Artificial Intelligence

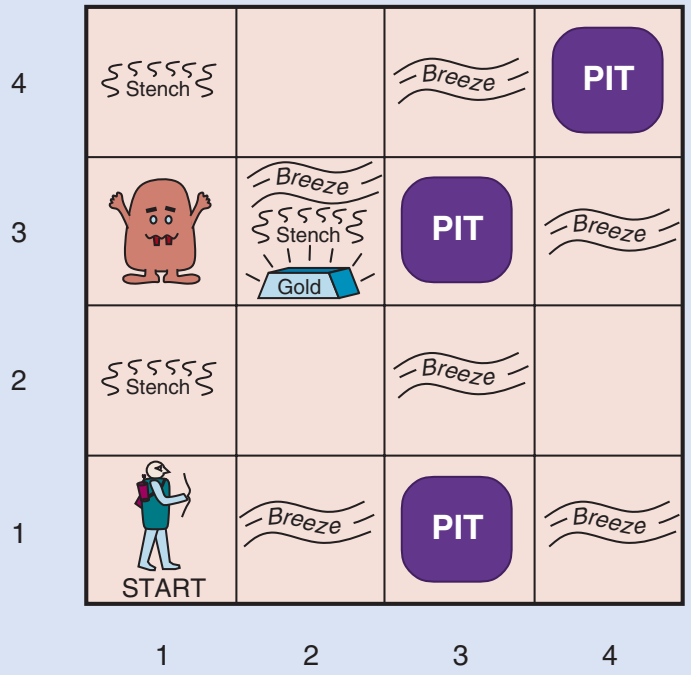
Logical Agents

Dr. Bilgin Avenoğlu

Logical Agents - Introduction

- In AI, knowledge-based agents use a process of reasoning over an internal representation of knowledge to decide what actions to take.
- The problem-solving agents know things, but only in a very limited, inflexible sense.
  - They know what actions are available and what the result of performing a specific action from a specific state will be, but they don't know general facts.
  - A route-finding agent doesn't know that it is impossible for a road to be a negative number of kilometers long.
  - An 8-puzzle agent doesn't know that two tiles cannot occupy the same space.
- The knowledge they have is very useful for finding a path from the start to a goal, but not for anything else.

Wumpus World



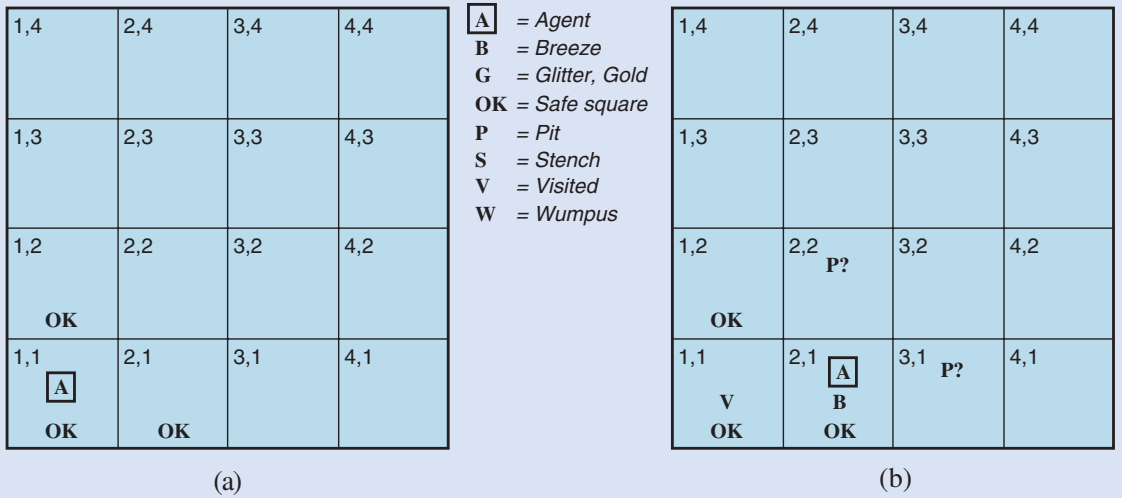


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After moving to [2,1] and perceiving [*None*, *Breeze*, *None*, *None*, *None*].

