Artificial Intelligence Foundation – JC3001

Lecture 17: Logic Agents II

Prof. Aladdin Ayesh (aladdin.ayesh@abdn.ac.uk)

Dr. Binod Bhattarai (binod.bhattarai@abdn.ac.uk)

Dr. Gideon Ogunniye, (g.ogunniye@abdn.ac.uk)

September 2025

Material adapted from:

Russell and Norvig (AIMA Book): Chapters 7 and 9

Course Progression

- Part 1: Introduction
  1. Introduction to AI ✓
  2. Agents ✓
- Part 2: Problem-solving
  1. Search 1: Uninformed Search ✓
  2. Search 2: Heuristic Search ✓
  3. Search 3: Local Search
  4. Search 4: Adversarial Search ✓
- Part 3: Reasoning and Uncertainty
  1. Reasoning 1: Constraint Satisfaction ✓
  2. **Reasoning 2: Logic and Inference**
  3. Probabilistic Reasoning 1: BNs
  4. Probabilistic Reasoning 2: HMMs
- Part 4: Planning
  1. Planning 1: Intro and Formalism
  2. Planning 2: Algos and Heuristics
  3. Planning 3: Hierarchical Planning
  4. Planning 4: Stochastic Planning
- Part 5: Learning
  1. Learning 1: Intro to ML
  2. Learning 2: Regression
  3. Learning 3: Neural Networks
  4. Learning 4: Reinforcement Learning
- Part 6: Conclusion
  1. Ethical Issues in AI
  2. Conclusions and Discussion

Objectives

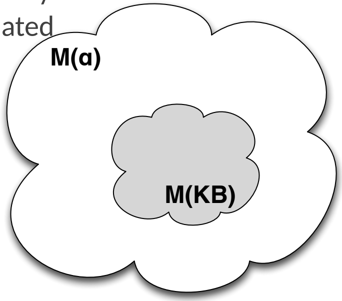
- Knowledge-based agents ✓
- Logic - models and entailment
- Propositional and Lifted Inference
  - Resolution
  - Forward and Backward Chaining

- Logic
  - Forward Chaining
  - Backward Chaining

- **Logics** are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences in the language
- Semantics define the “meaning” of sentences;
  i.e., define truth of a sentence in a world
- E.g., 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 $x = 7$

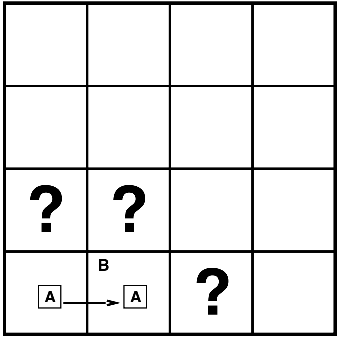
- **Entailment** means that one thing follows from another:
  $KB \models \alpha$
- Knowledge base KB entails sentence α
  if and only if
  α is true in all worlds where KB is true
- E.g. the KB containing "Arsenal won" and "Chelsea won"
  entails "Either Arsenal won or Chelsea won"
  E.g., $x + y = 4$ entails $4 = x + y$
- Entailment is a relationship between sentences (i.e. **syntax**) that is based on **semantics**

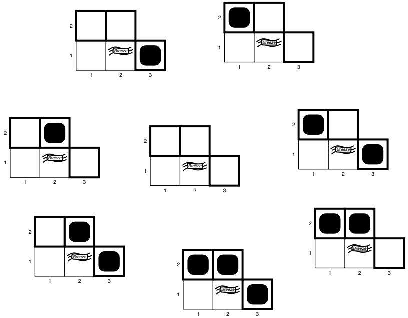
- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated
- We say m is a model of a sentence α if α is true in m
- $M(\alpha)$ is the set of all models of α
- Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$



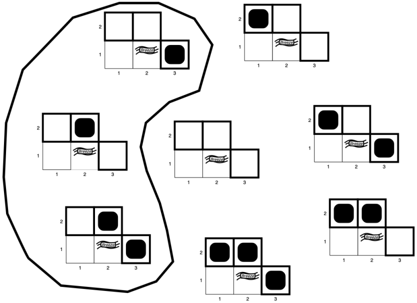
Entailment in the Wumpus World

- Situation after detecting nothing in $[1,1]$, moving right, breeze in $[2,1]$
- Consider possible models for **?**s assuming only pits $P_{1,2}$, $P_{2,2}$ and $P_{3,1}$
- 3 boolean choices $\Rightarrow$ 8 possible models

