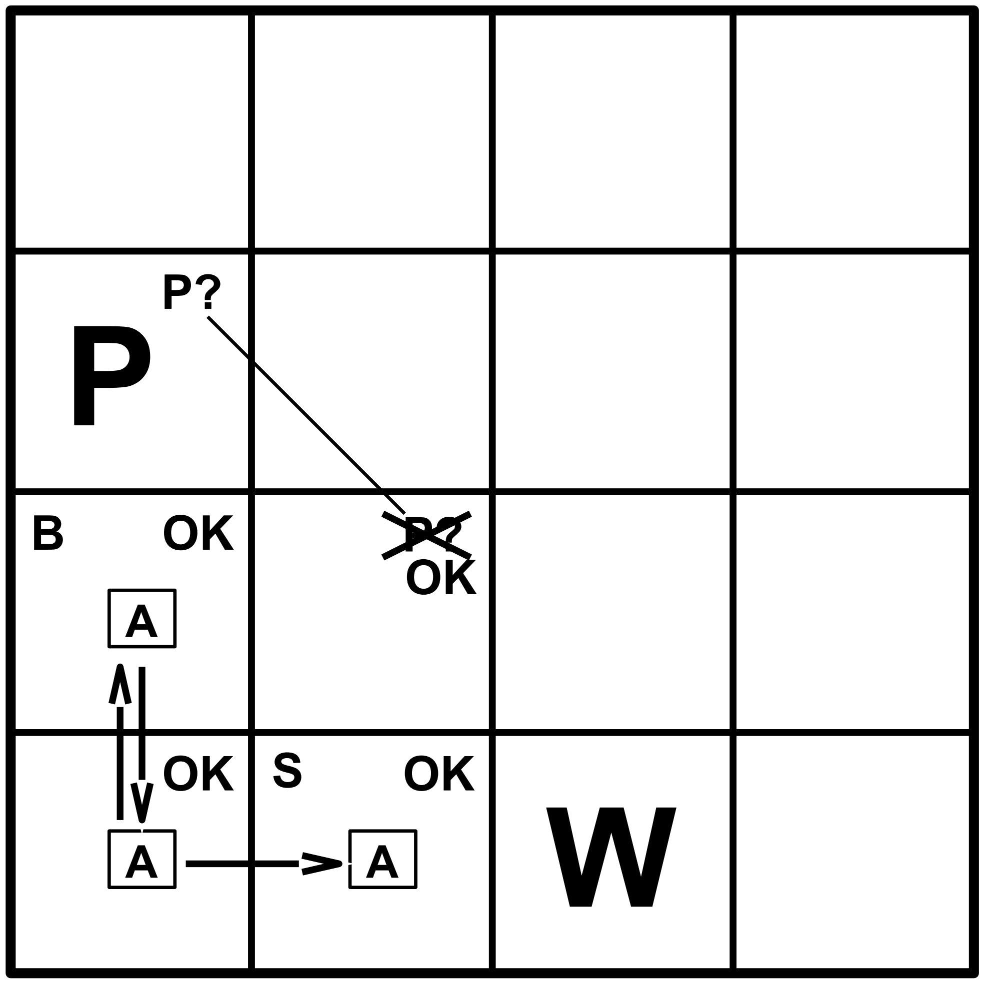
- Deduce possible pits in adjacent squares.

Exploring a wumpus world



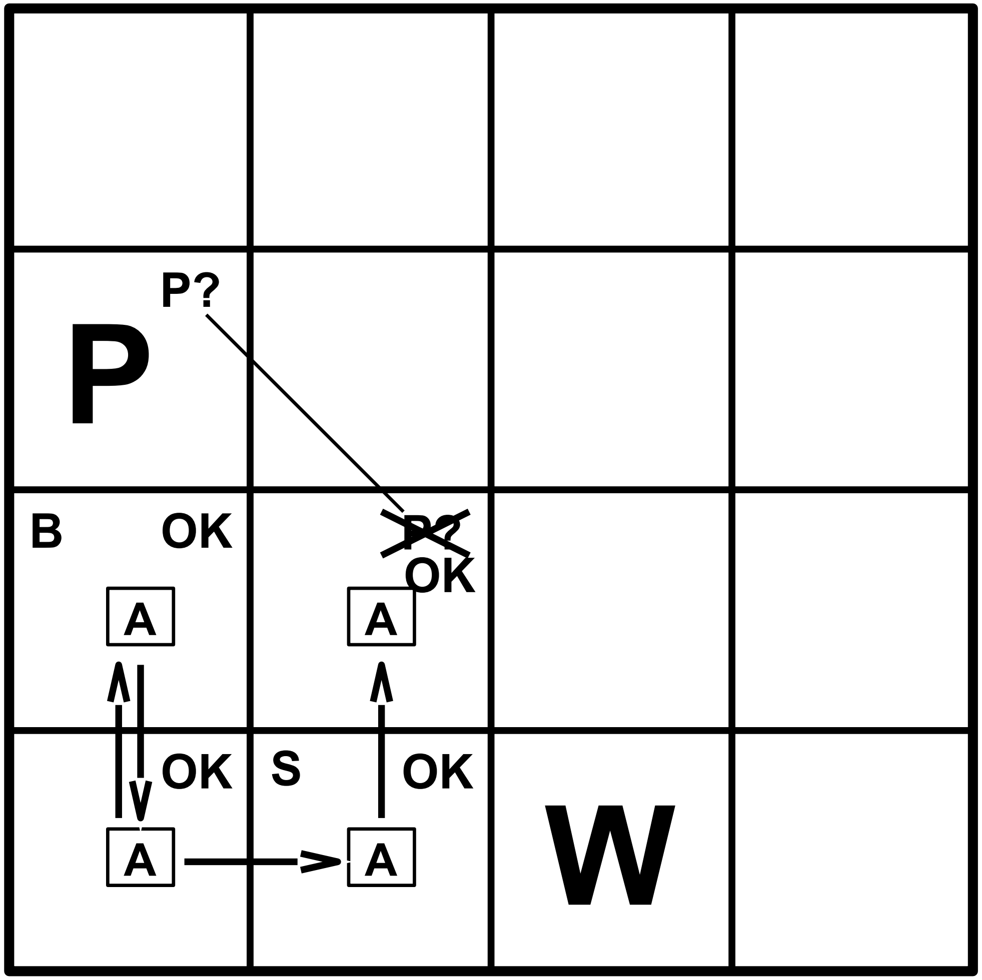
- Better go explore a safer place...maybe gather more info...
- Detect smell (S) in [1,2]... But no breeze!

