Lectures 7: Logical Agents

Artificial Intelligence

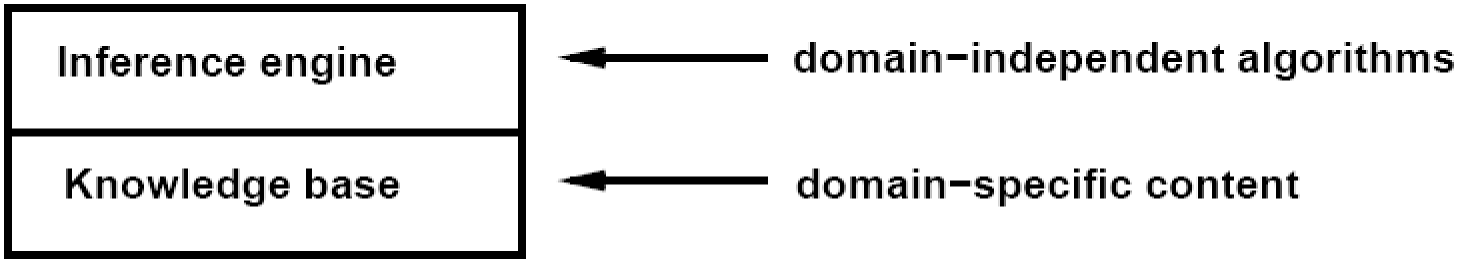
CS-6364

Logical Agents

<u>Main concepts</u>: *representation of knowledge and reasoning processes*

- *Knowledge Bases*
- *Logic in general: models and entailment*
- *Propositional Logic*
- *Equivalence, validity, satisfiability*
- *Inference rules and theorem proving*
  - *Forward chaining*
  - *Backward chaining*
  - *resolution*
- *Reasoning Patterns*

Knowledge Bases



- Knowledge base = set of sentences in **a formal language**

The two most important components of AI systems are: Knowledge base and reasoning

A KB is a set of representations of facts; representations are called sentences. Unlike in DB, the KB representation is such that it allows reasoning.

The key issues that need to be addressed are: how to represent the knowledge, and to find inference rules that allow us to reason on the KB.

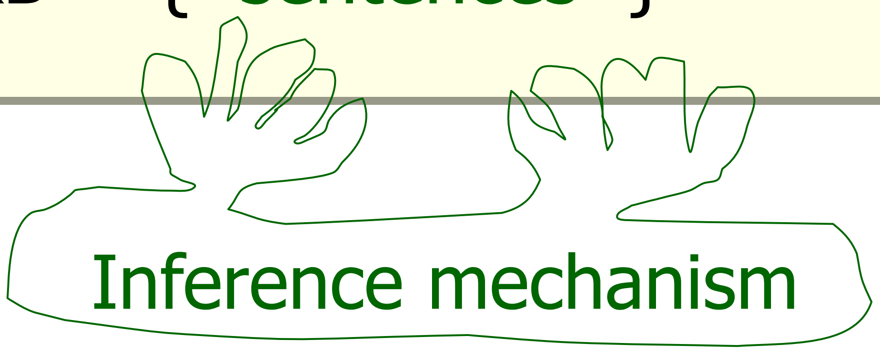
The agent perceives the outside world and puts more facts on the KB (TELLs the KB), and then makes decisions about future actions by ASKing itself what to do – answers should follow from the KB

What do you do with knowledge?

- A way of adding sentences to the knowledge base TELL
- A way of querying what is known ASK
- Determining what follows from the KB to satisfy a query is the job of *the inference mechanism*.

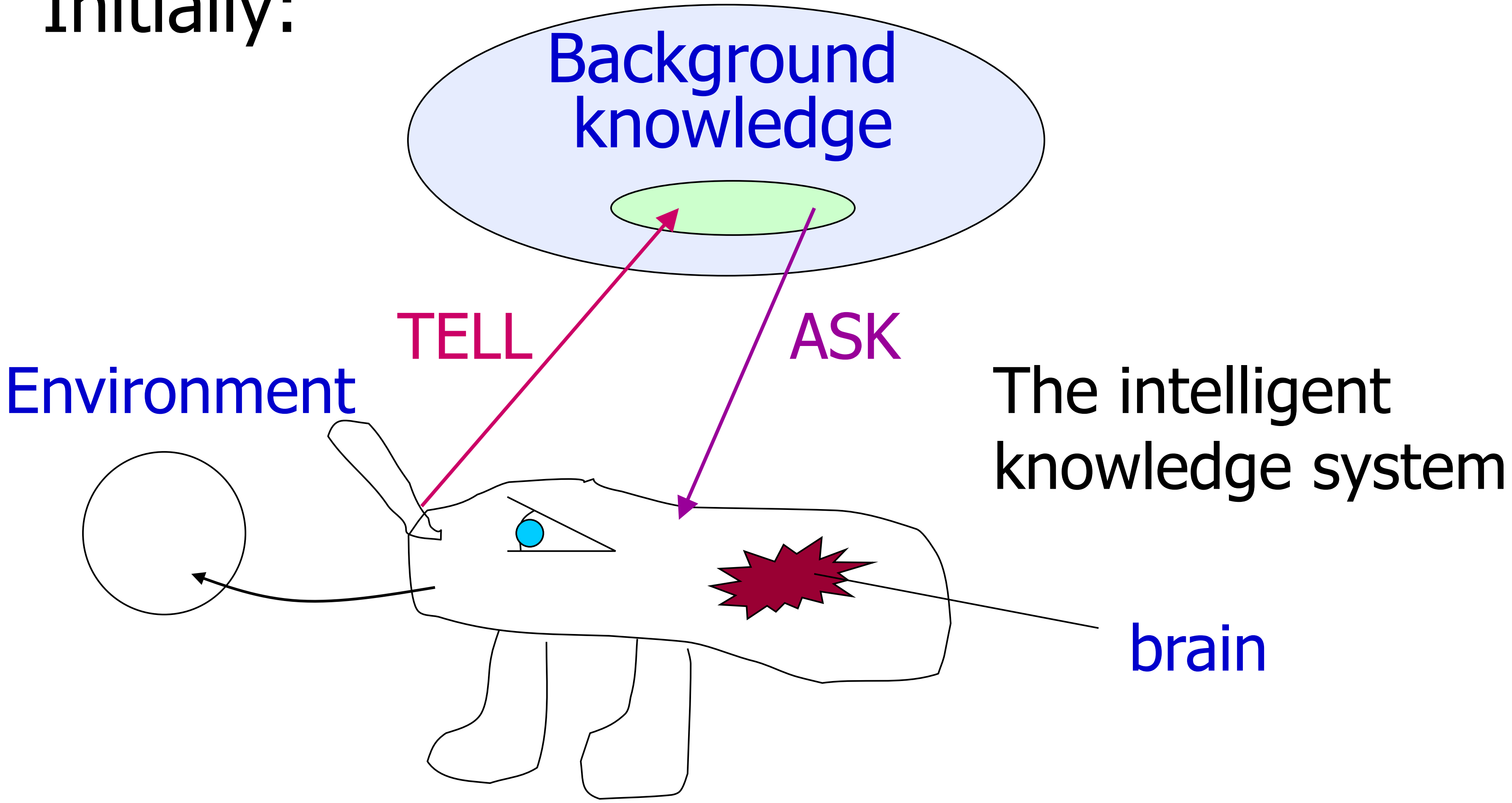
$\Downarrow$

- KB = { sentences }



More about knowledge

Initially:



A generic 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

Knowledge-Based Agents

One can distinguish several knowledge levels:

<u>Level</u>	<u>Primitives</u>
Epistemological level	Concept types, inheritance and structuring relations
Logical level	Propositions, predicates, logical operators
Implementation level	Atoms, pointers, data structures

<u>Example:</u>

Epistemological level: Golden Gate Bridge links San Francisco and Marin County.

Logical level: Links (GG Bridge, SF, Marin)

Implementation level: *is where the agent architecture is run. It is where sentences are physically implemented in the most efficient way*.

Motivating example

The Wumpus world

What is Wumpus?

- based on an early computer game that explores a cave consisting of rooms connected by passageways.

➢ The Wumpus is a beast that eats anyone who enters the room

➢ To make things worse, some rooms contain bottomless pits that will trap anyone who wonders in these rooms

What is good in the Wumpus world?

The occasional heap of gold

➢ Performance Measure ⇒ *+1000 for picking up the gold, -1000 for falling into a pit or being eaten by the wumpus, -10 for using the arrow, -1 for each action*

Agent's actuators

- Go forward
- Turn right 90°
- Turn left 90°

+ Grab (to pick up an object that is in the same square as the agent)

+ Shoot (to fire an arrow in straight line in the direction the agent is facing)

+ Climb (to leave the cave !!)

5 sensors

1) in the <u>square containing the wumpus</u> directly adjacent to the agent, the perception is <u>stench</u>
2) in the squares adjacent to pit it perceives a <u>breeze</u>
3) in the square where gold is, it perceives a <u>glitter</u>
4) when the agent walks into a wall, it perceives a <u>bump</u>
5) when the agent is killed, it gives out a woeful scream that can be perceived anywhere in the cave

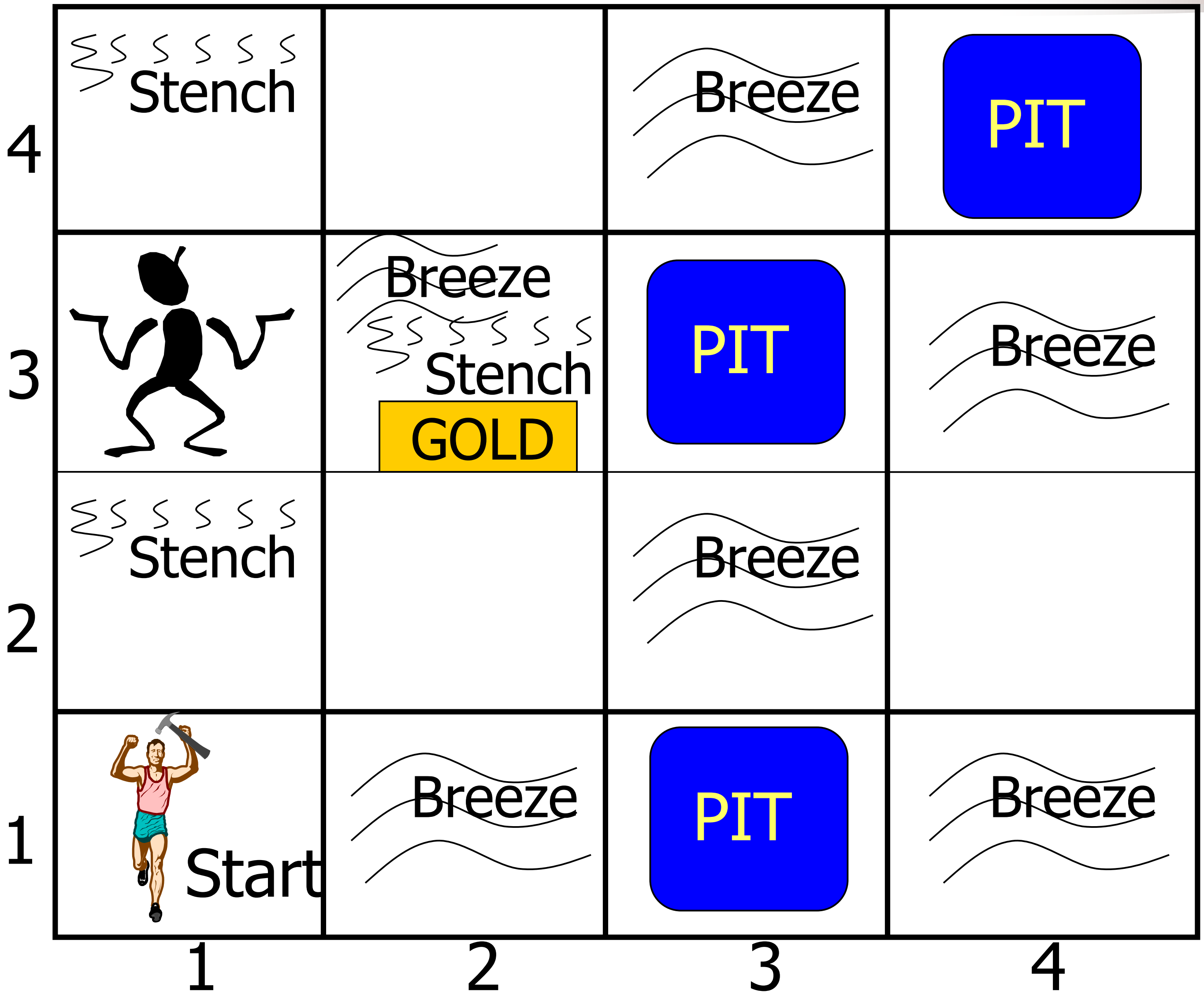
How are the percepts delivered?

The agent gets them in the form of a list of 5 symbols

- Example: there is a stench, a breeze, a glitter, but no bump or scream:

  [Stench, Breeze, Glitter, None, None]

A typical Wumpus world



Wumpus world characterization

- Fully 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
- Single-agent? Yes – Wumpus is essentially a natural feature

The agent's goal

Find the gold and bring it back to the start as quickly as possible! Without being killed!

$\Rightarrow$ Utility functions

- 1000 points awarded for climbing out of the cave while carrying the gold!
- 1 point penalty for every action taken
- 10,000 penalty points for being killed

Acting and reasoning in the wumpus world

- we know now the rules
- *but we do not know how the wumpus agent should act*

Initially:

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2	3,2	4,2
1,1 $\boxed{\textbf{A}}$ **OK**	2,1 **OK**	3,1	4,1

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

After one move

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2	3,2	4,2
1,1 **A** **OK**	2,1 **OK**	3,1	4,1

Initially

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2 **P?**	3,2	4,2
1,1 **V** **OK**	2,1 **A** **B** **OK**	3,1 **P?**	4,1

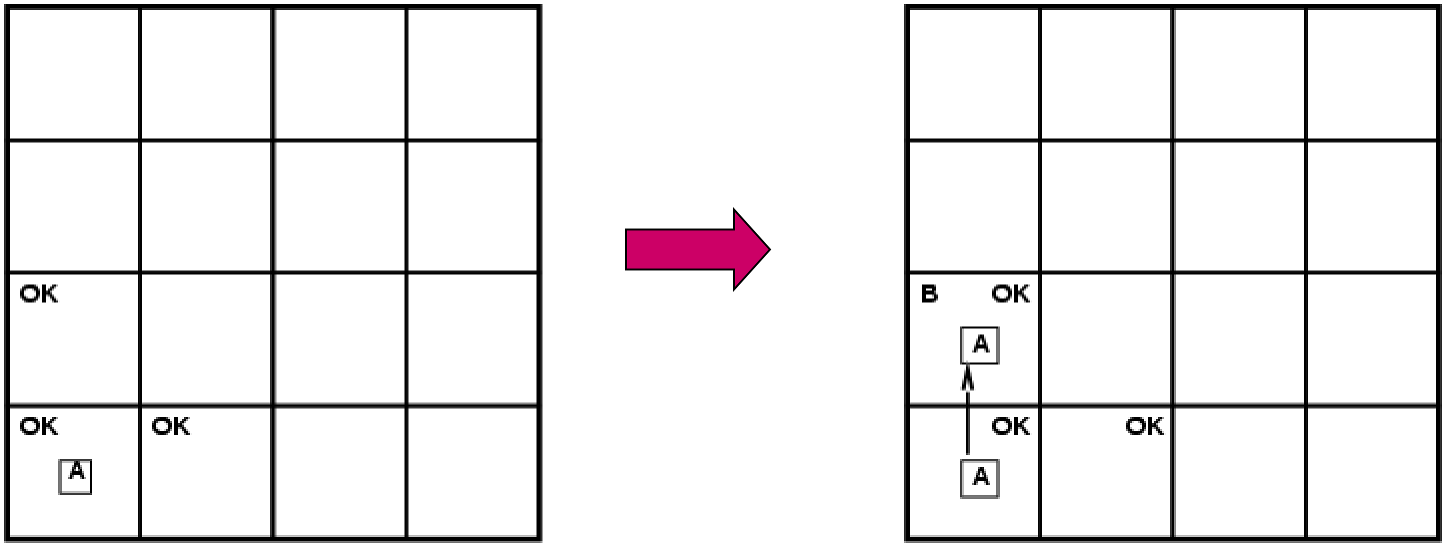
After one move

Two later stages

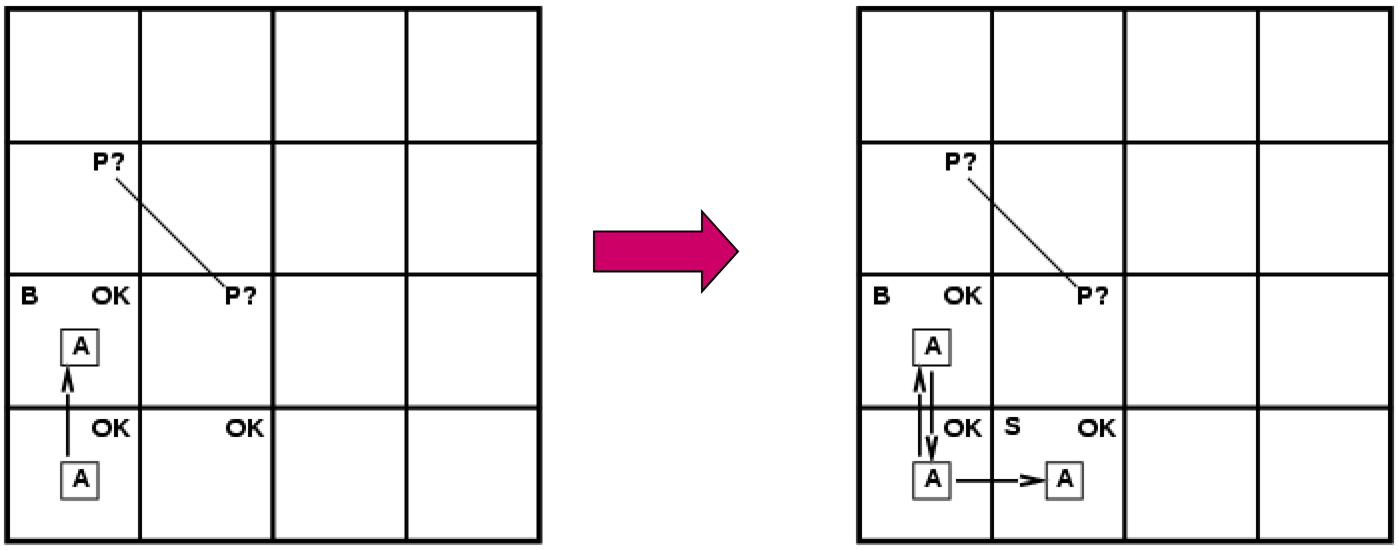
1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 V S OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

