

TDT4136 Introduction to Artificial Intelligence

Lecture 5: Logical Agents, Propositional Logic

Chapter 7 in the textbook

Keith Downing

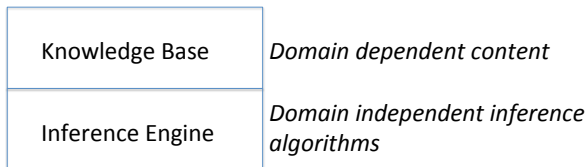
Norwegian University of Science and Technology

2024

Outline

Knowledge-based agents

- **Knowledge base (KB):** A set of sentences that describe facts about the world in some formal (representational) language
- **Inference engine:** A set of procedures that use the representational language to infer new facts from known ones as well as to answer a variety of KB queries.



Operation on the Knowledge Base

Two important operations on the KB:

- add new knowledge to KB
- ask questions about the knowledge in the KB. Questions are "asked" /triggered in two ways
 - a direct question from the user that doesn't require reasoning but just **retrieval** from the KB
 - a question representing a lack of knowledge required to solve a problem. This knowledge is implicit in the KB and need to be **inferred**.

- What happens in your mind when you are asked the following question?
 - Question: Is Stockholm in Sweden?
- What happens in your mind when you are asked the following question?
 - Question: Does a vegan person eat snail?
- Different?

A simple knowledge-based agent

- Knowledge base = set of **sentences** in a **formal** language
- Declarative approach to building an agent:
 - **Tell** it what it needs to know
 - Then it can **Ask** itself what to do—answers should "follow from" the KB

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*))

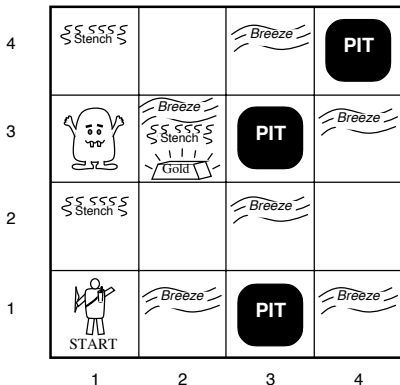
action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))

TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))

t ← *t* + 1

return *action*

An "informal" look into Wumpus World



Environment

Squares adjacent to wumpus are smelly

Squares adjacent to pit are breezy

One room contains a gold

Glitter iff gold is in the same square

Shooting kills wumpus (screams) if agent is facing it

When an agent walks into a wall, it perceives a bump.

The goal of the agent is to grab the gold and bring it to square [1,1]

Wumpus World PEAS description

Performance measure

- +100 points for walk out w/gold
- 100 points for dying
- 1 point for each action
- 10 points for using arrow

Actions and percepts

Actions: Left turn by 90° , Right turn by 90° , Forward, Grab, Climb, Shoot

Percept: [Stench?, Breeze?, Glitter?, Bump?, Scream?]

First movement

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
















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

$[\neg \text{Stench}, \neg \text{Breeze}, \neg \text{Glitter}, \neg \text{Bump}, \neg \text{Scream}]$
 $[\neg \text{Stench}, \text{Breez}, \neg \text{Glitter}, \neg \text{Bump}, \neg \text{Scream}]$

- Knowledge Base: Rules of game/environment
- Location (always starts at): [1,1]
- Percept: $[\neg \text{Stench}, \neg \text{Breeze}, \neg \text{Glitter}, \neg \text{Bump}, \neg \text{Scream}]$
- Action: Move forward to cell [2,1]. Outcome: Location [2,1]
- New Percept: $[\neg \text{Stench}, \text{Breeze}, \neg \text{Glitter}, \neg \text{Bump}, \neg \text{Scream}]$
- Infer: There must a pit in [2,2] or [3,1]
- Knowledge Base: Rules of game/environment + inference, i.e., There must be pit in [2,2] or [3,1]
- Action: Return to [1,1] to try next safe cell

Next move

1,4	2,4	3,4	4,4
1,3 W!	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

4				
3		  		
2				
1				
	1	2	3	4

- Knowledge Base: Rules of game/environment + inference, i.e., There must be pit in [2,2] or [3,1]
- Action: Move to cell [1,2]. Outcome: Location [1,2]
- Percept: [Stench, \neg Breeze, \neg Glitter, \neg Bump, \neg Scream]
- Infer: No pit in [2,2]
- Knowledge Base: Rules of game/environment + inference, i.e., There is pit in [3,1]

Wumpus World PEAS description

Environment characteristics

Deterministic?: : Yes. Outcomes are exactly specified

Static?: Yes. Wumpus and pits do not move

Discrete?: Yes

Single agent?: Yes - assuming wumpus as a natural phenomena

Fully observable?: No, only local perception

Episodic?: No, previous actions affect the current and next actions.
Remembering what was observed (e.g., pit) is very useful

How to tell a machine to play the Wumpus game?

How to represent the knowledge into the agent ?

How does the agent do the inference based on built knowledge ?

We need a knowledge representation language.