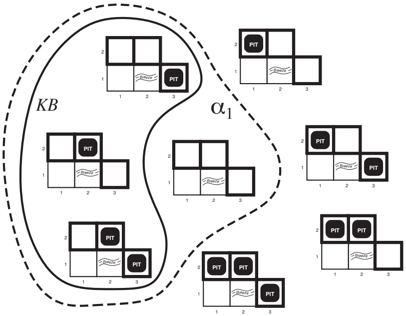




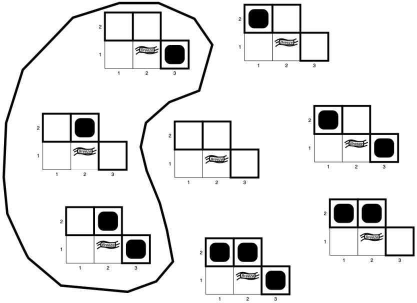
- $\alpha_1 =$ "$[1, 2]$ is safe"
- $\alpha_2 =$ "$[2, 2]$ is safe"



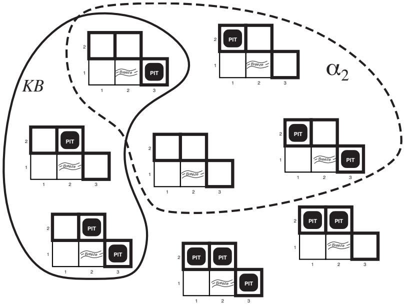
- KB = wumpus-world rules + observations



- KB = wumpus-world rules + observations
- α_1 = "$[1,2]$ is safe", $KB \models \alpha_1$, proved by **model checking**



- KB = wumpus-world rules + observations

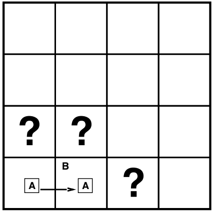


- KB = wumpus-world rules + observations
- α_2 = "$[2, 2]$ is safe", $KB \not\models \alpha_2$

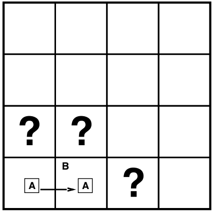
- Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$$\neg P_{1,1}$$
$$\neg B_{1,1}$$
$$B_{2,1}$$

- "Pits cause breezes in adjacent squares"



- 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 $[i, j]$.
  R_1: $\neg P_{1,1}$
  R_2: $\neg B_{1,1}$
  R_3: $B_{2,1}$
- "Pits cause breezes in adjacent squares"
  R_4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  R_5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- "A square is breezy if and only if there is an adjacent pit"



Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*

- Enumerate rows (different assignments to symbols)
- if KB is true in row, check that α is too

Inference by Enumeration

- Depth-first enumeration of all models is sound and complete
- Procedure is also linear on the number of models

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 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?(KB, α)
    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$(KB, \alpha, rest, model \cup \{P = false\})$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
$false$	$false$	$false$	$false$	$false$	$false$	$false$	$true$	$true$	$true$	$true$	$false$	$false$
$false$	$false$	$false$	$false$	$false$	$false$	$true$	$true$	$true$	$false$	$true$	$false$	$false$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$false$	$true$	$false$	$false$	$false$	$false$	$false$	$true$	$true$	$false$	$true$	$true$	$false$
$false$	$true$	$false$	$false$	$false$	$false$	$true$	$true$	$true$	$true$	$true$	$true$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$false$	$true$	$true$	$true$	$true$	$true$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$true$	$true$	$true$	$true$	$true$	$true$	$\underline{true}$
$false$	$true$	$false$	$false$	$true$	$false$	$false$	$true$	$false$	$false$	$true$	$true$	$false$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$false$	$true$	$true$	$false$	$true$	$false$

- How large will this table be?

