Chapter 7
Logical Agents

CS 461 – Artificial Intelligence
Pinar Duygulu
Bilkent University, Spring 2008

Slides are mostly adapted from AIMA and MIT Open Courseware

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

Introduction

- The **representation of knowledge** and the **reasoning processes** that bring knowledge to life are central to entire field of artificial intelligence
- Knowledge and reasoning are important to artificial agents because they enable successful behaviors that would be very hard to achieve otherwise (no piece in chess can be on two different squares at the same time)
- Knowledge and reasoning also play a crucial role in dealing with partially observable environments (inferring hidden states in diagnosing diseases, natural language understanding)
- Knowledge also allows flexibility.

Knowledge bases

- Knowledge base = set of sentences in a formal language
- Each sentence is expressed in a knowledge representation language and represents some assertions about the world
- There should be a way to add new sentences to KB, and to query what is known
- Declarative approach to building an agent (or other system):
  - `TELL` it what it needs to know
  - Then it can `ASK` itself what to do - answers should follow from the KB
- Both tasks may involve inference – deriving new sentences from old
- In logical agents – when one `ASK`s a question to KB, the answer should follow from what has been `TELL`ed
- 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

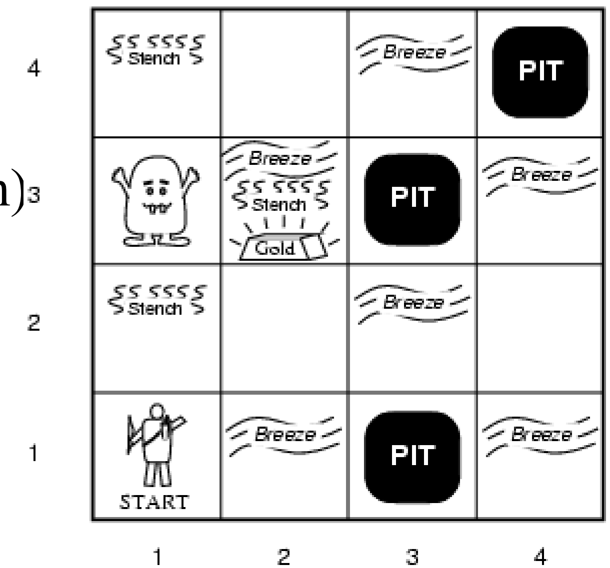
```
function KB-AGENT(percept) returns an action
   static: KB, a knowledge base
           t, a counter, initially 0, indicating time

   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 : maintain the background knowledge
- Each time the agent program is called it does three things
  - `TELL`s the KB what it perceives
  - `ASK` the KB what action it should perform
  - `TELL` the KB that the action is executed
- 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
  - Deduce appropriate actions
- Declarative versus procedural approaches

Wumpus World PEAS description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly (stench)
  - 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

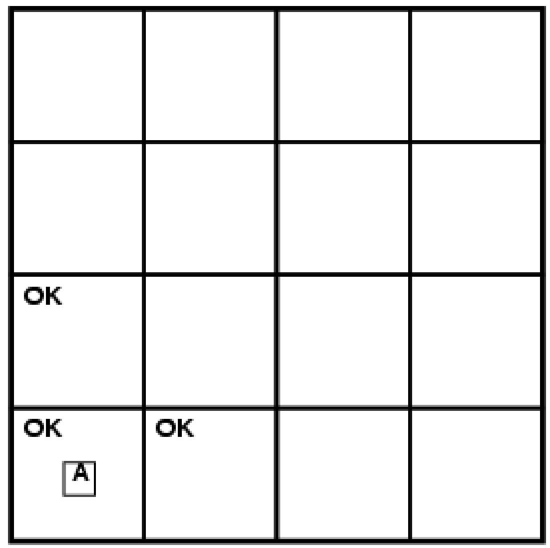


- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

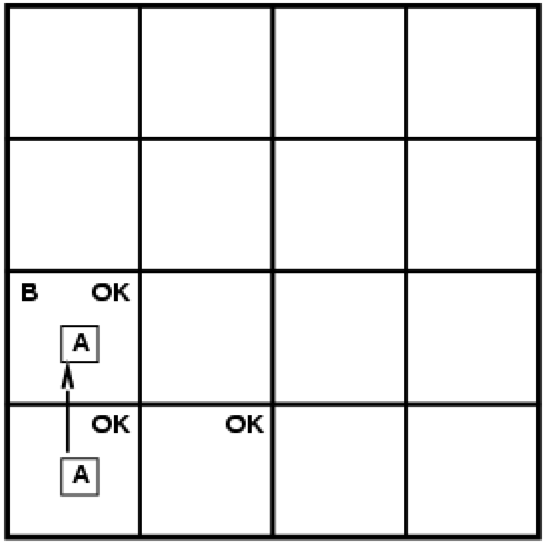
Wumpus world characterization

- 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

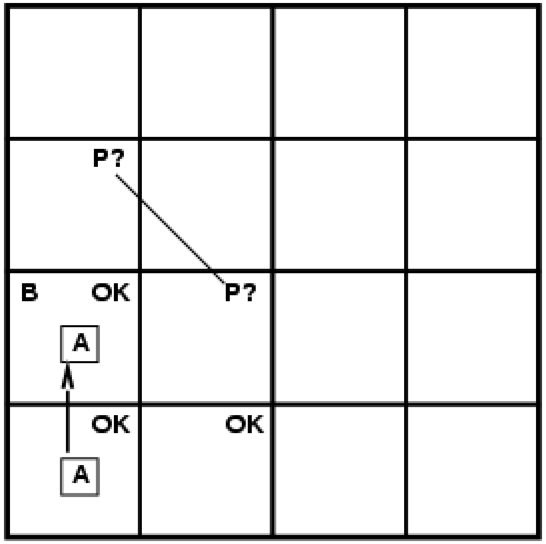
Exploring a wumpus world



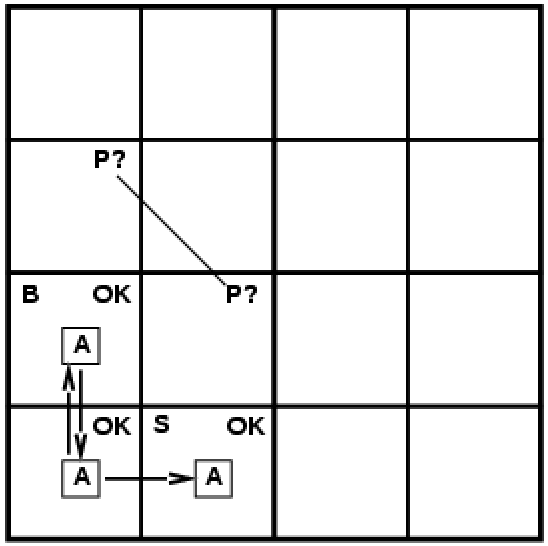
Exploring a wumpus world



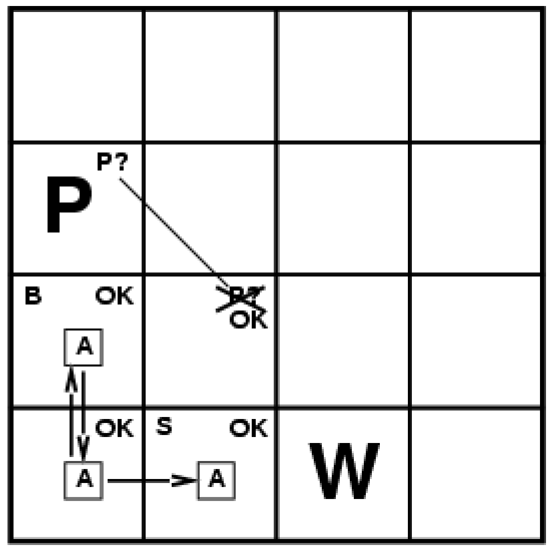
Exploring a wumpus world



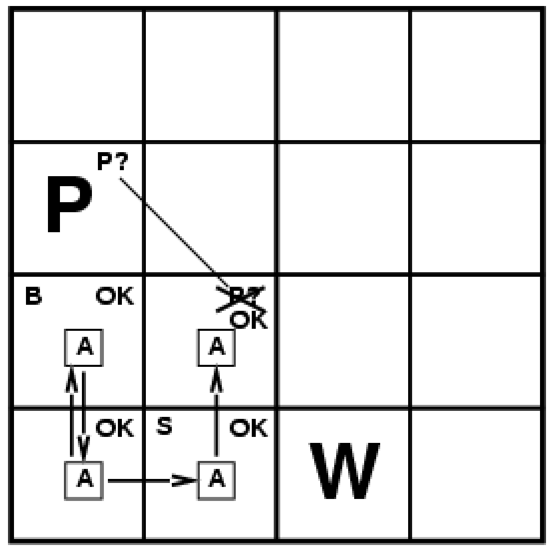
Exploring a wumpus world



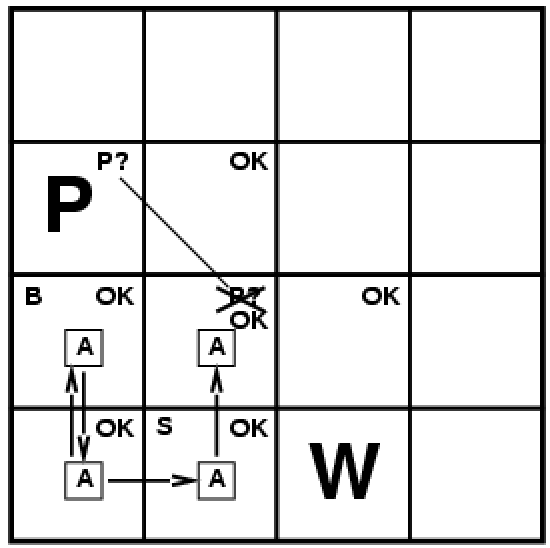
Exploring a wumpus world



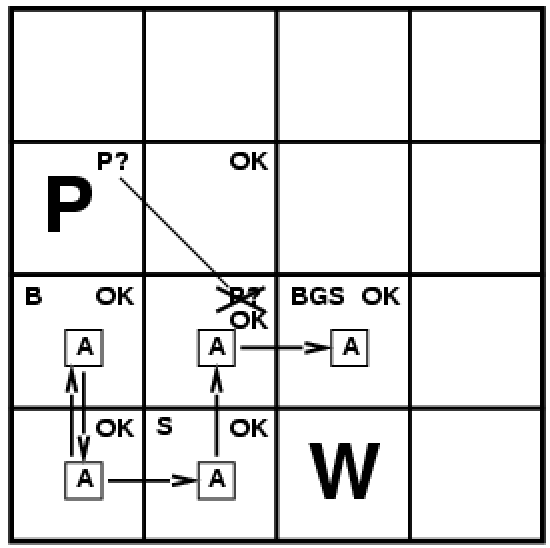
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



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 $x = 0$, $y = 6$

- Possible world – model
- m is a model of α – the sentence α is true in model m

Entailment

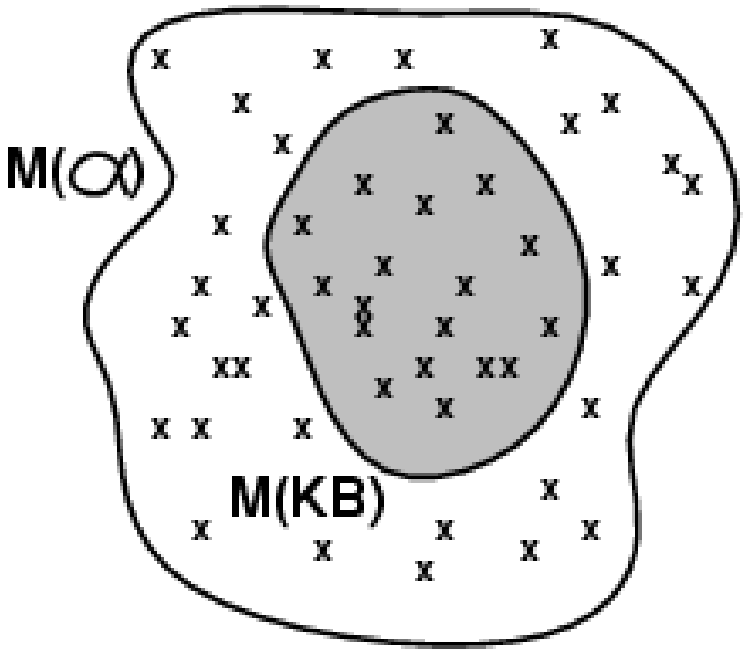
- Entailment means that one thing follows from another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true
  - If α true then *KB* must also be 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

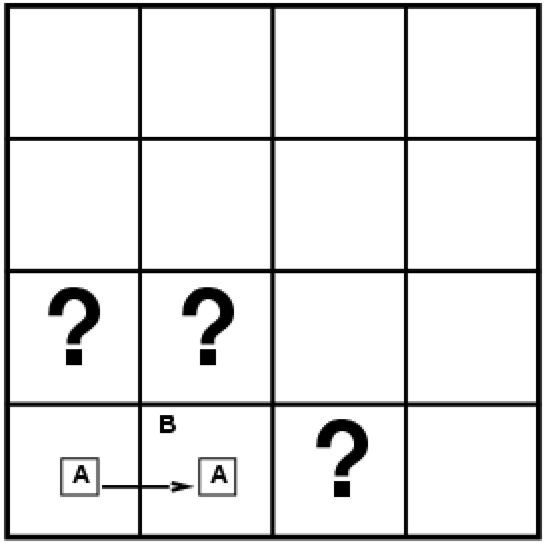
Models

- 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(α)* is the set of all models of α
- Then KB $\models$ α iff *M(KB)* $\subseteq$ *M*(α)
  - E.g. *KB* = Giants won and Reds won
    α = Giants won



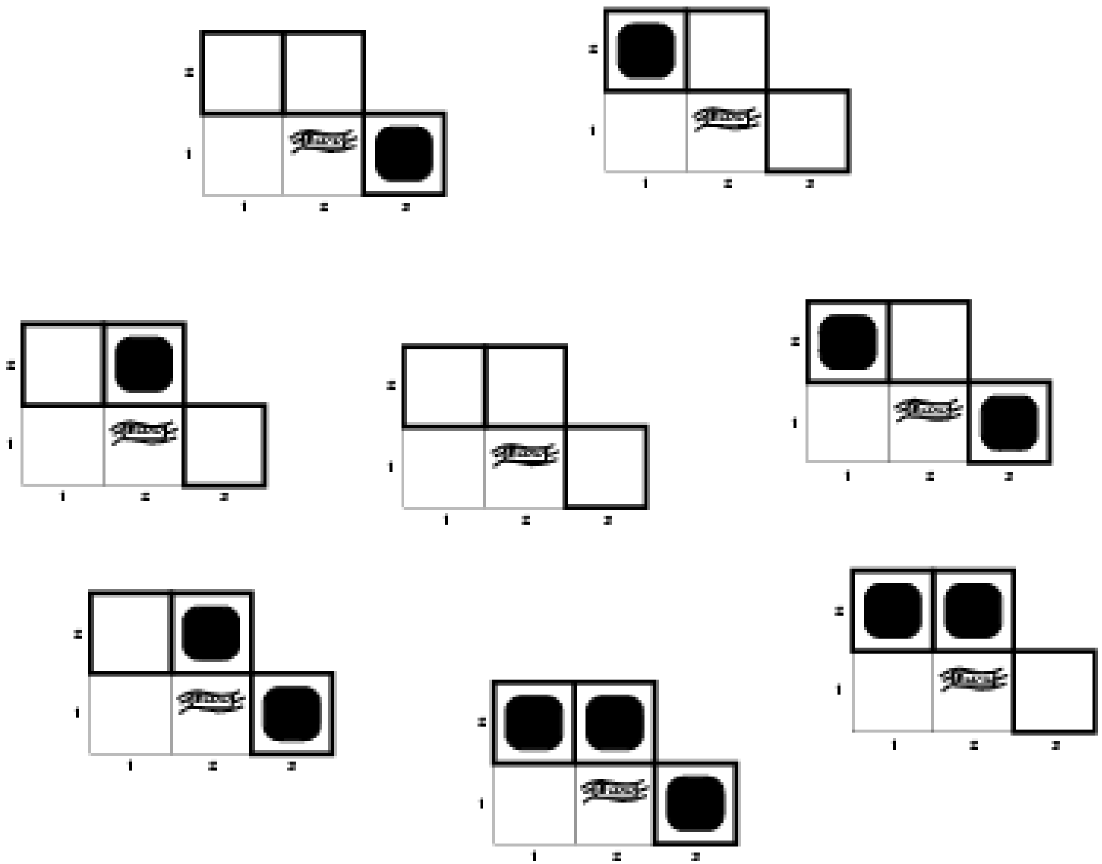
Entailment in the wumpus world

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



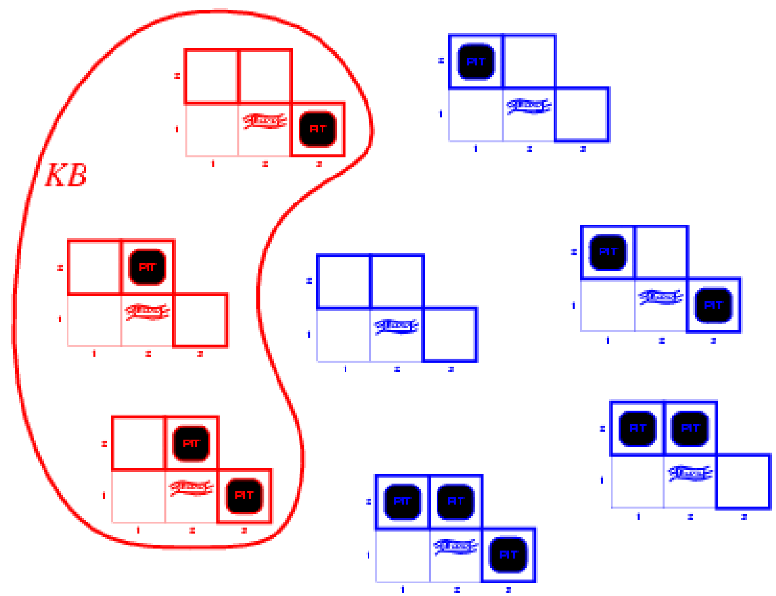
Wumpus models

3 Boolean choices ⇒ 8 possible models
for the adjacent squares [1,2], [2,2] and [3,1] to contain pits



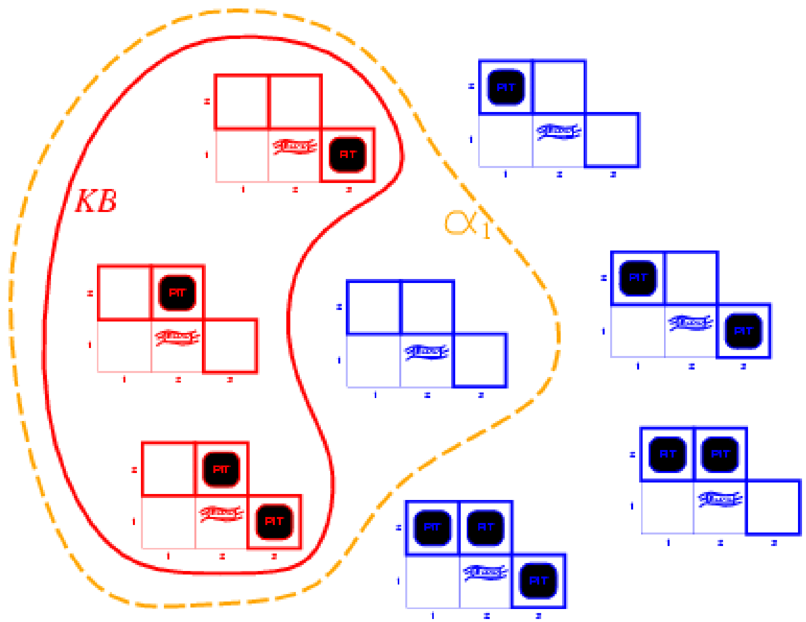
Wumpus models

Consider possible models for *KB* assuming only pits



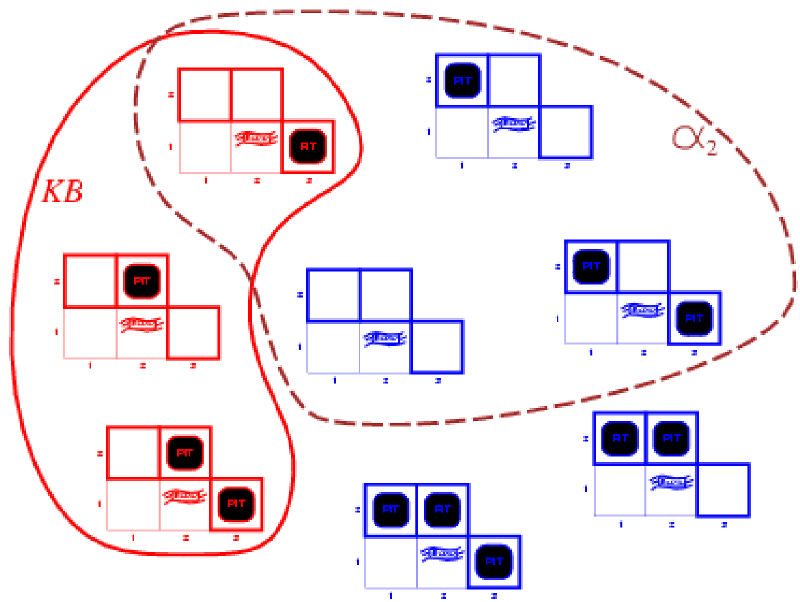
- *KB* = wumpus-world rules + observations
- KB is false in any model in which [1,2] contains a pit, because there is no breeze in [1,1]

Wumpus models



- Consider α_1 = "[1,2] is safe" = "There is no pit in [1,2]"
- *In every model KB is true* α_1 *is also true*
- *KB* $\models \alpha_1$, proved by model checking
- We can conclude that there is no pit in [1,2]

Wumpus models



- Consider α_2 = "[2,2] is safe" = "There is no pit in [2,2]"
- *In some models in which KB is true* α_2 *is false*
- *KB* $\not\models$ α_2
