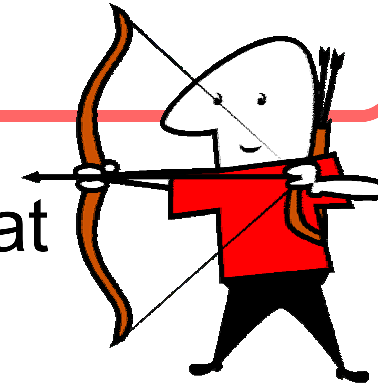
Introduction to Logical Agents (Propositional Logic)

Department of Electrical and Electronics Engineering
Spring 2005
Dr. Afşar Saranlı

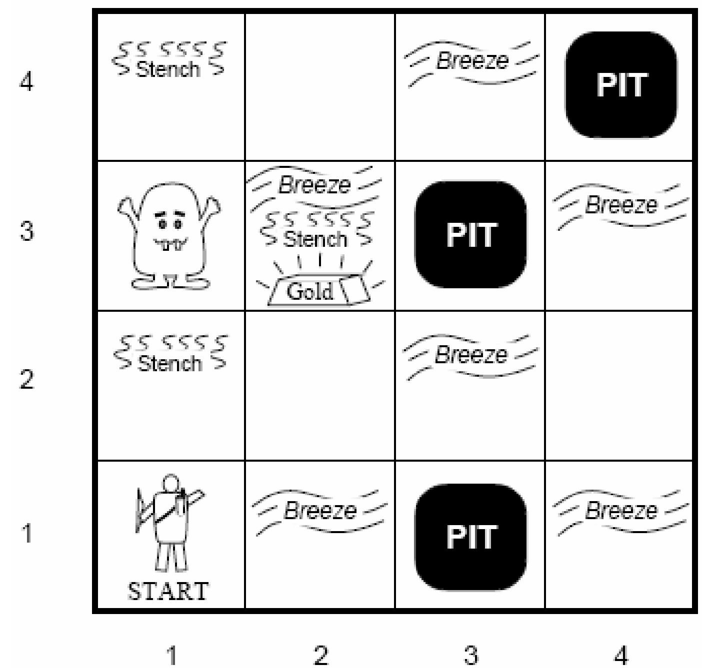
Thanks to Professor Andrew W. Moore (Carnegie Mellon University) http://www.cs.cmu.edu/~awm/tutorials
Also: Artificial Intelligence: A Modern Approach, 2nd Ed., Russel & Norvig

The “Wumpus” World

A very simple computer game (somewhat similar to minesweeper…)

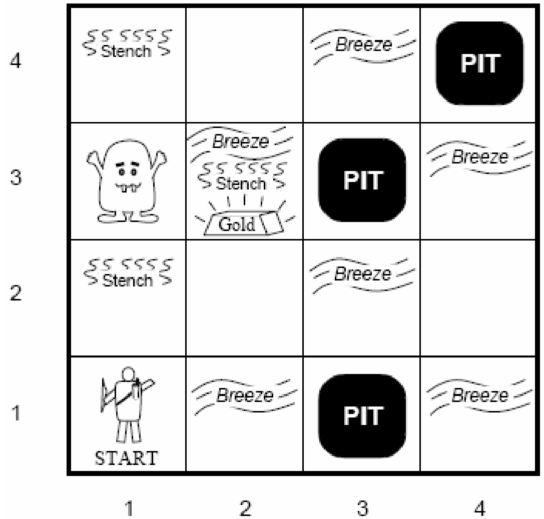


- An agent is exploring,
- Avoid deadly wumpus and pits,
- Find the gold!
- Given: Certain laws of the world + agent’s obervations



The “Wumpus” World – PEAS

- *Performance measure*
  - gold: +1000, death: -1000,
    -1 per step, -10 for using the arrow
- *Environment*
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter i gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
- *Actuators*
  - Left turn, Right turn, Forward, Grab, Release, Shoot
- *Sensors*
  - Breeze, Glitter, Smell

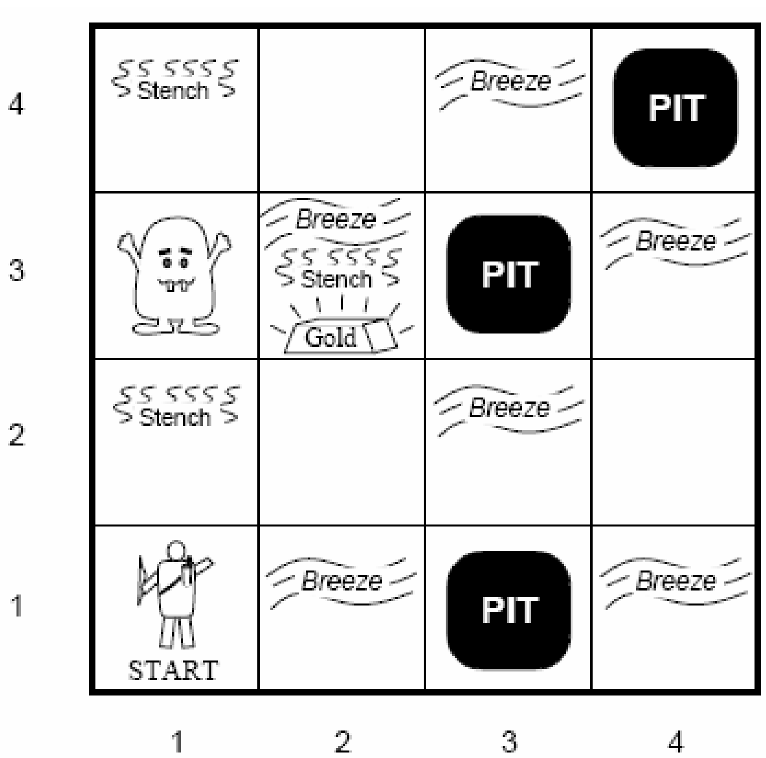


Wumpus World - characterization

- As viewed by the agent…
- 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 -- finite number of states
- Single-Agent ?? Yes -- wumpus is essentially a natural feature