Exploring a wumpus world

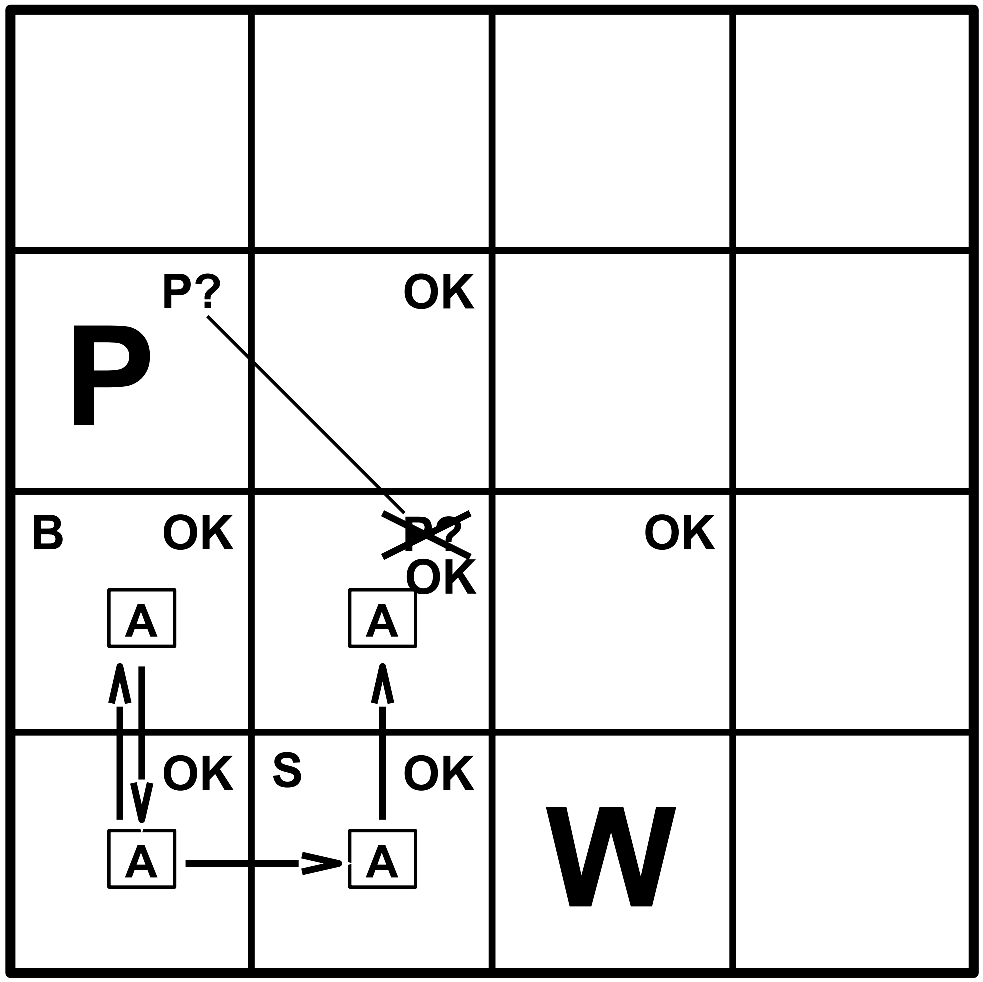


- Can deduce W in [1,3] (can't be in [2,2] because was no smell in [2,1]!)
- Can definitely place the Pit in [3,1]