Figure 7.4 Two later stages in the progress of the agent. (a) After moving to [1,1] and then [1,2], and perceiving [*Stench*, *None*, *None*, *None*, *None*]. (b) After moving to [2,2] and then [2,3], and perceiving [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

Propositional Logic

Definition 2.1 Let $Op = \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\}$ be the set of logical operators and Σ a set of symbols. The sets Op, Σ and $\{t, f\}$ are pairwise disjoint. Σ is called the *signature* and its elements are the *proposition variables*. The set of propositional logic formulas is now recursively defined:

- t and f are (atomic) formulas.
- All proposition variables, that is all elements from Σ, are (atomic) formulas.
- If A and B are formulas, then $\neg A$, (A), $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are also formulas.

Definition 2.2 We read the symbols and operators in the following way:

t:	"true"	
f:	"false"	
$\neg A$:	"not A"	(negation)
$A \wedge B$:	"A and B"	(conjunction)
$A \vee B$:	"A or B"	(disjunction)
$A \Rightarrow B$:	"if A then B"	(implication (also called *material implication*))
$A \Leftrightarrow B$:	"A if and only if B"	(equivalence)

Truth table

Definition 2.3 A mapping $I : \Sigma \rightarrow \{t, f\}$, which assigns a truth value to every proposition variable, is called an *interpretation*.

Table 2.1 Definition of the logical operators by truth table

A	B	(A)	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
t	t	t	f	t	t	t	t
t	f	t	f	f	t	f	f
f	t	f	t	f	t	t	f
f	f	f	t	f	f	t	t

- Because every proposition variable can take on two truth values, every propositional logic formula with n different variables has 2^n different interpretations.
- For unparenthesized formulas, the priorities are ordered as follows, ¬, ∧, ∨, ⇒, ⇔.

Semantic

- A logic must also define the semantics, or meaning, of sentences.
- The semantics defines the truth of each sentence with respect to each possible world.

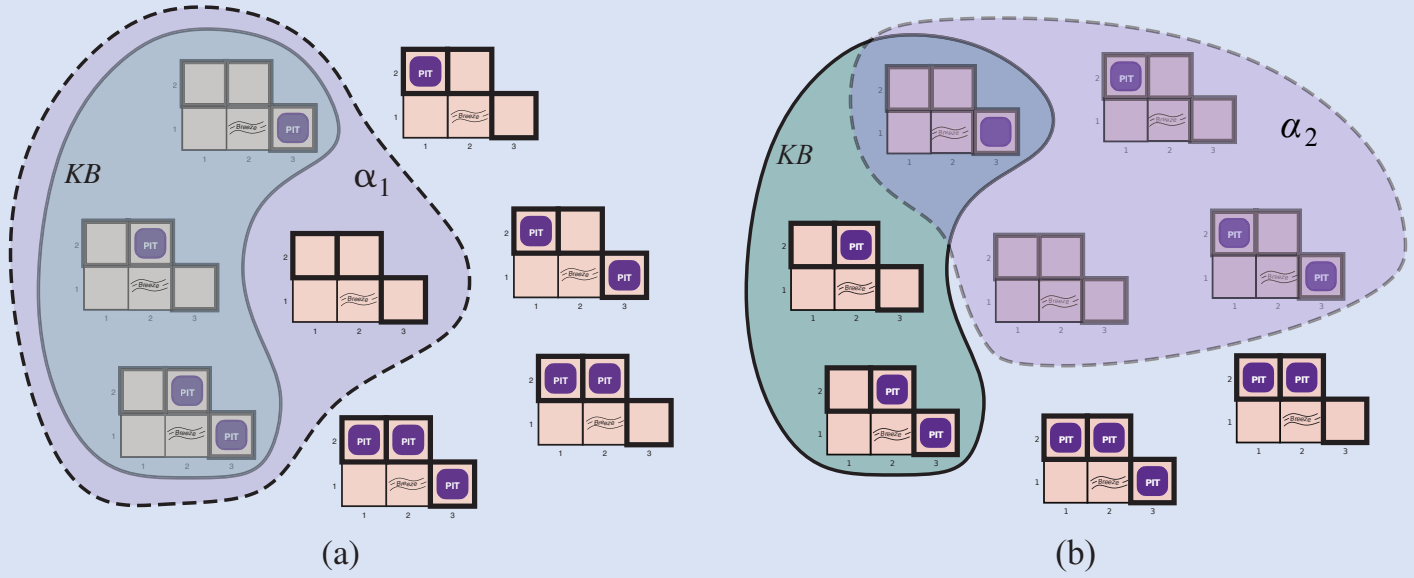


Figure 7.5 Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of α_1 (no pit in [1,2]). (b) Dotted line shows models of α_2 (no pit in [2,2]).