Exploring a wumpus world…

What we do not know…

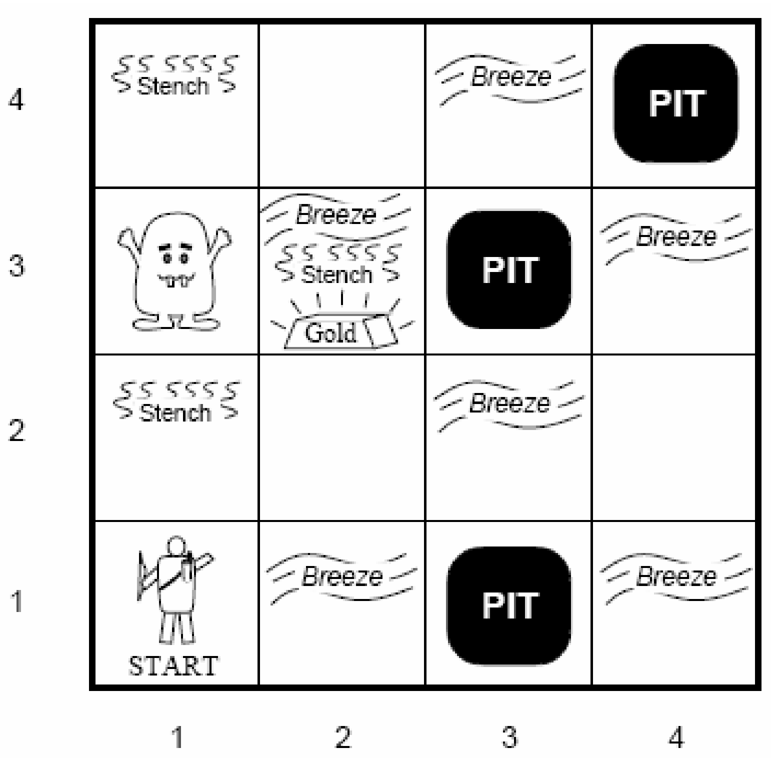


Here is what we know:

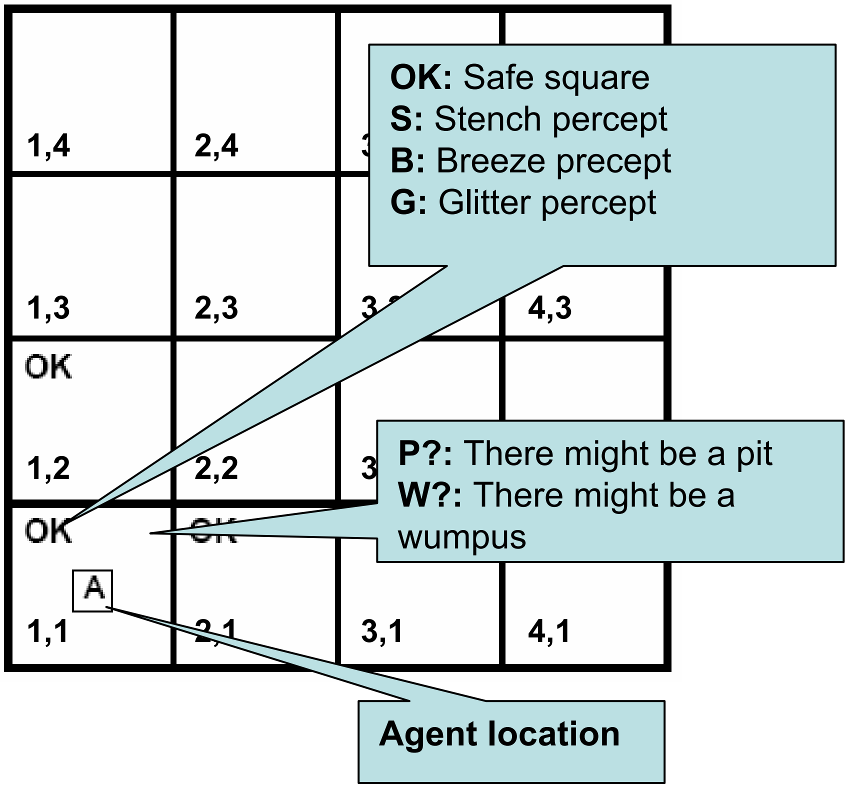
1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
OK 1,2	2,2	3,2	4,2
OK A 1,1	OK 2,1	3,1	4,1

Exploring a wumpus world…

What we do not know…



Here is what we know:



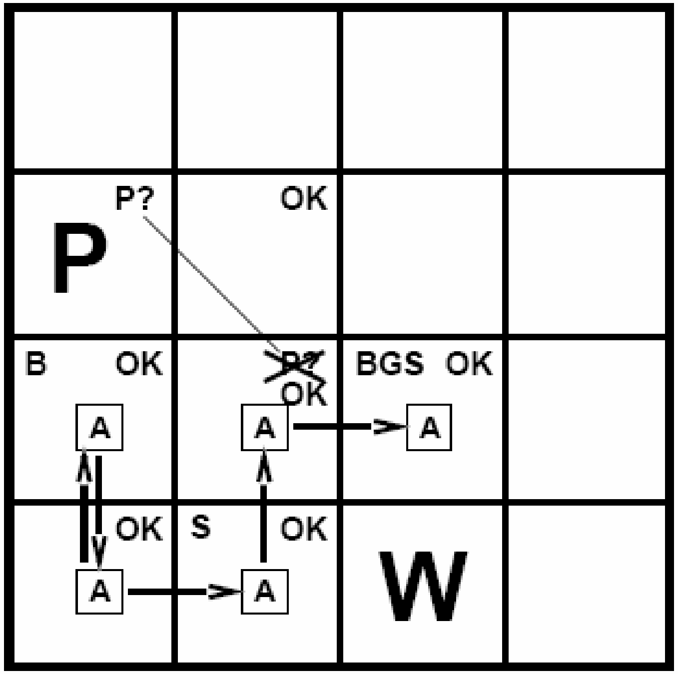
Exploring a wumpus world…

- Now another (unknown) world… How would you explore?
- Let us try!

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

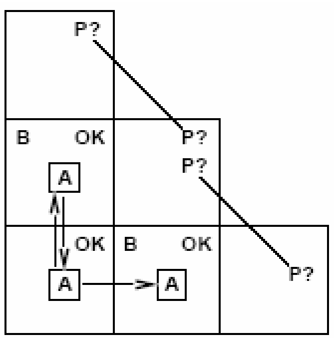
Exploring a wumpus world…

- Our final knowledge about this world…

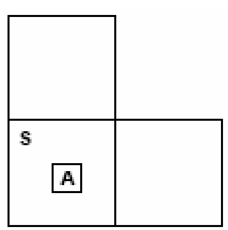


Exploring a wumpus world…

- Some tight spots



- Breeze in 1,2 and 2,1 !! → No safe actions
- However: Assuming pits uniformly distributed →
- 2,2 has pit with prob. 0.86 vs 0.31



- Stench in 1,1 → Cannot move!!
- Can use a strategy of *coercion*:
- Shoot straight ahead
- Wumpus was there → dead → safe
- Wumpus was not there → safe

So what is Logic anyways?

- Logics (!) are formal languages for representing information,
- such that conclusions can be drawn.
- Syntax determines the *structure* of *sentences* in the language,
- Semantics determines the *meaning* of these sentences.
- i.e., determines the truth of a sentence in a model (also called “possible world”)

So what is Logic anyways?

- Example: 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 $x=0$; $y=6$

So what is Logic anyways?

- *Declarative* versus *Procedural* system building…
- Logics are declarative languages.

- *Propositional Logic*: A very simple logic
- Also called: *Binary Logic*

Propositional Logic Sentences

- a Λ b
- Sunny V Cloudy
- ~(AmTired Λ AmEnergic)
- LectureBoring => InstructorFired
- ~(LectureBoring => InstructorFired)

Syntax involves *Propositional Symbols*, TRUE
FALSE, the *unary* "~" (Not) *operator* an
Λ, V, =>, <=> *operators*.

A more formal specification is usually
needed:
See "Backus-Naur" Form Grammar

Propositional Logic Sentences

- ~ (or ¬) (Not) *negation,*
- Λ (And) called a *conjunction*,

  arguments called *conjunts*
- V (Or) called a *disjunction*,
- => (implies) called an *implication*, or a *rule,*

  premise or antecedent => conclusion or consequent
- <=> (if and only if) called a *biconditional*

Propositional Logic Semantics

- Semantics of the language establishes the truth of all sentences
- We unary and binary connectives, we have

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*

Possible Model

- A Possible Model (also called “Possible World”)
- Given *n* propositional symbols, there are 2^n possible models (involving every combination of TRUE and FALSE assignments to the variables)

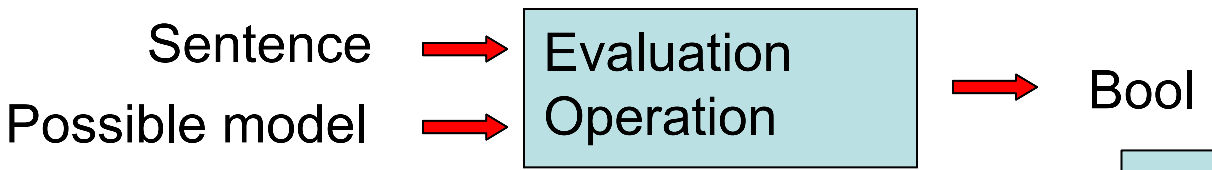
Possible Model

- A Possible Model (also called “Possible World”)
- Given *n* propositional symbols, there are 2^n possible models (involving every combination of TRUE and FALSE assignments to the variables)
- Example: Suppose three propositional symbols A, B, C. There are 8 possible models.