- We cannot conclude that there is no pit in [2,2]

Inference

- $KB \vdash_i \alpha$ = sentence α can be derived from KB by a procedure i (an inference algorithm)

- Soundness: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- An inference algorithm that derives only entailed sentences is sound or truth preserving (model checking is a sound procedure)
- Completeness: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
- An inference algorithm is complete if it can derive any sentence that is entailed

- Think set of all consequences of KB as a haystack and α as a needle. Entailment is like the needle being in the haystack, and inference is like finding it
- An unsound inference procedure essentially makes things up as it goes along – it announces the discovery of nonexistent needles
- For completeness, a systematic examination can always decide whether the needle is in the haystack which is finite

- If KB is true in the real world then any sentence α derived from KB by a sound inference procedure is also true in real world
  - The conclusions of the reasoning process are guaranteed to be true in any world in which the premises are true

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- Atomic sentences : consists of proposition symbols P_1, P_2
- Complex sentences : constructed from atomic sentences using logical connectives
  - 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)

Precedence

- Use parentheses to specify the precedence
- Otherwise the precedence from highest to lowest is: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- $A \Rightarrow B \Rightarrow C$ is not allowed

$\neg$	highest
$\wedge$	
$\vee$	
$\rightarrow$	
$\leftrightarrow$	lowest

$A \vee B \wedge C$	$A \vee (B \wedge C)$
$A \wedge B \rightarrow C \vee D$	$(A \wedge B) \rightarrow (C \vee D)$
$A \rightarrow B \vee C \leftrightarrow D$	$(A \rightarrow (B \vee C)) \leftrightarrow D$

- Precedence rules enable "shorthand" form of sentences, but formally only the fully parenthesized form is legal.
- Syntactically ambiguous forms allowed in shorthand only when semantically equivalent: $A \wedge B \wedge C$ is equivalent to $(A \wedge B) \wedge C$ and $A \wedge (B \wedge C)$

Propositional logic: Semantics

Semantics defines the rules for determining the truth of a sentence with respect to a particular model

Each model specifies true/false for each proposition symbol

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

True is true in every model, *False* is false in every model

The truth value of every other proposition symbol must be specified directly in the model

For the complex sentences

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

Important shorthand

$$S_1 \Rightarrow S_2 \equiv \neg S_1 \vee S_2$$

$$S_1 \Leftrightarrow S_2 \equiv S_1 \Rightarrow S_2 \wedge S_2 \Rightarrow S_1$$

Truth tables for connectives

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

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

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

Implication: if P is true then I am claming that Q is true, otherwise I am making no claim
The sentence is false, if P is true but Q is false

Biconditional: True whenever both P->Q and Q->P is true
(e.g. a square is breezy if and only if adjacent square has a pit: implication requires the presence of pit if there is a breeze, biconditional also requires the absence of pit if there is no breeze)

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].

Knowledge base includes:

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

R2: $\neg B_{1,1}$ No breeze in [1.1]

R3: $B_{2,1}$ Breeze in [2,1]

- "Pits cause breezes in adjacent squares"

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$$KB = R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5$$

Inference

- Decide whether $KB \models \alpha$
- First method: enumerate the models and check that α is true in every model in which KB is true
- $B_{1,1}$ $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{3,1}$
- 7 symbols : $2^7 = 128$ possible models

Truth tables for inference

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

KB = R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5

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

$\alpha 2 = P_{2,2}$

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

α1 is true in all models that KB is true

α2 is true only in two models that KB is true, but false in the other one

Inference by enumeration

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