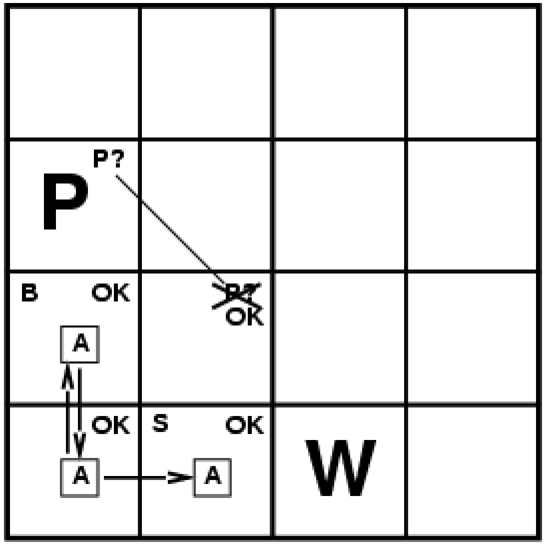
Exploring a wumpus world –once again

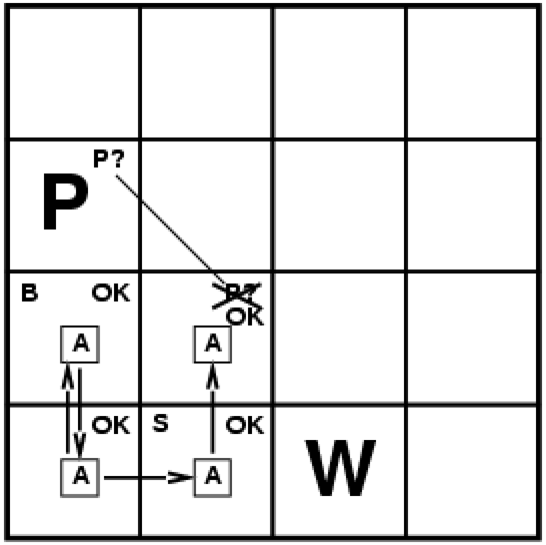


Exploring a wumpus world

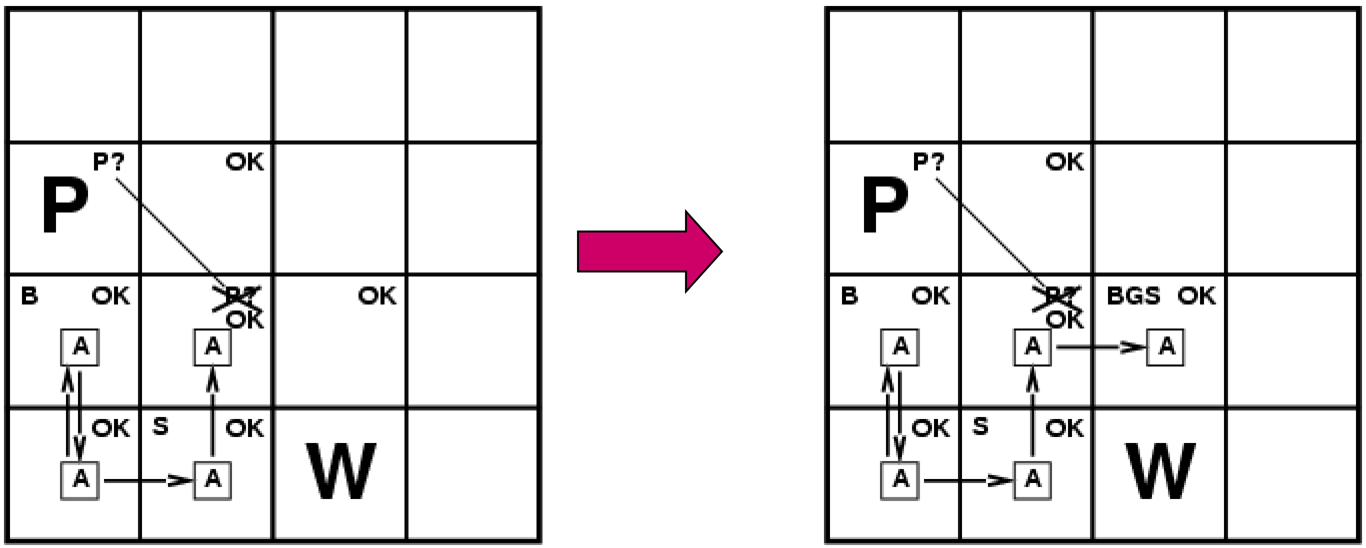


Exploring a wumpus world





Exploring a wumpus world



Testing

- We specify a full class of environments & insist the agent to do well over the whole class

- Choose the location of gold and wumpus randomly, with a uniform distribution.

- The probability of a square to contain a pit = 0.2

Results

- In 21% of the environments, there is no way the agent can get a positive score. Why?

The gold is in a pit or surrounded by pits.

- In some environments, the agent must choose between going home empty-handed or taking the chance that could lead either to death or gold!

- In most of the environments, the agent can safely retrieve the gold

Conclusions

- The actions are based on deductions and beliefs
- They are the connectors between knowledge bases and inference (reasoning)
- Together, K representation and reasoning support the operation of a knowledge-based agent

- *In each case where the agent draws a conclusion from the available information, that conclusion is guaranteed to be correct if the available information is correct*

Logic in general

Logics are formal languages for representing information such that conclusions can be drawn

- Syntax defines the sentences in the language
- Semantics define the "meaning" of sentences;
  - i.e., define truth of a sentence in a world

- E.g., the language of arithmetic
  - $x+2 \geq y$ is a sentence; $x2+y > \{\}$ is not a sentence
  -
  - $x+2 \geq y$ 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 $x = 0, y = 6$

KBs are expressed as sentences in a knowledge representation language. A knowledge representation language is defined by 2 aspects:

- the syntax describing all the possible configurations that constitute sentences
- the semantics describes the facts in the world to which sentences refer

Semantics and Logical Reasoning

- SEMANTICS defines the *truth* of each sentence with respect to each *possible world*
- *Each sentence must be either true or false*
- *Model = possible world*

The notion of truth being defined, we can define logical reasoning.
*- The relation of **logical entailment** between two sentences*
$\alpha \models \beta$; α entails β ; in every model in which α is true, then β must be true as well.

Logical reasoning is based on the concept of **entailment** - a sentence follows logically from another sentence.

Entailment

- Entailment means that one thing follows from another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true
  - E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"
  - E.g., x+y = 4 entails 4 = x+y
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

Models

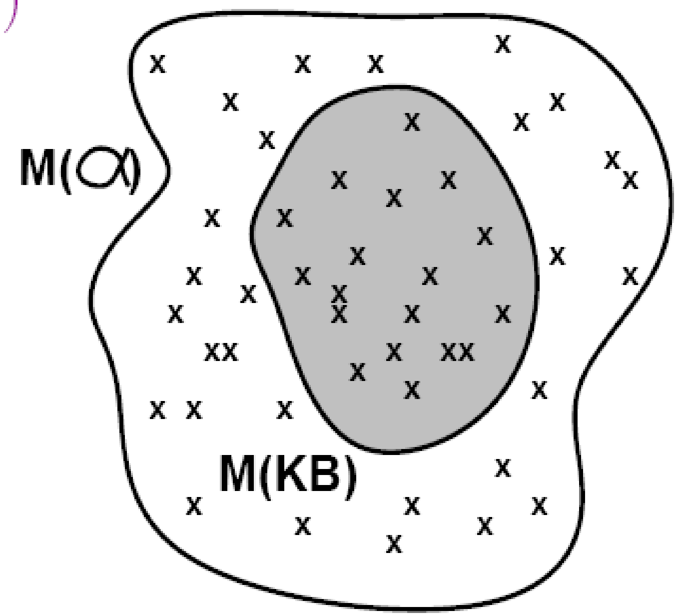
Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

We say m is a model of a sentence α if α is true in m

$M(\alpha)$ is the set of all models of α

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

E.g. KB = Giants won and Reds won
α = Giants won



Apply entailment to the Wumpus

- The agent has detected nothing in [1,1], and breeze in [2,1]. The KB is made up of these 2 percepts as well as the Wumpus rules
- The agent is interested in knowing whether the adjacent squares [1,2], [2,2] and [3,1] contain pits.
- Each of the three squares might or might not contain a pit – there are $2^3=8$ possible models.

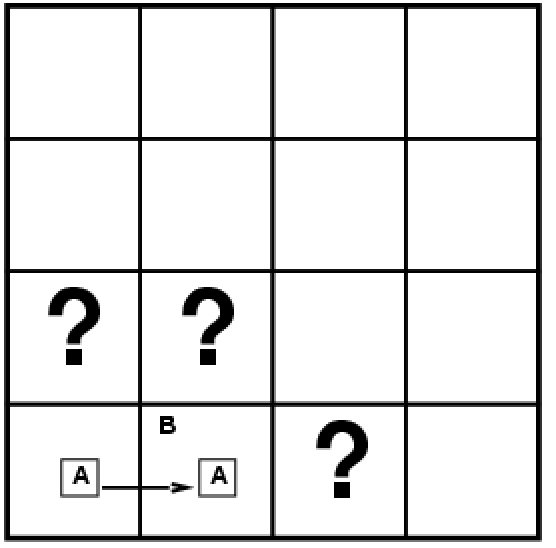
1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2 **P?**	3,2	4,2
1,1 **V** **OK**	2,1 **A** **B** **OK**	3,1 **P?**	4,1

Entailment in the wumpus world

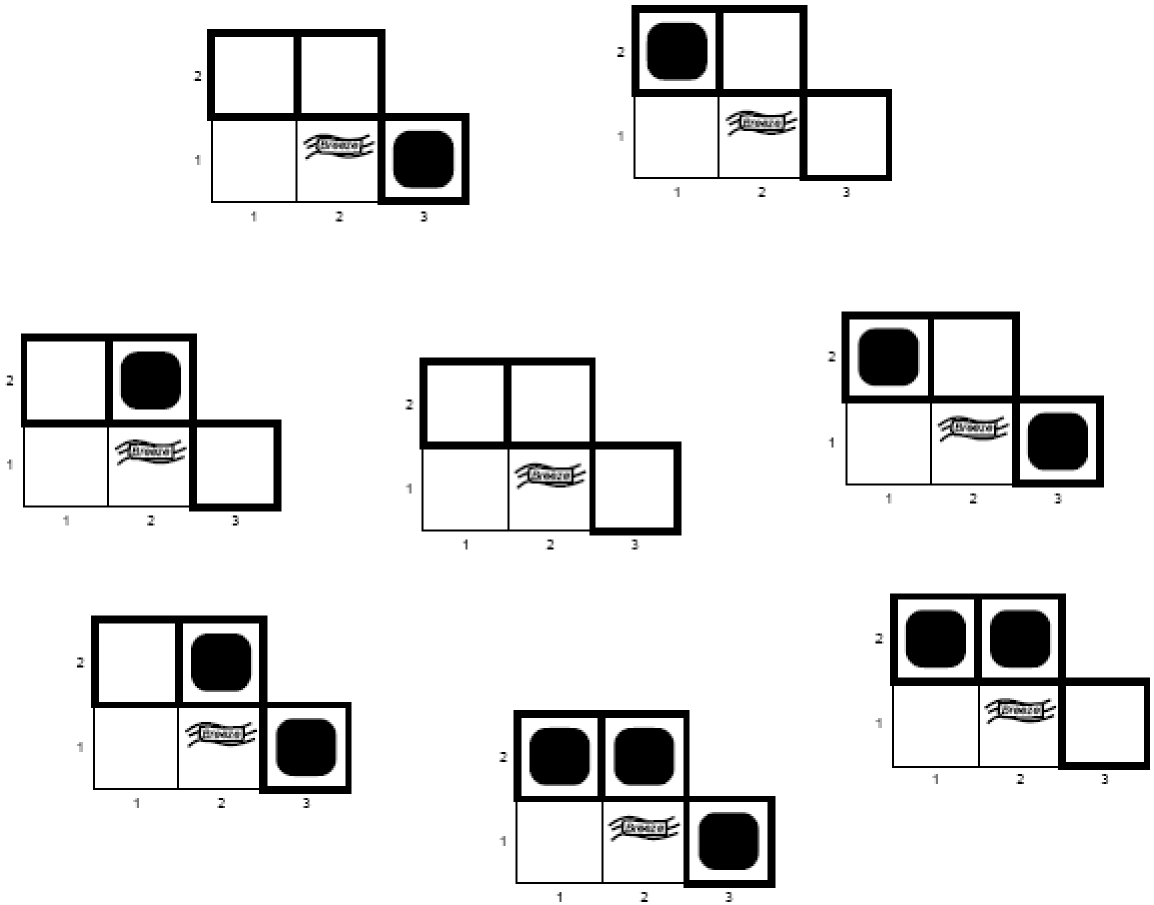
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for *KB* assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models