A	B	C
FALSE	FALSE	FALSE
FALSE	FALSE	**TRUE**
FALSE	**TRUE**	FALSE
FALSE	**TRUE**	**TRUE**
TRUE	FALSE	FALSE
TRUE	FALSE	**TRUE**
TRUE	**TRUE**	FALSE
TRUE	**TRUE**	**TRUE**

Eval(Sentence, Possible Model)

- We evaluate a *sentence* for a *possible model.*



We can express TRUE / FALSE by 1 / 0

$$\text{Eval}(\text{“}a \wedge \sim(b \vee c)\text{”}, \begin{array}{|c|c|c|}\hline a & b & c \\ \hline 1 & 0 & 1 \\ \hline \end{array}) \rightarrow 0$$

Knowledge Base

- What is it?
- It is a collection of our *knowledge about the world*,
- In logic: KB is a collection of zero or more *logic sentences*

KB_1

$a \vee b$
$\sim c \vee a$

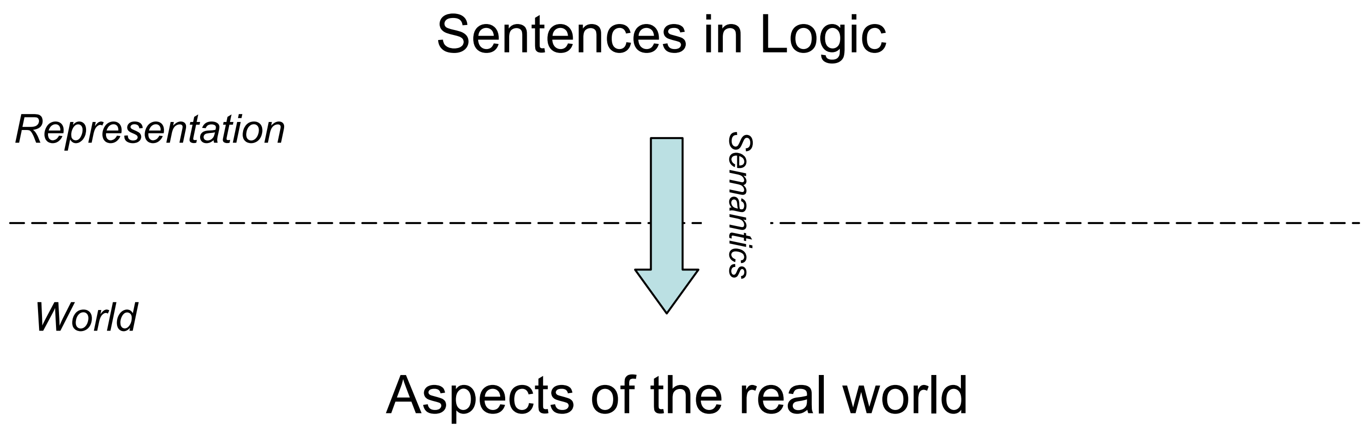
KB_2

$a \vee b$
$\sim a \wedge \sim b$
$c \wedge \sim c$

KB_3

Knowledge Base

- Why do we care?
- Because we can *represent* the real world with this stuff !! (Umm, to some extend anyway…)



Logical Entailment

- *Entailment* means one thing follows from another:

$$KB \models S$$

- And pronounced

  "knowledge base KB logically entails S"

- And means

  "all possible models that make KB evaluate to TRUE also makes S evaluate to TRUE"

Logical Entailment

- $KB \models S$ pronounced
  "knowledge base KB logically entails S"
- And means
  "all possible models that make KB evaluate to TRUE also makes S evaluate to TRUE"

KB$_1$

a V b
~c V a

a	*b*	*c*	*KB evals to TRUE?*	*a Λ c*
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Question:
Does KB $\models$ a Λ c ?

Logical Entailment

- $KB \models S$ pronounced
  "knowledge base KB logically entails S"
- And means
  "all possible models that make KB evaluate to TRUE also makes S evaluate to TRUE"

KB$_1$

a V b
~c V a

a	*b*	*c*	*KB evals to TRUE?*	*a Λ c*
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Question:
Does KB ⊨ a Λ c ?

Answer:
No

Because there are model in which KB is True but a Λ c is False!

Logical Entailment

- $KB \models S$ pronounced
  "knowledge base KB logically entails S"
- And means
  "all possible models that make KB evaluate to TRUE also makes S evaluate to TRUE"

KB$_1$

a V b
~c V a

a	***b***	***c***	***KB evals to TRUE?***	***a V c***
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Question:
Does KB $\models$ a V c ?

Logical Entailment

- $KB \models S$ pronounced
  "knowledge base KB logically entails S"
- And means
  "all possible models that make KB evaluate to TRUE also makes S evaluate to TRUE"

KB$_1$

a V b
~c V a

a	b	c	KB evals to TRUE?	a V c
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Question:
Does KB $\models$ a V c ?

Answer:
No

Because there is a model in which KB is True but a V c is False!

Logical Entailment

- $KB \models S$ pronounced
  "knowledge base KB logically entails S"
- And means
  "all possible models that make KB evaluate to TRUE also makes S evaluate to TRUE"

KB$_1$

a V b
~c V a

a	b	c	KB evals to TRUE?	a V (b Λ ~c) V (~b V c)
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Question:

Does
KB ╞ a V (b Λ ~c) V (~b V c) ?