The objective of knowledge representation is to express the knowledge about the world in a computer-tractable form




Various Knowledge Representation Languages

Fundamental concepts in logic

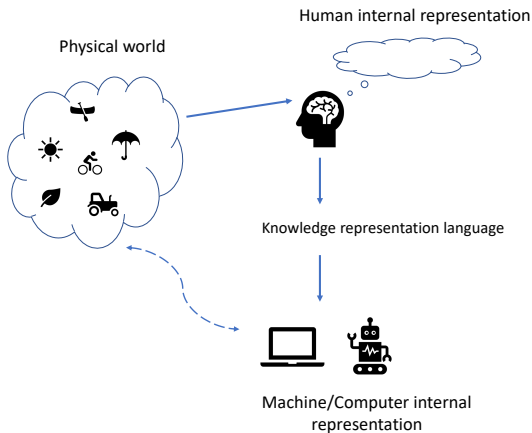
- Syntax and semantics
- Possible worlds and Model
 - an assignment of a symbol is either true or false in boolean systems
 - a *possible world* is an assignment of all the Symbols in the "system"
 - a possible world for each assignment, hence a set of possible worlds representing all possible assignment combinations
 - a *model* (of sentence S): a symbol assignment(i.e, a possible world) that makes S true
 - a *model* (of KB) is a possible world where the KB(i.e., each sentence in it) is true
- *Entailment*. $\langle \text{Sentence1} \rangle \models \langle \text{Sentence2} \rangle$
 - A entails B
 - B follows from A
 - B is true whenever A is true

Computers and logic

- Computers can use logic in order to, e.g., prove mathematical theorems and to diagnose failures
- But, first what is logic and how it works.

Human - layman	Human- computer scientist	Computer
Natural language	Knowledge Representation Language	Programming language (e.g., Prolog)
		
Sissel is a student. Sophie is a teacher. Sissel plays piano Sophie plays piano if she is happy	student(Sissel) teacher (Sophie) playsPiano(Sissel) happy(Sophie) → playsPiano(Sophie)	student(Sissel). teacher (Sophie). playsPiano(Sissel). playsPiano(Sophie):- happy(Sophie).
Does Sissel play piano? Does Sophie play piano?	playPiano(Sissel) playPiano(Sophie) (answer these, i.e., find if these are True or False)	?- playPiano(Sissel). ?- playPiano(Sophie).

Physical World and Internal Representations



What is Logic

- One of the oldest disciplines in history
- Dates back to Aristoteles



"EVERYONE'S USING YOUR THEOREM, PYTHAGORAS.
I TOLD YOU YOU SHOULD HAVE PATENTED IT."

If there had been Facebook[®]
in Ancient Greece, we would
never have had philosophy.

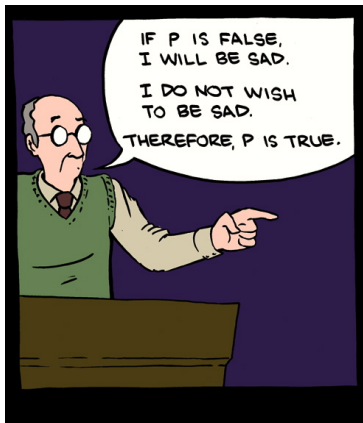


People use logic all the time

- People use logic in order to talk about observations, to define concepts, and to formalize theories

People use logic all the time -cont.

- People use logic to prove something

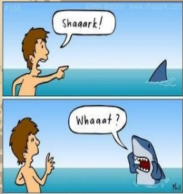


There. Now you can skip 99% of philosophical debates.

People use logic all the time -cont.

- Using logical reasoning we can derive new information from what we already new.

EXAMPLE

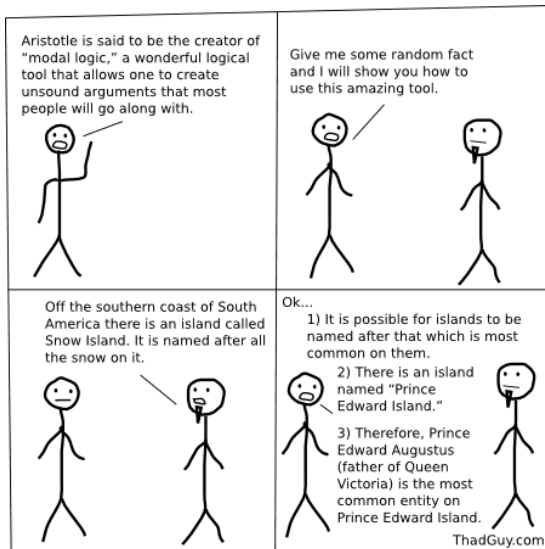


All fish swim.
(Major Premise)

Every shark is a fish.
(Minor Premise)

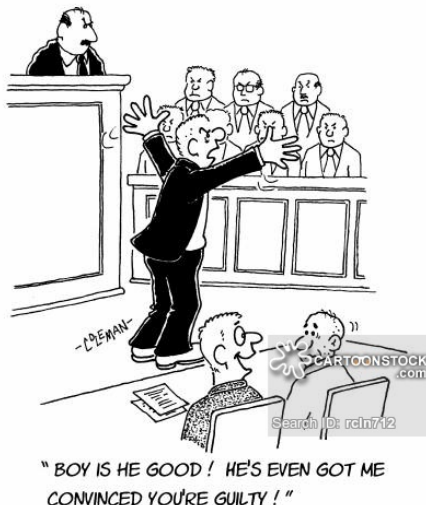
Therefore every shark swim.
(Conclusion)

People use logic all the time -cont.



People use logic all the time -cont.

- We use logical proofs to convince others of our conclusions..



People use logic all the time -cont.