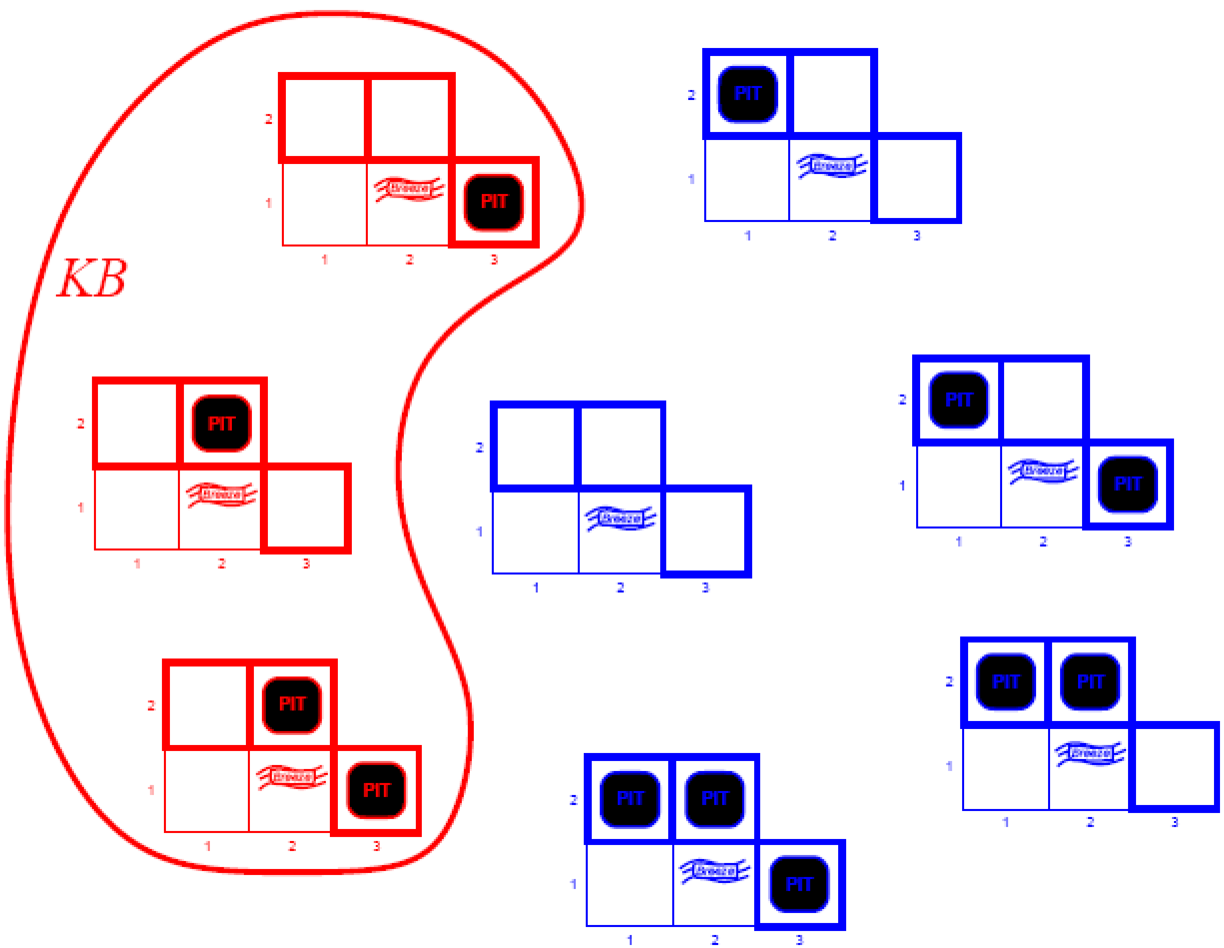


Wumpus Models



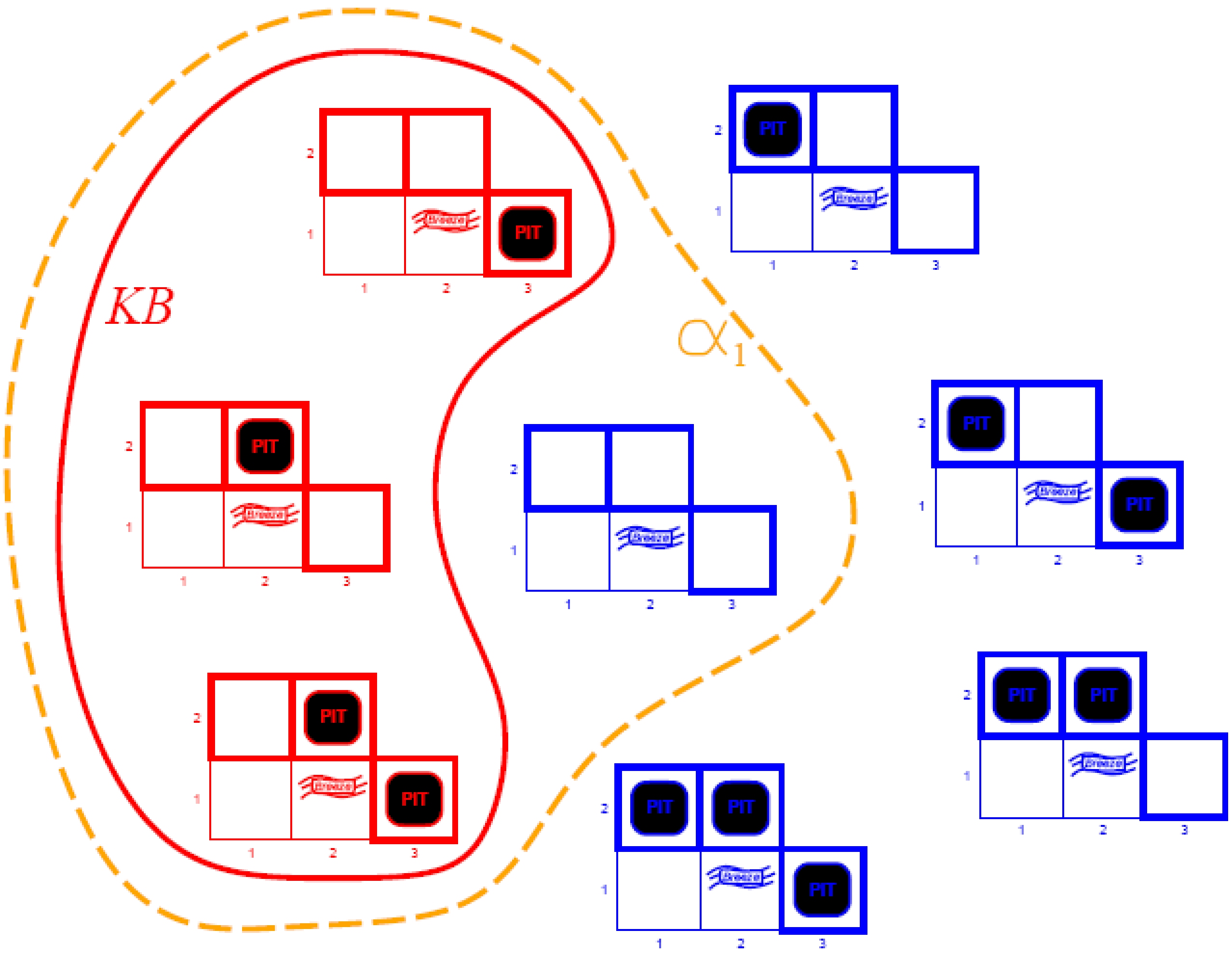
Given observations of nothing in [1,1] and breeze in [2,1]. The agent is interested in knowing whether the adjacent squares [1,2], [2,2] and [3,1] contain pits: 8 possibilities

KB=wumpus-word rules + observations



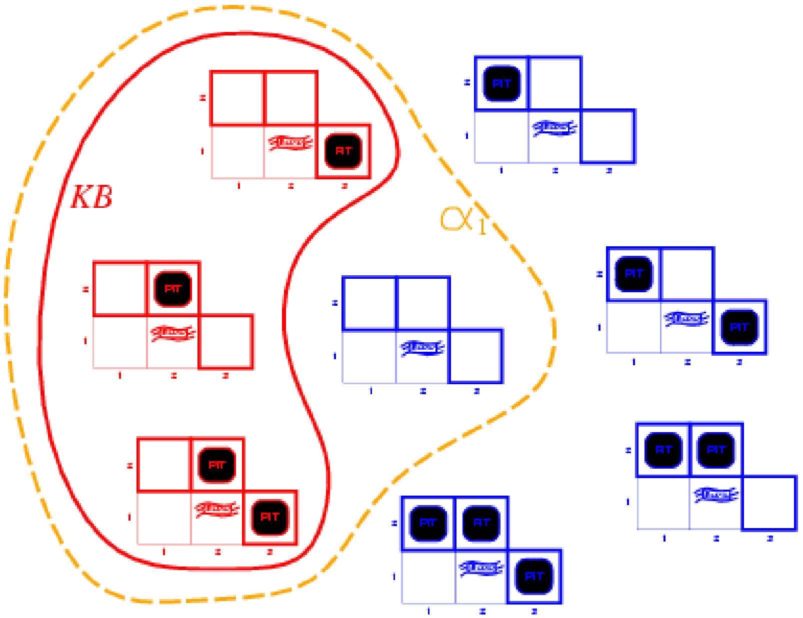
Given observations of nothing in [1,1] and breeze in [1,2].
Breeze in [1,2] translates in 3 statements in the KB: (a) a pit could exist only in [1,3]; (b) a pit could exist only in [2,2]; and (c) we could have a pit in both [1,3] and [2,2]

Model 1



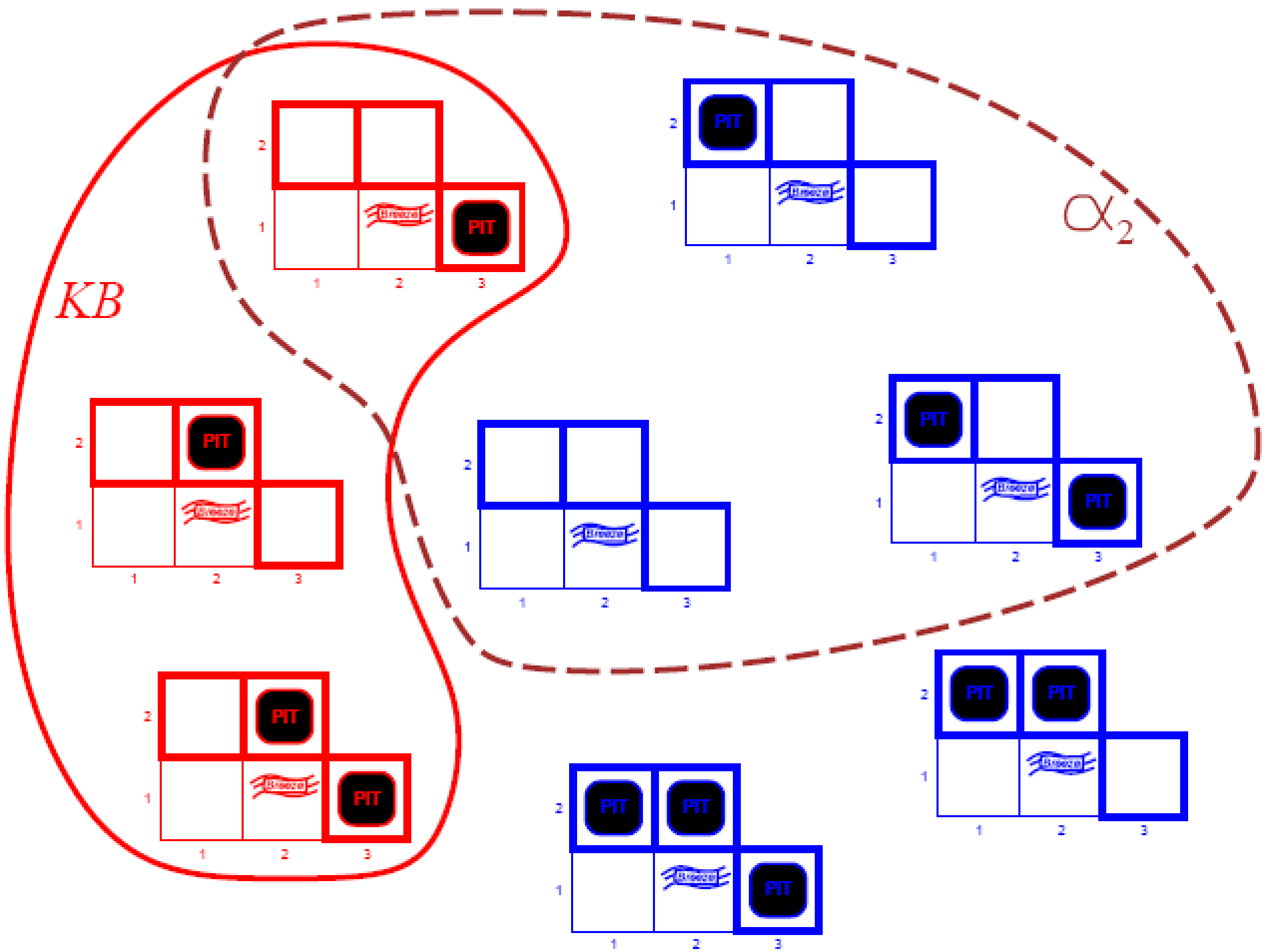
Models of the KB and α1, i.e. no pit in [1,2]

Wumpus models



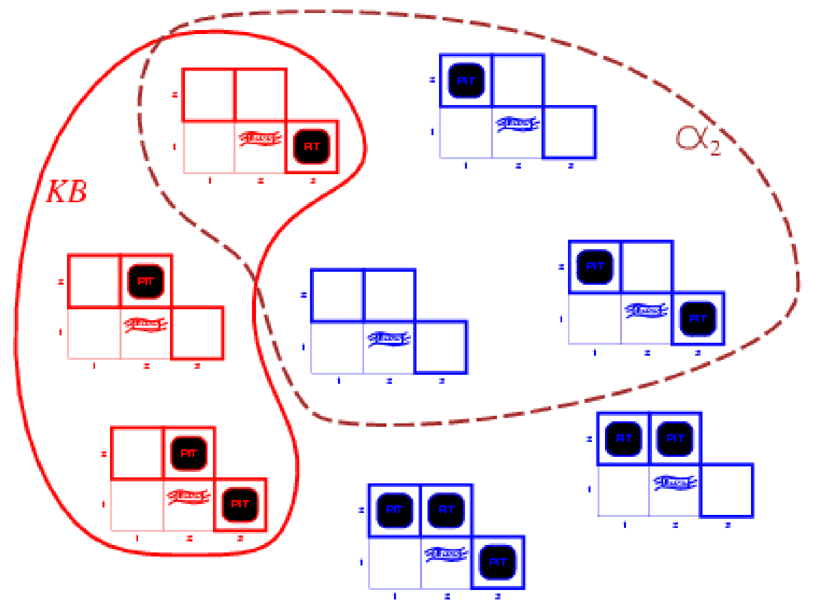
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", *KB* $\models \alpha_1$, proved by model checking

Model 2



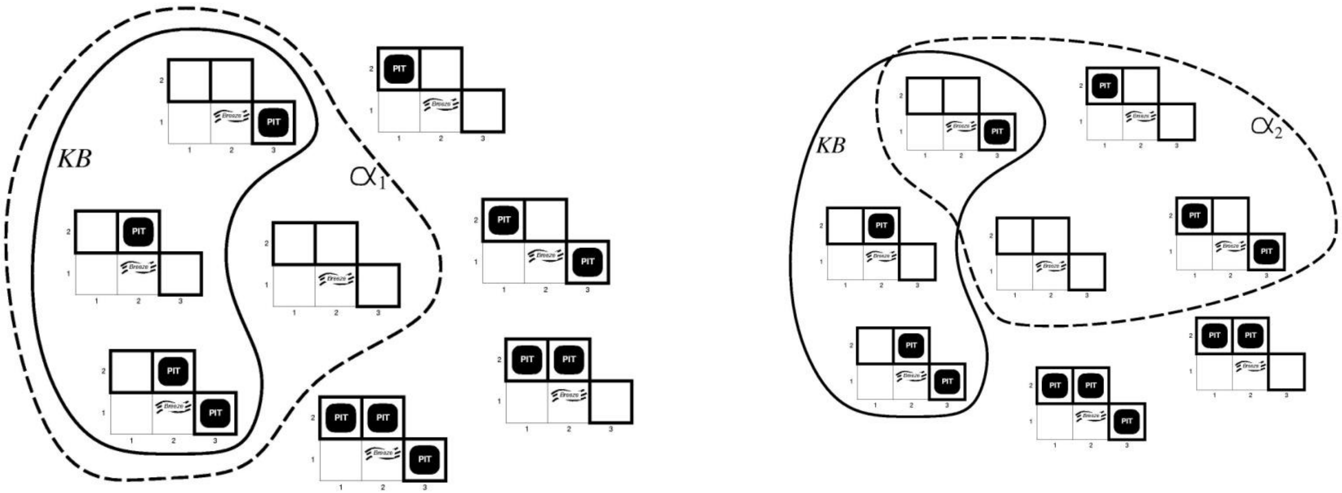
Models of the KB and α2, i.e. no pit in [2,2]

Wumpus models



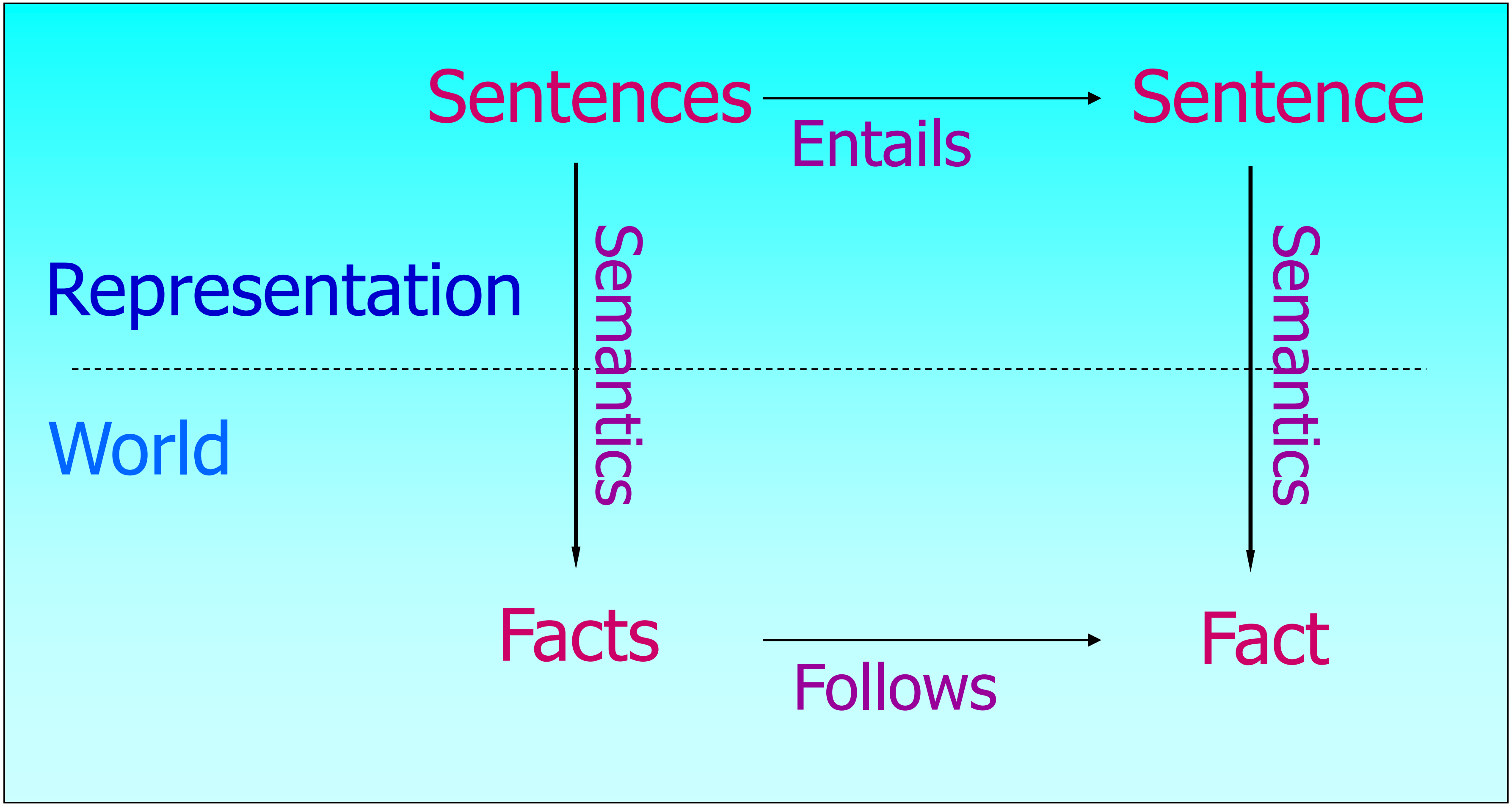
- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Model checking



- The KB is false in models that contradict what the agent knows – for example, in any models in which [1,2] contains a pit, it is a contradiction, because there is no breeze in [1,1]
- Let us consider two possible conclusions:
  - α_1 = "There is no pit in [1,2]"
  - α_2 = "There is no pit in [2,2]"
- We see that in every model in which KB is true, α_1 is also true, hence
- KB $|= \alpha_1$
- But in some models for which KB is true, α_2 is false, hence KB $|\neq \alpha_2$

Entailment and reasoning



Inference mechanisms

An inference mechanism that derives only entailed sentences is called **sound,** or **truth preserving**.