```
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))
```

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

Logical equivalence

- Two sentences are logically equivalent iff they are true in same models:
- $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

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

Validity and satisfiability

A sentence is valid if it is true in all models,

e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Valid sentences are tautologies

Every valid sentence is equivalent to *True*

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

Every valid implication sentence describes a legitimate inference

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

If a sentence is true in a model m, then we say m satisfies the sentence, or a model of the sentence

A sentence is unsatisfiable if it is true in no models

e.g., $A \wedge \neg A$

α is valid iff $\neg \alpha$ is unsatisfiable, α is satisfiable iff $\neg \alpha$ is not valid

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

Examples

Sentence	Valid?	Interpretation that make sentence's truth value = **f**
$smoke \rightarrow smoke$ $smoke \vee \neg smoke$	valid	
$smoke \rightarrow fire$	satisfiable, not valid	$smoke =$ **t**, $fire =$ **f**
$(s \rightarrow f) \rightarrow (\neg s \rightarrow \neg f)$	satisfiable, not valid	$s =$ **f**, $f =$ **t** $s \rightarrow f =$ **t**, $\neg s \rightarrow \neg f =$ **f**
contrapositive $(s \rightarrow f) \rightarrow (\neg f \rightarrow \neg s)$	valid	
$b \vee d \vee (b \rightarrow d)$ $b \vee d \vee \neg b \vee d$	valid	

Satisfiability

- Related to constraint satisfaction
- Given a sentence S, try to find an interpretation i where S is true
- Analogous to finding an assignment of values to variables such that the constraint hold
- Example problem: scheduling nurses in a hospital
  - Propositional variables represent for example that Nurse1 is working on Tuesday at 2
  - Constraints on the schedule are represented using logical expressions over the variables
- Brute force method: enumerate all interpretations and check

Example Problem

Imagine we knew that:

- If today is sunny, then Tomas will be happy ($S \rightarrow H$)
- If Tomas is happy, the lecture will be good ($H \rightarrow G$)
- Today is sunny (S)

Should we conclude that the lecture will be good?

Checking Interpretations

- Start by figuring out what set of interpretations make our original sentences true.
- Then, if G is true in all those interpretations, it must be OK to conclude it from the sentences we started out with (our knowledge base).
- ***In a universe with only three variables, there are 8 possible interpretations in total.***

S	H	G
t	t	t
t	t	f
t	f	t
t	f	f
f	t	t
f	t	f
f	f	t
f	f	f

Checking Interpretations

- Only one of these interpretations makes all the sentences in our knowledge base true:
- S = true, H = true, G = true.

S	H	G	S→ H	H→G	S
t	t	t	t	t	t
t	t	f	t	f	t
t	f	t	f	t	t
t	f	f	f	t	t
f	t	t	t	t	f
f	t	f	t	f	f
f	f	t	t	t	f
f	f	f	t	t	f

Checking Interpretations

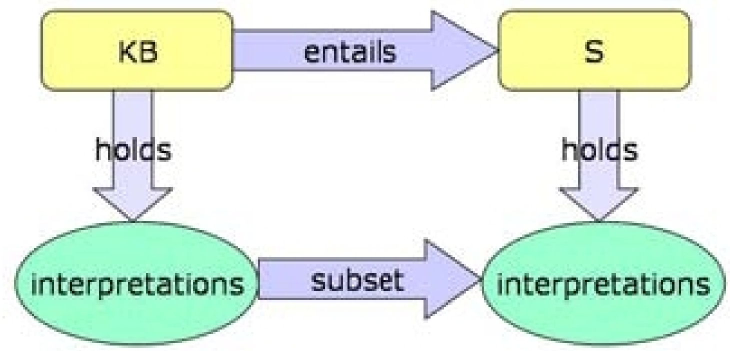
- it's easy enough to check that G is true in that interpretation, so it seems like it must be reasonable to draw the conclusion that the lecture will be good.

S	H	G	S →H	H→G	S	G
t	t	t	t	t	t	t
t	t	f	t	f	t	f
t	f	t	f	t	t	t
t	f	f	f	t	t	f
f	t	t	t	t	f	t
f	t	f	t	f	f	f
f	f	t	t	t	f	t
f	f	f	t	t	f	f

Computing entailment

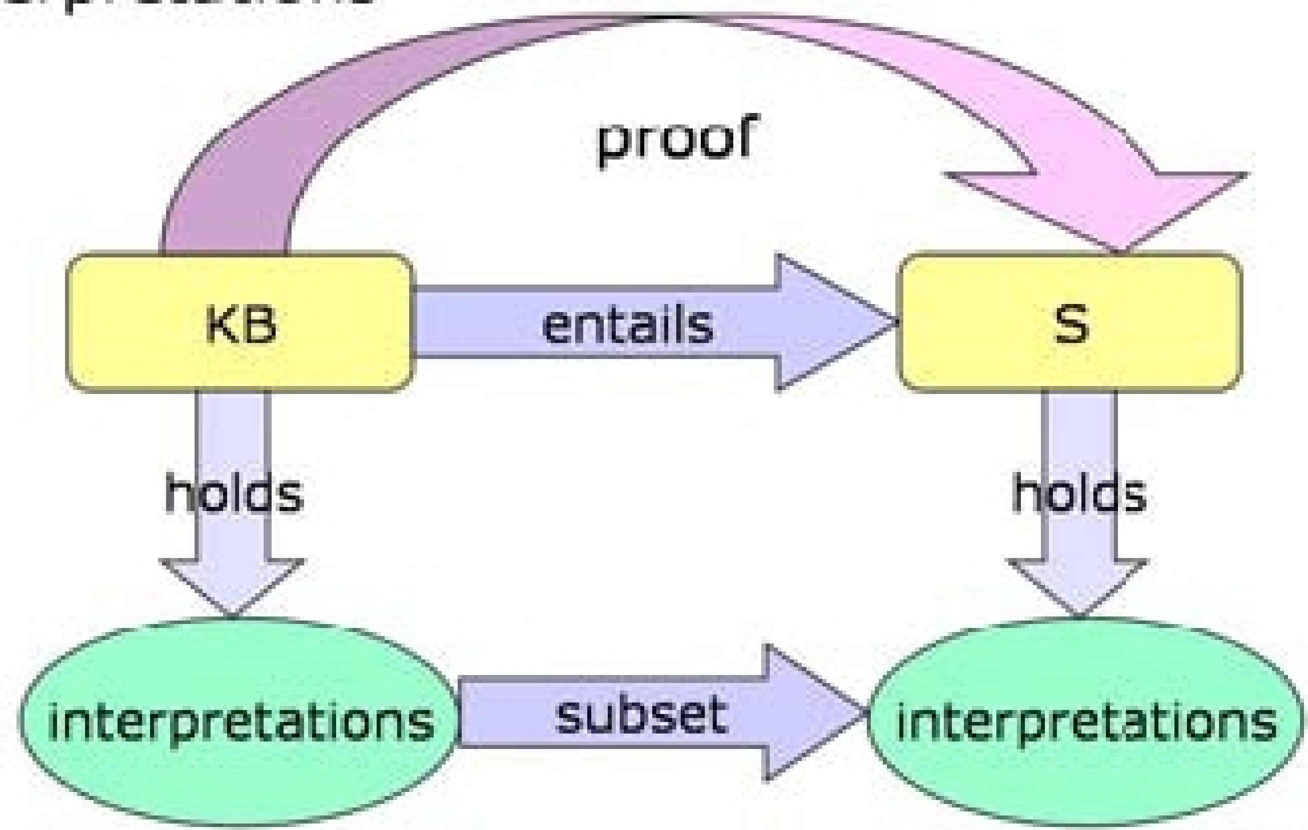
A knowledge base (KB) *entails* a sentence S iff every interpretation that makes KB true also makes S true

- enumerate all interpretations
- select those in which all elements of KB are true
- check to see if S is true in all of those interpretations



Entailment and Proof

A proof is a way to test whether a KB entails a sentence, without enumerating all possible interpretations



Proof

- Proof is a sequence of sentences
- First ones are premises (KB)
- Then, you can write down on the next line the result of applying an inference rule to previous lines
- When S is on a line, you have proved S from KB

- If inference rules are *sound*, then any S you can prove from KB is entailed by KB

- If inference rules are *complete*, then any S that is entailed by KB can be proved from KB

Logical equivalence

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

Natural deduction

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \qquad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Biconditional Elimination

Some inference rules:

$$\frac{\alpha \rightarrow \beta \quad \alpha}{\beta}$$

Modus ponens

$$\frac{\alpha \rightarrow \beta \quad \neg \beta}{\neg \alpha}$$

Modus tolens

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

And-introduction

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

And-elimination

Example

Prove S

Step	Formula	Derivation
1	$P \wedge Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \wedge R) \rightarrow S$	Given

Example

Prove S

Step	Formula	Derivation
1	$P \wedge Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \wedge R) \rightarrow S$	Given
4	P	1 And-Elim

Example

Prove S

Step	Formula	Derivation
1	$P \wedge Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \wedge R) \rightarrow S$	Given
4	P	1 And-Elim
5	R	4,2 Modus Ponens

Example

Prove S

Step	Formula	Derivation
1	$P \wedge Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \wedge R) \rightarrow S$	Given
4	P	1 And-Elim
5	R	4,2 Modus Ponens
6	Q	1 And-Elim

Example

Prove S

Step	Formula	Derivation
1	$P \wedge Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \wedge R) \rightarrow S$	Given
4	P	1 And-Elim
5	R	4,2 Modus Ponens
6	Q	1 And-Elim
7	$Q \wedge R$	5,6 And-Intro

Example

Prove S

Step	Formula	Derivation
1	$P \wedge Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \wedge R) \rightarrow S$	Given
4	P	1 And-Elim
5	R	4,2 Modus Ponens
6	Q	1 And-Elim
7	$Q \wedge R$	5,6 And-Intro
8	S	7,3 Modus Ponens

Example from Wumpus World

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

KB = R1 $\wedge$ R2 $\wedge$ R3 $\wedge$ R4 $\wedge$ R5

Prove $\alpha 1 = \neg P_{1,2}$

Example from Wumpus World

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R6 : $B_{1,1} \Leftrightarrow (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ Biconditional elimination

Example from Wumpus World

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R6 : $B_{1,1} \Leftrightarrow (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ Biconditional elimination

R7 : $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ And Elimination

Example from Wumpus World

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R6 : $B_{1,1} \Leftrightarrow (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ Biconditional elimination

