Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #24

Artificial Intelligence for Data Science

Week-7:
Introduction to Knowledge Representation and Logic [Part-III]

Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

Inference by Proof

2

We say there is a proof of φ from Γ in an inference system $\mathcal{R}$ if we can derive φ by applying the rules of $\mathcal{R}$ repeatedly to Γ and its derived sentences

Example: a proof of P from $\{(P \lor H) \land \neg H\}$

1. $(P \lor H) \land \neg H$ — *by assumption*
2. $P \lor H$ — *by $\land$-elimination applied to (1)*
3. $\neg H$ — *by $\land$-elimination applied to (1)*
4. P — *by unit resolution applied to (2),(3)*

We can represent a proof more visually as a proof tree:

Example:

$$\frac{\dfrac{(P \lor H) \land \neg H}{P \lor H} \qquad \dfrac{(P \lor H) \land \neg H}{\neg H}}{P}$$

Rule-Based Inference in Propositional Logic

3

More precisely, there is a proof of φ from Γ in $\mathcal{R}$ if

1. $\varphi \in \Gamma$ or,
2. there is a rule in $\mathcal{R}$ that applies to Γ and produces φ or,
3. there is a proof of each $\varphi_1, \ldots, \varphi_m$ from Γ in $\mathcal{R}$ and a rule that applies to $\{\varphi_1, \ldots, \varphi_m\}$ and produces φ

Then, the inference system $\mathcal{R}$ is specified as follows:

$$\Gamma \vdash_{\mathcal{R}} \varphi \quad \textit{iff} \quad \textit{there is a proof of } \varphi \textit{ from } \Gamma \textit{ in } \mathcal{R}$$

4

- An inference algorithm that derives only entailed sentences is called **sound or truth-preserving**. Soundness is a highly desirable property.

- An inference algorithm is **complete** if it can derive any sentence that is entailed.

Soundness of R

The given system $\mathcal{R}$ is sound because all of its rules are

Example: the Resolution rule $\dfrac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$

	α	β	γ	$\neg\beta$	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
1.	*false*	*false*	*false*	*true*	*false*	*true*	*false*
2.	*false*	*false*	*true*	*true*	*false*	*true*	*true*
3.	*false*	*true*	*false*	*false*	*true*	*false*	*false*
4.	*false*	*true*	*true*	*false*	<u>*true*</u>	<u>*true*</u>	*true*
5.	*true*	*false*	*false*	*true*	<u>*true*</u>	<u>*true*</u>	*true*
6.	*true*	*false*	*true*	*true*	<u>*true*</u>	<u>*true*</u>	*true*
7.	*true*	*true*	*false*	*false*	*true*	*false*	*true*
8.	*true*	*true*	*true*	*false*	<u>*true*</u>	<u>*true*</u>	*true*

All the interpretations that satisfy both $\alpha \vee \beta$ and $\neg\beta \vee \gamma$ (4,5,6,8) satisfy $\alpha \vee \gamma$ as well

Another Inference System: Resolution

- The previous inference system is difficult to implement efficiently in practice
- Resolution is a more efficient inference system (in practice)
- The main reason is that
  - it requires formulas to be in a special form: CNF
  - it has only one inference rule

Resolution

7

Literal: prop. symbol (P) or negated prop. symbol ($\neg P$)

Clause: set of literals $\{l_1, \ldots, l_k\}$ (understood as $l_1 \vee \cdots \vee l_k$)

Conjunctive Normal Form: set of clauses $\{C_1, \ldots, C_n\}$ (understood as $C_1 \wedge \cdots \wedge C_n$)

Resolution rule for CNF:

$$\frac{l_1 \vee \cdots \vee l_k \vee P \qquad \neg P \vee m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_n}$$

Resolution is sound and complete for CNF KBs

E.g.,

$$\frac{P \vee Q \qquad \neg Q}{P} \qquad \frac{P \vee Q \qquad R \vee \neg Q \vee \neg S}{P \vee R \vee \neg S} \qquad \frac{P \vee Q \qquad \neg Q \vee P \vee R}{P \vee R}$$