The KB can be thought of as a set of sentences or as a single sentence that asserts all the individual sentences. The KB is false in models that contradict what the agent knows—for example, the KB is false in any model in which [1,2] contains a pit, because there is no breeze in [1,1]. There are in fact just three models in which the KB is true, and these are shown surrounded by a solid line in Figure 7.5. Now let us consider two possible conclusions:

α_1 = "There is no pit in [1,2]." α_2 = "There is no pit in [2,2]."

We have surrounded the models of α_1 and α_2 with dotted lines in Figures 7.5(a) and 7.5(b), respectively. By inspection, we see the following:

in every model in which *KB* is true, α_1 is also true.

Hence, $KB \models \alpha_1$: there is no pit in [1,2]. We can also see that

in some models in which *KB* is true, α_2 is false.

Hence, *KB* does not entail α_2: the agent *cannot* conclude that there is no pit in [2,2]. (Nor can it conclude that there *is* a pit in [2,2].)[4]

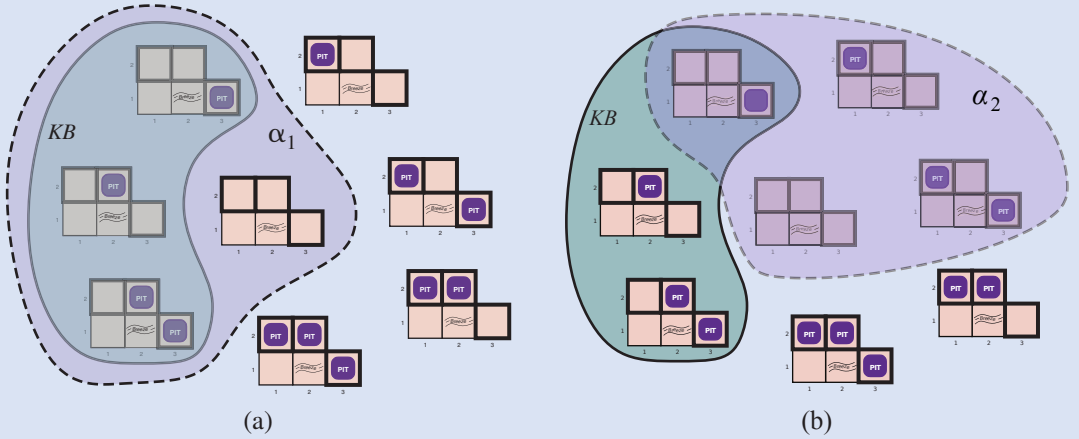
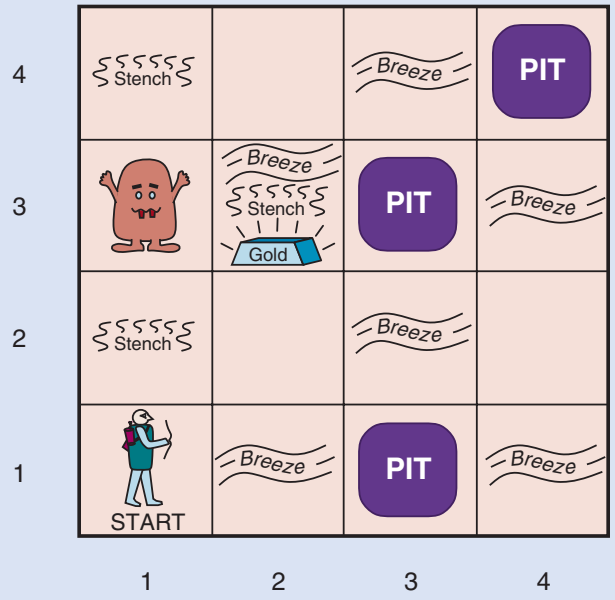


Figure 7.5 Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of α_1 (no pit in [1,2]). (b) Dotted line shows models of α_2 (no pit in [2,2]).

