TDT4136 Introduction to Artificial Intelligence

Summary of Lecture 7: Logical Agents, Propositional Logic

Chapter 7 in the textbook

Liyuan Xing

Adjunct Associate Professor @ NAIL/IDI/NTNU
Machine Learning Engineer @ TrønderEnergi

2022

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.

Knowledge Base	*Domain dependent content*
Inference Engine	*Domain independent inference algorithms*

Operation on the Knowledge Base

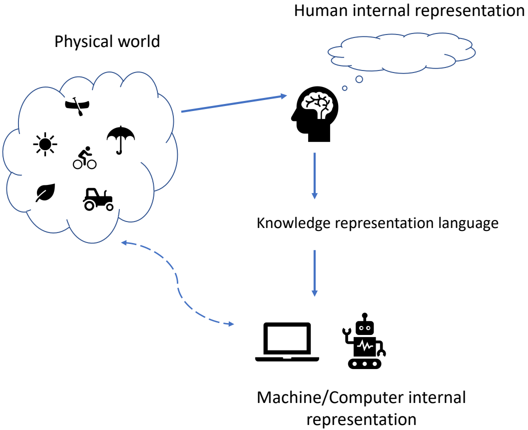
Two important operations on the KB:

- 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**.

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.* $\langle Sentence1 \rangle \models \langle Sentence2 \rangle$
  - *A entails B*
  - *B follows from A*
  - *B is true whenever A is true*

Physical World and Internal Representations

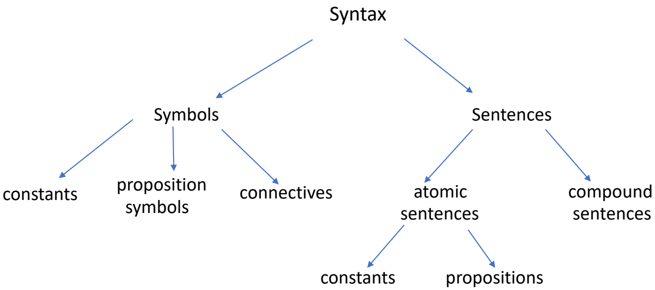


Syntax of Propositional Logic

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


Syntax
Symbols
Sentences
constants
proposition symbols
connectives
atomic sentences
compound sentences
constants
propositions

Atomic Sentences

- **Proposition:** a declarative statement about the world that is either true or 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)

Semantics

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

Entailment

- 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

Entailment may be obtained in various ways:

- Model checking/enumeration - uses truth tables
- Inference rules approach - uses chain of inference rules directly
- Resolution and Proof by Contradiction - uses resolution inference rule and CNF sentences
- Forward and Backward chaining - uses Modus Ponens rule and Horn clauses

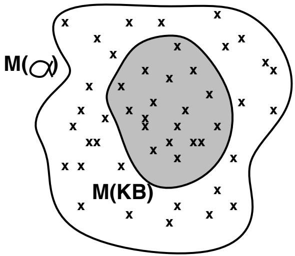
Model checking

Entailment through checking models:

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

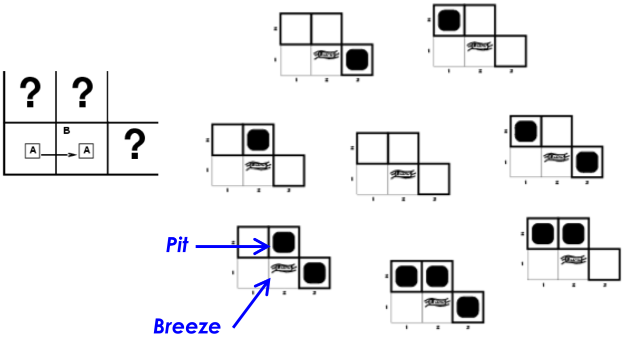
KB $\models \alpha$ iff M(KB) $\subseteq$ M(α).

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



Wumpus Models

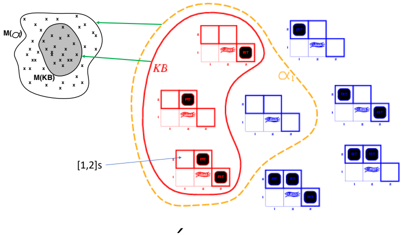
The reduced Wumpus World. All 8 possible worlds/models are:



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 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: 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: $A \wedge B \rightarrow C, \quad A \wedge B$
α: C

Does KB $\models \alpha$, i.e.,
$[(A \wedge B \rightarrow C) \wedge (A \wedge B)] \models C$??

World	A	B	C	A ∧ 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

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**

Theorem Proving

- 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 $\beta \models \alpha$

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

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

Inference rules

Syllogism rules

- Modus ponens (method of affirming)

$$\frac{\begin{array}{l} A \rightarrow B \\ \text{A} \end{array}}{\text{B}}$$

- Modus Tollens (method of denying)

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

- Hypothetical Syllogism

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

- Disjunctive syllogism (Unit resolution)

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

Some other inference rules

- And-elimination

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

- And-introduction

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

- Or-introduction

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

- Implication Creation

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

- Implication Distribution

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

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.