Logical Entailment

- *KB* $\models$ *S* pronounced

  "knowledge base KB logically entails S"

- And means

  "all possible models that make

  TRUE also makes S evalu

Que

D

KB $\models$ a V (b Λ ~c) V (~b V c) ?

Answer:

Yes

We don't care about models that make KB false. The can make S true or false

KB$_1$

a V b

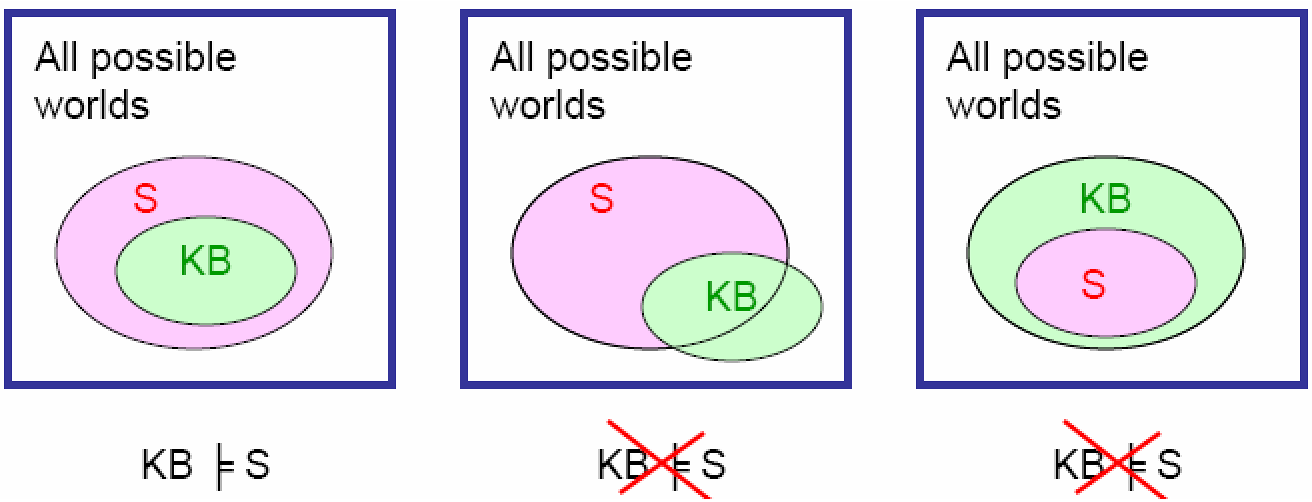
~c V a

a	*b*	*c*	*KB evals to TRUE?*	*a V (b Λ ~c) V (~b V c)*
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Because every model that makes KB True also makes the above statement True.

Logical Entailment

- $KB \models S$ pronounced
  "knowledge base KB logically entails S"
- And means
  "all possible models that make KB evaluate to TRUE also makes S evaluate to TRUE"



Inference

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

Consequences of KB are a haystack; α is a needle.
Entailment = needle in haystack; inference = finding it

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 $KB \vdash_i \alpha$

Inference by Enumeration

- Enumerate all possible models
- For each model for which KB is true, check that α is true too.
- Depth-First enumeration *Sound* and *Complete.*
- What is the problem?

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

- Will grow exponentially with the number of symbols!!
- $O(2^n)$ for n symbols. Problem is *co-NP-complete.*

Or… Apply Inference Rules

- Apply inference rules to derive new statements from given ones
- Until… You hit α.

Also: *Modus-ponens*

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

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

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

Validity is connected to inference via the Deduction Theorem:

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

A sentence is satisfiable if it is true in **some** model

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

A sentence is unsatisfiable if it is true in **no** models

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

Satisfiability is connected to inference via the following:

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

i.e., prove α by *reductio ad absurdum*

Good old *"proof by contradiction"*

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 alg.
  - Typically require translation of sentences into a normal form

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

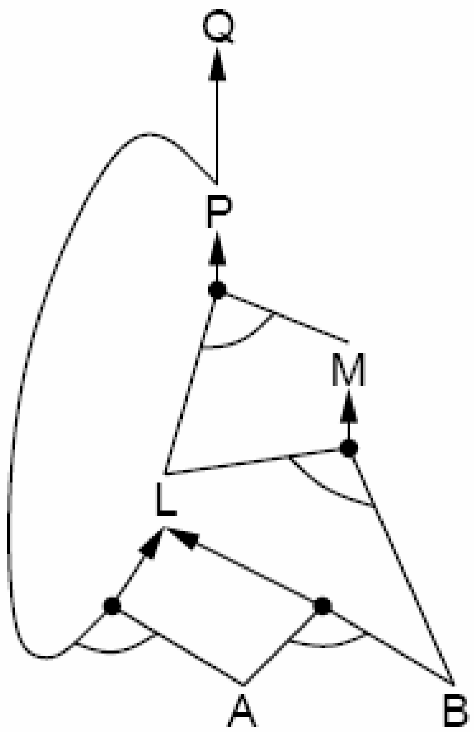
$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