A simple knowledge base

$P_{x,y}$ is true if there is a pit in $[x,y]$.
$W_{x,y}$ is true if there is a wumpus in $[x,y]$, dead or alive.
$B_{x,y}$ is true if there is a breeze in $[x,y]$.
$S_{x,y}$ is true if there is a stench in $[x,y]$.
$L_{x,y}$ is true if the agent is in location $[x,y]$.



- There is no pit in [1,1]:

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

- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:

$$R_2 : \quad B_{1,1} \quad \Leftrightarrow \quad (P_{1,2} \vee P_{2,1}) .$$
$$R_3 : \quad B_{2,1} \quad \Leftrightarrow \quad (P_{1,1} \vee P_{2,2} \vee P_{3,1}) .$$

- The preceding sentences are true in all wumpus worlds. Now we include the breeze percepts for the first two squares visited in the specific world the agent is in, leading up to the situation in Figure 7.3(b).

$$R_4 : \quad \neg B_{1,1} .$$
$$R_5 : \quad B_{2,1} .$$

A simple inference procedure

- Decide whether $KB \models \alpha$ for some sentence α.
  - Is $\neg P_{1,2}$ entailed by our KB?
- Our first algorithm for inference is a model-checking approach that is a direct implementation of the definition of entailment:
  - enumerate the models, and check that α is true in every model in which KB is true.
- Models are assignments of *true* or *false* to every proposition symbol.
- The relevant proposition symbols are
  - $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, and $P_{3,1}$.
  - With seven symbols, there are $2^7 = 128$ possible models;
  - in three of these, KB is true.
  - In those three models, $\neg P_{1,2}$ is true, hence there is no pit in [1,2].
  - $P_{2,2}$ is true in two of the three models and false in one, so we cannot yet tell whether there is a pit in [2,2].

$O(1)$ – constant-time
$O(\log_2(n))$ – logarithmic-time
$O(n)$ – linear-time
$O(n^2)$ – quadratic-time
$O(n^k)$ – polynomial-time
$O(k^n)$ – exponential-time
$O(n!)$ – factorial-time

A simple inference procedure

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

Figure 7.9 A truth table constructed for the knowledge base given in the text. *KB* is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

Propositional Theorem Proving

- Theorem proving: applying rules of inference directly to the sentences in our knowledge base to construct a proof of the desired sentence without consulting models.
- Logical equivalence: two sentences α and β are logically equivalent if they are true in the same set of models. We write this as $\alpha \equiv \beta$

$$\alpha \equiv \beta \quad \text{if and only if} \quad \alpha \models \beta \text{ and } \beta \models \alpha .$$

- A sentence is valid if it is true in all models.
  - *P* ∨ ¬*P* is valid. Valid sentences are also known as tautologies

For any sentences α and β, $\alpha \models \beta$ if and only if the sentence $(\alpha \Rightarrow \beta)$ is valid.

- A sentence is satisfiable if it is true in, or satisfied by, some model.
  - *(R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5)*, is satisfiable - there are three models in which it is true

Propositional Theorem Proving

Definition 2.5 A formula is called

- *Satisfiable* if it is true for at least one interpretation.
- *Logically valid* or simply *valid* if it is true for all interpretations. True formulas are also called *tautologies*.
- *Unsatisfiable* if it is not true for any interpretation.

Every interpretation that satisfies a formula is called a *model* of the formula.

Reductio ad absurdum - proof by contradiction

Validity and satisfiability are of course connected: α is valid iff $\neg\alpha$ is unsatisfiable; contrapositively, α is satisfiable iff $\neg\alpha$ is not valid. We also have the following useful result:

$\alpha \models \beta$ *if and only if the sentence* $(\alpha \wedge \neg\beta)$ *is unsatisfiable.*

Inference and proofs

- Inference rules are applied to derive a proof
  - a chain of conclusions that leads to the desired goal.
- The best-known rule is called Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

- Whenever any sentences of the form $\alpha \Rightarrow \beta$ and α are given, then the sentence β can be inferred.
  - if (*WumpusAhead* ∧ *WumpusAlive*) ⇒ *Shoot* and (*WumpusAhead* ∧ *WumpusAlive*) are given, then *Shoot* can be inferred.

Inference and proofs

- Another useful inference rule is And-Elimination: from a conjunction, any of the conjuncts can be inferred:

$$\frac{\alpha \wedge \beta}{\alpha}$$

- from (*WumpusAhead* ∧ *WumpusAlive*), *WumpusAlive* can be inferred.