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
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*

- How large will this table be?
  $2^7 = 128$ lines

- Unfortunately $O(2^n)$ for n symbols;
- Problem is co-NP-complete

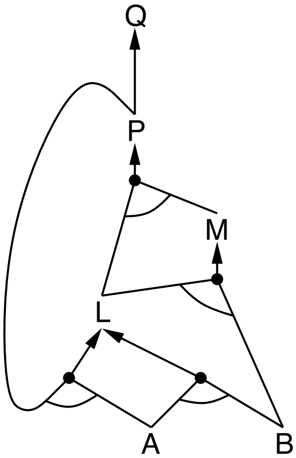
- Horn Form (restricted)
  $KB =$ **conjunction** of **Horn clauses**
- Horn clause =
  - proposition symbol; or
  - (conjunction of symbols) $\Rightarrow$ symbol
    E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
- **Modus Ponens** (for Horn Form): complete for Horn KBs

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

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

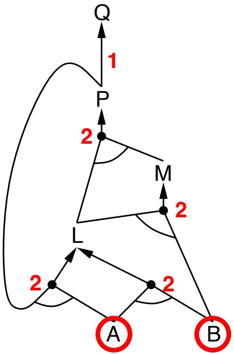
Idea: fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until query is found

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

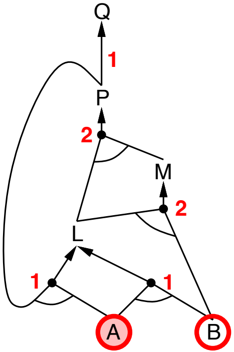


function PL-FC-Entails(KB, α) **returns** $true$ or $false$
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a proposition symbol
  $count \leftarrow$ a table, where $count[c]$ is the number of symbols in c's premise
  $inferred \leftarrow$ a table, where $inferred[s]$ is initially $false$ for all symbols
  $agenda \leftarrow$ a queue of symbols, initially symbols known to be true in KB
  while $agenda$ is not empty **do**
    $p \leftarrow$ Pop$(agenda)$
    if $p = \alpha$ **then return** $true$
    if $inferred[p] = false$ **then**
      $inferred[p] \leftarrow true$
    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 $agenda$
  return $false$

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



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



Forward Chaining Example

$P \Rightarrow Q$

$L \land M \Rightarrow P$

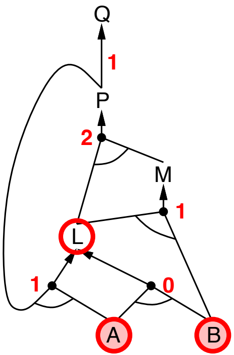
$B \land L \Rightarrow M$

$A \land P \Rightarrow L$

$A \land B \Rightarrow L$

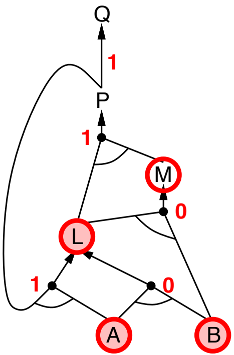
A

B

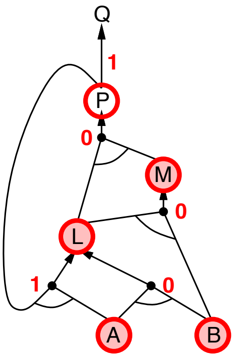


Forward Chaining Example

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

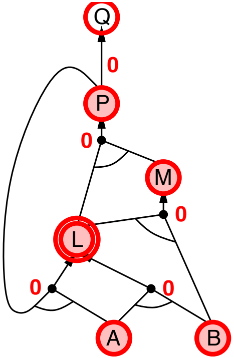


$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



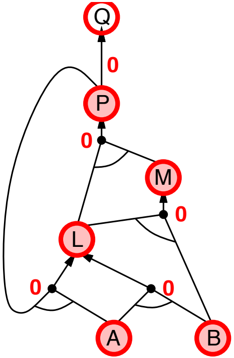
Forward Chaining Example

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



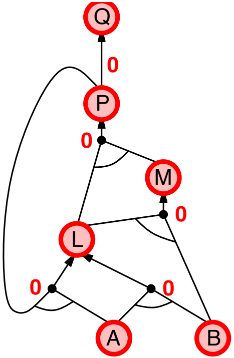
Forward Chaining Example

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



Forward Chaining Example

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



- 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

- Essentially And-Or search:
  - initial state = query
  - goal state = KB
  - actions = clauses
  - states = models of KB
  - if a plan exists q is true
    (plan contains the inference steps)