Conversion to CNF

8

Ex.: $A \Leftrightarrow (B \vee C)$

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

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

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

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

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

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

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

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

Resolution Procedure

- Proof by contradiction: show KB |= α by showing KB ∧ ¬ α unsatisfiable.
- Do the latter by deriving False from CNF of KB ∧ ¬ α

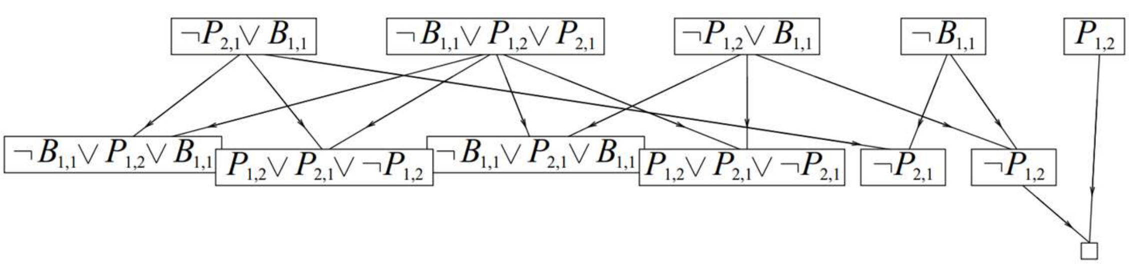
```
function PL-RESOLUTION(KB, α) returns true or false

  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

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

$$\text{CNF} = \{ \neg P_{1,2} \lor B_{1,1},\ \neg B_{1,1} \lor P_{1,2} \lor P_{2,1},\ \neg P_{1,2} \lor B_{1,1},\ \neg B_{1,1},\ P_{1,2} \}$$



Forward and Backward Chaining

11

Horn clause: prop. symbol (p) or implication $p_1 \wedge \cdots \wedge p_n \Rightarrow p$

Horn Form: set of Horn clauses $\{C_1, \ldots, C_n\}$ (understood as $C_1 \wedge \cdots \wedge C_n$)
E.g., $\{ \ C, \ B \Rightarrow A, \ C \wedge D \Rightarrow B \ \}$

Modus Ponens for Horn Form

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

Sound and complete for Horn Form KBs

Can be used with forward chaining or backward chaining

These algorithms are very natural and run in linear time

Forward chaining

12

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

Ex.: query is Q