Propositional Logic

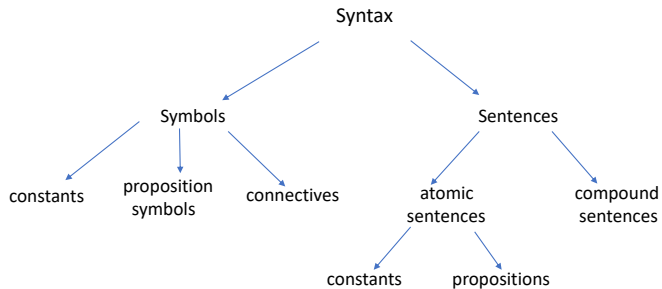
In this lecture we deal with **Propositional Logic**.

Syntax of Propositional Logic

Syntax - symbols, sentences

- Symbols (alphabet) consists of:
 - Constants: True, False
 - Proposition symbols : P, Q, . . .
 - Connectives: \neg , \wedge , \vee , \rightarrow , \Leftrightarrow
- Sentences can be atomic (constants and propositions) or compound sentences

Syntax



- **Proposition:** a declarative statement about the world that is either true or false
- Which of the followings are propositions:
 - Norway is in Europe.
 - Stockholm is capital city of Norway.
 - What is your name?
 - Do your homework.
 - This sentence is false.

Which of the followings are propositions?

- Norway is in Europe (true).
- What is your name? (not declarative)
- Do your homework (not declarative)
- This sentence is false (neither true nor false)

Compound Sentences

- Compound sentences: constructed from atomic and/or other compound sentences via 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, $S_1 \implies S_2$ is a sentence (implication)
 - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Example - Compound Sentences

Assume the following propositions:

P : It is sunny this afternoon

Q : it is colder than yesterday

Lg : the traffic light is green

Cg: the cars will go

How are the following compound sentences be represented in terms of the proposition symbols above?

- ① It is not sunny this afternoon and it is colder than yesterday.
- ② If the traffic light is green then the cars will go.
- ③ The cars will go only if traffic light is green

Example - Compound Sentences, cont.

How are the following compound sentences be represented in terms of the proposition symbols above, p , q , lg , cg ?

- 1 It is not sunny this afternoon and it is colder than yesterday.
- 2 If the traffic light is green then the cars will go.
- 3 The cars will go only if traffic light is green

Representation of these sentences in propositional logic:

- 1 $\neg P \wedge Q$
- 2 $Lg \implies Cg$
- 3 $Cg \implies Lg$

Semantics of **atomic** sentences are determined according to their truth values wrt interpretations.

An **interpretation** maps symbols to one of the two values: True (T), or False (F), depending on whether the symbol is **satisfied** in the "world".

- P: Light in the room is on (*True in Interpretation I*) then $\text{Value}(P, I) = \text{True}$,
- Q: It rains outside (False) then $\text{Value}(Q, I) = \text{False}$
- If P: Light in the room is on (*False in I'*) then $\text{Value}(P, I') = \text{False}$

Semantics of Connectives

Semantics of **compositional** sentences are determined using the standard rules of logic 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

Example - Compound Sentences with implication

Assume the following propositions:

L_g : the traffic light is green

C_g : the cars will go

Draw the truth table for implication and check if the logic representation of the following sentences are correct.

- 1 If the traffic light is green then the cars will go: $L_g \implies C_g$
- 2 The cars will go only if traffic light is green: $C_g \implies L_g$

- Intuitively, when we read in the newspaper that “RBK’ and “Brann” won’, we can immediately say ”RBK won”
- **Note that \models is NOT a part of the logical knowledge representation(KR) language, it is not a connective in any logic KR language)**
- \models belongs to a/the metalanguage that is used a level above the knowledge representation
- Entailment means that the truth of one sentence (α) follows from the truth of another (e.g., set of all sentences in KB).

Semantics of Inferring new information

Inference may be needed in order to answer a question, based on what the agent knows (i.e, the sentences in the KB).

Logically, this means whether KB "**entails**" the sentence (e.g. α) of which truth is asked in the question, i.e., $KB \models \alpha$?

In other saying, the logical agent can use the sentences in the Knowledge Base to draw conclusions that are *logically entailed* by those sentences.

How to design the reasoning procedure(s) that answers $KB \models \alpha$?

Entailment may be obtained in various ways:

- Model checking/enumeration
- Resolution and Proof by Contradiction
- Forward and Backward chaining

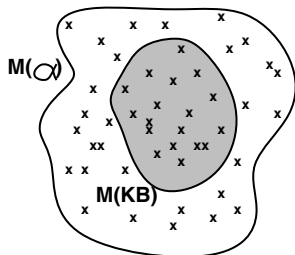
Model checking

Entailment through checking models:

Necessary for entailment of α : α is true in every model that KB is true.

$KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$.

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




Example: Wumpus gridworld


Propositional representation of Wumpus gridworld.