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

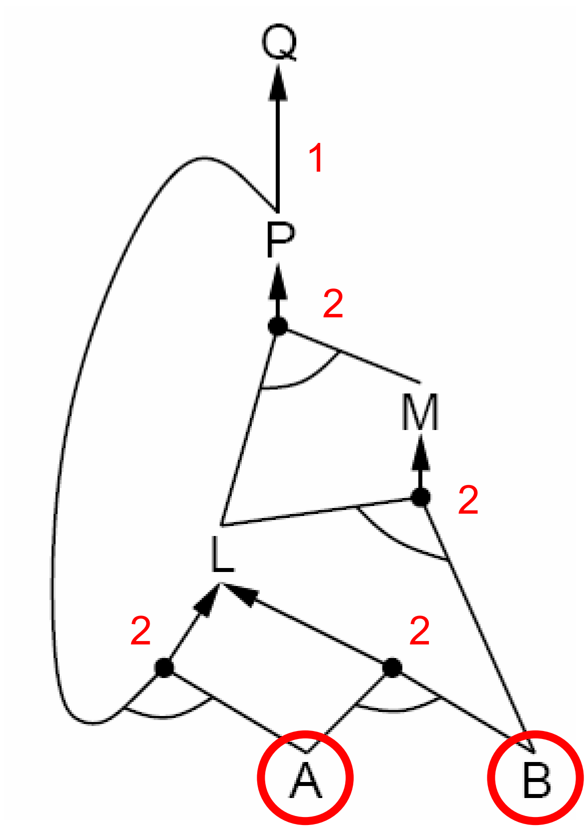
Example: Forward Chaining

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

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

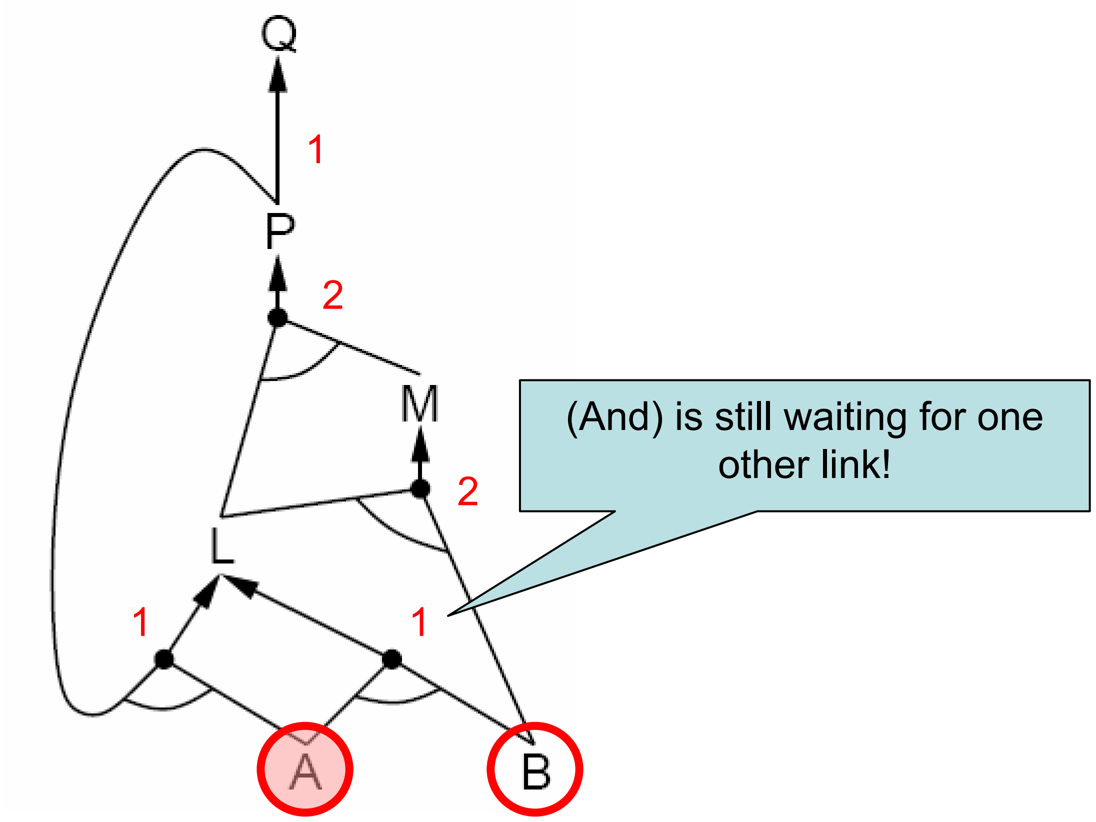


Example: Forward Chaining

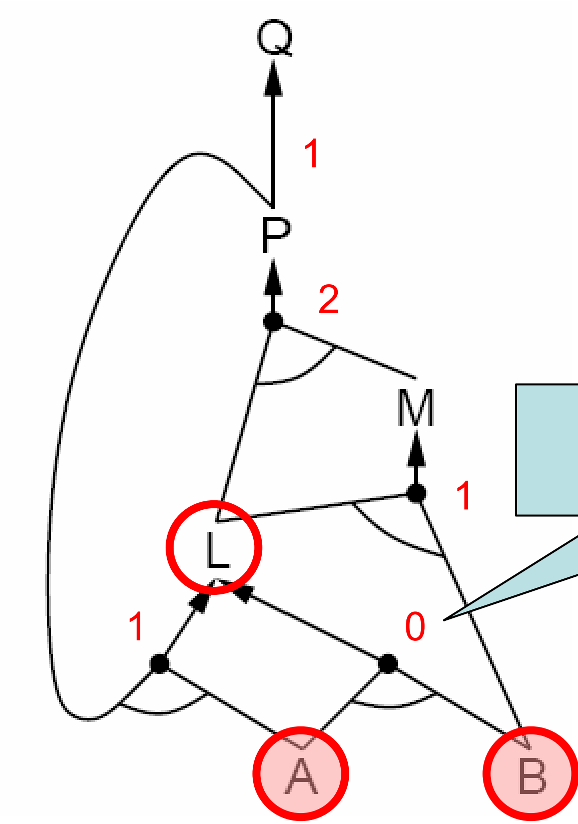


We start with facts in the KB.