Inference Rules

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Figure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary sentences of propositional logic.

Inference rules applied to wumpus world

We start with the knowledge base containing R_1 through R_5 and show how to prove $\neg P_{1,2}$, that is, there is no pit in [1,2]:

1. Apply biconditional elimination to R_2 to obtain

$$R_6 : \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) .$$

2. Apply And-Elimination to R_6 to obtain

$$\frac{\alpha \wedge \beta}{\alpha}$$

$$R_7 : \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) .$$

3. Logical equivalence for contrapositives gives

$$R_8 : \quad (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})) .$$

4. Apply Modus Ponens with R_8 and the percept R_4 (i.e., $\neg B_{1,1}$), to obtain

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

$$R_9 : \quad \neg(P_{1,2} \vee P_{2,1}) .$$

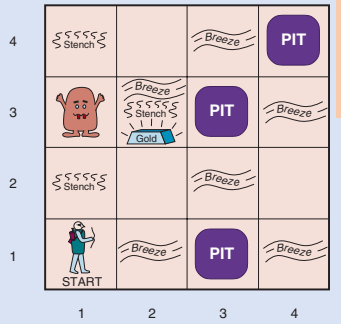
5. Apply De Morgan's rule, giving the conclusion

$$R_{10} : \quad \neg P_{1,2} \wedge \neg P_{2,1} .$$

That is, neither [1,2] nor [2,1] contains a pit.

$$\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}$$

$$\begin{aligned}
R_1 &: \quad \neg P_{1,1} . \\
R_2 &: \quad B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) . \\
R_3 &: \quad B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}) . \\
R_4 &: \quad \neg B_{1,1} . \\
R_5 &: \quad B_{2,1} .
\end{aligned}$$



Searching for proofs

- Searching for proofs is an alternative to enumerating models.
- Finding a proof can be more efficient because the proof can ignore irrelevant propositions, no matter how many of them there are.
  - The proof just given leading to $\neg P_{1,2} \wedge \neg P_{2,1}$ does not mention the propositions $B_{2,1}$, $P_{1,1}$, $P_{2,2}$, or $P_{3,1}$.
  - They can be ignored because the goal proposition, $P_{1,2}$, appears only in sentence R_2;
  - The other propositions in R_2 appear only in R_4 and R_2;
  - So R_1, R_3, and R_5 have no bearing on the proof.

$$
\begin{aligned}
R_1 &: \quad \neg P_{1,1} . \\
R_2 &: \quad B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) . \\
R_3 &: \quad B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}) . \\
R_4 &: \quad \neg B_{1,1} . \\
R_5 &: \quad B_{2,1} . \\
R_6 &: \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) . \\
R_7 &: \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) . \\
R_8 &: \quad (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})) . \\
R_9 &: \quad \neg (P_{1,2} \vee P_{2,1}) . \\
R_{10} &: \quad \neg P_{1,2} \wedge \neg P_{2,1} .
\end{aligned}
$$

Monotonicity

- One final property of logical systems is monotonicity, which says that the set of entailed sentences can only increase as information is added to the KB.
- For any sentences α and β ,

$$\text{if} \quad KB \models \alpha \quad \text{then} \quad KB \wedge \beta \models \alpha \,.$$

- Suppose the KB contains the additional assertion β
  - β : there are exactly eight pits in the world.
  - this might help the agent draw additional conclusions, but it cannot invalidate any conclusion α already inferred
    - such as the conclusion that there is no pit in [1,2].

Proof by resolution

- Any of the search algorithms (depth-first, breadth-first, iterative deepening etc.) can be used to find a sequence of steps that constitutes a proof like this.
- Search algorithms are complete in the sense that they will find any reachable goal,
  - but if the available inference rules are inadequate, then the goal is not reachable
    - no proof exists that uses only those inference rules.
  - for example, if we removed the biconditional elimination rule, the proof in the preceding example would not go through.
- Resolution: a complete inference algorithm when coupled with any complete search algorithm.