R7 : $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ And Elimination

R8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$ Equivalence for contrapositives

Example from Wumpus World

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R6 : $B_{1,1} \Leftrightarrow (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ Biconditional elimination

R7 : $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ And Elimination

R8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$ Equivalence for contrapositives

R9: $\neg (P_{1,2} \vee P_{2,1})$ Modus Ponens with R2 and R8

Example from Wumpus World

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R6 : $B_{1,1} \Leftrightarrow (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ Biconditional elimination

R7 : $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ And Elimination

R8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$ Equivalence for contrapositives

R9: $\neg (P_{1,2} \vee P_{2,1})$ Modus Ponens with R2 and R8

R10: $\neg P_{1,2} \wedge \neg P_{2,1}$ De Morgan's Rule

Example from Wumpus World

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R6 : $B_{1,1} \Leftrightarrow (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ Biconditional elimination

R7 : $((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$ And Elimination

R8: $(\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$ Equivalence for contrapositives

R9: $\neg (P_{1,2} \vee P_{2,1})$ Modus Ponens with R2 and R8

R10: $\neg P_{1,2} \wedge \neg P_{2,1}$ De Morgan’s Rule

R11: $\neg P_{1,2}$ And Elimination

Monotonicity

- The set of entailed sentences can only increase as information is added to the knowledge base

- If
- $KB \models \alpha$
- Then
- $KB \land \beta \models \alpha$

Proof systems

- There are many natural deduction systems; they are typically "proof checkers", sound but not complete
- Natural deduction uses lots of inference rules which introduces a large branching factor in the search for a proof.
- In general, you need to do "proof by cases" which introduces even more branching.

Prove R

1	$P \vee Q$
2	$Q \rightarrow R$
3	$P \rightarrow R$

Resolution

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

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

….

R11: $\neg B_{1,2}$

R12: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

R13: $\neg P_{2,2}$

R14: $\neg P_{1,3}$

R15: $(P_{1,1} \vee P_{2,2} \vee P_{3,1})$ biconditional elimination on R3, followed by a Modus Ponens with R5

R16: $(P_{1,1} \vee P_{3,1})$ Resolution with $\neg P_{2,2}$ in R13

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]

R17: $P_{3,1}$ Resolve with $\neg P_{1,1}$ in R1

Resolution

- Resolution rule:

$$\frac{\begin{array}{c}\alpha \vee \beta \\ \neg\beta \vee \gamma\end{array}}{\alpha \vee \gamma}$$

- Single inference rule is a sound and complete proof system

- Requires all sentences to be converted to conjunctive normal form

Conjunctive Normal form

- Conjunctive normal form (CNF) formulas:

$$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$$

  - $(A \vee B \vee \neg C)$ is a clause, which is a disjunction of literals
  - A, B, and ¬ C are literals, each of which is a variable or the negation of a variable.
  - Each clause is a requirement that must be satisfied and can be satisfied in multiple ways
  - Every sentence in propositional logic can be written in CNF

Converting to CNF

1. Eliminate arrows using definitions
2. Drive in negations using De Morgan's Laws

$$\neg(\phi \vee \varphi) \equiv \neg\phi \wedge \neg\varphi$$
$$\neg(\phi \wedge \varphi) \equiv \neg\phi \vee \neg\varphi$$

3. Distribute **or** over **and**

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

4. Every sentence can be converted to CNF, but it may grow exponentially in size

CNF Conversion Example

$$(A \vee B) \rightarrow (C \rightarrow D)$$

1. Eliminate arrows

$$\neg(A \vee B) \vee (\neg C \vee D)$$

2. Drive in negations

$$(\neg A \wedge \neg B) \vee (\neg C \vee D)$$

3. Distribute

$$(\neg A \vee \neg C \vee D) \wedge (\neg B \vee \neg C \vee D)$$

Resolution

- Resolution rule:

$$\frac{\alpha \vee \beta \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can't apply any more
- Resolution refutation is sound and complete
  - If we derive a contradiction, then the conclusion follows from the axioms
  - If we can't apply any more, then the conclusion cannot be proved from the axioms.

Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion

Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	•	4,8

Example

Prove R

1	P v Q
2	P → R
3	Q → R

false v R

¬ R v false

false v false

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	•	4,8

The power of false

Prove Z

1	P
2	¬ P

Step	Formula	Derivation
1	P	Given
2	¬ P	Given
3	¬ Z	Negated conclusion
4	•	1,2

Note that $(P \wedge \neg P) \rightarrow Z$ is valid

Any conclusion follows from a contradiction – and so strict logic systems are very brittle.

Example

Prove R

1	$(P \to Q) \to Q$
2	$(P \to P) \to R$
3	$(R \to S) \to \neg(S \to Q)$

Convert to CNF

- $\neg(\neg P \lor Q) \lor Q$
- $(P \land \neg Q) \lor Q$
- $(P \lor Q) \land (\neg Q \lor Q)$
- $(P \lor Q)$

- $\neg(\neg P \lor P) \lor R$
- $(P \land \neg P) \lor R$
- $(P \lor R) \land (\neg P \lor R)$

- $\neg(\neg R \lor S) \lor \neg(\neg S \lor Q)$
- $(R \land \neg S) \lor (S \land \neg Q)$
- $(R \lor S) \land (\neg S \lor S) \land (R \lor \neg Q) \land (\neg S \lor \neg Q)$
- $(R \lor S) \land (R \lor \neg Q) \land (\neg S \lor \neg Q)$

Example

Prove R

1	$(P \to Q) \to Q$
2	$(P \to P) \to R$
3	$(R \to S) \to \neg (S \to Q)$

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9	$\neg Q$	6,8
10	P	1,9
11	R	3,10
12	•	7,11

Resolution

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals

clauses

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

- Resolution inference rule (for CNF):

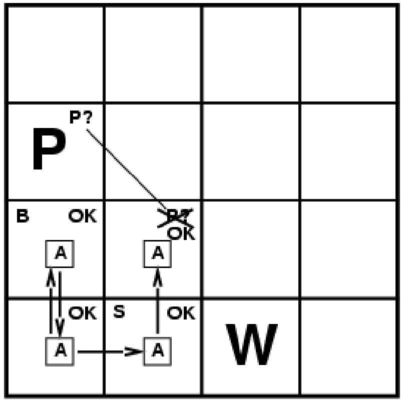
$$\frac{l_i \vee \ldots \vee l_k, \qquad m_1 \vee \ldots \vee m_n}{l_i \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary 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

Conversion to CNF



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

3. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

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

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

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

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

4. Apply distributivity law ($\land$ over $\lor$) and flatten:

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

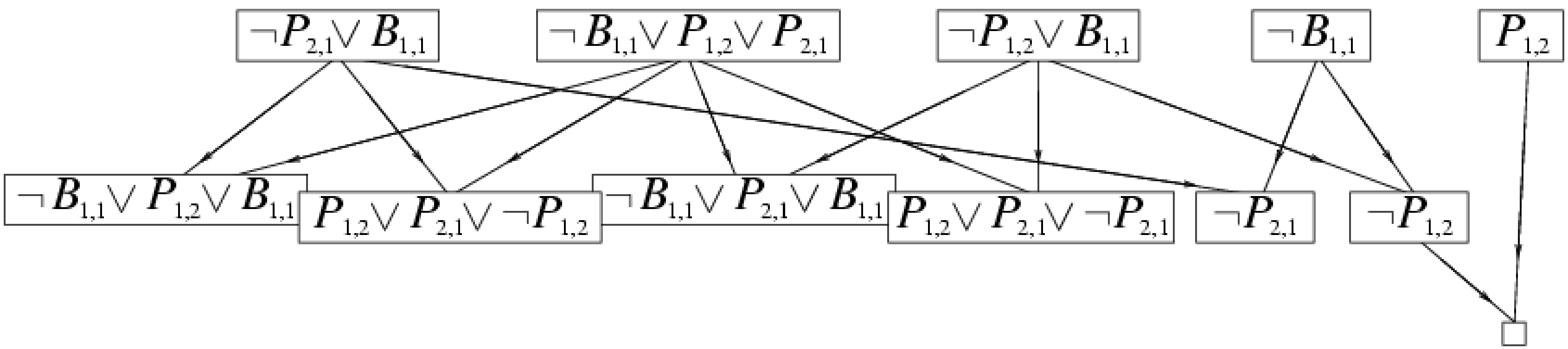
Resolution algorithm

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