Exploring a wumpus world

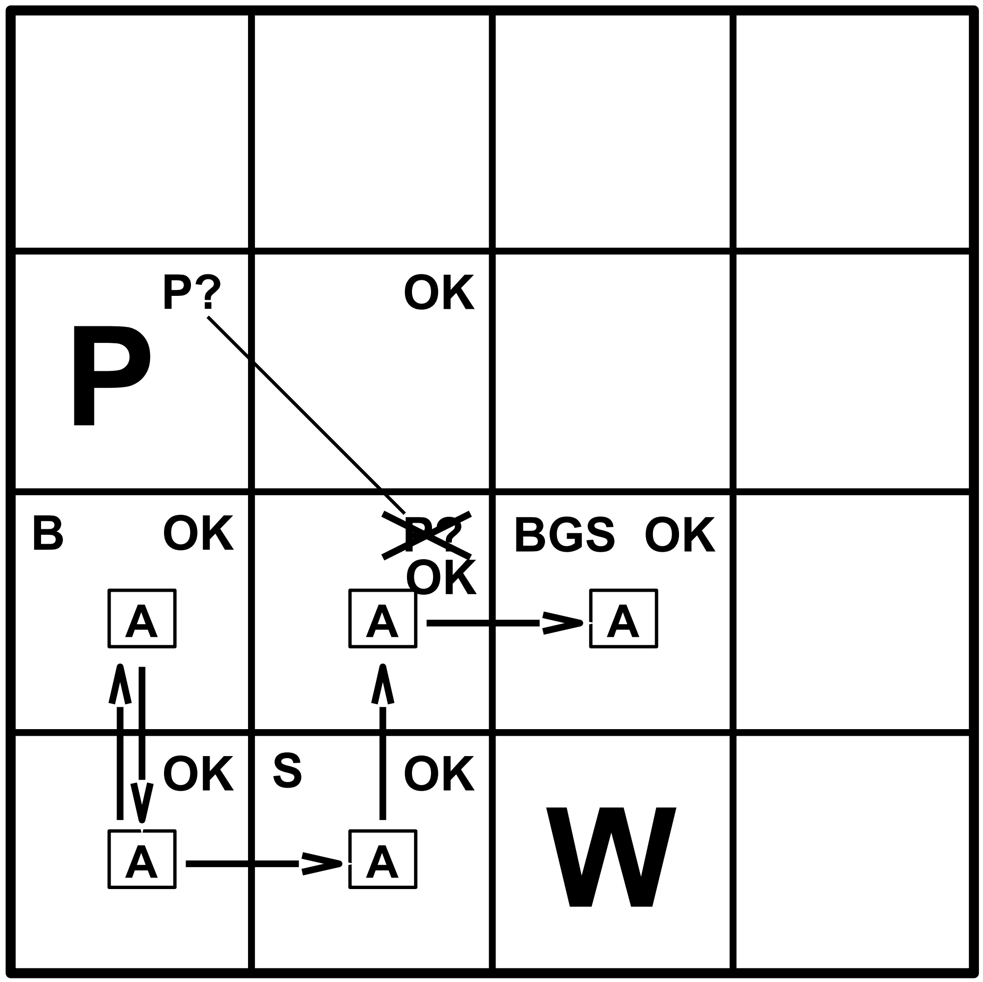


Exploring a wumpus world



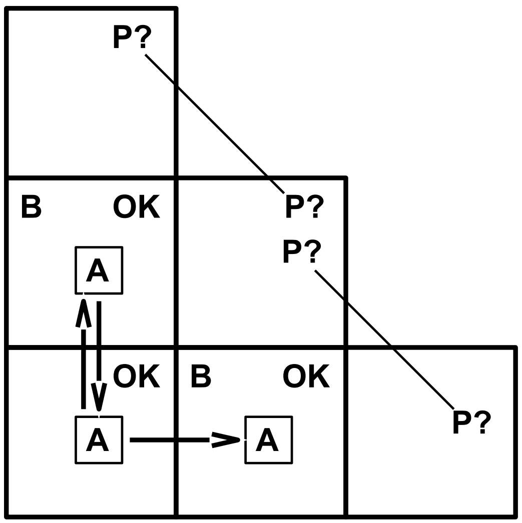
- Now that [2,2] determined OK, can go there.
- Nothing sensed → deductions about adjacent

Exploring a wumpus world

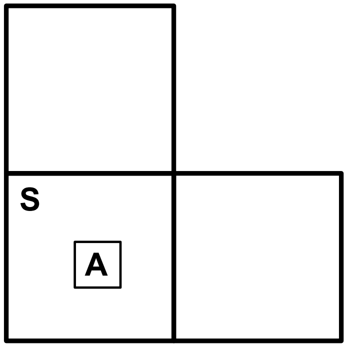


- And then on to [2,3]. Detect Glitter! Grab Gold!
- Then head back out to exit.

Tight spots: Can't *always* reason safely



- Breeze in (1,2) and (2,1)
  ⇒ no safe actions!
- Make educated guess:
  Assuming pits uniformly distributed, (2,2) has pit w/ prob 0.86, vs. 0.31



- Smell in (1,1)
  ⇒ cannot move!
- Can use a strategy of coercion:
  Act: shoot straight ahead
  - Wumpus was there → dead
  - Wumpus not there → safe

Introduction to Logic

Let's start with some basics: definitions

- Logics are formal languages for representing information
  - such that conclusions can be drawn
- Syntax defines the format of legal sentences in the language
- Semantics define the "meaning" of sentences
  - i.e., define truth of a sentence with respect to a particular world (state)

Example: The language of arithmetic

- Syntax: $x + 2 \geq y$ is a legal sentence; $x2 + y >$ is not
- Semantics:
  - $x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y
  - $x + 2 \geq y$ is true in a world where $x = 7, y = 1$
  - $x + 2 \geq y$ is false in a world where $x = 0, y = 6$

Logical *Entailment*

- Entailment means that one thing follows from another:
  - KB |= α
  - Knowledge base KB entails sentence α
    if and only if
    α is true in all worlds where KB is true

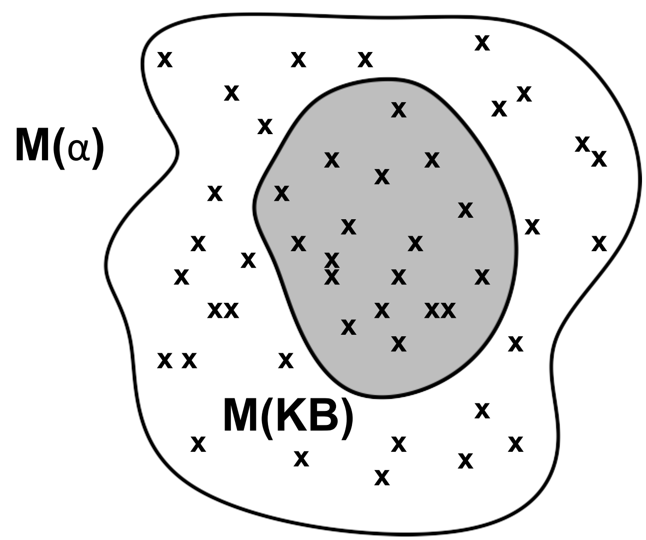
- Example: the KB containing "the Giants won" and "the Reds won" entails
  α ="Either the Giants won or the Reds won"
- → α is true in all worlds in which KB is true.

- Example: x + y = 4 entails 4 = x + y

- Entailment our first element of *reasoning!*
  - is a relationship between sentences (i.e., syntax)
  - Idea that one sentence (logical fact) *follows logically* from another sentence

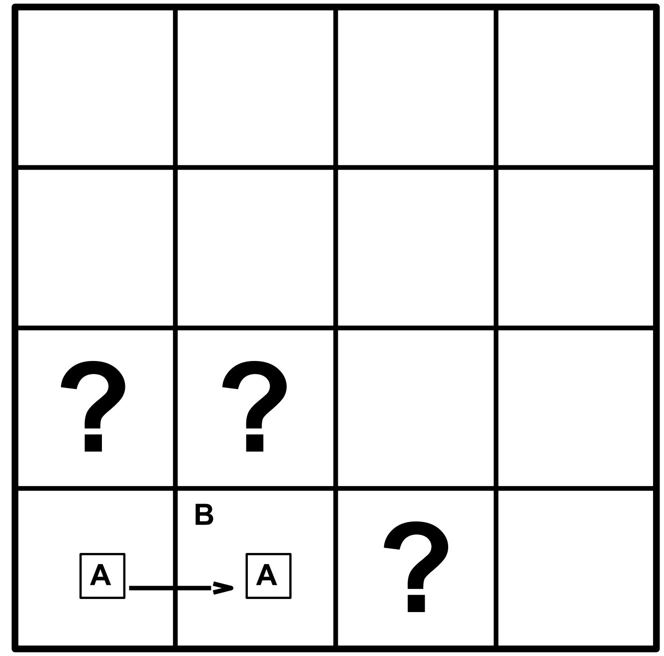
Models

- What about "*in a world* where x is true"? A "world"? What's that?
- Model = a possible "world".
  - Formally structured expression of world state with respect to which truth can be evaluated
  - Basically a collection of logical sentences describing a world or state
  - We say m is a model of a sentence α if α is true in m
  - Notation: M (α) is the set of all models of α



- Example:
  - KB = Giants won and Reds won
  - α = Giants won
  - Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$
    - KB entails α iff, in every model where α is true, KB is also true.
    - Note that KB is the stronger statement here: the "tighter" set of possible models.

