CSC520 - Artificial Intelligence
Lecture 12

Dr. Scott N. Gerard

North Carolina State University

Feb 18, 2025

Agenda

- Proof by resolution
- Forward and backward chaining
- First Order Logic
- Models and Interpretation
- Existential and Universal Quantification

Proof Methods

Proof methods can be roughly divided into two kinds

- Model checking
  - Involve truth table enumeration (always exponential in n)
  - Improved backtracking, e.g., Davis–Putnam–Logemann–Loveland (DPLL)
  - Heuristic search in model space, e.g., min-conflicts-like hill-climbing algorithms
- 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 translation of sentences into a normal form

Logical Equivalence

Two sentences α and β are logically equivalent iff they are true in the same set of models
$\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	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
  - $True$, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- Validity is connected to inference via the Deduction Theorem
  - $KB \models \alpha$ iff $(KB \Rightarrow \alpha)$ is *valid*
- A sentence is satisfiable if it is true in *some* model
  - $A \vee B$, C
- A sentence is unsatisfiable if it is true in *no* models
  - $A \wedge \neg A$
- Satisfiability is connected to inference
  - $KB \models \alpha$ iff $(KB \wedge \neg\alpha)$ is unsatisfiable
  - Known as proof by refutation or contradiction

Inference Rules

- All logical equivalences are inference rules
- $\dfrac{\alpha \Rightarrow \beta, \alpha}{\beta}$ – Modus Ponens
- $\dfrac{\alpha \wedge \beta}{\alpha}$ – And-elimination

Conjunctive Normal Form

- A sentence expressed as conjunction of disjunctions is said to be in conjunctive normal form
  - E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$
- Any sentence in propositional logic can be converted to CNF
  - E.g., $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

$$
\begin{array}{rl}
(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) & \text{biconditional elimination} \\
(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1}) & \text{implication elimination} \\
(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1}) & \text{De Morgan} \\
(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) & \text{distributivity}
\end{array}
$$

Proof By Resolution

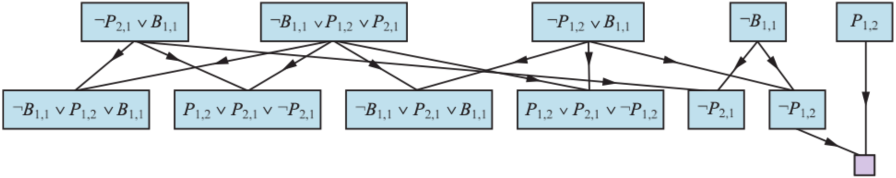
- Resolution inference rule
  - takes two clauses with at least one pair of complementary literals
  - produces a new clause containing all literals of the two original clauses except the two complementary literals

- E.g. $\dfrac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$

- E.g. $\dfrac{P_{1,1} \vee P_{3,1}, \qquad \neg P_{1,1} \vee \neg P_{2,2}}{P_{3,1} \vee \neg P_{2,2}}$
- Resolution is sound and complete for propositional logic
- Idea: To show $KB \models \alpha$, we show that $KB \wedge \neg\alpha$ is unsatisfiable

Proof by Resolution

- $KB : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) \wedge \neg B_{1,1}$
- $KB : (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1}) \wedge \neg B_{1,1}$
- Would like to prove: $\alpha = \neg P_{1,2}$



Resolution Algorithm

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

inputs: KB, a knowledge base, α, a sentence

clauses $\leftarrow$ the set of clauses in CNF in $KB \wedge \neg\alpha$

new $\leftarrow \{\}$

while true **do**

for 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*

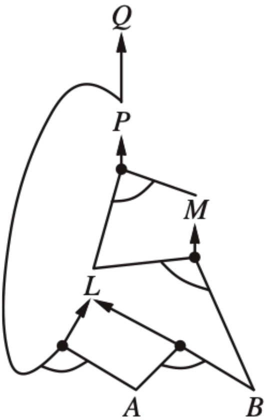
Forward and Backward Chaining

- Horn clause is either (a) a proposition, or (b) conjunction of propositions $\Rightarrow$ proposition
  - E.g., C, $B \Rightarrow A$, $C \wedge D \Rightarrow B$
- Alternately, Horn clause is disjunction of literals of *which at most one is positive*
  - E.g., C, $\neg B \vee A$, $\neg C \vee \neg D \vee B$
- Suppose KB is conjunction of Horn clauses
  - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
- On such KB forward or backward chaining can be used which runs in linear time
- Modus Ponens for inferencing

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n \quad \alpha_1 \wedge \alpha_2 \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Forward Chaining

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



Forward Chaining Algorithm

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

inputs: KB, a knowledge base, q, a proposition symbol

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

$inferred \leftarrow$ a table, $inferred[c]$ is initially false for all symbols

$queue \leftarrow$ a queue of symbols, initially symbols that are true in KB

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

$p \leftarrow$ POP($queue$)