An inference mechanism is **complete** if it derives any sentence that is entailed by the KB.

Facts and rules are reflected as sentences in KB

- They are considered true

- Every way of generating a new sentence which is also true is part of the inference mechanism

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from *KB* by procedure *i*

Soundness: *i* is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: *i* is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

$KB \models \alpha$

- Then we say that sentence α is entailed (or inferred) by KB

- Thus an inference procedure can do 2 things:

  - Given a KB, it can generate new sentences α
  - Given a KB and a sentence α it can determine whether α was entailed by KB or not!

- An inference procedure that derives only entailed sentences is called sound or truth-preserving.

- Convention An inference procedure *i* can be described by the sentences it derives. If *i* can derive α from KB, KB $\models_i \alpha$: "Alpha is derived from KB by i"

A very simple logic: Propositional Logic

- Define a logic through:
  - Syntax
    - Defines the *allowable* sentences
  - Semantics
    - Defines the rules that determine *the truth value* of sentences
  - Inference
    - Reasoning patterns

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)

Backus-Naur Representation of Propositional Logic Grammar

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
& \mid & \neg\ \mathit{Sentence} \\
& \mid & \mathit{Sentence} \wedge \mathit{Sentence} \\
& \mid & \mathit{Sentence} \vee \mathit{Sentence} \\
& \mid & \mathit{Sentence} \;\Rightarrow\; \mathit{Sentence} \\
& \mid & \mathit{Sentence} \;\Leftrightarrow\; \mathit{Sentence} \\
\\
\text{OPERATOR PRECEDENCE} & : & \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

Figure 7.7 A BNF (Backus–Naur Form) grammar of sentences in propositional logic, along with operator precedences, from highest to lowest.

Ambiguity of Propositional logic Grammar

- Example: $P \land Q \lor R$ can be interpreted:

  $(P \land Q) \lor R$ or

  $P \land (Q \lor R)$

- Solution: Precedence of operators

  $\neg \, , \land \, , \lor \, , \Rightarrow \, , \Leftrightarrow$

  higher precedence → lower precedence

- Example: $\neg\, P \lor Q \land R \Rightarrow S$ is equivalent to

  $(\neg\, (P) \lor (Q \land R)) \Rightarrow S$

Propositional Logic Semantics

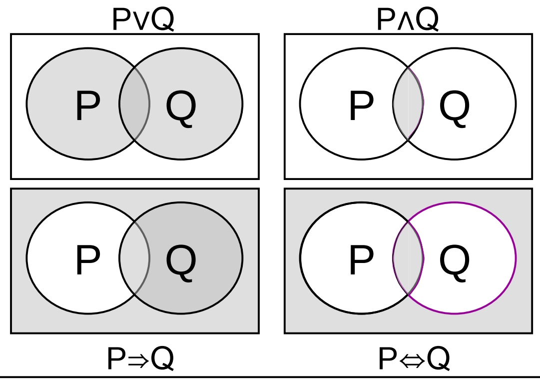
<u>Semantics</u> results from the meaning of proposition symbols, constants and logical connectives.

A Sentence is:

- Valid if is true for <u>all</u> interpretations
- Satisfiable if is true for <u>some</u> interpretations
- Unsatisfiable if is false for <u>all</u> interpretations

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*

Figure 7.8 Truth tables for the five logical connectives.

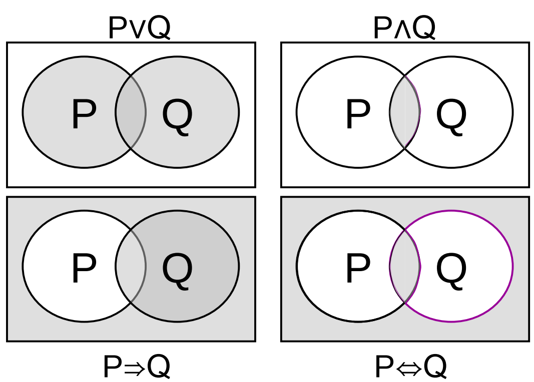


Models of complex sentences in terms of the models of their components. In each diagram, the shaded parts correspond to the models of the complex sentences.

Propositional logic: Semantics

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
i.e., is false iff S_1 is true and S_2 is false
$S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true



Models of complex sentences in terms of the models of their components. In each diagram, the shaded parts correspond to the models of the complex sentences.

Validity and inference

Truth tables may be used for
- definition of connectives
- test for valid sentences

- For example for sentence

$$((P \lor H) \land \neg H) \Rightarrow P$$

P	H	$P \lor H$	$(P \lor H) \land \neg H$	$((P \lor H) \land \neg H) \Rightarrow P$
False	False	False	False	True
False	True	True	False	True
True	False	True	True	True
True	True	True	False	True

A simple Knowledge Base

- Construct a knowledge base for the Wumpus world using the semantics of propositional logic:
  - *First chose the vocabulary of proposition symbols*:

- For each i,j:

  $P_{i,j}$ is true if there is a pit at [i,j]

  $B_{i,j}$ is true if there is a breeze at [i,j]

- *The KB contains the following sentences:*

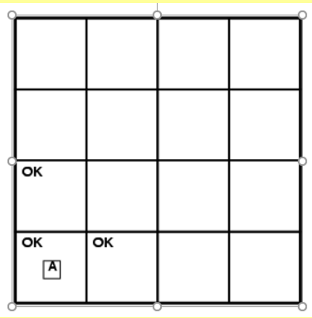
  $R_1: \neg P_{1,1}$

  $R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

  $R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

  $R_4: \neg B_{1,1}$

  $R_5: B_{2,1}$



Logical inference

$\Rightarrow$ The process of implementing entailment between sentences;

KB |= α

First algorithm for inference is a direct implementation of the definition of entailment:

1. enumerate the models and then
2. check if α is true in every model in which KB is true.

Let us write the KB:

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

$\neg P_{1,1}$

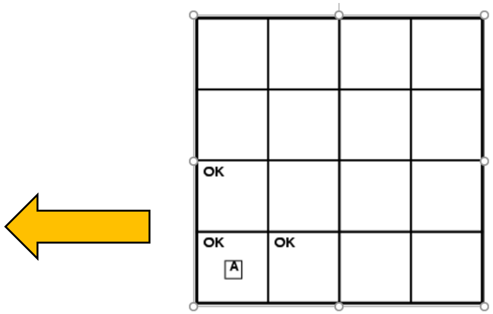
$\neg B_{1,1}$

$B_{2,1}$

c. "Pits cause breezes in adjacent squares"

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

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



Let us suppose that α_1 is "there is a pit in 3,1", represented as $P_{3,1}$

Truth tables for inference

What propositional symbols we have in the KB?

$B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, $P_{3,1}$

First algorithm for INFERENCE = ***model checking***

1. Enumerate the models
2. Check that is true in every model where KB is true

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

For *n* symbols, we shall have 2^n models!!!

- Depth-first enumeration of all models is sound and complete

TT-Entails?

For n symbols, time complexity is $O(2n)$, space complexity is $O(n)$

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 $\leftarrow$ a list of the proposition symbols in *KB* and α
   return TT-CHECK-ALL (*KB*, α, *symbols, []*)

PL-TRUE? : returns true if a sentence holds within a model

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 *false // when KB is false*
  else do
      P $\leftarrow$ FIRST(*symbols*); rest $\leftarrow$ REST(*symbols)*
      return TT-CHECK-ALL (*KB*, α, *rest, model* $\cup$ *{P = True})* **and**
                TT-CHECK-ALL (*KB*, α, *rest, model* $\cup$ *{P = False})*

Example

- Wumpus world
- We have the following KB, describing the rooms [1,1],[2,1][3,1],[1,2][2,2]:

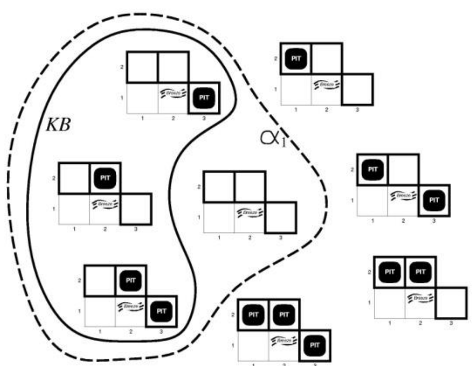
R_1: $\neg P_{1,1}$ R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R_4: $\neg B_{1,1}$ R_5: $B_{2,1}$

- The symbols are:
  - $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{2,1}$, $P_{2,2}$, $P_{3,1}$, $P_{1,2}$
- We have 2^7=128 possible models
- In 3 of these models the KB is true:

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1



$\alpha_1 = \neg P_{1,2}$

TT-CHECK-ALL (*KB*, α, *rest*, EXTEND*(P,true,model)*)

TT-CHECK-ALL (*KB*, α, *rest*, EXTEND*(P,false,model)*)

Equivalence, validity and satisfiability

- Logical equivalence: two sentences α and β are logically equivalent *if they are true in the same set of models.*

We write: $\alpha \equiv \beta$ if $\alpha \models \beta$ and $\beta \models \alpha$

- Validity: a sentence is valid if it is true in all models. Valid sentences are also known as *tautologies.*
  - *Deduction theorem:* for any α and β, $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.

- Satisfiability: a sentence is satisfiable if it is true in some model.
  - "Reductio ad absurdum" $\alpha \models \beta$ *if and only if the sentence* $(\alpha \wedge \neg \beta)$ *is unsatisfiable.*

Logic Equivalences

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

Validity and satisfiability

□ Definition A sentence is **valid** or necessarily true if and only if it is true under all possible interpretations in all possible worlds

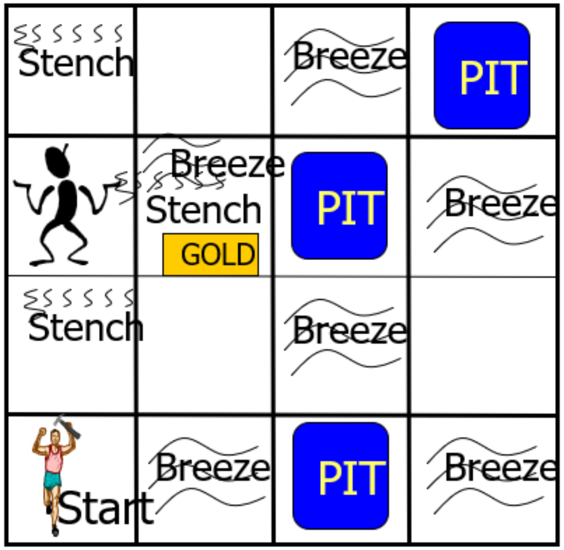
→ regardless of what it is supposed to mean

→ regardless of the state of affairs in the universe being described

□ Example "There is a pit at [2,2] or there is not a pit at [2,2]"

→ A sentence is **satisfiable** if and only if there is some interpretation in some world for which it is true.

Example: "There is a wumpus at [1,3]" - is satisfiable because there is a configuration for which the wumpus is at [1,3]]



Rules of inference for Propositional Logic

- We have seen that <u>truth tables</u> provide for a way of establishing the soundness of an inference

 There are <u>patterns</u> of inferences that occur over and over again + their soundness needs to be shown once and for all

$\Rightarrow$ such a pattern is an **inference rule**

Notation $\alpha \vdash \beta$

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

This is not a sentence, but rather an inference rule

1) Modus Ponens

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

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

$\rightarrow$Example: (Rain $\Rightarrow$ Wet and Rain then Wet may be inferred)

2) And-Elimination

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

→ From a conclusion, you can infer any of the conjuncts

Example: (WumpusAhead ∧ WumpusAlive) ⇒ WumpusAhead may be inferred)

3) And-Introduction

$$\frac{\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

→ From a set of sentences, you can infer their conjunction

Example: (WumpusAhead, WumpusAlive, Gold)⇒ WumpusAhead ∧ WumpusAlive ^ Gold may be inferred)

4) Or-Introduction

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

$\rightarrow$ From a sentence, you can infer its disjunction with anything else at all

Example: (WumpusAhead) $\Rightarrow$ WumpusAhead $\vee$ Gold *may be inferred)*

5) Double Negation Elimination

$$\frac{\neg(\neg\alpha\ )}{\alpha}$$

→ From a doubly negated sentence, you can infer a positive sentence.)

Example: $\neg(\neg$ WumpusAhead $) \Rightarrow$ WumpusAhead *may be inferred*

6) Unit Resolution

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

$\rightarrow$ From a disjunction, if one of the disjuncts if false, then you can infer the other one is true

Example: (WumpusAhead$\vee$ Gold, $\neg$WumpusAhead)$\Rightarrow$ Gold *may be inferred*

Unit Resolution

Takes a clause – a disjunction of literals – and produces a new clause

$$\frac{l_1 \vee \ldots \vee l_k, m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

where m and l_i are complementary literals, i.e. m = $\neg l_i$

Unit resolution may be generalized to **<u>the full resolution rule</u>**:

$$\frac{l_1 \vee \ldots \vee l_k, m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary literals

<u>Special case</u>: clauses of length=2

$$\frac{P_{11} \vee P_{31}, \neg P_{11} \vee P_{22}}{P_{31} \vee P_{22}}$$

<u>Factoring</u>: the resulting clause should contain only one copy of each literal

7) Resolution

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

or equivalently

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

$\rightarrow$ This is the most difficult. Because β cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

Example: (WumpusAhead$\vee$ Gold, $\neg$WumpusAhead $\vee$Breeze)$\Rightarrow$ Gold $\vee$ Breeze *may be inferred*

Logic Equivalences = Inference rules

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

Proof methods

Proof methods are divided into two kinds:

1. Applications 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 translation of sentences into a normal form
2. Model checking
   - Truth table enumeration (always exponential in *n*)
   - Improved backtracking
   - Heuristic search in model space (e.g. hill-climbing)

Conjunctive Normal Form

- *Resolution applies only to disjunctions of literals*

Problem: it seems to be relevant only to KBs and queries consisting of disjunctions → How can it lead to complete inference procedure for all propositional logic?