```
function PL-RESOLUTION(KB, α) returns true or false
    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
```

Resolution example

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$ $\alpha = \neg P_{1,2}$



Forward and backward chaining

- Horn Form (restricted)

  KB = conjunction of Horn clauses
  - Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) $\Rightarrow$ symbol
  - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
- Modus Ponens (for Horn Form): complete for Horn KBs

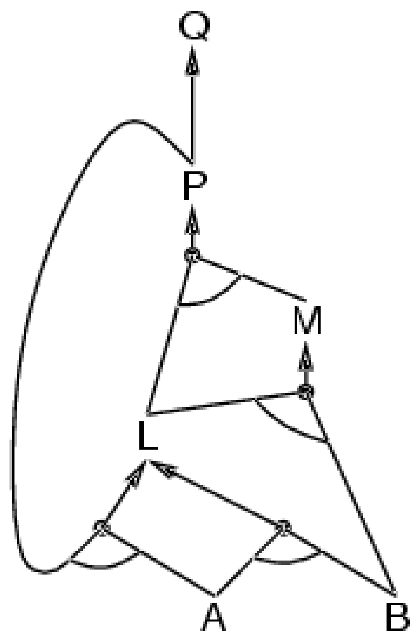
$$\frac{\alpha_1, \ldots ,\alpha_n, \qquad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

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

Forward chaining

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

$$
\begin{aligned}
&P \Rightarrow Q \\
&L \wedge M \Rightarrow P \\
&B \wedge L \Rightarrow M \\
&A \wedge P \Rightarrow L \\
&A \wedge B \Rightarrow L \\
&A \\
&B
\end{aligned}
$$



Forward chaining algorithm

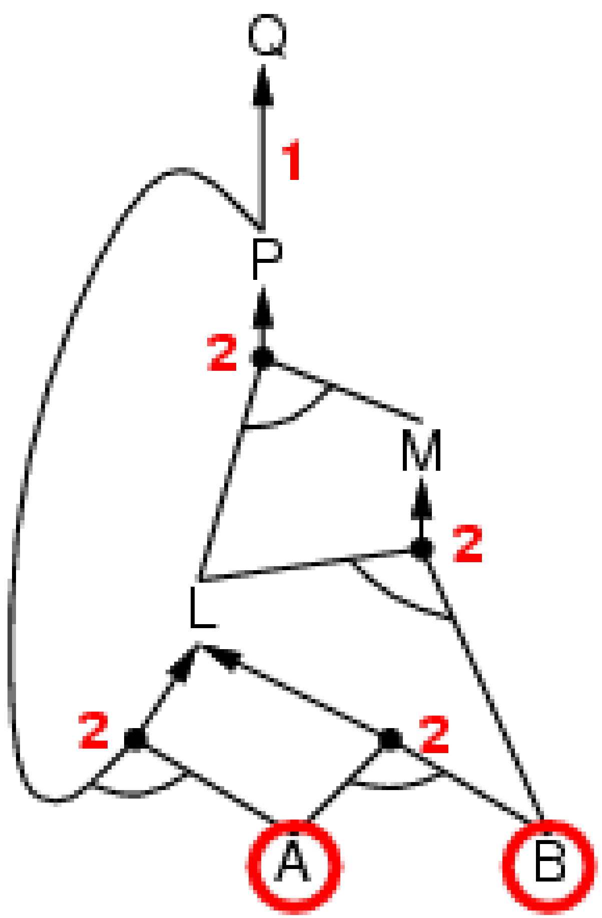
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  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
```