Example: Forward Chaining

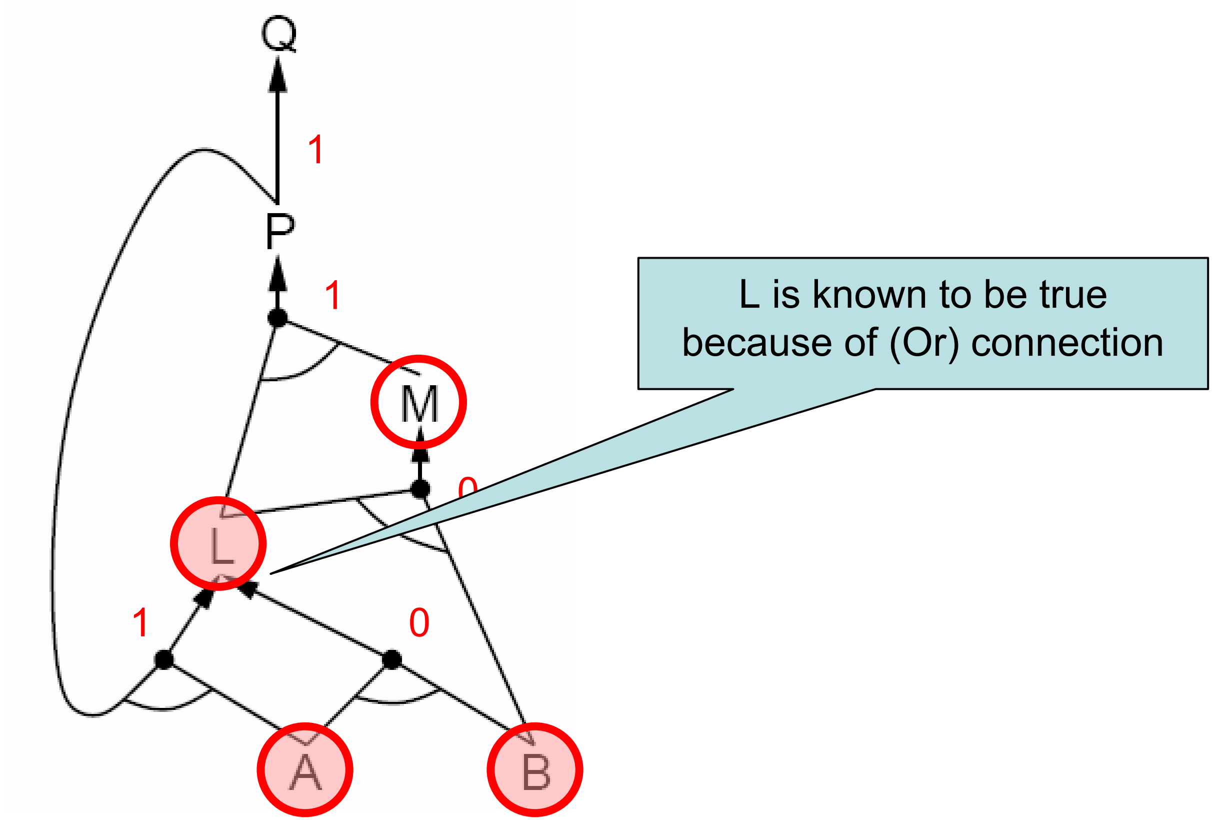


Example: Forward Chaining

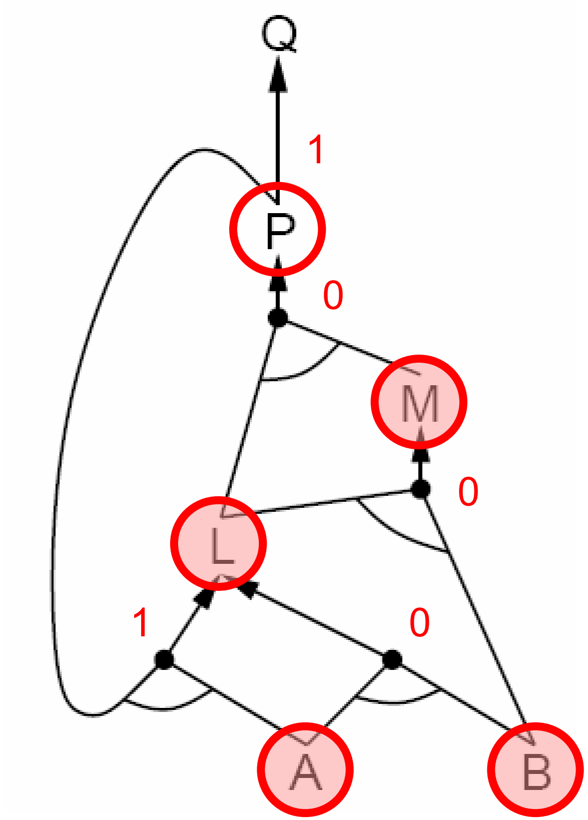


Both conjuncts satisfied.