- Rules of Wumpus game and representation of a game.
- KB = wumpus-world rules + observations
- 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 $[i,j]$.
- “Pits cause breezes in adjacent squares” = “A square is breezy if and only if there is an adjacent pit”

Wumpus world after the first 2 moves

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

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

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

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

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

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

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

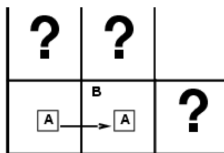
$$R_4 : \neg B_{1,1}$$

$$R_5 : B_{2,1}$$

- $KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

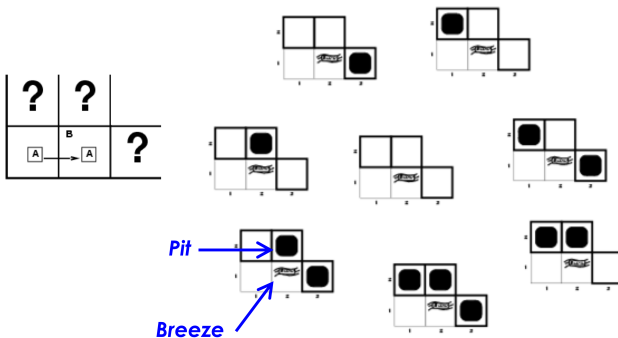
Models in reduced Wumpus gridworld

Consider possible models for KB assuming only pits and a **reduced** Wumpus world with only 5 squares and pits:



Wumpus Models

The reduced Wumpus World. All 8 possible worlds/models are:



Decision in Wumpus world by Model Checking

Still reduced Wumpus gridworld example.

- Goal: Decide whether KB says "no pit in [1,2]"
- Let $\alpha = \neg P_{1,2}$
- Does $\text{KB} \models \neg P_{1,2}$?

Models within the red line are consistent with our KB:

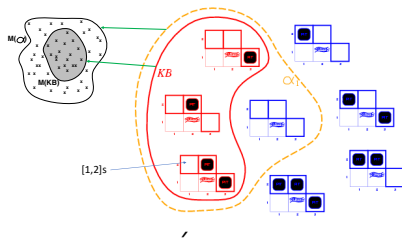
$$R_1 : \neg P_{1,1}$$

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

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

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

$$R_5 : B_{2,1}$$



By model checking: $\text{KB} \models \alpha$
Hence [1,2] is safe!

Model Checking through Truth Tables

Truth Table is a simple method for model enumeration and checking.

KB: $A \wedge B \rightarrow C$, $A \wedge B$

α : C

Does $\text{KB} \models \alpha$, i.e.,

$[(A \wedge B \rightarrow C) \wedge (A \wedge B)] \models C$??

World	A	B	C	$A \wedge B$	$A \wedge B \rightarrow C$
0	0	0	0	0	1
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	0	1
4	1	0	0	0	1
5	1	0	1	0	1
6	1	1	0	1	0
7	1	1	1	1	1

- $M[(A \wedge B \rightarrow C) \wedge (A \wedge B)] = \{7\} \subseteq \{1, 3, 5, 7\} = M(C)$.
- Yes

Example Model enumeration in Wumpus world

- After visiting $(1,1)$ and $(2,1)$

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

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

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

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

$$R_5 : B_{2,1}$$

- $KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

- In this Wumpus world: 7 symbols (in KB). We can get $2^7 = 128$ models

Example Model Enumeration in Wumpus world- cont.

if KB is true in a row, check whether α is too.

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
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
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
true	true	true	true	true	true	true	false	true	true	false	true	false

Answer: $KB \models \neg P_{1,2}$. No pit in $P_{1,2}$.

Truth-table enumeration algorithm

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($KB, \alpha, symbols, \{ \}$)

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*

if EMPTY?($symbols$) **then**

if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)

else return *true* // when KB is false, always return true

else

$P \leftarrow$ FIRST($symbols$)

$rest \leftarrow$ REST($symbols$)

return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)

and

 TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))

Inference by Model Enumeration

- Enumeration is sound and complete
- The truth table is exponential in the number of propositional symbols (we checked all rows/assignments)
- Model checking complexity:
If KB and α and contain n symbols:
 - Time complexity: $O(2^n)$
 - Space complexity: $O(n)$
- **We need effective/smarter ways of doing inference**

- To build a proof of the desired/goal/question sentence without dealing with model enumeration/truth table
- A proof is a chain of application of **inference rules** - and logical equivalences.
- We'll see what these inference rules are - soon.

First, some concepts related to Entailment

- Logical Equivalence. Two sentences α and β are logically equivalent if $\alpha \models \beta$ and $\beta \models \alpha$
- Validity. A sentence is valid if it is true in all models.
 - How is validity relevant to Entailment?
 - Answer: This relates to the relationship between entailment and implication.
 - Decision of whether $\alpha \models \beta$: answer is "yes" iff the sentence $\alpha \implies \beta$ is valid - true in all models. This is called **Deduction Theorem**
- Satisfiability is about a specific relationship between a sentence and a(some) model.
 - a model *satisfies* a sentence if the sentence is true for this model
- How is satisfiability relevant to entailment?
- Decision of whether $\alpha \models \beta$: answer is "yes" iff the sentence $\alpha \wedge \neg\beta$ is unsatisfiable - not true in any model.
- This is the basis of an inference procedure called **proof by refutation (or contradiction)**

Some/standard Logical equivalences

Two sentences are **logically equivalent** iff true in same models:

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

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \implies \beta) \equiv (\neg\beta \implies \neg\alpha) \quad \text{contraposition}$$

$$(\alpha \implies \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$

$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \implies \beta) \wedge (\beta \implies \alpha)) \quad \text{biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{De Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$

$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

Inference Rules approach

- How to make the process more efficient?
- KB is true on only a smaller subset
- **Solution:** check only entries for which KB is True.
- That is, infer new logical sentences from the knowledge base and see if they match a query
- This is the idea behind the inference rules approach
- *Inference rules* represent sound inference patterns repeated in inferences

Properties of an Inference Procedure and connection to Entailment

- Assume an **inference procedure** i that
 - derives a sentence α from the KB, i.e., $KB \vdash_i \alpha$
- **Soundness**: An inference procedure is sound
 - If whenever $KB \vdash_i \alpha$, then it is also true that $KB \models \alpha$

Sound inference?