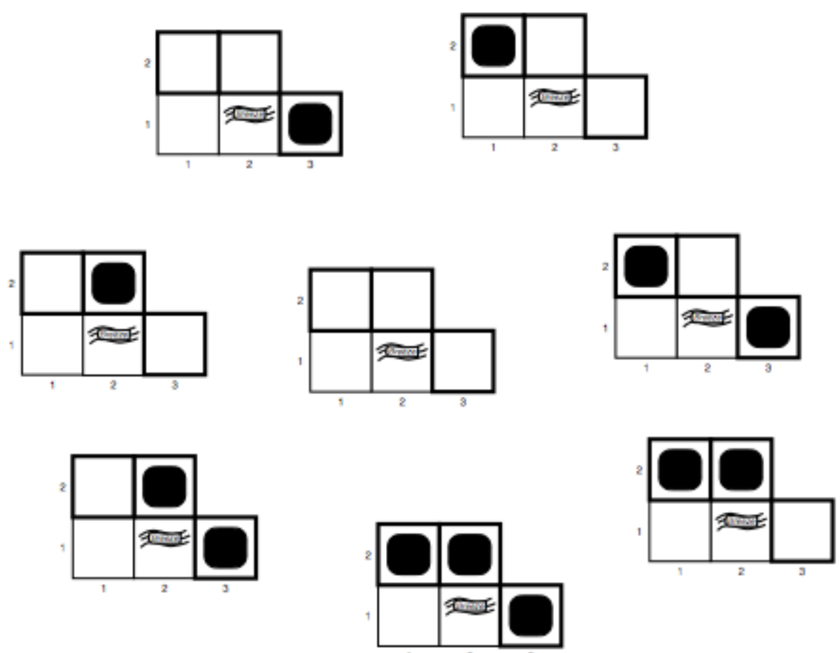
Example: Entailment in the wumpus world



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

Consider possible models for ?s assuming only pits:

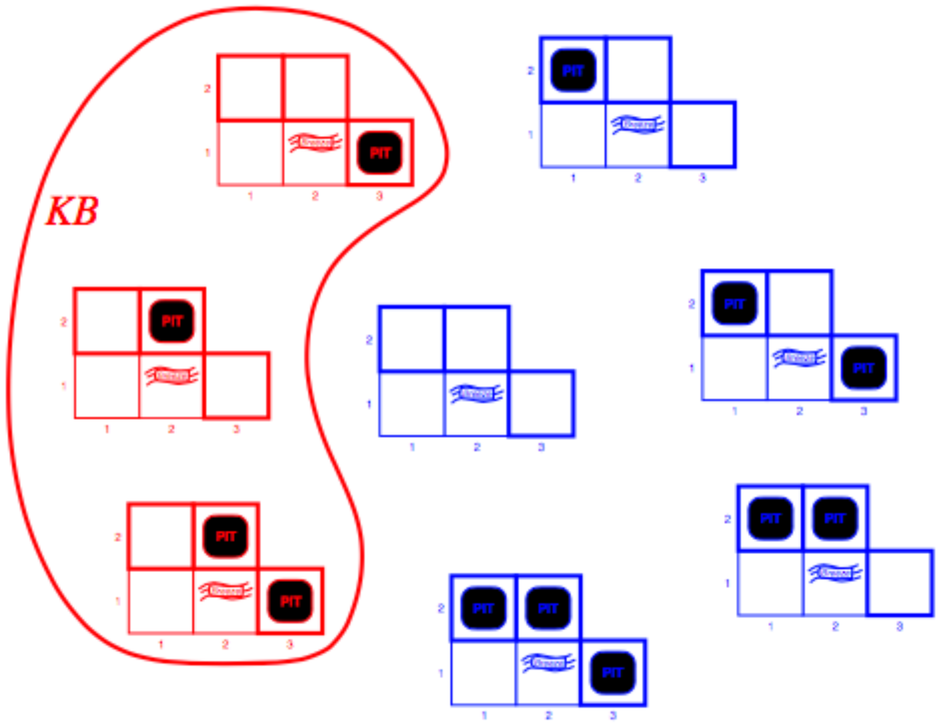
- Each square could contain a pit...or not
  - → 3 Boolean choices
  - → 8 possible models



Note:
The *full* model set for this world is large!
→ contains all possible combinations of possible contents for *every* square on board.

We are just looking at the subset dealing with the squares at the frontier of our exploration. Efficient!