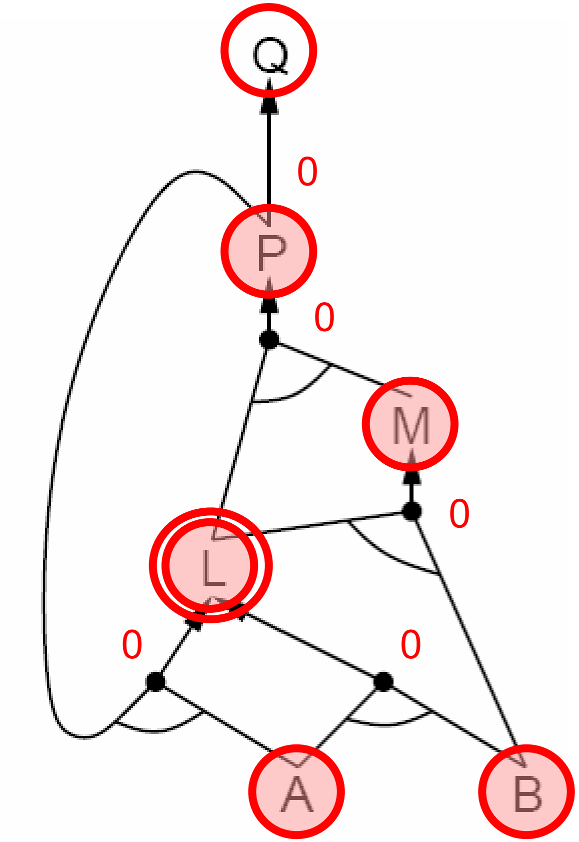
Example: Forward Chaining



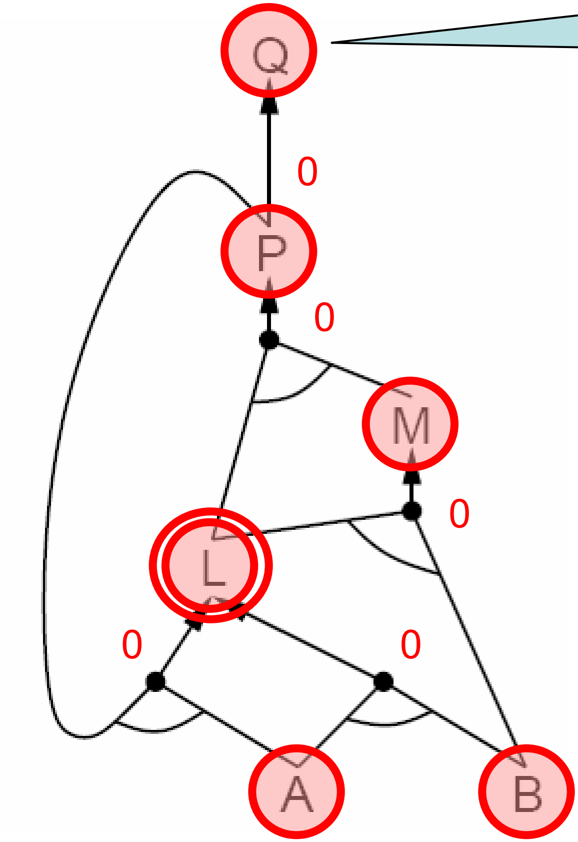
Example: Forward Chaining



Example: Forward Chaining



Example: Forward Chaining

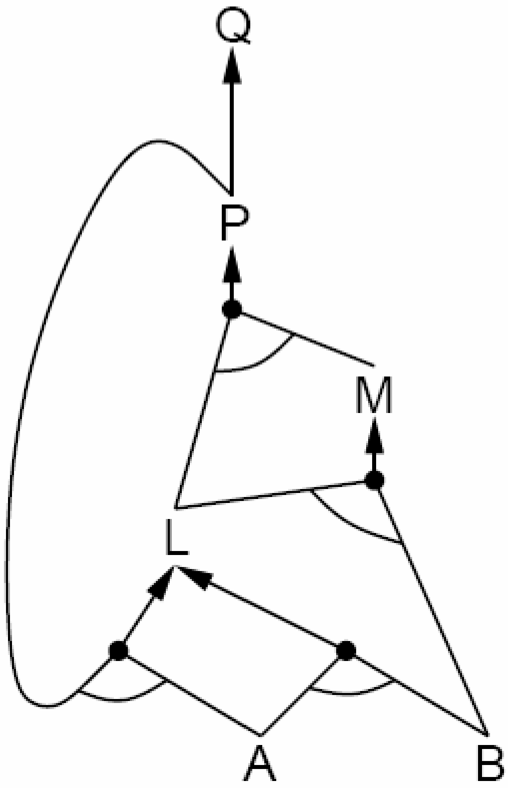


We have proved that Q is True!

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



Forward vs Backward Chaining

- FC is data-driven, cf. 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