- **Completeness**: An inference procedure is complete
 - If whenever $KB \models \alpha$ then it is also true that $KB \vdash_i \alpha$
- Sound and complete inference procedures are desirable

Syllogism rules

- Modus ponens (method of affirming)

$$\begin{array}{l} A \rightarrow B \\ A \\ \hline B \end{array}$$

- Modus Tollens (method of denying)

$$\begin{array}{l} A \rightarrow B \\ \neg B \\ \hline \neg A \end{array}$$

- Hypothetical Syllogism

$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C \\ \hline A \rightarrow C \end{array}$$

- Disjunctive syllogism (Unit resolution)

$$\begin{array}{l} A \vee B \\ \neg A \\ \hline B \end{array}$$

Some other inference rules

- And-elimination

$$\frac{A_1 \wedge A_2 \wedge \dots \wedge A_n}{A_i}$$

- And-introduction

$$\frac{A_1, A_2, \dots, A_n}{A_1 \wedge A_2 \wedge \dots \wedge A_n}$$

- Or-introduction

$$\frac{A_i}{A_1 \vee A_2 \vee \dots \vee A_i \vee \dots \vee A_n}$$

- Implication Creation

$$\frac{A}{B \implies A}$$

- Implication Distribution

$$\frac{A \implies (B \implies C)}{(A \implies B) \implies (A \implies C)}$$

Inference rules approach

- Inference rule approach: Apply an inference rule that matches with the knowledge in KB. Do this until satisfying the query, e.g., P1.
- Starting with a KB.
 - ASK(P1): is P1 true given what is in KB?
- Derive P1 from the KB
 1. Use inference rules to add new statements
 2. Use **logical equivalence** to rewrite existing statements

Inference rules approach - Example.

KB: $P \implies Q, Q \implies R$

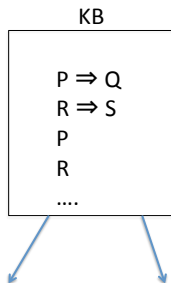
Question: Does $\text{KB} \models (P \implies R)$?

Using Inference rules:

- ① $P \implies Q$ (Premise)
- ② $Q \implies R$ (Premise)
- ③ $P \implies (Q \implies R)$ (Implication Creation: 2)
- ④ $(P \implies Q) \implies (P \implies R)$ (Implication Distribution: 3)
- ⑤ $P \implies R$ (Modus Ponens: 4, 1)

Problems with Inference rules approach

- There may be more than one rule that can apply at a certain stage.



- One solution: **Resolution** is a single inference rule that yields a complete inference algorithm

Unsatisfiability and proof by Inference rule Resolution

- A sentence is **satisfiable** if it is true in **some** models
- A sentence is **unsatisfiable** if it is true in **no** models
e.g., $A \wedge \neg A$
- **Unsatisfiability** is connected to inference via the following:
 $KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable
i.e., **proof** α by contradiction

Resolution rule and "Proof by Contradiction"

- Instead of showing $KB \models \alpha$, we show that $KB \wedge \neg\alpha$ is not satisfiable.
- Disproving $KB \wedge \neg\alpha$
proves the entailment $KB \models \alpha$
- The inference rule called *Resolution Rule* is used for this purpose

Resolution Rule

- Example:

$$\frac{(A \vee B), (\neg A)}{B}$$

- Resolution inference rule

$$\frac{l_1 \vee \dots \vee l_{i-1} \vee \mathbf{l_i} \vee l_{i+1} \vee \dots \vee l_k, m_1 \vee \dots m_{j-1} \vee \mathbf{m_j} \vee m_{j+1} \vee \dots \vee m_n}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

where l_i and m_j are complementary literals (e.g., $l_i = \neg m_j$)

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 $KB \wedge \neg\alpha$
 $new \leftarrow \{ \}$

while *true* **do**

for each pair of clauses 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*

$clauses \leftarrow clauses \cup new$

Conjunctive Normal Form

However to apply resolution technique its requires to represent KB as well as any sentence α that we wish to derive in a special format known as **Conjunctive Normal Form** (CNF): conjunction of clauses.

- A **clause** is an expression of the form $l_1 \vee l_2 \vee \dots \vee l_k$ where each l_i is a literal - a disjunction of literals.
- Example CNF: $(A \vee B) \wedge (\neg A \vee \neg C \vee D)$ - conjunction of disjunctions
- A **literal** is either a propositional symbol or the negation of a symbol.
- Every propositional sentence is equivalent to a conjunction of clauses.

Conversion to CNF

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

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

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

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

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

3. Move \neg inwards using de Morgan's rule:

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

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

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

Resolution procedure for Wumpus problem

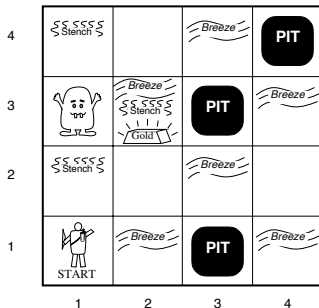
Our knowledge base: $R2 \wedge R4$

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

Now, we want to verify that there is no pit in $[1,2]$. $\alpha = \neg P_{1,2}$

$KB \models \alpha$?