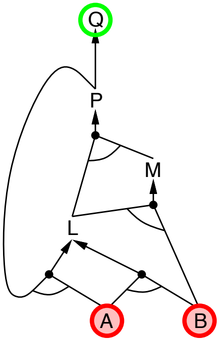
function And-Or-Graph-Search($problem$)
  return Or-Search($problem$.Initial-State, $problem$, [])

---

function Or-Search($state$, $problem$, $path$)
  if $problem$.Goal-Test($state$) **then return** [ ] ▷ *the empty plan*
  if $state$ is on $path$ **then return** $failure$
  for each $action$ in $problem$.Actions($state$) **do**
    $plan \leftarrow$ And-Search(Results($state$, $action$), $problem$, $[state|path]$)
    if $plan \neq failure$ **then return** $[action|plan]$
  return $failure$
function And-Search($states$, $problem$, $path$)
  for each s_i **in** $states$ **do**
    $plan \leftarrow$ Or-Search(s_i, $problem$, $path$)
    if $plan = failure$ **then return** $failure$
  return[**if** s_1 **then** $plan_1$ **else if** s_2 **then** $plan_2$ **else** $\ldots$ **if** s_{n-1} **then** $plan_{n-1}$ **else** $plan_n$]

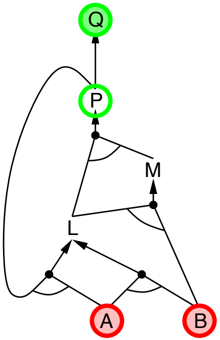
Backward Chaining Example

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

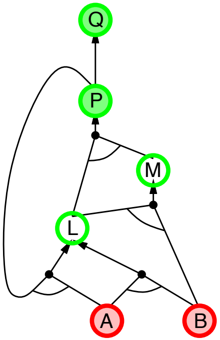


Backward Chaining Example

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

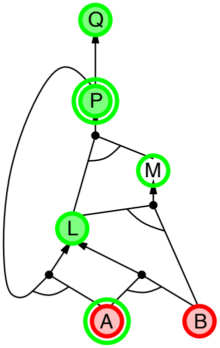


$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B



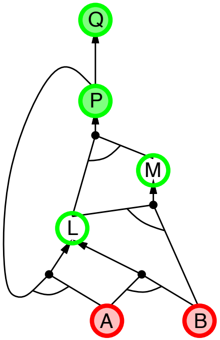
Backward Chaining Example

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

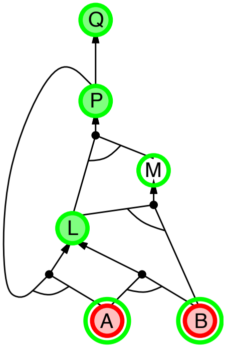


Backward Chaining Example

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

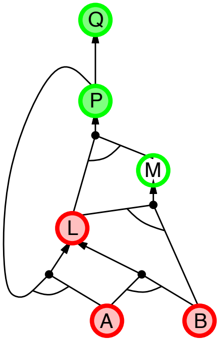


$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

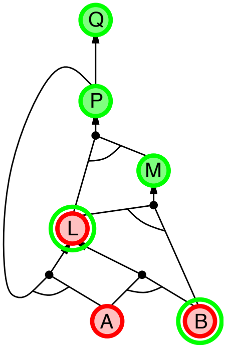


Backward Chaining Example

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

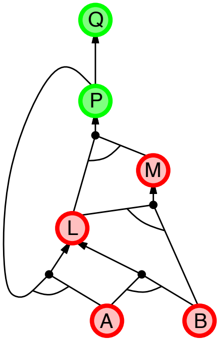


$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



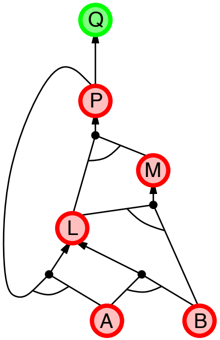
Backward Chaining Example

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



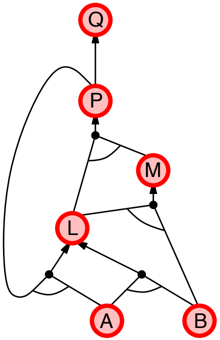
Backward Chaining Example

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



Backward Chaining Example

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



Forward 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 the size of the KB

To continue in the next session.