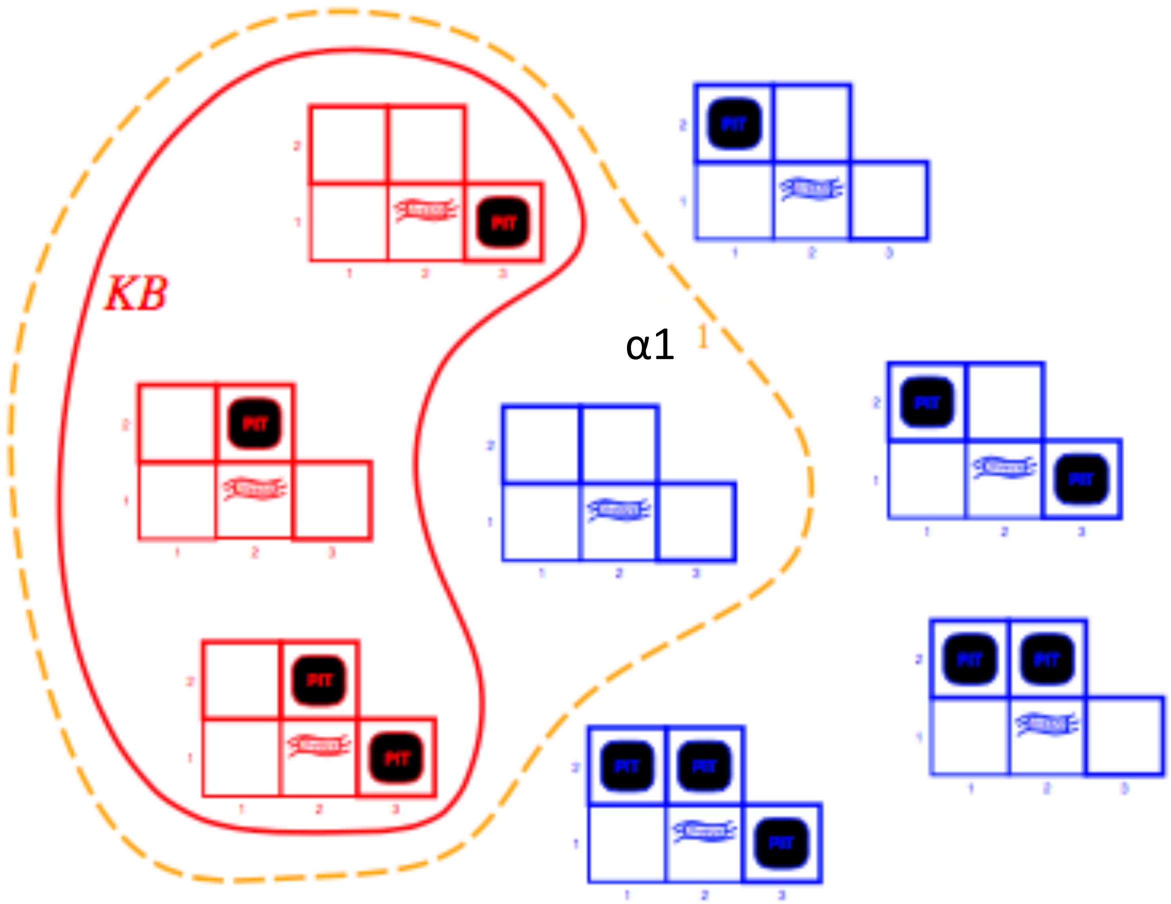
Wumpus models



- KB = wumpus world rules + observations (percepts)
  - Percepts = breeze([1,2]) , nothing([1,1])

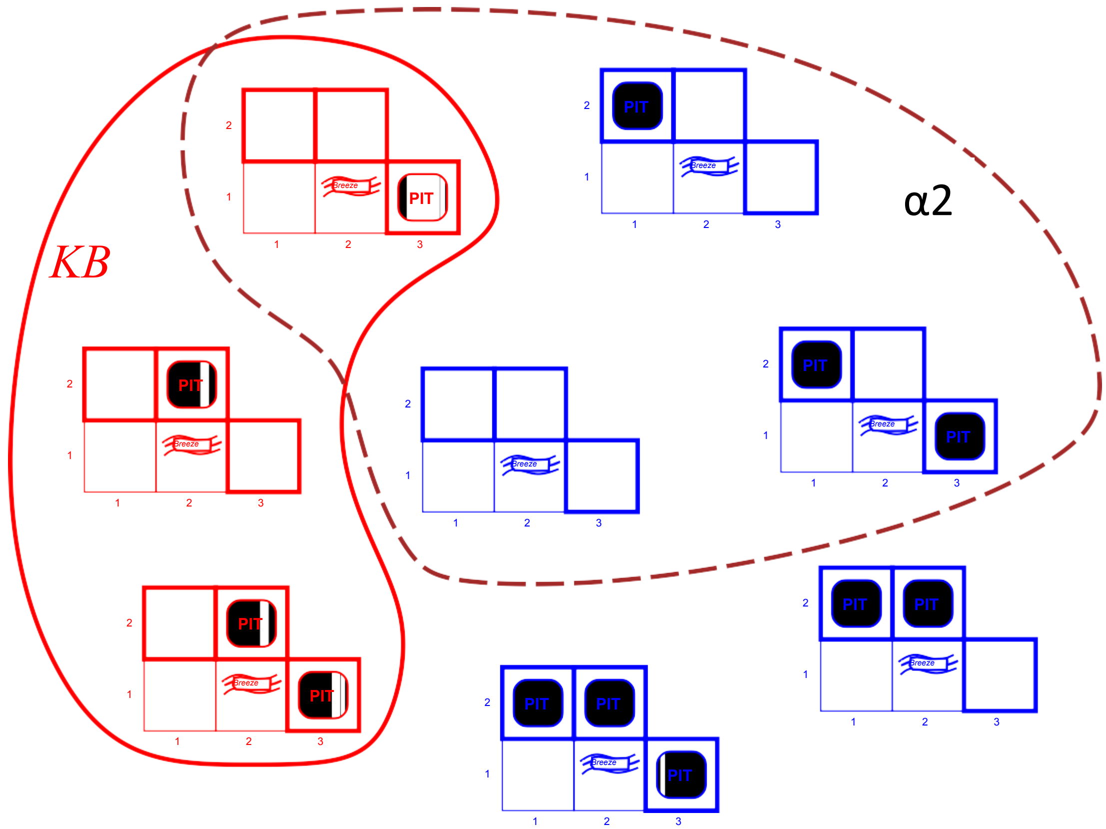
- Solid red line = all of the models in which KB is *true* = *M(KB)*
  - The state of the world represented by KB is consistent with the model

Wumpus models



- KB = wumpus-world rules + percepts
- Assertion α1 = “[2,1] is safe”
  - Dotted line is M(α1) = Set of all models in which α1 holds true.
- Then we can say that KB |= α1
  - In every model in which KB is true, α1 is also true. ➔ Proof by model checking
  - Thus: α1 is consistent with KB → “α1 is *derivable* from KB via model checking”

Wumpus models



Now let's consider another case:

- KB = wumpus-world rules + observations again, same as before
- α2 = "[2,2] is safe"

- Model checking shows that KB does **not** entail α2
- Can not conclude there is no pit in [2,2]
  - But also doesn't prove that there *is* one. Logical facts are simply inconclusive.

Logical Inference

- Model checking is one possible algorithm for logical inference
  - Plan: generate and test. Brute force.
    - Generate all possible models that could exist
    - Check that goal proposition (i.e. α) is true in all models in which KB is true
- KB $|-_i$ α → "sentence α can be derived from KB by procedure i"
    - KB $|-_{mc}$ α1 = "goal fact α1 can be derived from KB by model checking"
- Metaphor: "Logical consequences" of KB are a haystack; α is a needle.
  - Entailment = needle is in haystack: KB |= α (it's in there somewhere)
  - inference = finding the needle, i.e., proving the entailment
- Soundness: Inference algorithm i is sound if
  whenever KB $|-_i$ α, it is also true that KB |= α
  - Desirable! Unsound inference algo shows things entailed that aren't!
- Completeness: i is complete if
  whenever KB |= α, it is also true that KB $|-_i$ α
  - Desirable! A complete inference algo can derive any sentence (goal fact) that is entailed.

Propositional logic: Syntax

- Thus far: General logical concepts. Let's get concrete...
- Propositional logic is the simplest logic
  - Very basic, illustrates foundational ideas
  - So simple → also quite limiting. We'll need more power eventually...

- The proposition symbols P1, P2 simplest possible atomic sentences
  - The basic building blocks of propositional logic
  - Each represent a specific fact (e.g. $W_{1,2}$) that can be true or false

- Can be combined to form more complex sentences:
  - If S is a sentence, $\neg S$ is a sentence (negation)
  - If S1 and S2 are sentences, $S1 \land S2$ is a sentence (conjunction)
  - If S1 and S2 are sentences, $S1 \lor S2$ is a sentence (disjunction)
  - If S1 and S2 are sentences, $S1 \Rightarrow S2$ is a sentence (implication)
  - If S1 and S2 are sentences, $S1 \Leftrightarrow S2$ is a sentence (biconditional)

Propositional Logic: Semantics

- Each model specifies true/false for *each* proposition symbol
- Ex:

$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
False	False	True

← Pit in [3,1]. No pit in [2,2] and [1,2]

- With these three symbols: 8 possible models. Easily enumerated.