For this, we need to show that $KB \wedge \neg \alpha$ is unsatisfiable.

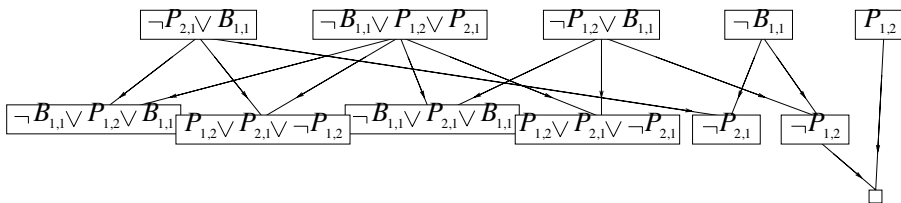


RR process on Wumpus example

We have converted the KB into CNF:

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

$\alpha = \neg P_{1,2}$, we take also its negation.



Problem with Resolution Refutation

- Resolution is complete but can be exponential in space and time.
- If we can reduce all clauses to a special forms called **Horn Clauses**, deciding entailment becomes linear in the size of the knowledge base (KB)
- Inference with Horn clauses can be done through the **forward chaining** and **backward chaining** algorithms

Horn Clauses

- Definite clause: Disjunction of literals of which exactly one is positive; the rest are negative
- Horn Clause: Disjunction of literals of which at most one is positive

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

$$\equiv [\neg A \vee \neg B \vee \neg C] \vee D \text{ (Associativity)}$$

$$\equiv \neg[A \wedge B \wedge C] \vee D \text{ (De Morgan's Law)}$$

$$\equiv [A \wedge B \wedge C] \rightarrow D \text{ (Implication Introduction)}$$

Premise \rightarrow Consequent

Modus Ponens in Forward and Backward Chaining

- Modus ponens is perfect for **Definite Clause** KBs.
- Forward and Backward algorithms rely on Modus Ponens:

$$\frac{\alpha_1, \dots, \alpha_n, \quad \alpha_1 \wedge \dots \wedge \alpha_n \implies \beta}{\beta}$$

Forward and backward chaining

- Forward chaining (data driven)

Idea: Whenever the premises of a rule are satisfied, infer the conclusion.

- Backward chaining (goal driven)

Idea: To prove the fact that appears in the conclusion of a rule prove the premises of the rule.

Both procedures are complete for KBs in the Definite clause form.

Forward chaining

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

$P \implies Q$ (Query)

$L \wedge M \implies P$

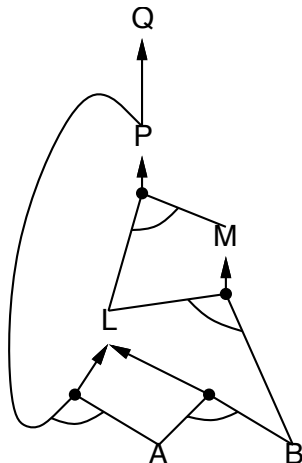
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



Forward chaining algorithm

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*

inputs: KB , the knowledge base, a set of propositional definite clauses

q , the query, a proposition symbol

$count \leftarrow$ a table, where $count[c]$ is initially the number of symbols in clause c 's premise

$inferred \leftarrow$ a table, where $inferred[s]$ is initially *false* for all symbols

$queue \leftarrow$ a queue of symbols, initially symbols known to be true in KB

while $queue$ is not empty **do**

$p \leftarrow \text{POP}(queue)$

if $p = q$ **then return** *true*

if $inferred[p] = \text{false}$ **then**

$inferred[p] \leftarrow \text{true}$

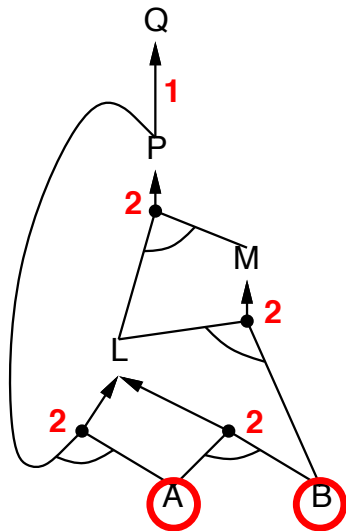
for each clause c in KB where p is in c .PREMISE **do**

 decrement $count[c]$

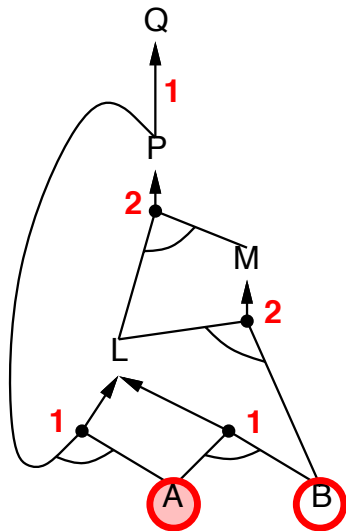
if $count[c] = 0$ **then** add c .CONCLUSION to $queue$