- Fo

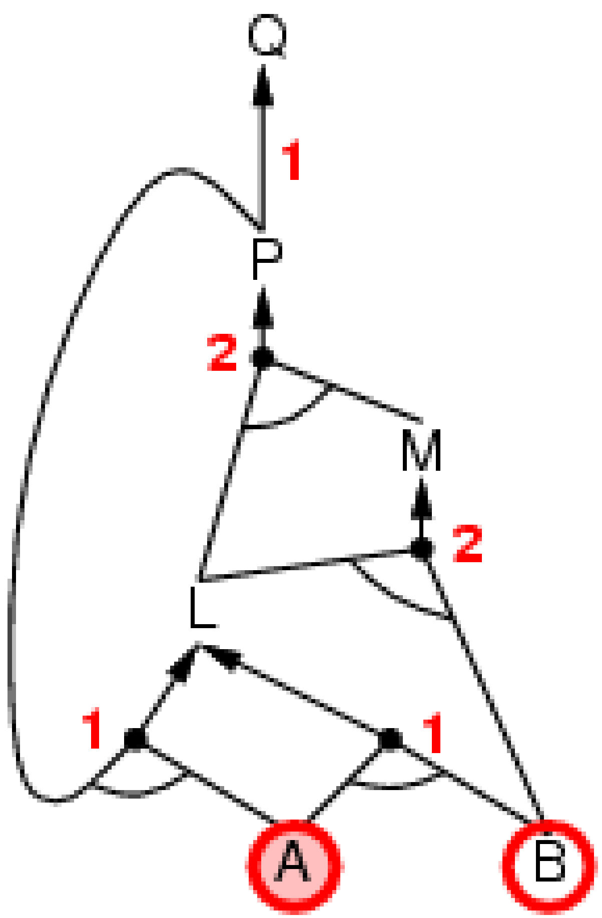
Forward chaining example

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



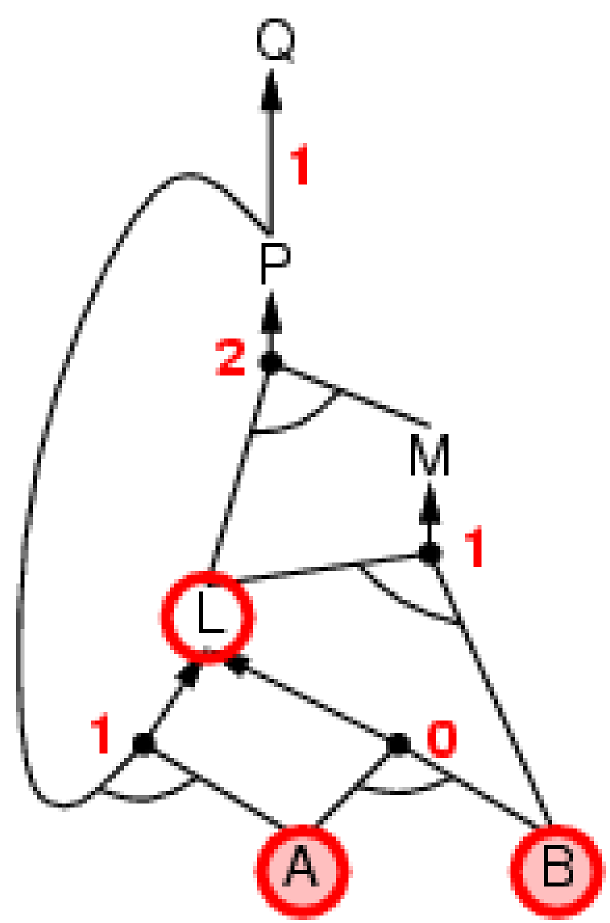
Forward chaining example

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



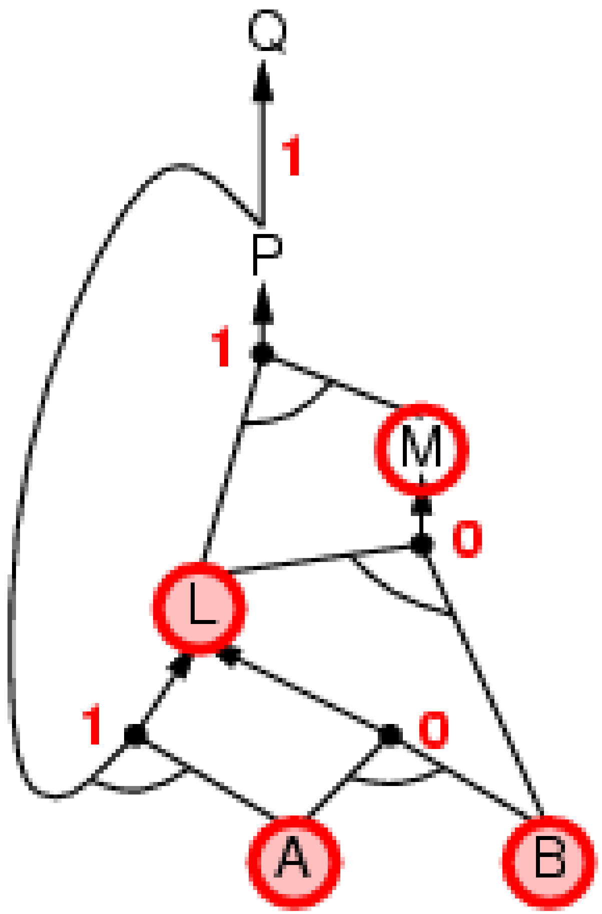
Forward chaining example

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



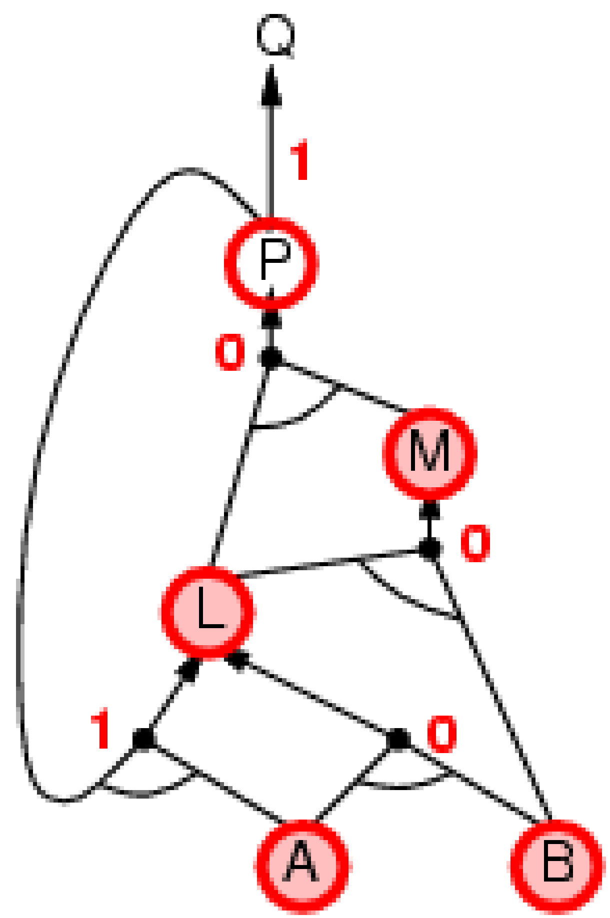
Forward chaining example

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



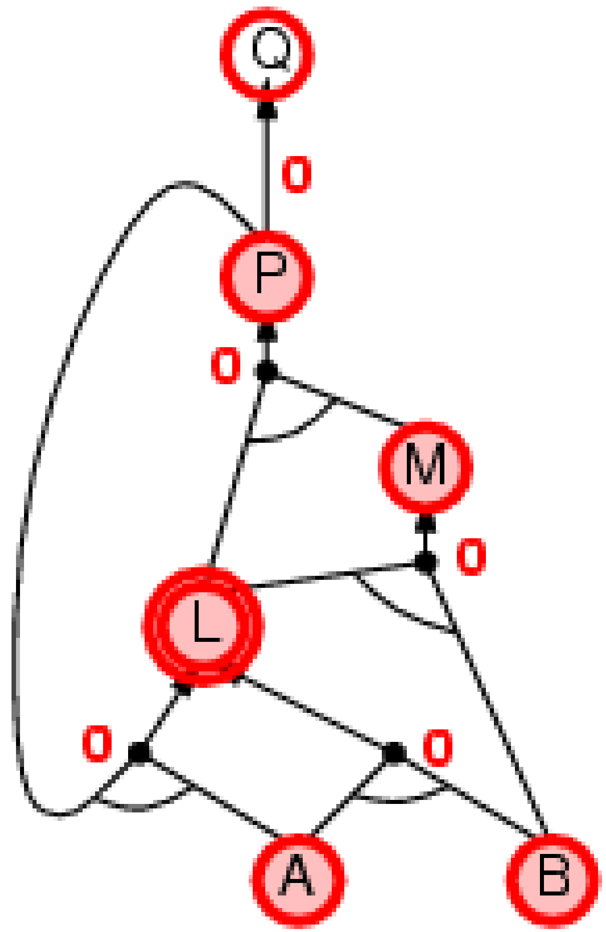
Forward chaining example

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



Forward chaining example

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



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

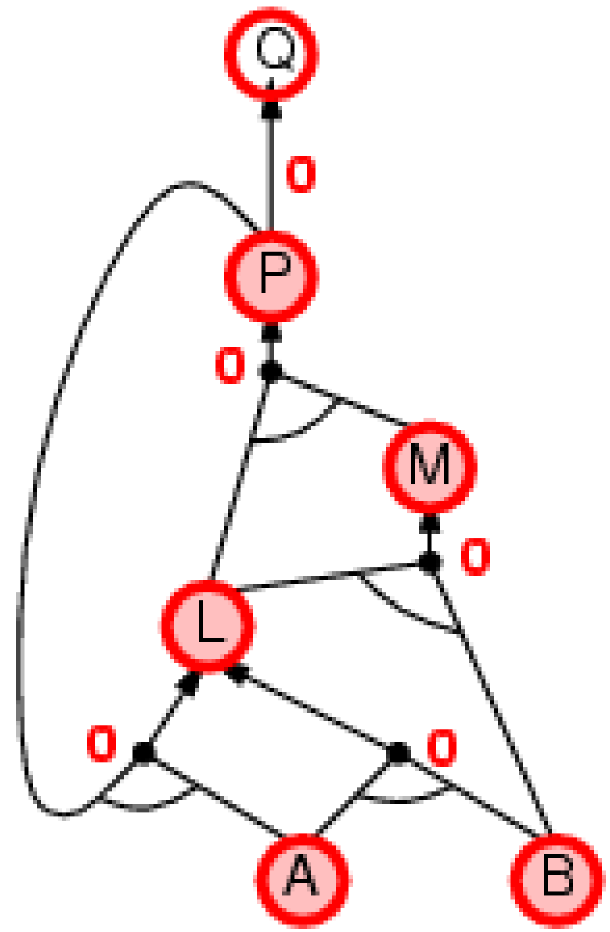
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