- Every sentence of propositional logic is logically equivalent to a conjunction of disjunctions of literals
- *CNF: A sentence is expressed as a conjunction of disjunctions of literals is said to be in conjunctive normal form (CNF)*
- K-CNF: has exactly k literals per clause

$$(l_{11} \vee \ldots \vee l_{1k}) \wedge (l_{n1} \vee \ldots \vee l_{nk})$$

Conversion to CNF

Take the sentence, e.g. $B_{11} \Leftrightarrow (P_{12} \vee P_{21})$

1. **Eliminate $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$**
   $(B_{11} \Rightarrow (P_{12} \vee P_{21})) \wedge ((P_{12} \vee P_{21}) \Rightarrow B_{11})$
2. **Eliminate $\Rightarrow$ by replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$**
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg(P_{12} \vee P_{21}) \vee B_{11})$
3. **CNF requires $\neg$ to appear only in literals;** we move $\neg$ inwards by repeated application of the following equivalences:

   $\neg(\neg\alpha) \equiv \alpha$ (double-negation elimination)

   $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ (de Morgan)

   $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ (de Morgan)

   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge ((\neg P_{12} \wedge \neg P_{21}) \vee B_{11})$
4. **Apply the distributivity law:** $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$
   $(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11})$

Inference Rules for the Wumpus

- Start with *the KB contains the following sentences:*

  $R_1: \neg P_{1,1}$

  $R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

  $R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

  $R_4: \neg B_{1,1}$

  $R_5: B_{2,1}$

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 P?	2,2	3,2	4,2
1,1 A V OK	2,1 B OK	3,1	4,1

- Try to prove $\neg P_{1,2}$

- 1/ Apply biconditional elimination to R_2

$(\alpha \Leftrightarrow \beta) \equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination

$$R_6: (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

- 2/ Apply AND-elimination to R_6

Proof

- 2/ Apply AND-elimination to R_6

And-Elimination

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

R_6: $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

R_7: $(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

- 3/ Apply contraposition to R_7

$(\alpha \Rightarrow \beta) \equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition

R_8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$

- 4/ Apply Modus-Ponens to R_8 and R_4: $\neg B_{1,1}$

Modus Ponens

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

R_9: $\neg (P_{1,2} \vee P_{2,1})$

What did we find?

- 5/ Apply De Morgan's rule to R_9

R_{10}: $\neg P_{1,2} \wedge \neg P_{2,1}$

The Conclusion!!!! : neither [1,2] nor [2,1] contains a pit

How did we do it????

- 6/ Apply AND-Elimination rule to R_{10}

And-Elimination

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

R_{11}: $\neg P_{1,2}$

In summary:

- KB

$R_1: \neg P_{1,1}$

$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$R_4: \neg B_{1,1}$

$R_5: B_{2,1}$

- Query: $\neg P_{1,2}$

Inference rules:

biconditional elimination

AND-elimination

Modus Ponens

contraposition

De Morgan

Inferences:

$R_6: (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_7: ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_8: (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$

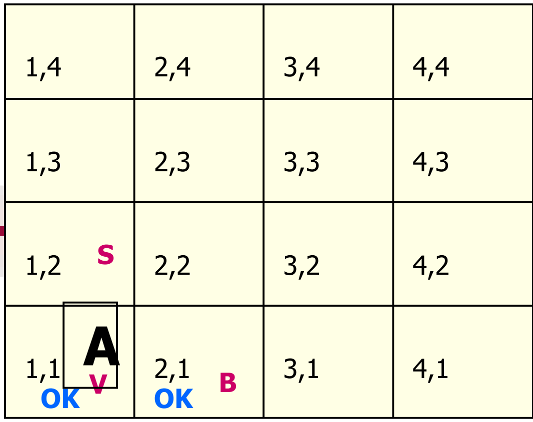
$R_9: \neg (P_{1,2} \vee P_{2,1})$

$R_{10}: \neg P_{1,2} \wedge \neg P_{2,1}$

$R_{11}: \neg P_{1,2}$

Searching for proofs is an alternative to enumerating models!!! It is more efficient because it ignores irrelevant propositions

Resolution



- Another inference rule
- *Example*: in the wumpus world show how this new inference rule works!

The agent returns from [2,1] to [1,1] and then goes to [1,2], when it perceives a stench, no breeze

$R_{11}: \neg B_{1,2}$

$R_{12}: B_{1,2} \Leftrightarrow (P_{11} \vee P_{22} \vee P_{13})$

We can entail there is no pit in [2,2] or [1,3] similarly as we entailed there is no pit in [1,2]

$R_{13}: \neg P_{22}$

$R_{14}: \neg P_{13}$

$P_{11} \vee P_{22} \vee P_{13} \Rightarrow B_{1,2}$

$\neg B_{1,2} \Rightarrow \neg(P_{11} \vee P_{22} \vee P_{13})$

$\Rightarrow \neg P_{11} \wedge \neg P_{22} \wedge \neg P_{13}$

Resolution (continued)

R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$\rightarrow$Apply biconditional elimination to R_3

$(\alpha \Leftrightarrow \beta) \equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$

biconditional elimination

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

followed by Modus Ponens applied to R_5: $B_{2,1}$

R_{15}: $(P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Now apply resolution: the literal $\neg P_{22}$ in R_{13} resolves with the literal P_{22} in R_{15} to give

R_{16}: $P_{1,1} \vee P_{3,1}$

If there is a pit in one of [1,1], [2,2] and [3,1], and it is not in [2,2] then it is in [1,1] or [3,1]

Similarly, the literal $\neg P_{11}$ in R_1 resolves with the literal P_{11} in R_{16} to give

R_{17}: $P_{3,1}$

If there is a pit in [1,1] or [3,1] and it is not in [1,1] then it is in [3,1]

Unit resolution rule:

$$\frac{l_1 \vee \ldots \vee l_k ,\ \neg l_i}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

A Resolution Algorithm

- Inference procedures based on resolution work by using the principle of **proof by contradiction**.
  - "Reductio ad absurdum" *α \|= β if and only if the sentence ($\alpha \wedge \neg \beta$) is unsatisfiable.*

In order to show that KB \|= α we show that (KB $\wedge$ α) is unsatisfiable!

We do this by proving a contradiction.

Resolution algorithm

Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  $clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  $new \leftarrow \{\ \}$
  loop do
    for each C_i, C_j **in** *clauses* **do**
      $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** *false*
    $clauses \leftarrow clauses \cup new$

The pair of clauses contain complimentary literals

Resolution

Conjunctive Normal Form (CNF—universal)

$$\textbf{conjunction} \text{ of } \underbrace{\textbf{disjunctions} \text{ of } \textbf{literals}}_{\textbf{clauses}}$$

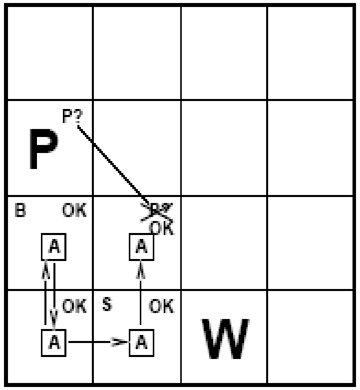
E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Resolution inference rule (for CNF): complete for propositional logic

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where ℓ_i and m_j are complementary literals. E.g.,

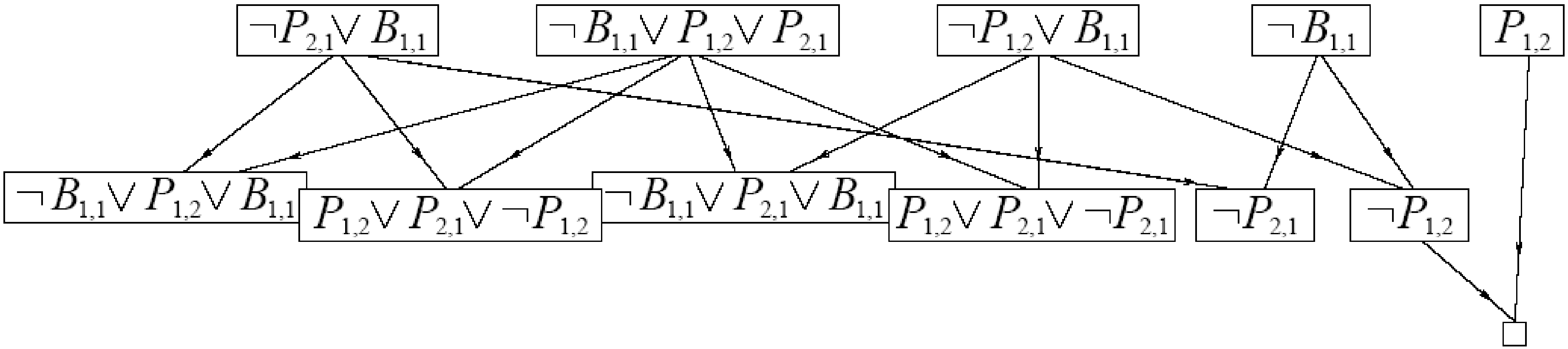
$$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



Resolution is sound and complete for propositional logic

Resolution example

$$KB = \left\{ (B_{1,1} \;\Leftrightarrow\; (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \quad \alpha = \neg P_{1,2} \right.$$



Resolution Closure

- We want to show that PL-RESOLUTION is complete.

The resolution closure *RC(S)* is a set of clauses *S* derivable from the repeated application of the resolution rule to the clauses of *S* or their derivatives.

RC(S) is what PL-RESOLUTION computes as the final value of the variable clauses.

If a set of clauses C is unsatisfiable, Then RC(C) contains the NULL Clause!

Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

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

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
      for each Ci, Cj in clauses do
         resolvents ← PL-RESOLVE(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

Horn Clauses

- A Horn clause is a disjunction of literals of which at most one is positive
  - Example: $(\neg L_{11} \vee \neg Breeze \vee B_{11})$
- *Every Horn clause can also be written as an implication whose premise is a conjunction of positive literals and whose conclusion is a simple positive literal*

$$L_{11} \wedge Breeze \Rightarrow B_{11}$$

Deciding entailment with Horn clauses can be done in time that is <u>linear</u> in the size of the knowledge base.

Inference with Horn Clauses can be done through the forward-chaining and the backward-chaining algorithms.

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 algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known in KB