return *false*

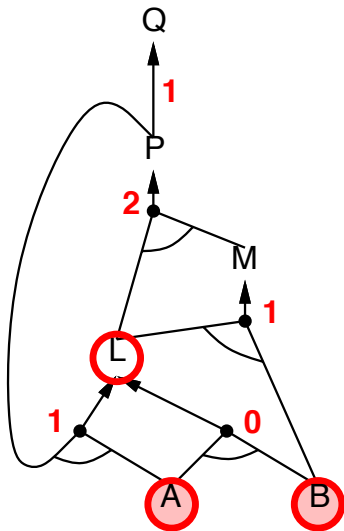
Forward chaining example



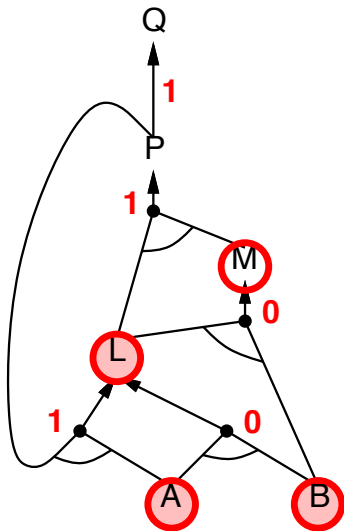
Forward chaining example



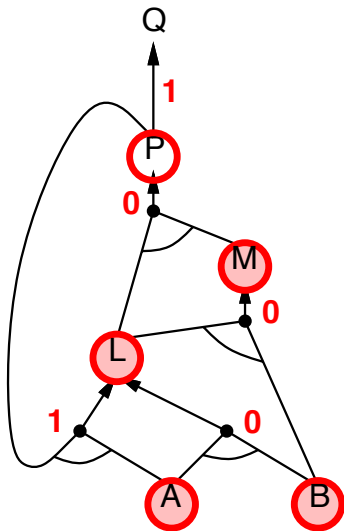
Forward chaining example



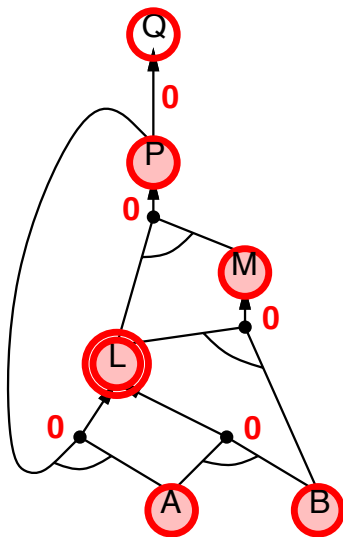
Forward chaining example



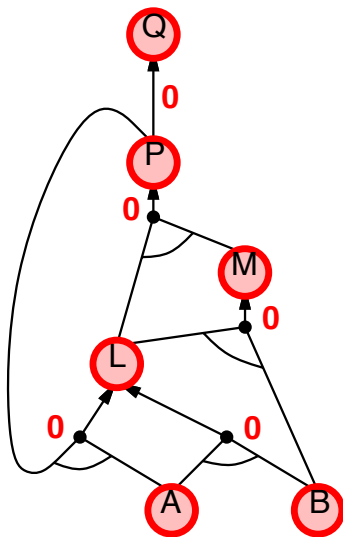
Forward chaining example



Forward chaining example



Forward chaining example



Backward chaining

Idea: work backwards from the query q :

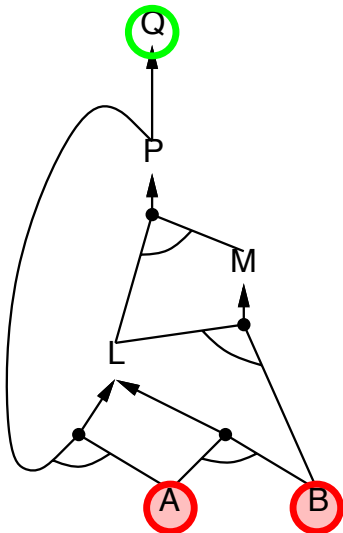
- to prove q by BC,
 - check if q is known already, or
 - prove by BC all premises of some rule concluding q

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



$P \Rightarrow Q$ (Query)

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

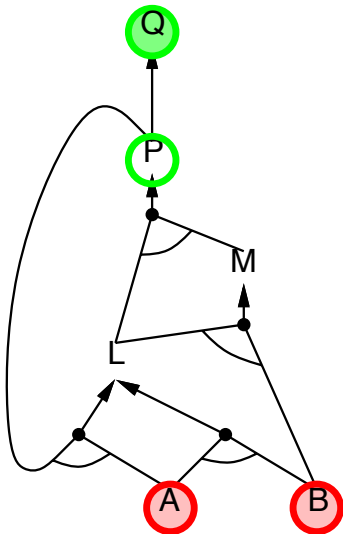
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

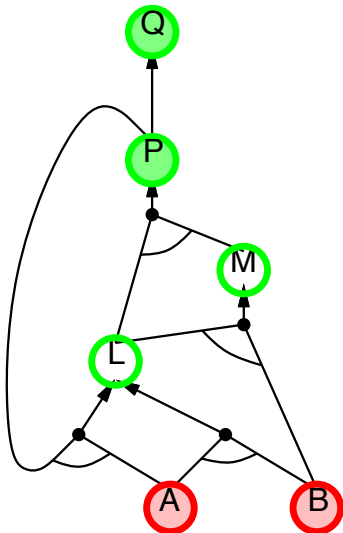
Backward chaining example


$$P \implies Q \text{ (Query)}$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \implies L$$
$$A \wedge B \Rightarrow L$$