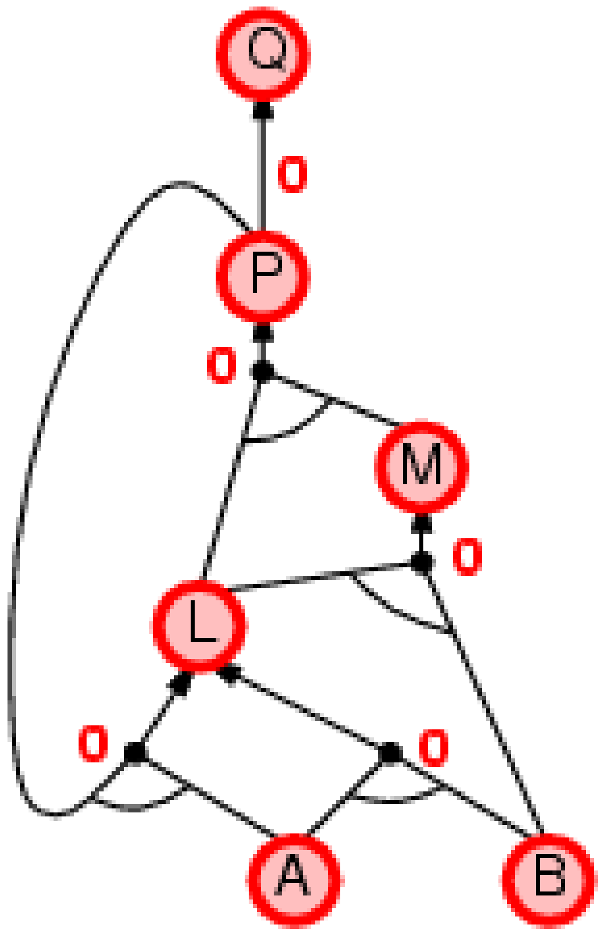
A

B



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



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 the query q:

to prove q by BC,

check if q is known already, or

prove by BC all premises of some rule concluding q

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

Avoid repeated work: check if new subgoal

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

Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

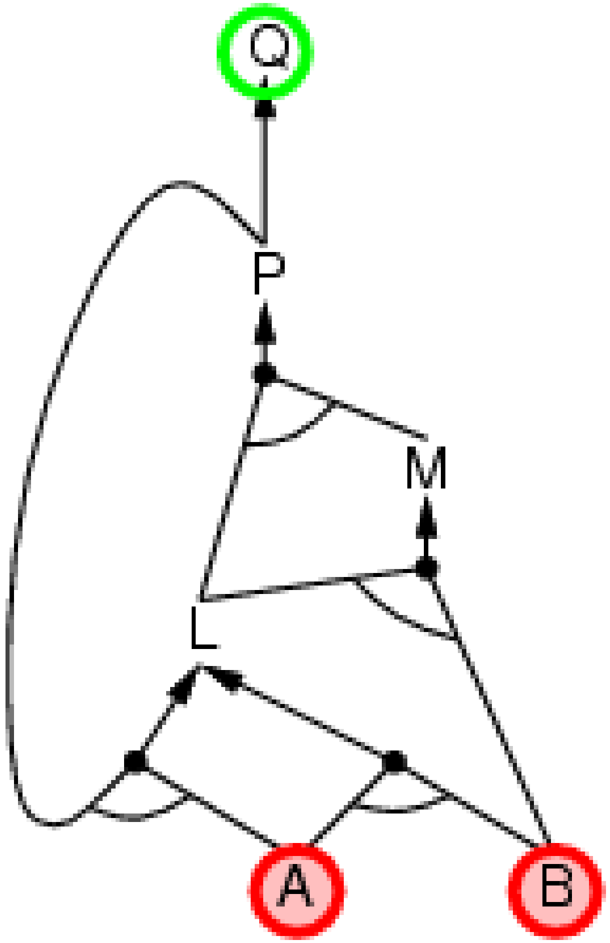
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

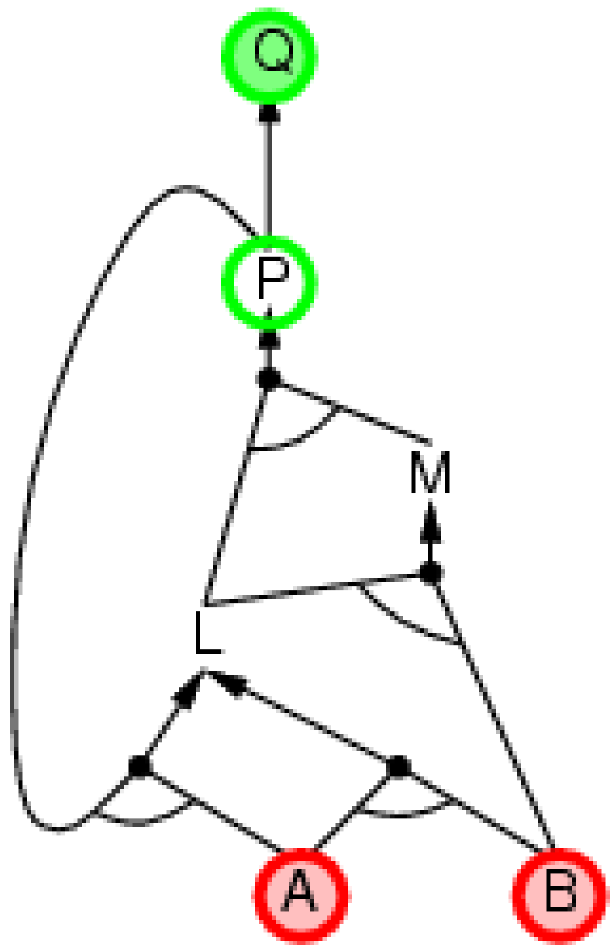
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

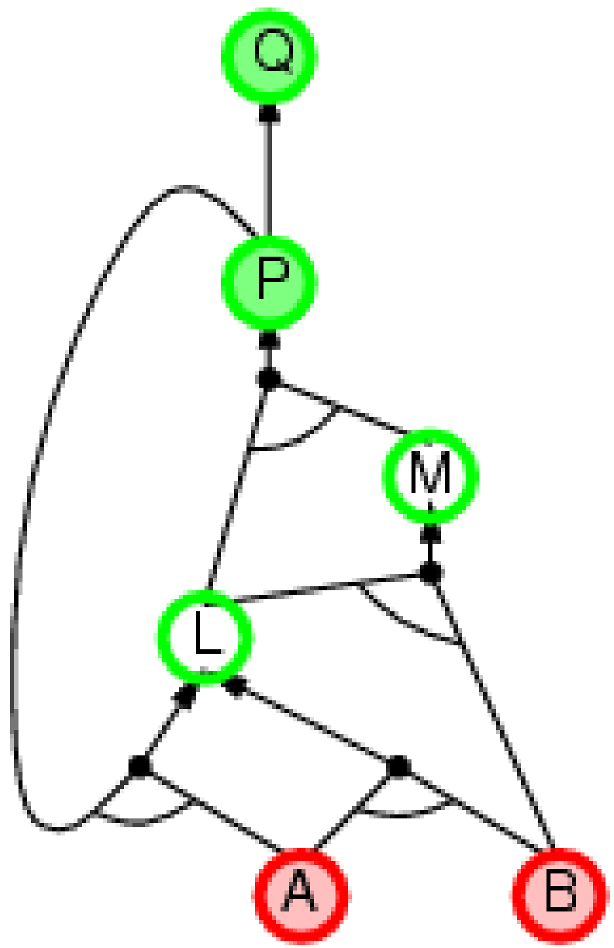
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