   while agenda is not empty do
       p ← POP(agenda)
       unless inferred[p] do
           inferred[p] ← true
           for each Horn clause c in whose premise p appears do
               decrement count[c]
               if count[c] = 0 then do
                   if HEAD[c] = q then return true
                   PUSH(HEAD[c], agenda)
   return false
```

AND-OR Graphs

- Multiple links joined by an arc indicate a conjunction: *every link must be proven*
- Multiple links without an arc indicate a disjunction: any link can be proved

Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

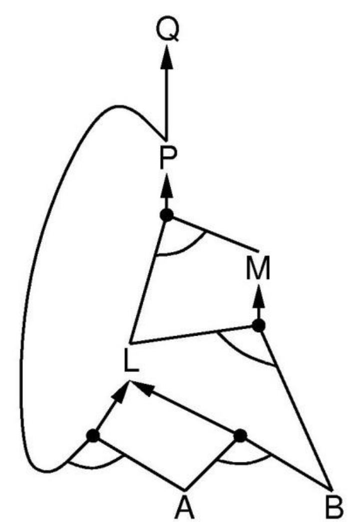
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

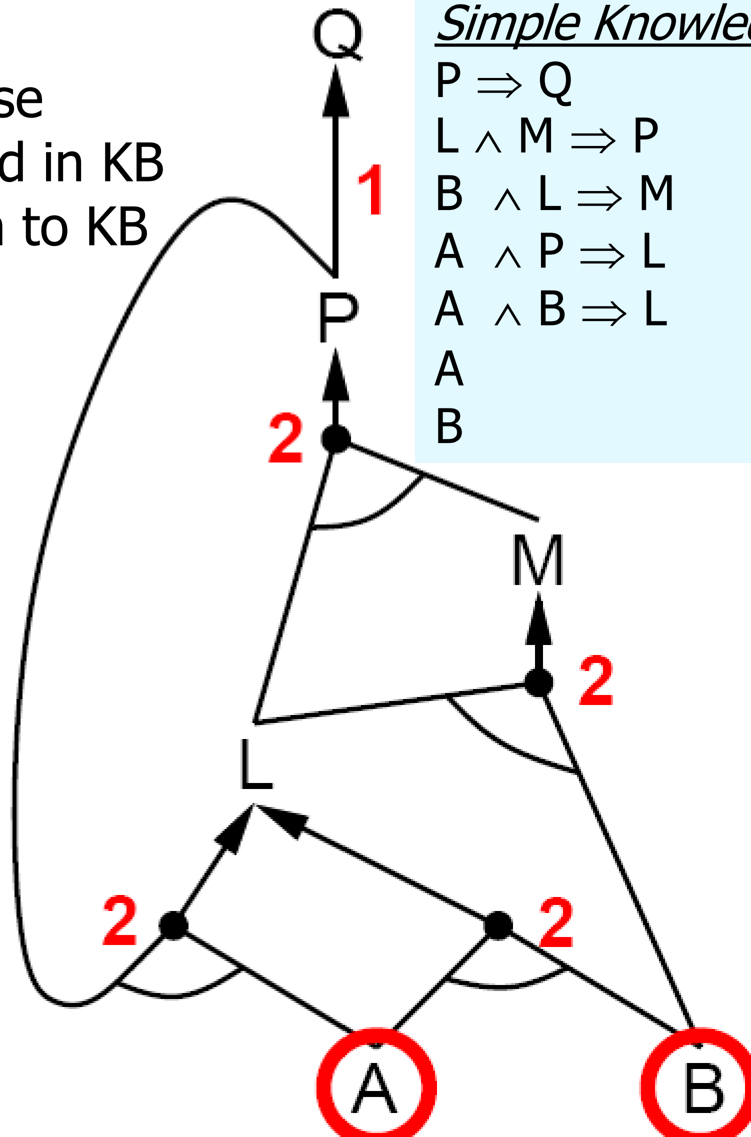
A

B



Forward chaining example

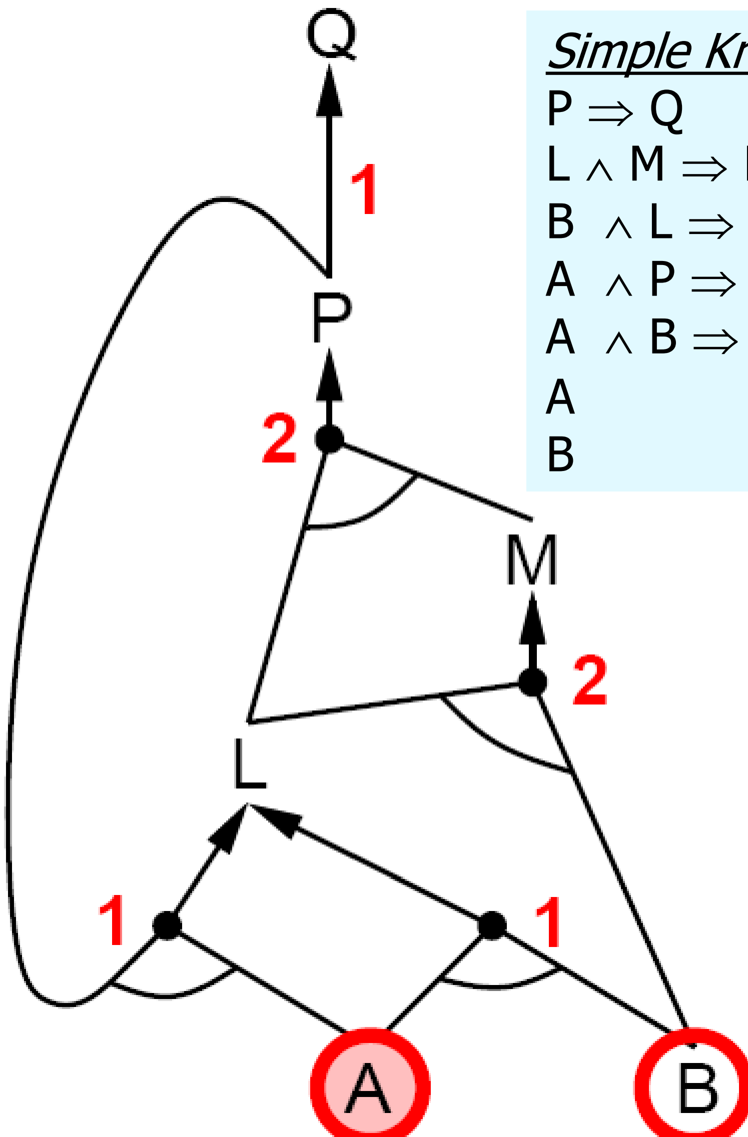
<u>The idea</u>:
1/ Fire any rule whose premises are satisfied in KB
2/ Add its conclusion to KB
Repeat 1 & 2 until Query is found



<u>Simple Knowledge base of Horn Clauses:</u>

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

Forward chaining example(2)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

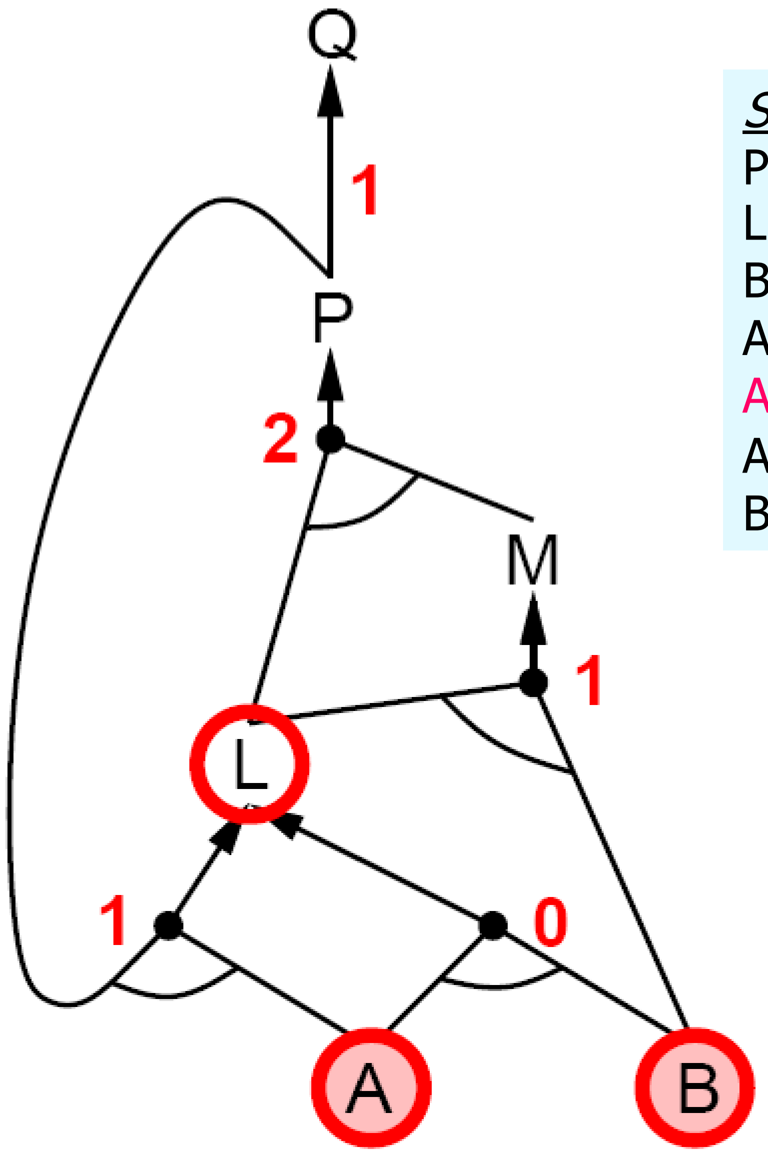
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward chaining example(3)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

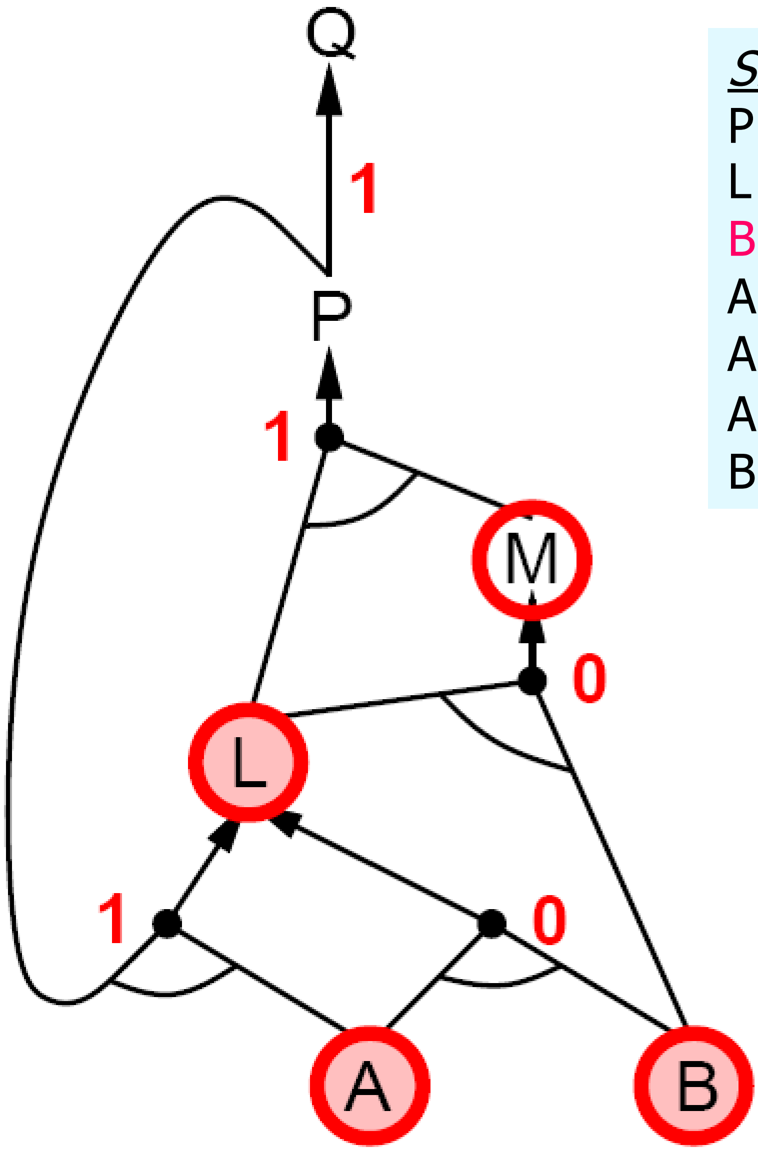
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward chaining example(4)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

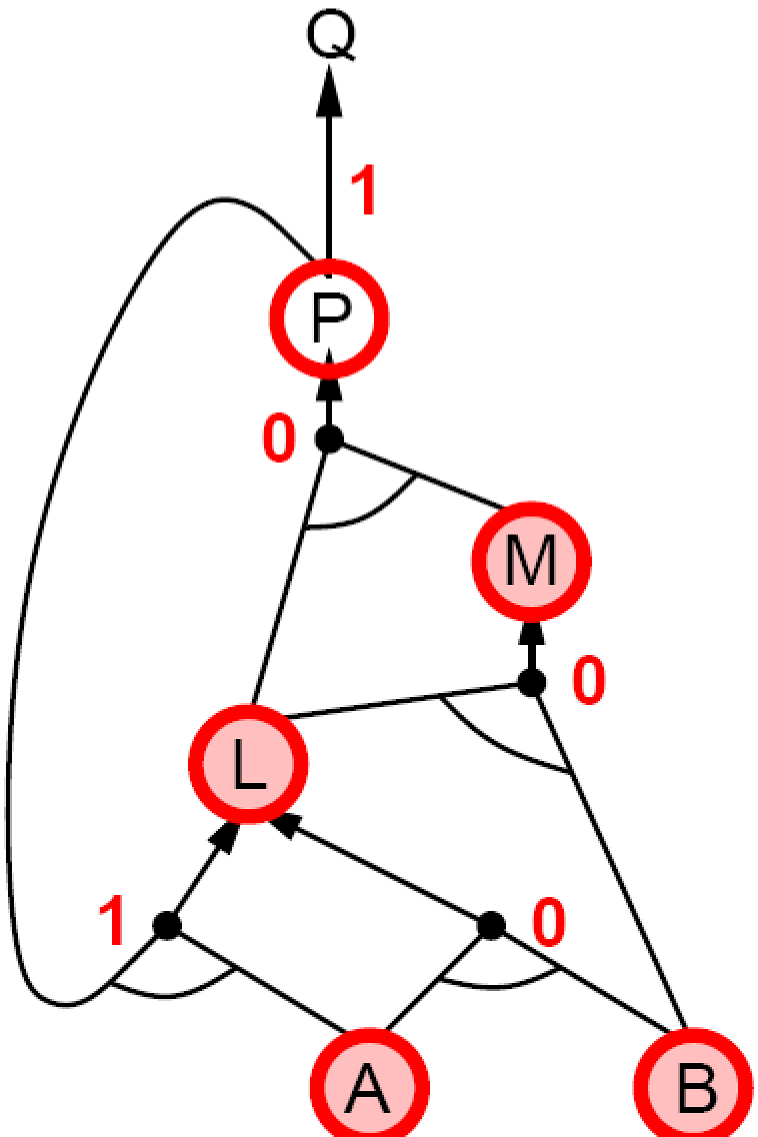
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward chaining example(5)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

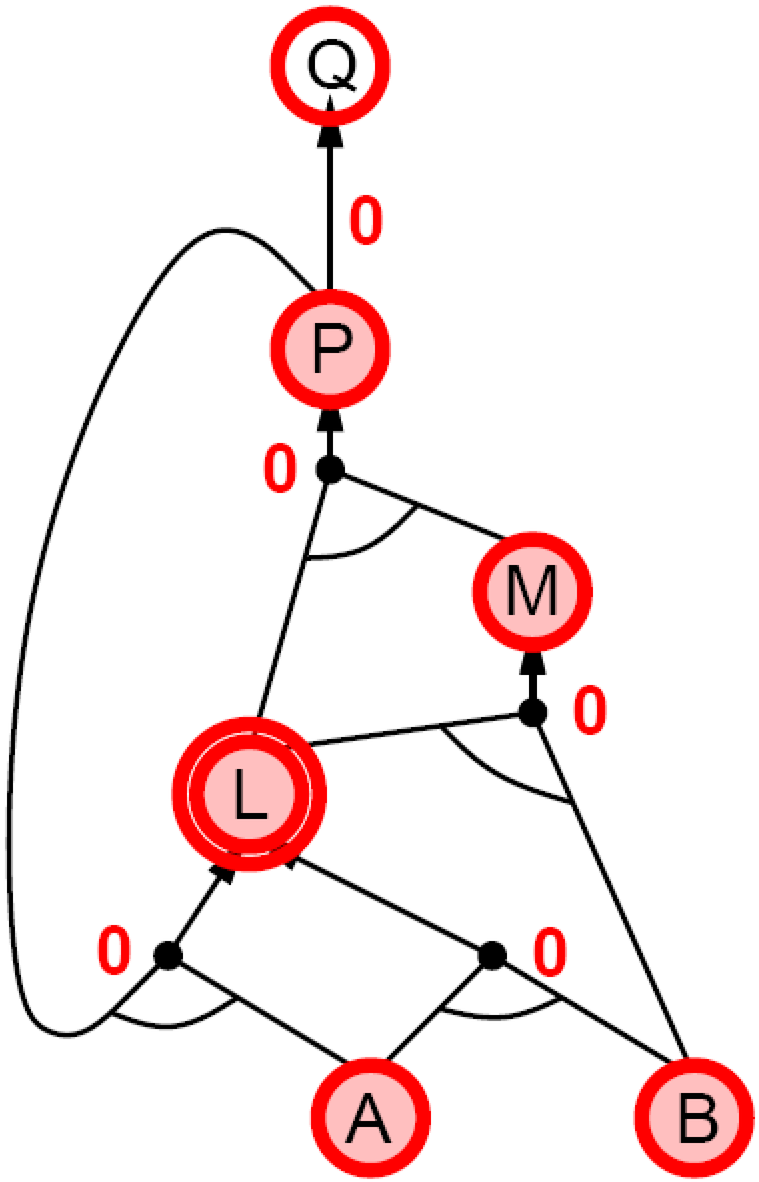
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward chaining example(6)



<u>*Simple Knowledge base of Horn Clauses:*</u>

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

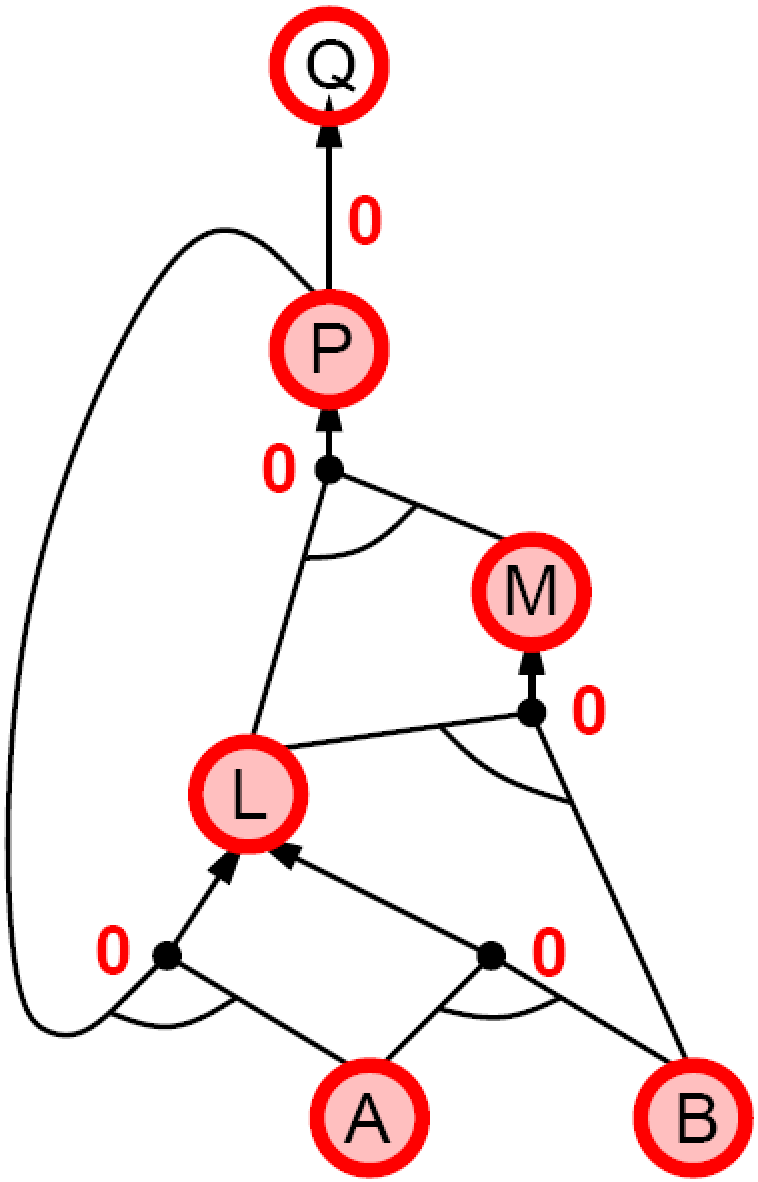
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

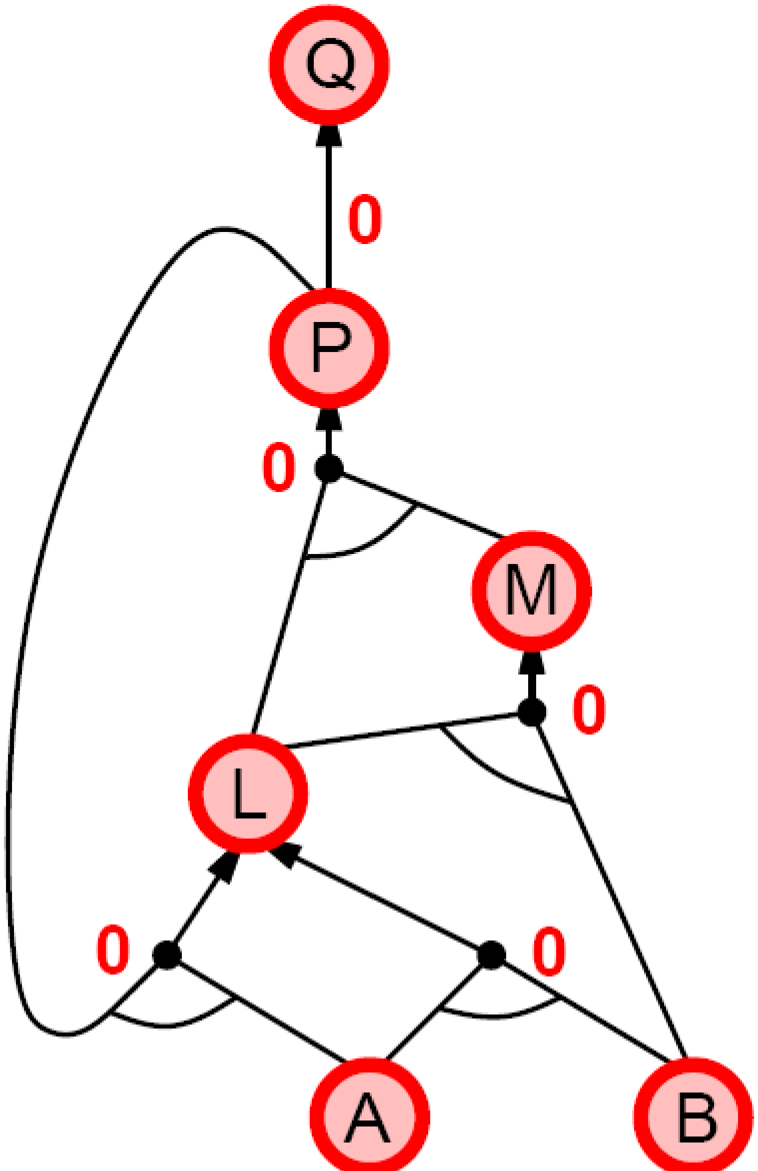
Forward chaining example(7)



Simple Knowledge base of Horn Clauses:

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

Forward chaining example(8)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Proof of completeness

FC derives every atomic sentence that is entailed by *KB*

1. FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final state as a model *m*, assigning true/false to symbols