A

$$B$$

Backward chaining example



$P \Rightarrow Q$ (Query)

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

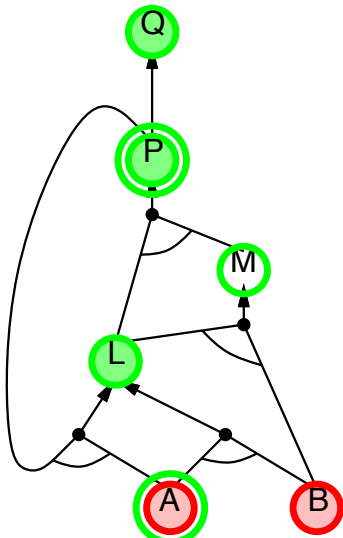
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example



$P \Rightarrow Q$ (Query)

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

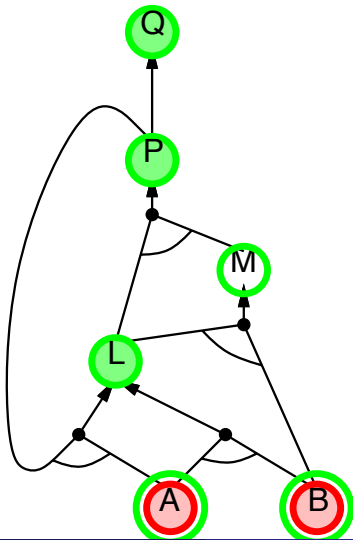
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example



$P \Rightarrow Q$ (Query)

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

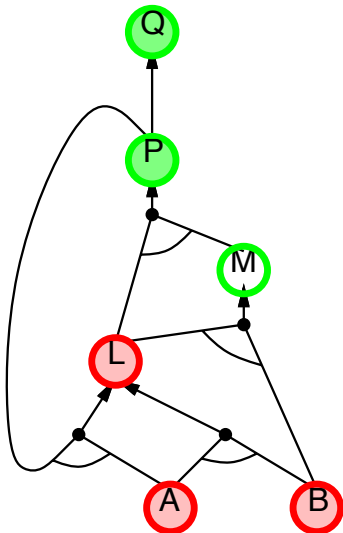
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example



$P \implies Q$ (Query)

$L \wedge M \implies P$

$B \wedge L \implies M$

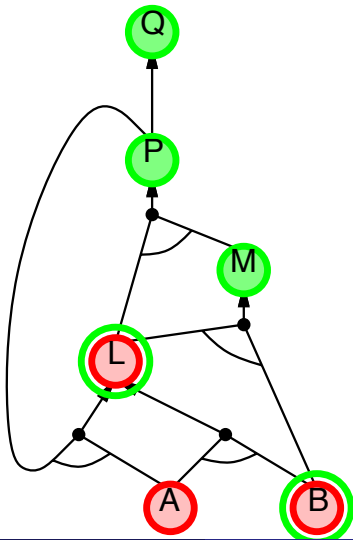
$A \wedge P \implies L$

$A \wedge B \implies L$

A

B

Backward chaining example



$P \Rightarrow Q$ (Query)

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

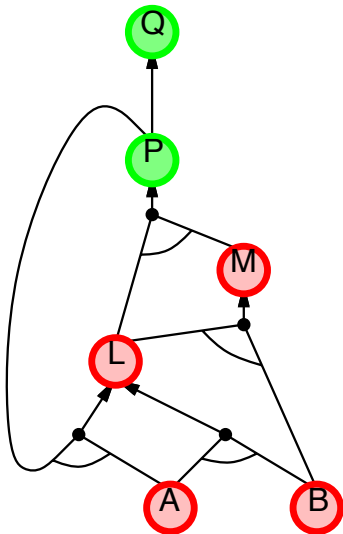
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example



$P \Rightarrow Q$ (Query)

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

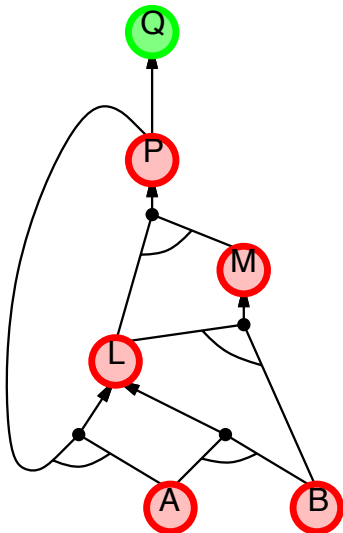
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Backward chaining example



$P \implies Q$ (Query)

$L \wedge M \implies P$

$B \wedge L \implies M$

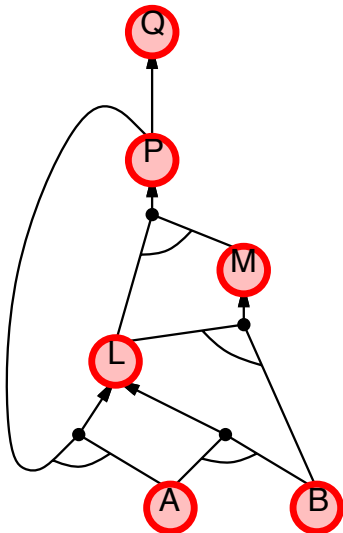
$A \wedge P \implies L$

$A \wedge B \implies L$

A

B

Backward chaining example



$P \implies Q$ (Query)

$L \wedge M \implies P$

$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B

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