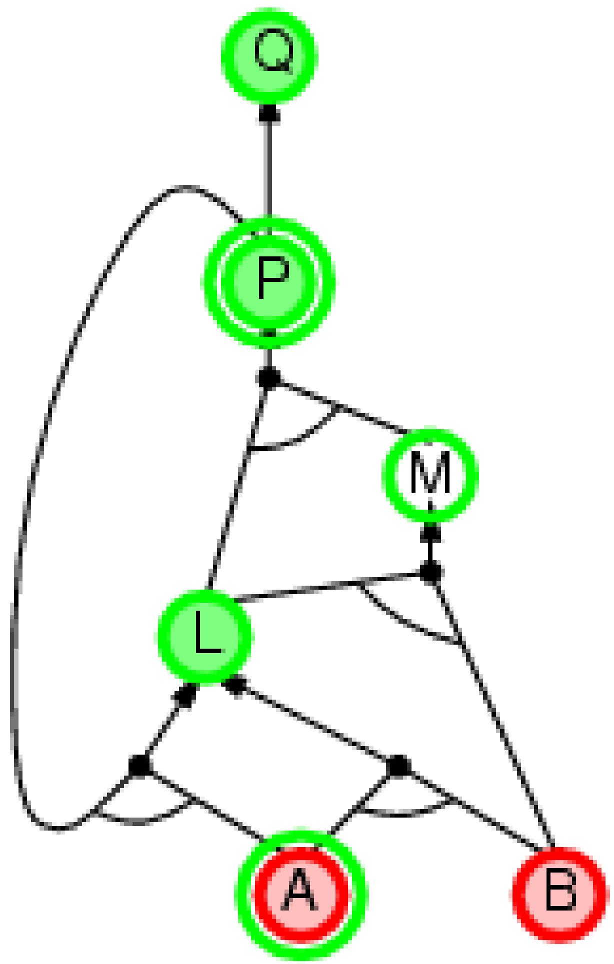
A

B



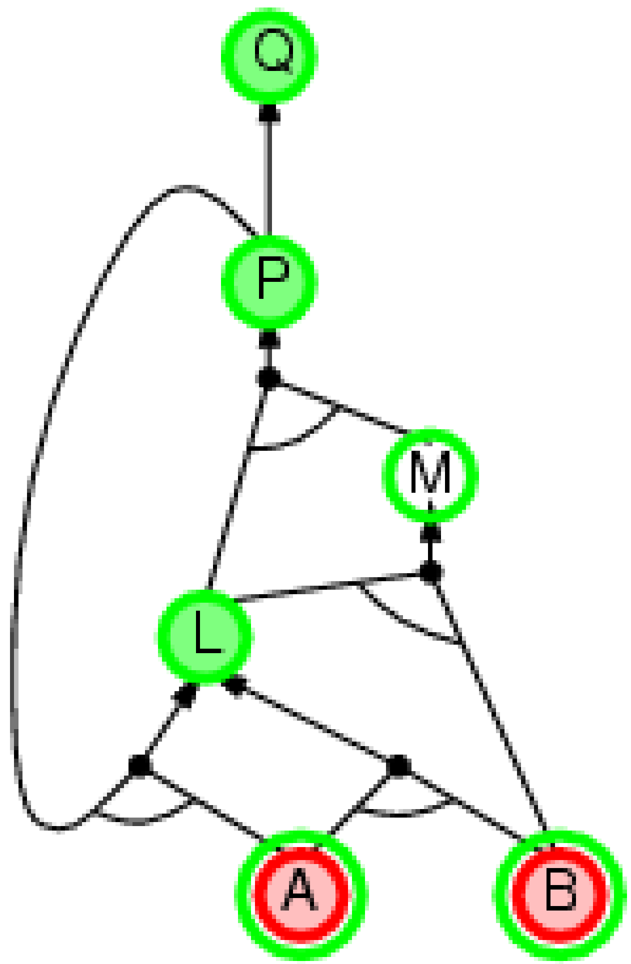
Backward chaining example

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



Backward chaining example

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



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

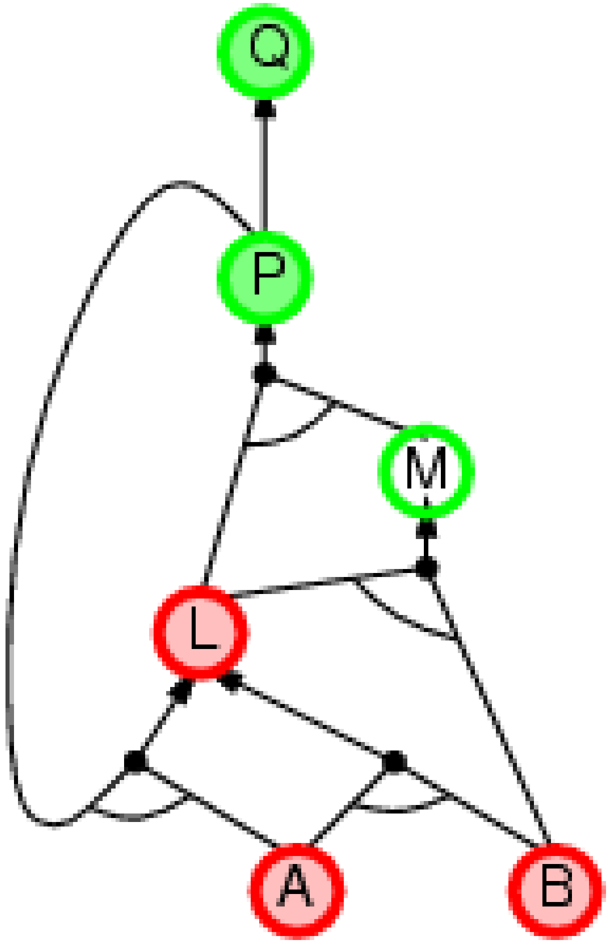
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

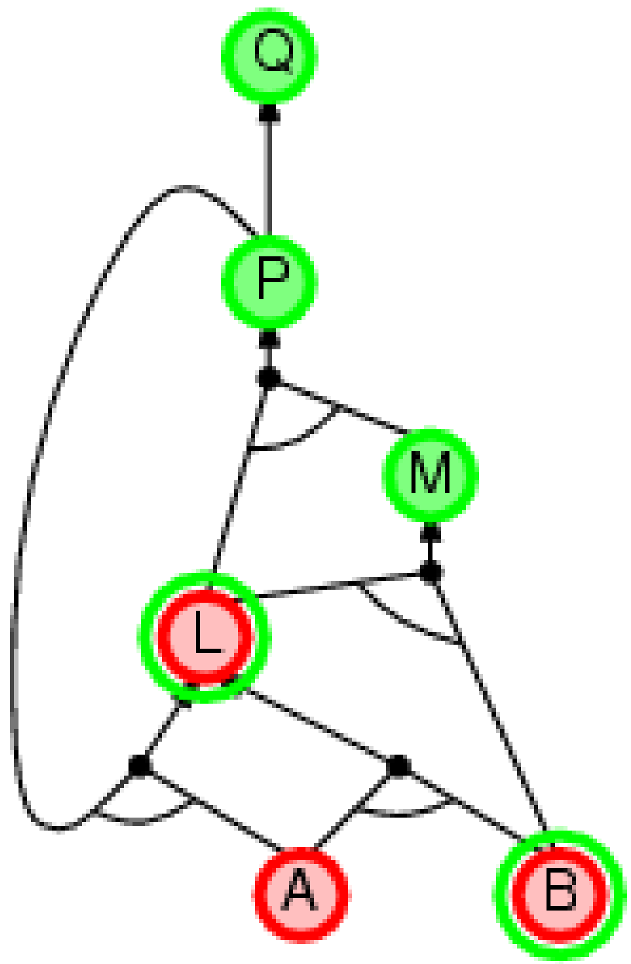
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

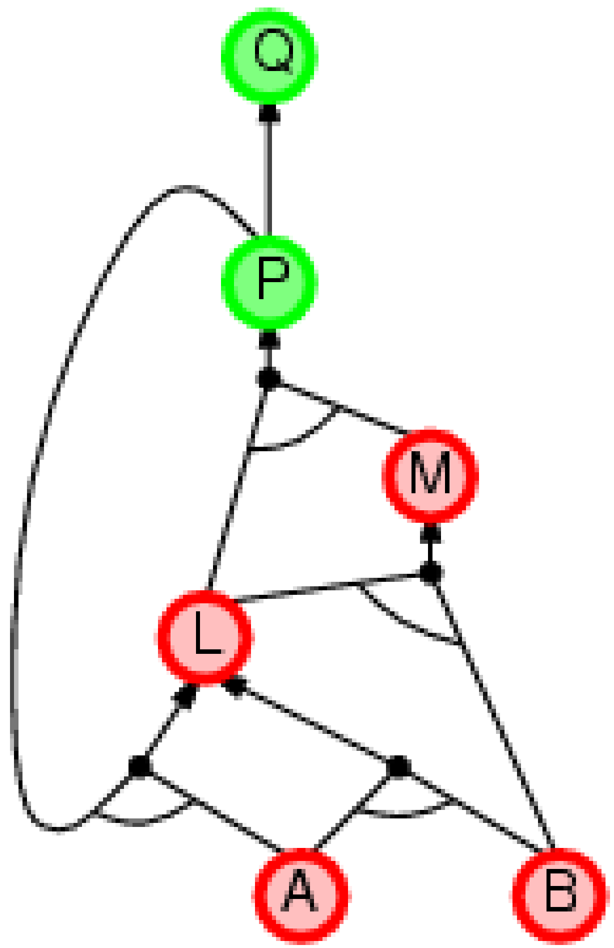
A

B



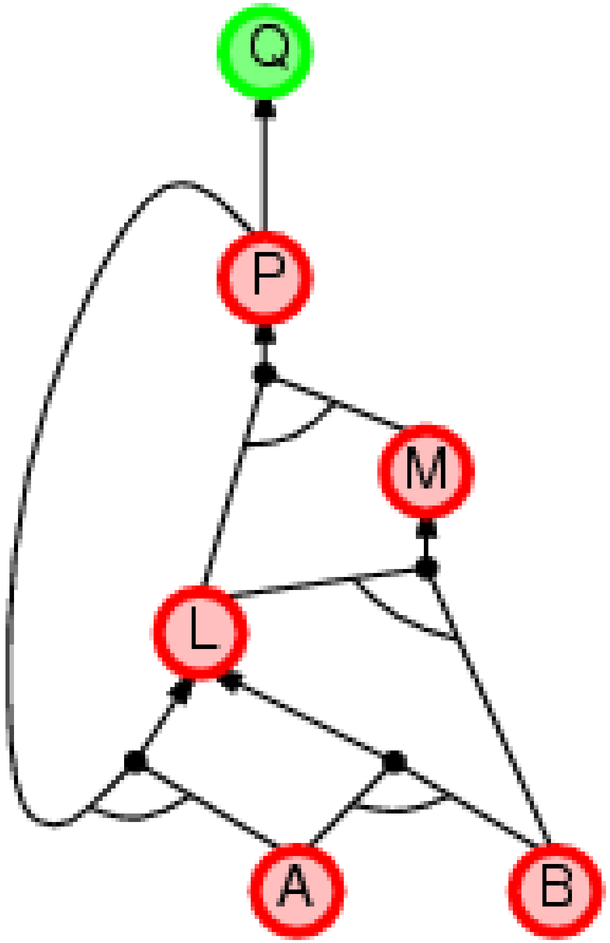
Backward chaining example

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



Backward chaining example

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



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

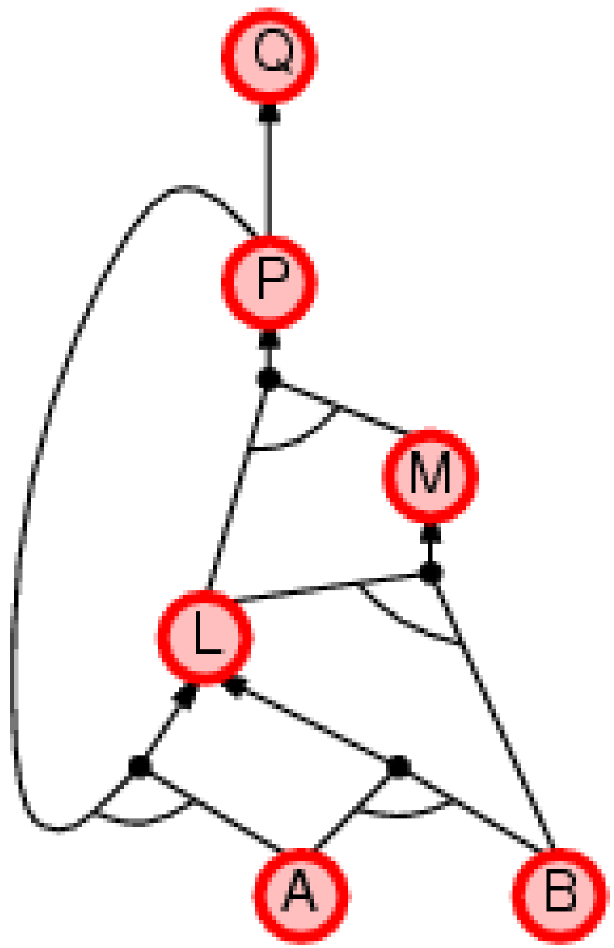
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

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

Proof methods

- Proof methods divide into (roughly) two kinds:
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
      Can use inference rules as operators in a standard search algorithm
    - Typically require transformation of sentences into a normal form
  - Model checking
    - truth table enumeration (always exponential in n)
    - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms

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