- Semantics: Rules for evaluating truth with respect to some model m
- For logical sentences S_i:

$\neg S$	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
i.e.,	is false iff	S_1	is true and	S_2	is false
(!!) i.e.,	is true if	S_1	is false and	S_2	T or F
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true and	$S_2 \Rightarrow S_1$	is true

- Simple recursive process evaluates arbitrary sentence
  - E.g.: $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = true \wedge (false \vee true) = true \wedge true = true$

Complete truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	
false	*true*	*true*	*false*	*true*	*true*	*true* *false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

- Interesting to note:
  - Implication (⇒). Non-intuitive: False *only* when P is true and Q is false.
  - Biconditional (⇔). "co-variance": True when both have same truth state.

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].
- Then: $\neg P_{1,1} \wedge \neg B_{1,1} \wedge B_{2,1}$ ➔ "no pit or breeze in [1,1], breeze in [2,2]
- How about: "Pits cause breezes in adjacent squares"?
  - Not possible in propositional logic. Can only state *specific facts.*
- $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$, $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$, etc. etc.
- "A square is breezy if and only if there is an adjacent pit" – stated *for each square*!

Truth tables for inference

Proposition Symbols

Models

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

- Wumpus KB (what we know):

R1: $\neg P_{1,1}$ no pit in [1,1]

R2: $B_{1,1} \Leftrightarrow (P_{2,1} \vee P_{1,2})$ B[1,1] only if pit in...

R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R4: $\neg B_{1,1}$ no breeze in [1,1]

R5: $B_{2,1}$ breeze in [2,1]

- Model Checking for entailment:
- KB is true if all rules (Rs) are true
  - True in just three models
- Some α is true if consistent across *all* true KB models
  - $\alpha = P_{2,1}$ → false in all three → deduce no pit [2,1]
  - $\alpha = P_{2,2}$ → Inconclusive...

Inference by enumeration

Depth-first enumeration of all models is sound and complete

```
function TT-Entails?(KB, α) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
            α, the query, a sentence in propositional logic

    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 ))
```

$O(2^n)$ for n symbols; problem is co-NP-complete

Propositional Theorem Proving

- So far: The only algorithm for proving entailment is model-checking
  - Have set of logical sentences KB, want to know if $\alpha = P_{1,2}$ is entailed
  - → generated 2^{Pi} models, check $M(KB) \subseteq M(\alpha)$
  - Gets expensive fast as the number logical facts (P_i) grows!

- **Propositional Theorem Proving**
  - Construct a proof of a sentence without consulting models
  - *Search* through a space of possible symbols transformations to connect KB with α.
  -

- Need three key concepts first:
  - Validity. A sentence is valid only if true in *all* models (tautology).
    - Ex. True, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
    - Gives us deduction theorem: A |= B if and only if $A \Rightarrow B$ is valid.
    - Can decide if A |= B by checking the $A \Rightarrow B$ true in all models!

  - Satisfiability. A sentence satisfiable if it's true in some model.
    - Earlier KB (R_1 through R_5) was satisfiable because true in 3 models.

Propositional Theorem Proving

- Last concept: Logical equivalence
  - To logical sentences A and B are *equivalent* if M(A)=M(b).
  - Meaning: A *equivalent* to B iff each entails the other → A |=B and B |=A
  - There are many equivalences established by standard rules of logic:

$(\alpha \land \beta)$	$\equiv$	$(\beta \land \alpha)$	commutativity of $\land$
$(\alpha \lor \beta)$	$\equiv$	$(\beta \lor \alpha)$	commutativity of $\lor$
$((\alpha \land \beta) \land \gamma)$	$\equiv$	$(\alpha \land (\beta \land \gamma))$	associativity of $\land$
$((\alpha \lor \beta) \lor \gamma)$	$\equiv$	$(\alpha \lor (\beta \lor \gamma))$	associativity of $\lor$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \lor \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \land \beta)$	$\equiv$	$(\neg\alpha \lor \neg\beta)$	De Morgan
$\neg(\alpha \lor \beta)$	$\equiv$	$(\neg\alpha \land \neg\beta)$	De Morgan
$(\alpha \land (\beta \lor \gamma))$	$\equiv$	$((\alpha \land \beta) \lor (\alpha \land \gamma))$	distributivity of $\land$ over $\lor$
$(\alpha \lor (\beta \land \gamma))$	$\equiv$	$((\alpha \lor \beta) \land (\alpha \lor \gamma))$	distributivity of $\lor$ over $\land$

Propositional Theorem Proving

- Validity and satisfiability are connected. Useful:
  - A is valid iff ¬A is unsatisfiable; A is satisfiable iff ¬A is not valid.
  - Thus: $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable
  - Basis for proof by contradiction! → Assume α false, show unsatisfiable

- Plus we have a number of standard logical inference rules:
  - Modus Ponens: $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$
  - And Elimination: $\dfrac{\alpha \wedge \beta}{\beta}$

- Plus: all of the logical equivalences can be used as inference rules

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ (implication elimination)

becomes

$$\frac{(\alpha \Rightarrow \beta)}{(\neg\alpha \vee \beta)} \quad \text{and} \quad \frac{(\neg\alpha \vee \beta)}{(\alpha \Rightarrow \beta)}$$

Propositional Theorem Proving Example:

that is, there is no pit in [1,2]. First, we apply biconditional elimination to R_2 to obtain

$$R_6: \quad (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) .$$

Then we apply And-Elimination to R_6 to obtain

$$R_7: \quad ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) .$$

Logical equivalence for contrapositives gives

$$R_8: \quad (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1})) .$$

Now we can apply Modus Ponens with R_8 and the percept R_4 (i.e., $\neg B_{1,1}$), to obtain

$$R_9: \quad \neg(P_{1,2} \lor P_{2,1}) .$$

Finally, we apply De Morgan's rule, giving the conclusion

$$R_{10}: \quad \neg P_{1,2} \land \neg P_{2,1} .$$

That is, neither [1,2] nor [2,1] contains a pit.

Wumpus KB

R1: $\neg P_{1,1}$

R2: $B_{1,1} \Leftrightarrow (P_{2,1} \lor P_{1,2})$

R3: $B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

R4: $\neg B_{1,1}$

R5: $B_{2,1}$

- Found this proof "manually", by hand
  - Needed cleverness and insight to find goal in directed manner.