KB is

$P \implies Q$

$L \wedge M \implies P$

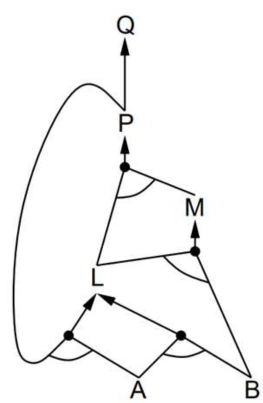
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



Forward chaining algorithm

13

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*
  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**
    $p \leftarrow$ POP(*agenda*)
    unless *inferred*[p] **do**
      inferred[p] $\leftarrow$ *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*

Forward chaining example

14

KB is

$P \implies Q$

$L \wedge M \implies P$

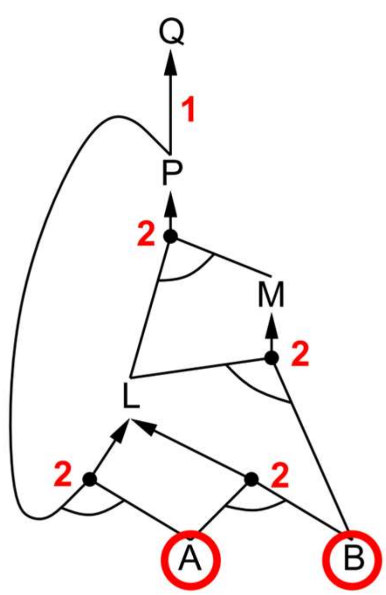
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



Forward chaining example

15

KB is

$P \implies Q$

$L \wedge M \implies P$

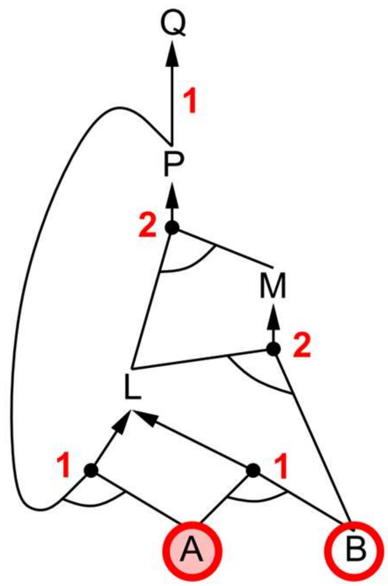
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



Forward chaining example

16

KB is

$P \implies Q$

$L \wedge M \implies P$

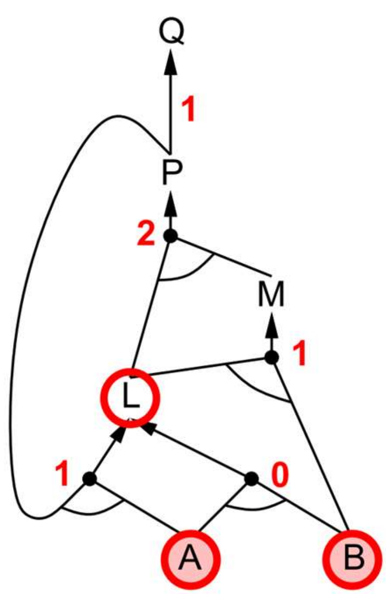
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



Forward chaining example

17

KB is

$P \implies Q$

$L \wedge M \implies P$

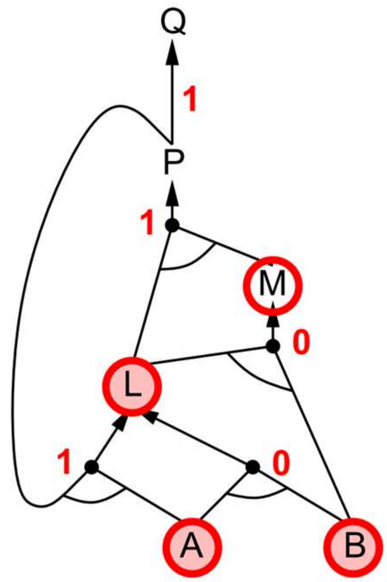
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



Forward chaining example

18

KB is

$P \implies Q$

$L \wedge M \implies P$

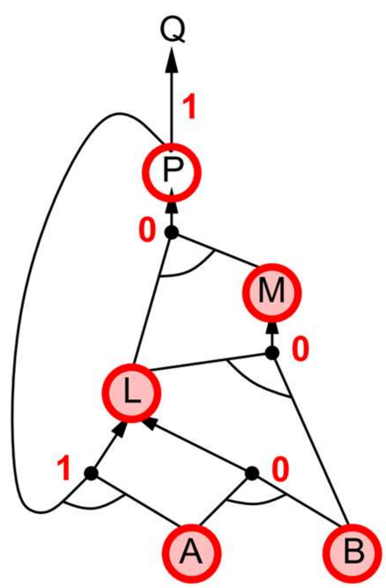
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



Forward chaining example

19

KB is

$P \implies Q$

$L \wedge M \implies P$

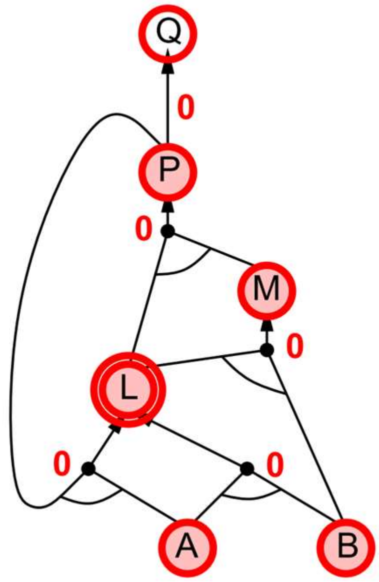
$B \wedge L \implies M$

$A \wedge P \implies L$

$A \wedge B \implies L$

A

B



20

Questions?

Slide Content taken from
Prof. Cesare Tinelli, Prof. Stuart Russell, and
Prof. Jim Martin