if p = q **then return** true

if $inferred[p] = false$ **then**

$inferred[p] \leftarrow true$

for clause $c \in KB$ where $p \in c$.PREMISE **do**

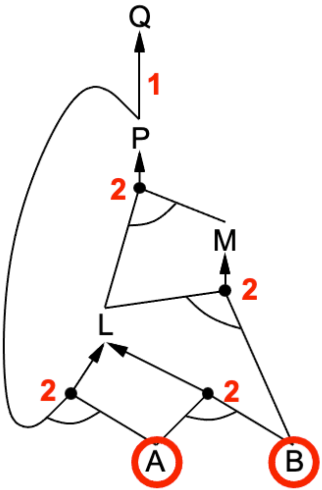
decrement $count[c]$

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

return false

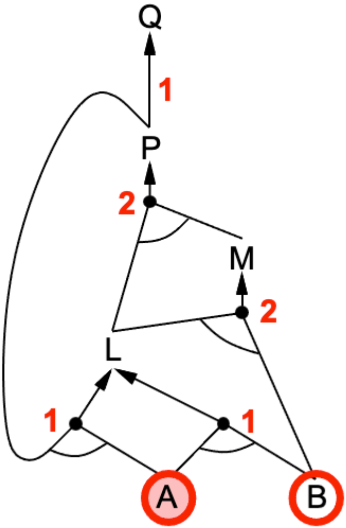
Forward Chaining

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



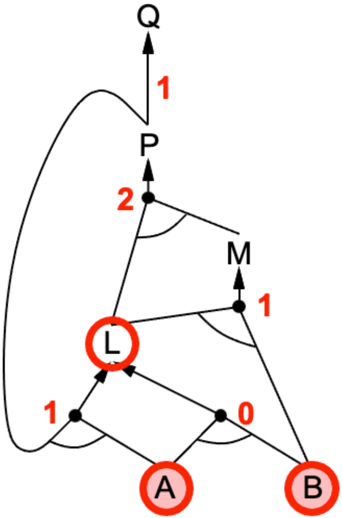
Forward Chaining

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



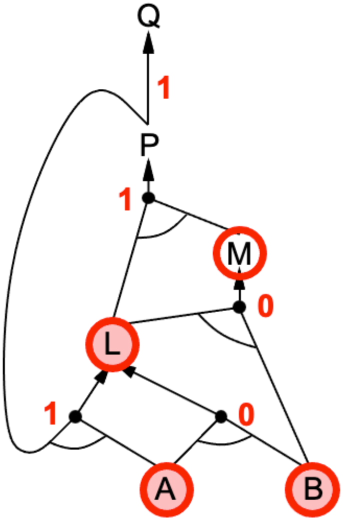
Forward Chaining

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



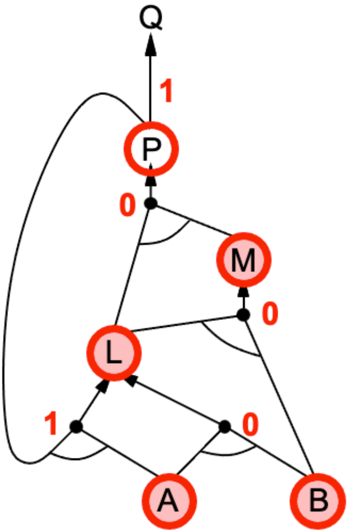
Forward Chaining

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



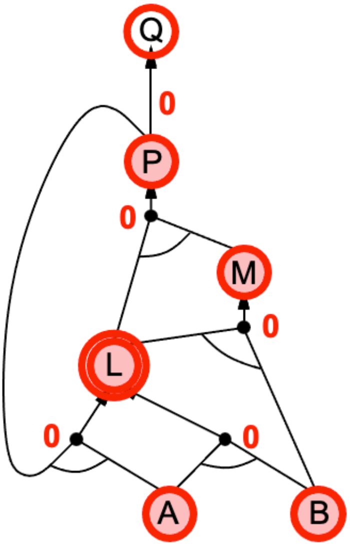
Forward Chaining

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



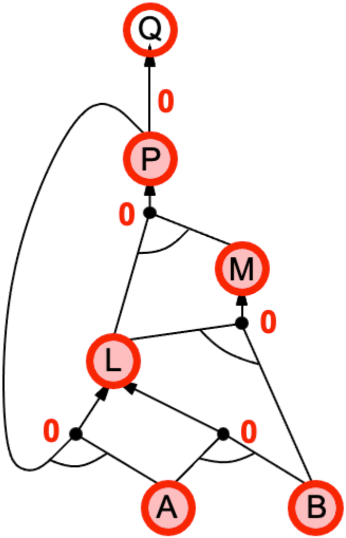
Forward Chaining

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



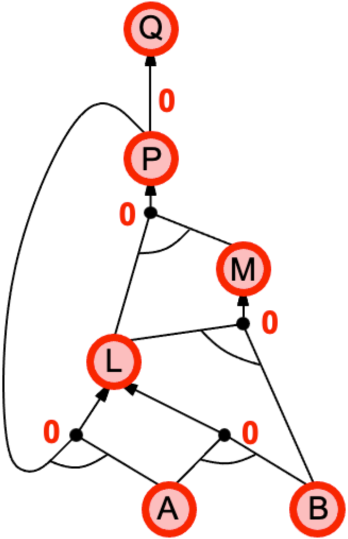
Forward Chaining

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



Forward Chaining

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



Backward Chaining

- Idea: work backwards from the Query q
  - Check if q is known already
  - Prove by backward chaining all premises of some rule that concludes q
- Avoid loops by checking if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal is already proved true or false

Propositional Logic Limitation

- Propositional logic has several strengths
  - Declarative: specifies the facts (what instead of how)
  - Allows partial, disjunctive and negated information
  - Compositional: complex sentences are constructed from elementary sentences
  - Context-independent: meaning is independent of the context
- However, propositional logic has limited expressive power
  - E.g. cannot say: *“Pits cause breeze in adjacent squares”*