- Could apply *any search* algo! Brute force!
  - Initial state: initial KB
  - Actions: applying all inference rules to all sentences → new Kbi
  - Result: Add bottom half of inference rule to KB_i to get Kb_{i+1}
  - Goal: goal state is when some KB_i generated contains target fact/query

Propositional Theorem Proving

- *Searching* for proofs in inference space is alternative to model checking
  - Often *much* more efficient: ignores facts (P_i's) irrelevant to target goal
  - Especially useful when the model space is complex (lots of P_i's)

- Searching for proofs is sound ... but is it complete?
  - Search algorithms like IDS are complete...*if a goal is reachable*.
  - Highly dependent on completeness of set of inference rules
    - Missing some critical inference rule → proof will not succeed.

- **Resolution Theorem Proving** solves this problem
  - Proof with a *single* inference rule (resolution)
  - Guaranteed complete algorithm if used with any complete search algorithm
  - But: requires all of KB to be clauses (see book disc.)
    - Clause = a *disjunction* of literals, e.g. $P_1 \vee P_2 \vee P_3 \vee P_4$
    - Luckily: any set of propositional logic can be turned into conjunctive normal form
  - *For any sentences A and B in propositional logic, a resolution theorem prover can decide if A |= B.*

Conversion to CNF

Plan: Apply various equivalences to "massage" into CNF

Example:

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

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

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

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

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

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

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

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

Not always super easy! But can be brute-forced with search!

Resolution Theorem Proving

- Basically: works by removing (resolving) contradictory literals.
- Example: Given KB:
  - R1: $\neg P_{1,1}$
  - R2: $\neg P_{1,3}$
  - R3: $\neg P_{2,2}$
  - R4: $P_{1,1} \vee P_{3,1} \vee P_{2,2}$
- Then:
  - R1 resolves with R4 to give R5: $(\neg P_{1,1} \vee P_{1,1}) \vee P_{3,1} \vee P_{2,2} = P_{3,1} \vee P_{2,2}$
  - R2 resolves with R5 to give R6: $P_{3,1}$
  - – At end of resolution we have inferred a specific fact!

- Full Resolution inference rule:

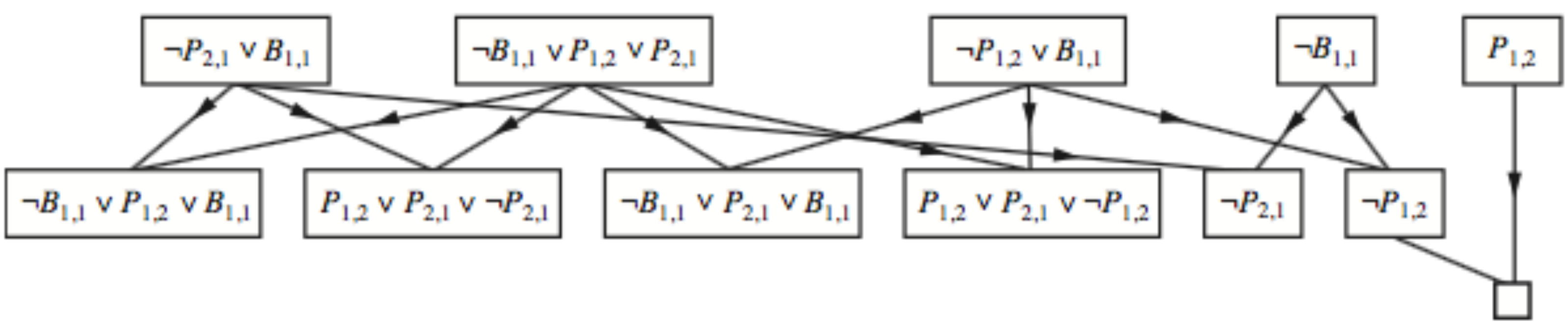
$$\frac{(a_1 \vee a_2 \vee a_3 .. \vee a_n) \wedge (m_1 \vee m_2 \vee m_3 \vee ... \vee m_n)}{(a_1 \vee a_3 .. \vee a_n) \wedge (m_1 \vee m_3 \vee ... \vee m_n)}$$

where a_2 and m_2 are complementary literals.

- So each resolution step:
  - – Considers a logical sentence in CNF (i.e. two clauses in your KB)
  - – Resolves to two new clauses → each with complementary literals removed.

Algorithm: Resolution Theorem Proving

- Idea: Proof by contradiction
  - Want to show that KB |= α → so show that (KB ∧ ¬α) is unsatisfiable
- Plan:
  - Convert (KB ∧ ¬α) into CNF
  - Exhaustively apply resolution to all pairs of clauses with complementary literals
  - Continue process until:
    - There are no new resolutions to make
      - Could not show unsatisfiability → KB does **not** entail α
    - Two clauses resolve to *empty clause*
      - $a_1 \vee a_1$ resolves to { } = essentially "false"
      - Unsatisfiability is shown → KB |= α
- Example:



Inference with Horn Clauses

- Resolution theorem is complete ... but also complex
  - Many practical cases: Don't need all this power (and complexity!)

- Inference with Horn clauses
  - If your KB can be expressed within a restricted rule format
  - Horn clause: Disjunction in which *at most* one element is positive
    - Ex: $(\neg L_{1,1} \lor \neg Breeze \lor B_{1,1})$; $\neg B_{2,2}$
    - No positive literals = goal clause
    - Can be rewritten as implications: $(L_{1,1} \lor Breeze) \Rightarrow B_{1,1}$
    - LHS= premise (body); RHS = consequent (head)

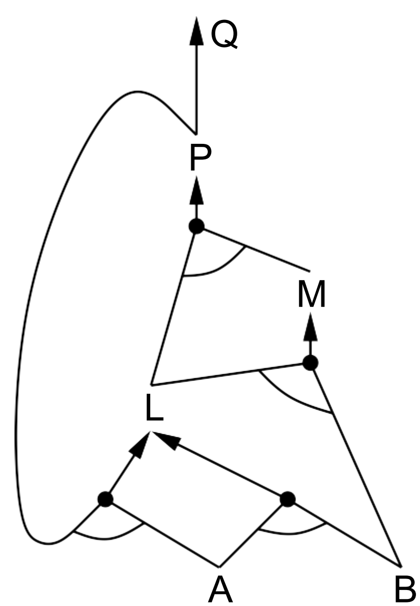
- Can be use in forward/backward chaining proof algorithm
  - These algorithms are very natural and run in linear time !