3. Every clause in the original *KB* is true in *m*

   $a_1 \wedge \ldots \wedge a_k \Rightarrow b$

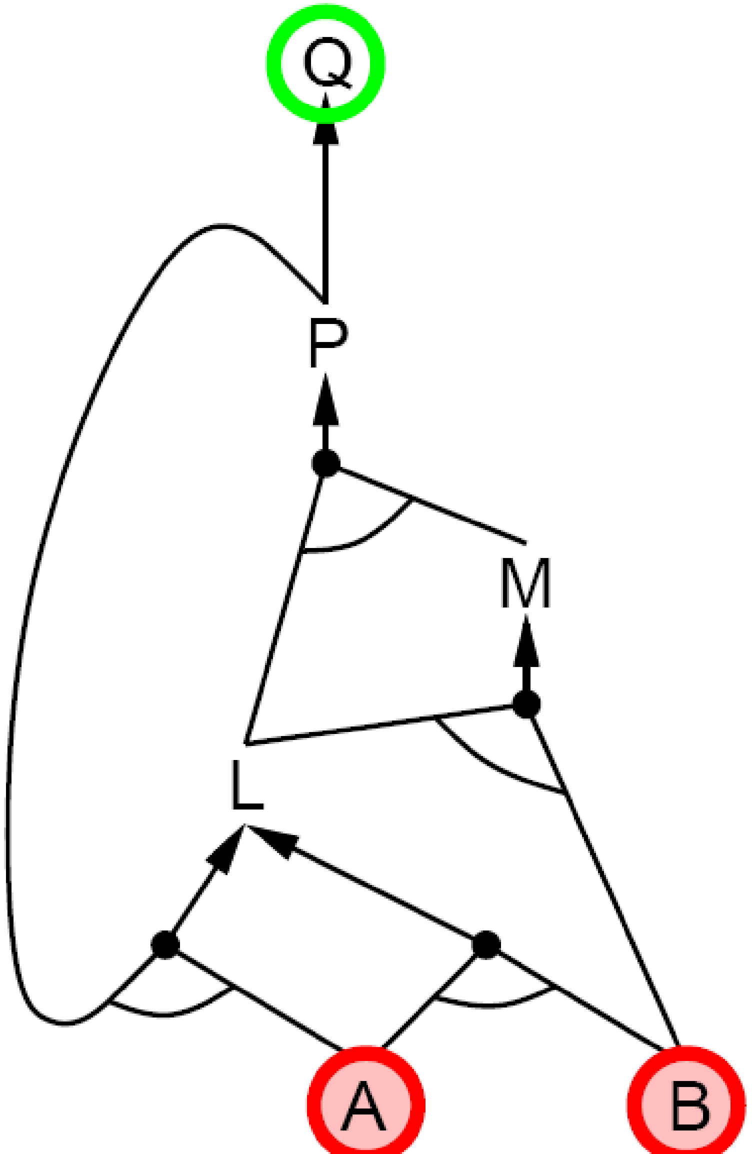
4. Hence *m* is a model of *KB*
5. If $KB \models q$, q is true in every model of *KB*, including *m*

Backward Chaining

- Idea: work backwards from 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 sub-goal is already on the goal stack

- Avoid repeated work: check if new sub-goal
  1. Has already been proven true, or
  2. Has already failed

Backward chaining example



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

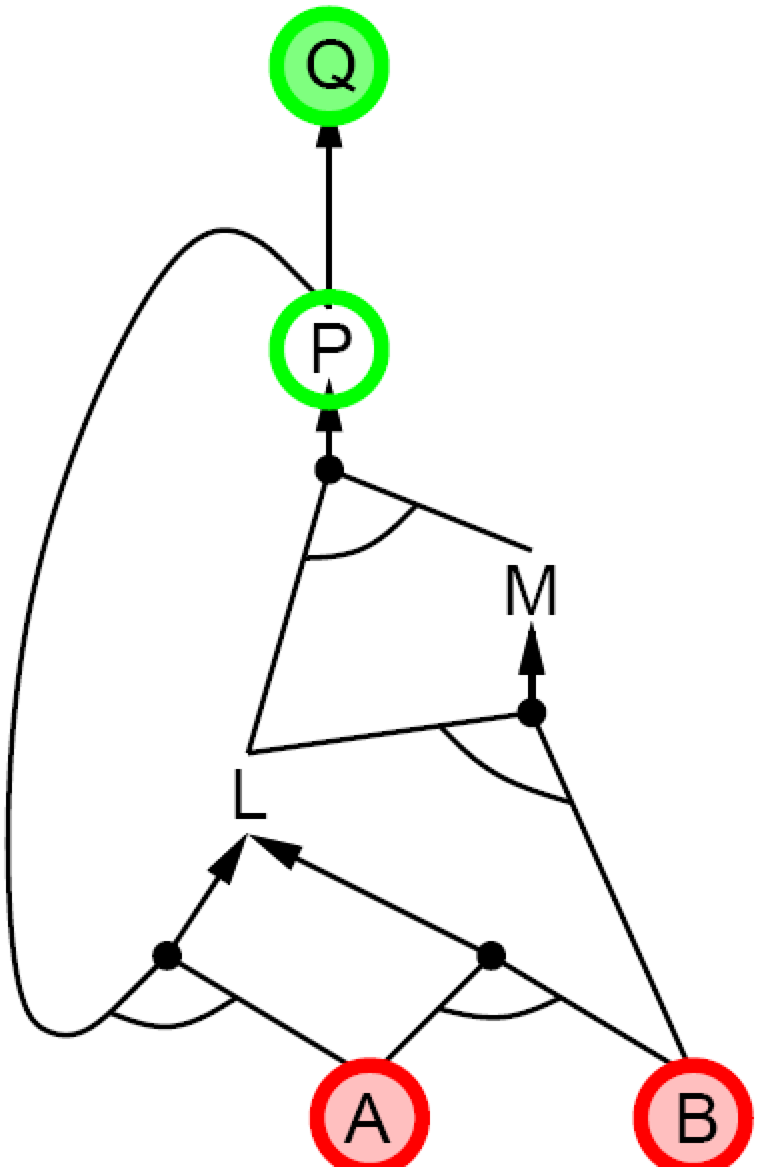
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

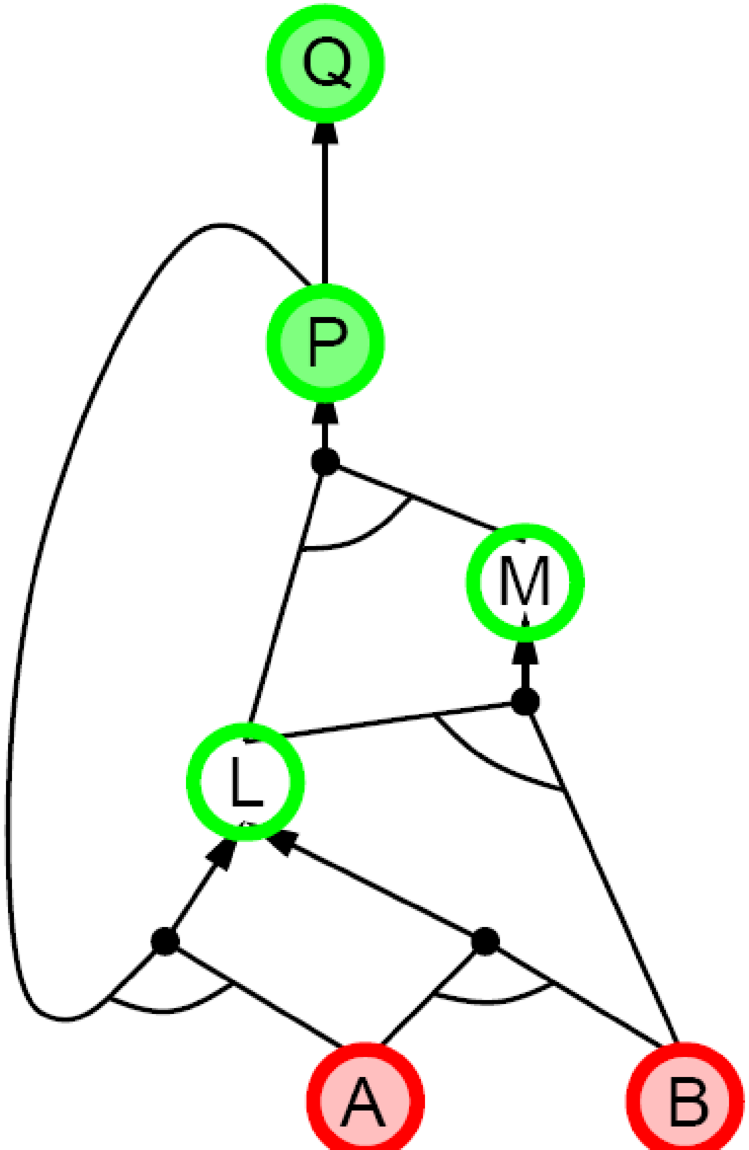
Backward chaining example(2)



Simple Knowledge base of Horn Clauses:

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

Backward chaining example(3)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

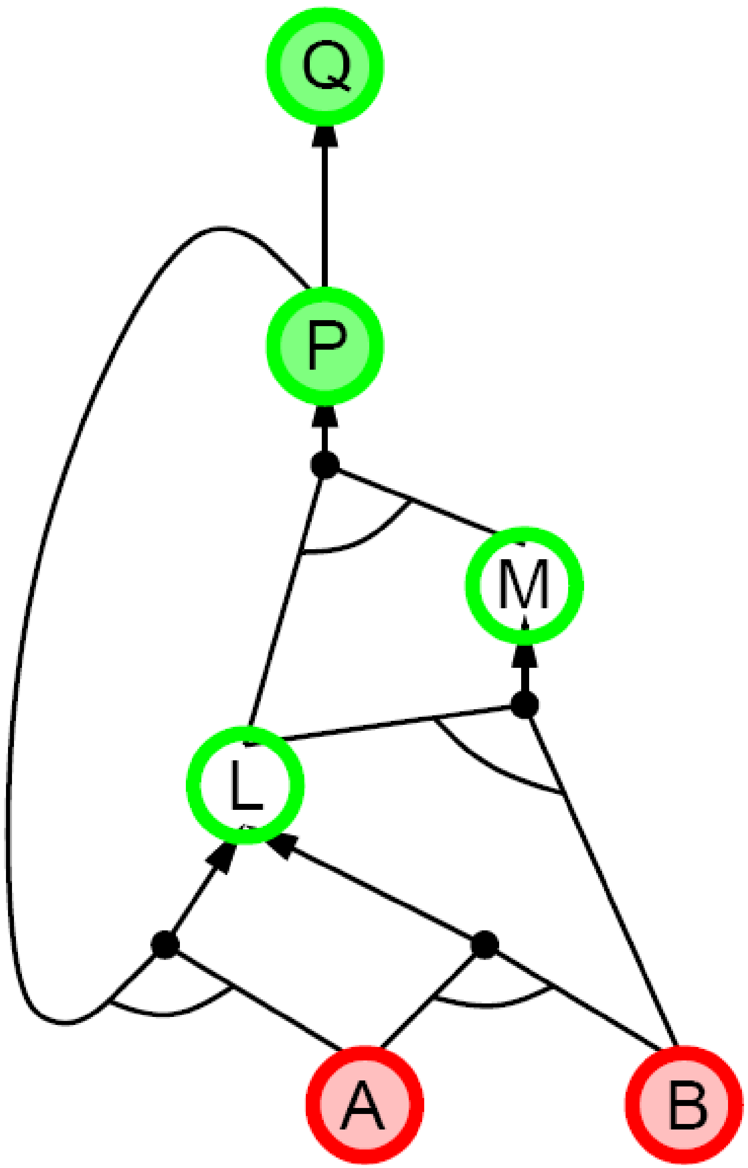
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

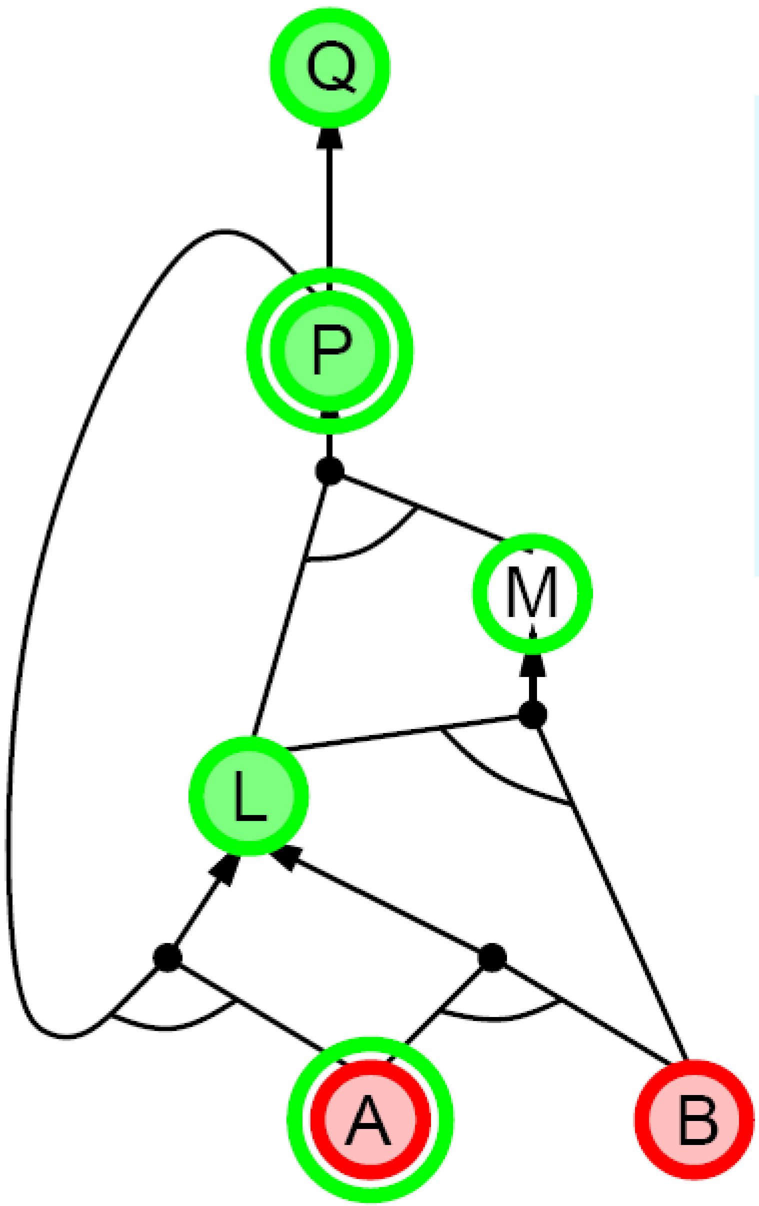
Backward chaining example(4)



Simple Knowledge base of Horn Clauses:

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

Backward chaining example(5)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

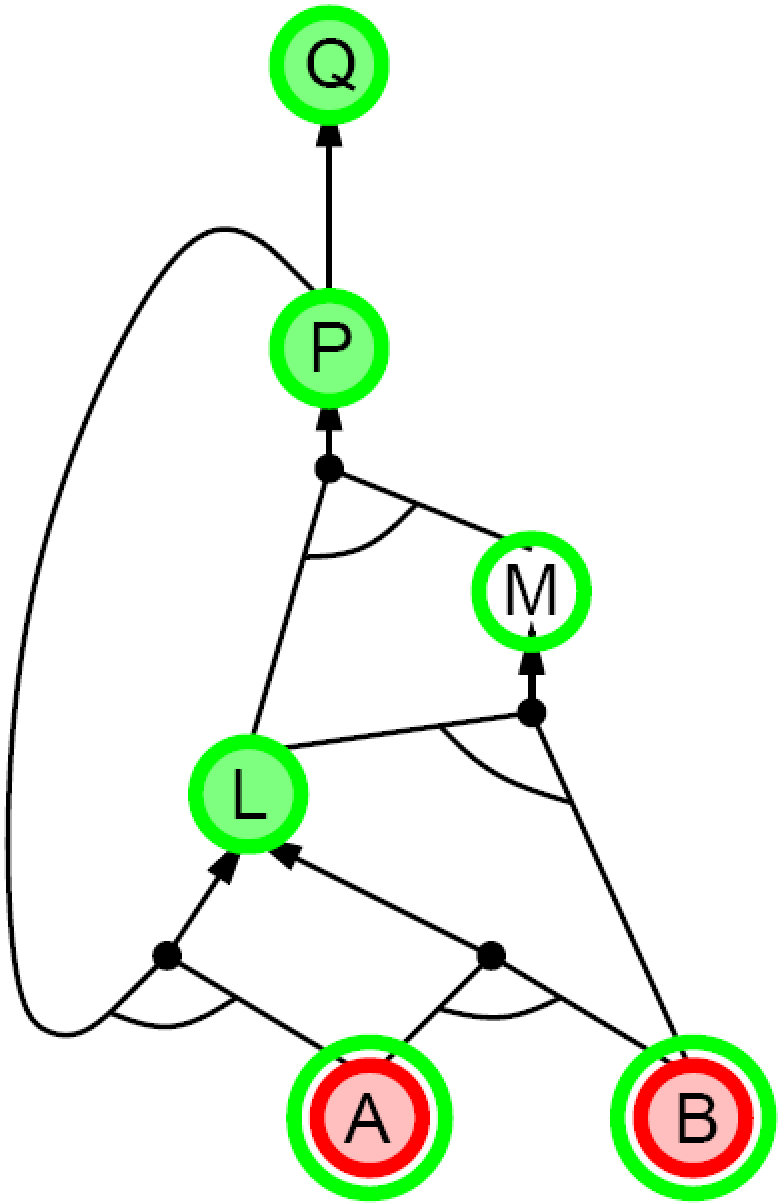
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example(6)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

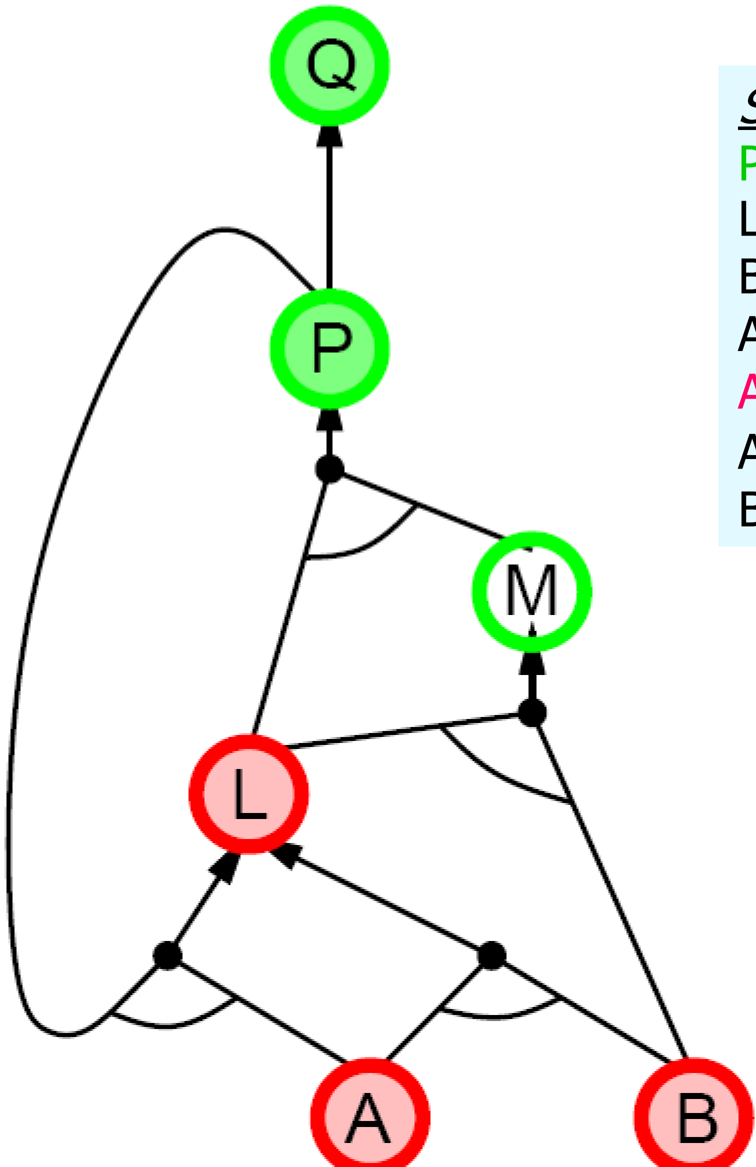
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example(7)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

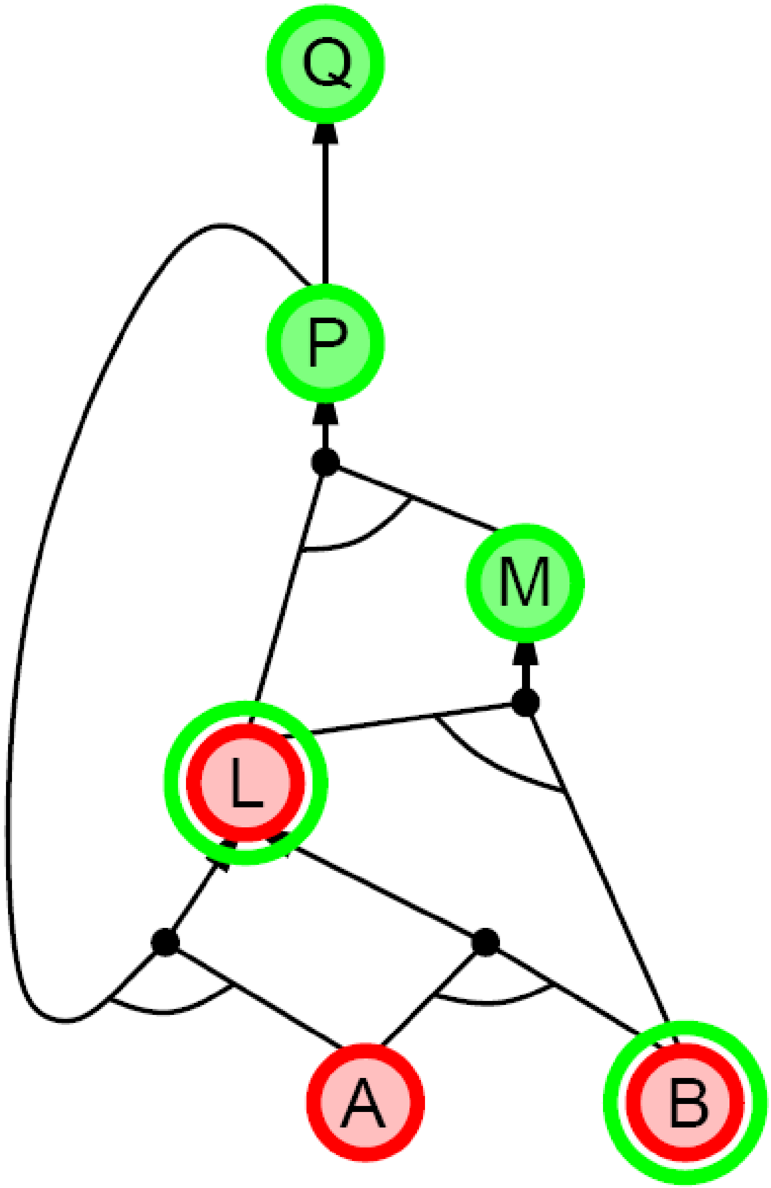
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example(8)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

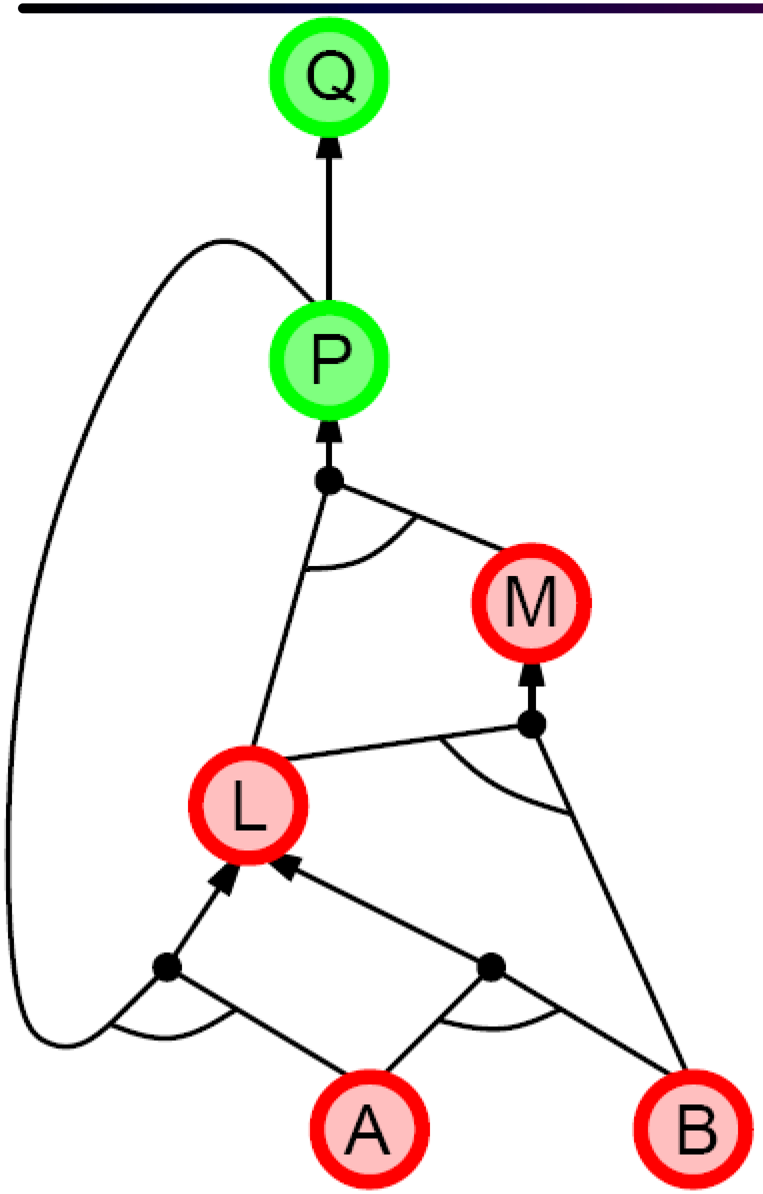
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example(9)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

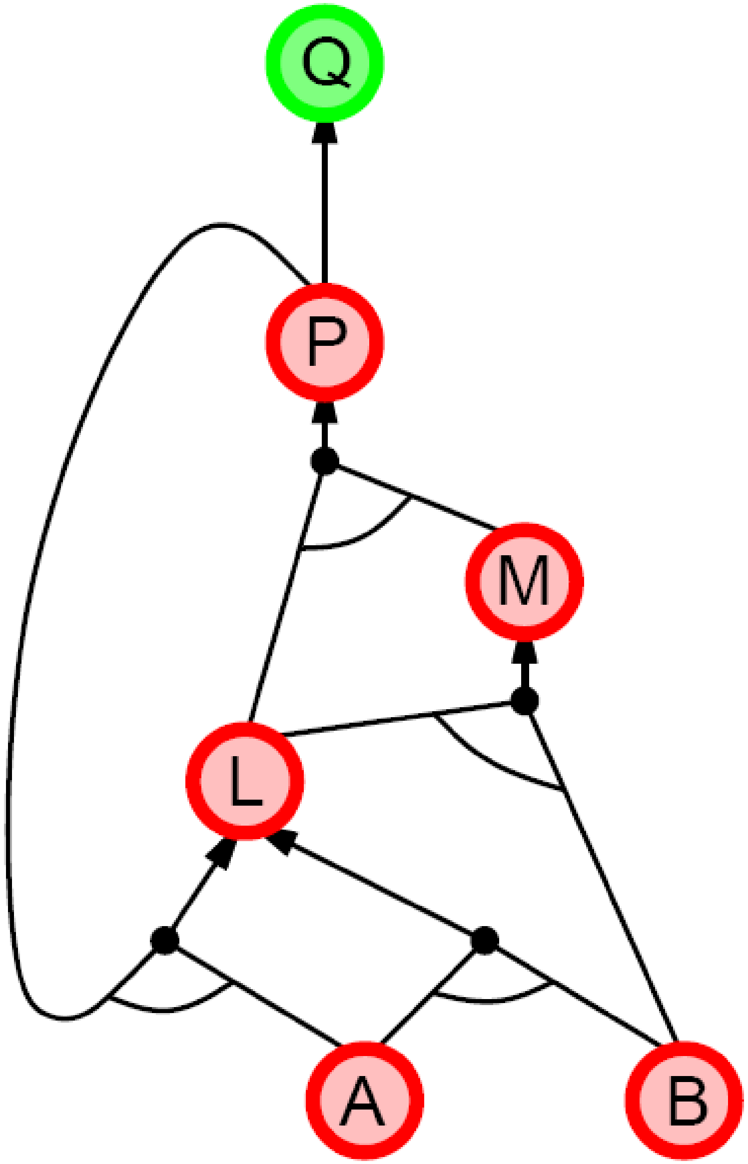
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

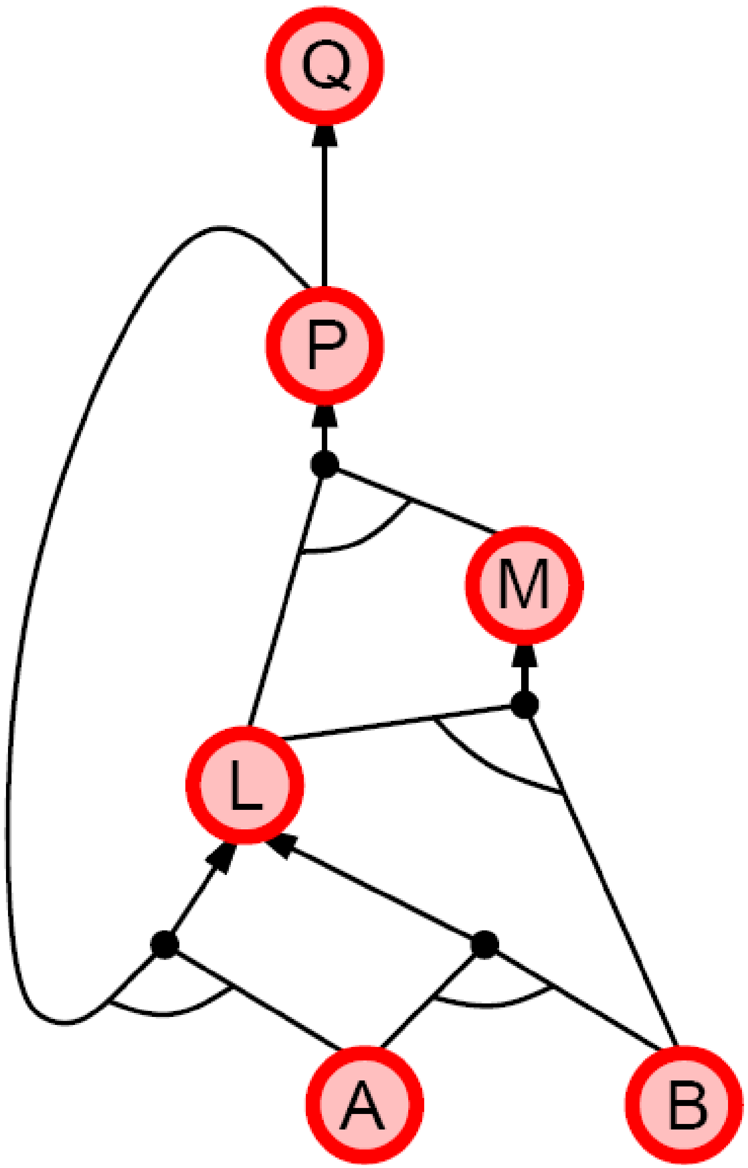
Backward chaining example(10)



Simple Knowledge base of Horn Clauses:

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

Backward chaining example(11)



Simple Knowledge base of Horn Clauses:

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward 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