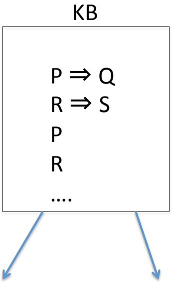
KB: $P \implies Q, Q \implies R$
Question: Does KB $\models (P \implies R)$?

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 ands "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

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

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 $KB \wedge \neg c$
  $new \leftarrow \{\,\}$
  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 $l_1 \vee\ l_2 \vee$?.. 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 $(\alpha \implies \beta) \wedge (\beta \implies \alpha)$.

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

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:

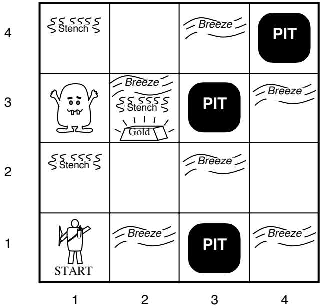
$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

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 $\models \alpha$?

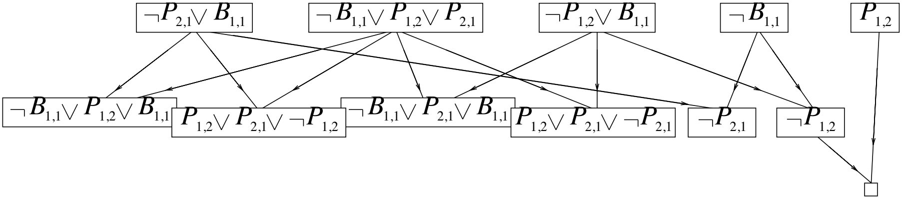
For this, we need to show that KB $\wedge \neg\alpha$ is unsatisfiable.



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

$\neg A \lor \neg B \lor \neg C \lor D$

$\equiv [\neg A \lor \neg B \lor \neg C] \lor D$ (Associativity)

$\equiv \neg[A \land B \land C] \lor D$ (De Morgan's Law)

$\equiv [A \land B \land C] \rightarrow D$ (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:

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

Forward and backward chaining

- Forward chaining (data driven)
  Idea: Whenever the premises of a rule are satisfied, infer the conclusion.

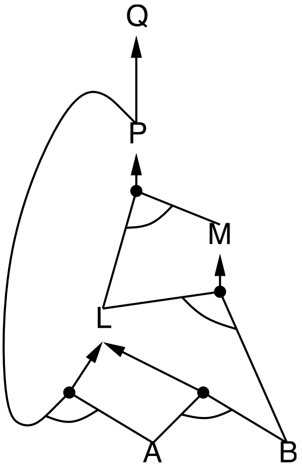
- Backward chaining (goal driven)
  Idea: To prove the fact that appears in the conclusion of a rule prove the premises of the rule.

  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

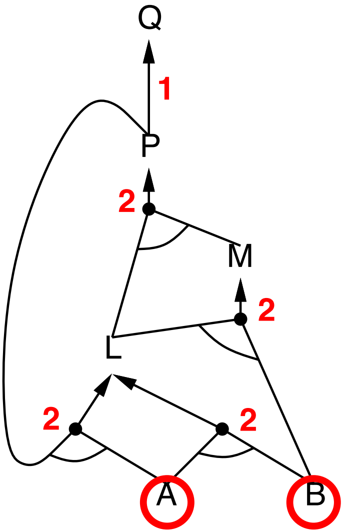


Forward chaining algorithm

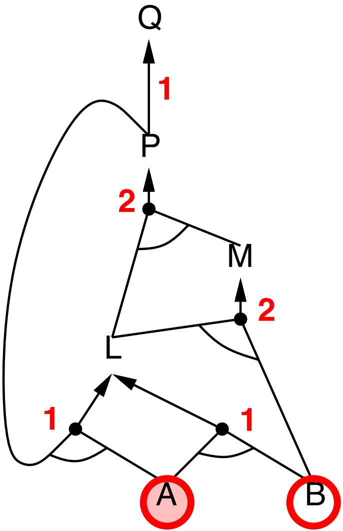
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while queue is not empty do
     p ← POP(queue)
     if p = q then return true
     if inferred[p] = false then
        inferred[p] ← 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



Backward chaining

Idea: work backwards from the query q:

to prove q by BC,

check if q is known already, or

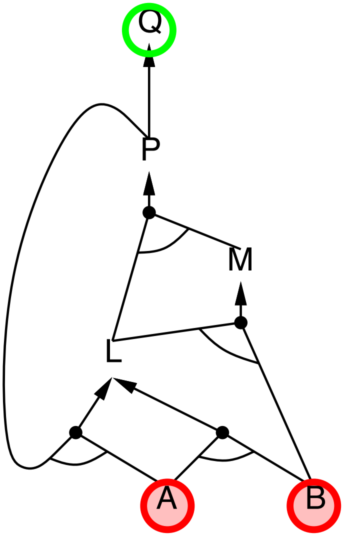
prove by BC all premises of some rule concluding q

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



$P \implies Q$ (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