Forward Chaining

- Idea: Work forward from the known facts to try to reach the target goal
  - Start with known facts → true by definition
  - Repeat:
    - fire any rule whose premises are satisfied in the KB,
    - add its conclusion to the KB
  - Until: query is found (proved!); or no more facts added to KB (stalled, failed)

- Visually: Can represent the H-clauses in the KB as a directed graph.
  - Forward chaining: start with facts and traverse the graph

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



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

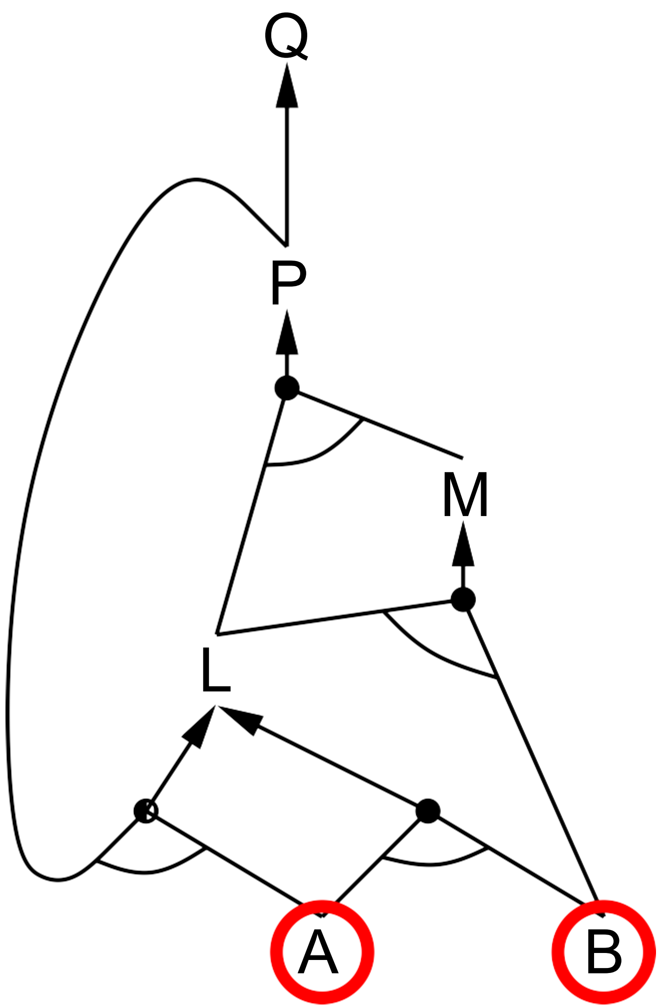
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward Chaining

- Idea: work backwards from the query *q*:
- The Plan: A simple (recursive!) algorithm
  - Initialize: Push *q* on the "proof stack" = things to be proven
  - Repeat:
    - Pop next fact q_i to prove off proof stack
    - Check if q_i is known to be true (fact in KB). If so, continue
    - Else search KB for rule R_j with head = q_i (a way to prove q_i)
    - Add premises of R_j to the proof stack
  - Until:
    - Proof stack is empty (success); or
    - no change in proof stack

  - 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

- Visually: Can represent the H-clauses in the KB as a directed graph.
  - Backward chaining: start with target goal and traverse the graph
  - Done if/when all leaves of search are facts

Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

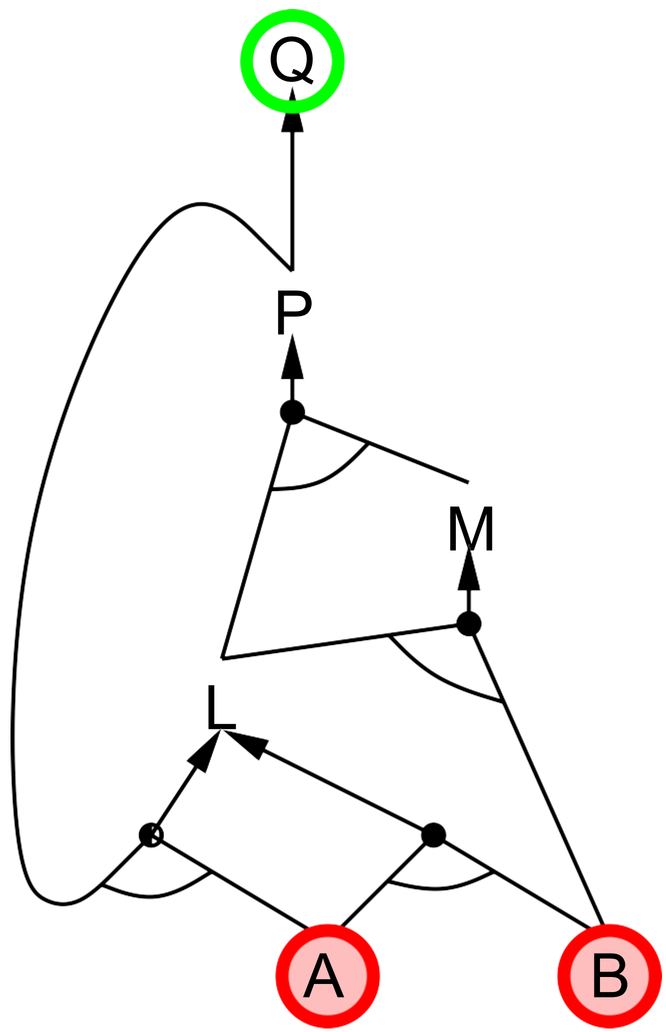
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Stack:

Summary: Inference Approaches

- Model Checking
  - Simple, complete ... But exponential in number of symbols (features) in KB
- Proposition Theorem proving by inference rules (Modus Ponens, etc.)
  - Implemented as search though proof space to find goal
  - Could be incomplete!
- Resolution theorem proving
  - Universal and guaranteed complete
  - ... but also arduous and complex
- Forward/Backward Chaining with Horn clauses
  - Possible in contexts where rules can be massaged in to Horn-clause form
  - Complete, straightforward, and efficient (linear time in size of KB)
  - FC: data-driven. Good for routine, automatic, continuous processing
    - Non-goal directed, e.g., dynamic facial recognition, routine decision-making
    - May do lots of inferring that is *irrelevant* to proving some goal
  - BC: goal-driven. Good for answering specific questions (posed as goals)
    - Complexity often much less than linear in size of KB
    - Basis for Prolog language

Summary: Inference Approaches

- 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

- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

- Propositional logic lacks expressive power

α β $\subseteq$ $\neg$ $\Rightarrow$ $\models$ $\wedge$ $\vee$
$\Leftrightarrow$