A complete Backtracking Algorithm

- The David-Putnam algorithm: first described in a paper by Davis, Longemann and Loveland – we call it the **DPLL algorithm**
- *It is a depth-first enumeration of all possible models.*
- *The input = a sentence in CNF (a set of clauses)*
- It implements 3 improvements over TT-ENTAILS? (slide 52)
  1. <u>Early termination</u>: it detects if a sentence must be TRUE or FALSE even with a partially completed model.
  2. <u>Pure symbol heuristic</u>: a "pure" symbol is a symbol that appears with the same "sign" in all clauses. If a sentence has a model, the pure symbol is assigned a value to make the symbols TRUE (otherwise the sentence would be FALSE). In determining the "purity" of a symbol, the algorithm can ignore clauses that are TRUE in the current models.
  3. <u>Unit clause heuristic</u>: a unit clause has 1 symbol. In DPLL it means clauses in which all symbols but 1 have been assigned FALSE. One unit clause can assign another one: <u>unit propagation</u>!

DPLL-Satisfiable?

function DPLL-Satisfiable?(*s*) **returns** *true* or *false*
inputs: *s*, a sentence in propositional logic

clauses ← a set of clauses in the CNF representation of *s*
symbols ← a list of the proposition symbols in *s*
return DPLL (*clauses*, *symbols*, *{})*

function DPLL (*clauses, symbols, model)* **returns** *true* or *false*
if every clause **in** *clauses is TRUE in model* **then return** *true*
if some clause **in** *clauses is FALSE in model* **then return** *false*
P,value ← FIND-PURE-SYMBOL(*clauses, symbols, model)*
if *P* is non-null **then return** DPLL (*clauses, symbols-P, model*∪*{P=value})*
P,value ← FIND-UNIT-CLAUSE(*clauses, model)*
if *P* is non-null **then return** DPLL(*clauses, symbols-P, model*∪*{P=value})*
P ← FIRST(*symbols*); rest ← REST(*symbols)*
return DPLL(*clauses, rest, model*∪*{P=true})* **or**
DPLL(*clauses, rest, model*∪*{P=false})*

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - syntax: formal structure of sentences
  - semantics: truth of sentences wrt models
  - entailment: necessary truth of one sentence given another
  - inference: deriving sentences from other sentences
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Resolution is complete for propositional logic
  Forward, backward chaining are linear-time, complete for Horn clauses
- Propositional logic lacks expressive power