Resolution

- The agent returns from [2,1] to [1,1] and then goes to [2,2], it perceives a stench, but no breeze.

$$R_{11}: \quad \neg B_{1,2}.$$
$$R_{12}: \quad B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3}).$$

By the same process that led to R_{10} earlier, we can now derive the absence of pits in [2,2] and [1,3] (remember that [1,1] is already known to be pitless):

$$R_{13}: \quad \neg P_{2,2}.$$
$$R_{14}: \quad \neg P_{1,3}.$$

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

We can also apply biconditional elimination to R_3, followed by Modus Ponens with R_5, to obtain the fact that there is a pit in [1,1], [2,2], or [3,1]:

$$R_{15}: \quad P_{1,1} \vee P_{2,2} \vee P_{3,1}.$$

Now comes the first application of the resolution rule: the literal $\neg P_{2,2}$ in R_{13} *resolves with* the literal $P_{2,2}$ in R_{15} to give the **resolvent**

$$R_{16}: \quad P_{1,1} \vee P_{3,1}.$$

In English; if there's a pit in one of [1,1], [2,2], and [3,1] and it's not in [2,2], then it's in [1,1] or [3,1]. Similarly, the literal $\neg P_{1,1}$ in R_1 resolves with the literal $P_{1,1}$ in R_{16} to give

$$R_{17}: \quad P_{3,1}.$$

In English: if there's a pit in [1,1] or [3,1] and it's not in [1,1], then it's in [3,1].

$$R_1: \quad \neg P_{1,1}.$$
$$R_2: \quad B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}).$$
$$R_3: \quad B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}).$$
$$R_4: \quad \neg B_{1,1}.$$
$$R_5: \quad B_{2,1}.$$
$$R_6: \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}).$$
$$R_7: \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}).$$
$$R_8: \quad (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})).$$
$$R_9: \quad \neg(P_{1,2} \vee P_{2,1}).$$
$$R_{10}: \quad \neg P_{1,2} \wedge \neg P_{2,1}.$$

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{De Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

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

Unit resolution

- Literal : a variable or a negated variable

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

where each ℓ is a literal and ℓ_i and m are **complementary literals** (i.e., one is the negation of the other). Thus, the unit resolution rule takes a **clause**—a disjunction of literals—and a literal and produces a new clause. Note that a single literal can be viewed as a disjunction of one literal, also known as a **unit clause**.

The unit resolution rule can be generalized to the full **resolution** rule

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where ℓ_i and m_j are complementary literals. This says that resolution takes two clauses and produces a new clause containing all the literals of the two original clauses *except* the two complementary literals. For example, we have

$$\frac{P_{1,1} \vee P_{3,1}, \qquad \neg P_{1,1} \vee \neg P_{2,2}}{P_{3,1} \vee \neg P_{2,2}}.$$

Unit resolution

$$\frac{(A_1 \vee \cdots \vee A_m \vee B), \quad (\neg B \vee C_1 \vee \cdots \vee C_n)}{(A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n)}. \tag{2.2}$$

We call the literals B and $\neg B$ complementary. The resolution rule deletes a pair of complementary literals from the two clauses and combines the rest of the literals into a new clause.

You can resolve only one pair of complementary literals at a time. For example, we can resolve P and $\neg P$ to deduce but you can't resolve on both P and Q at once to infer R.

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

Conjunctive normal form - CNF

- A resolution-based theorem prover can, for any sentences α and β in propositional logic, decide whether $\alpha \vDash \beta$.
- The resolution rule applies only to clauses (that is, disjunctions of literals),
  - relevant only to knowledge bases and queries consisting of clauses.

Theorem 2.4 *Every propositional logic formula can be transformed into an equivalent conjunctive normal form.*

Conjunctive normal form - CNF

- converting the sentence $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ into CNF

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. CNF requires $\neg$ to appear only in literals, so we "move $\neg$ inwards" by repeated application of the following equivalences from Figure 7.11:

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

In the example, we require just one application of the last 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. Now we have a sentence containing nested $\wedge$ and $\vee$ operators applied to literals. We apply the distributivity law from Figure 7.11, distributing $\vee$ over $\wedge$ wherever possible.

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

$$\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}$$

A resolution algorithm

- To show that $KB \models \alpha$, we show that $(KB \wedge \neg\alpha)$ is unsatisfiable.
  - Remember that we do this by proving a contradiction.
- First, $(KB \wedge \neg\alpha)$ is converted into CNF.
- Then, the resolution rule is applied to the resulting clauses.
  - Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it is not already present.
  - The process continues until one of two things happens:
    - there are no new clauses that can be added, in which case KB does not entail α;
    - two clauses resolve to yield the empty clause, in which case KB entails α.

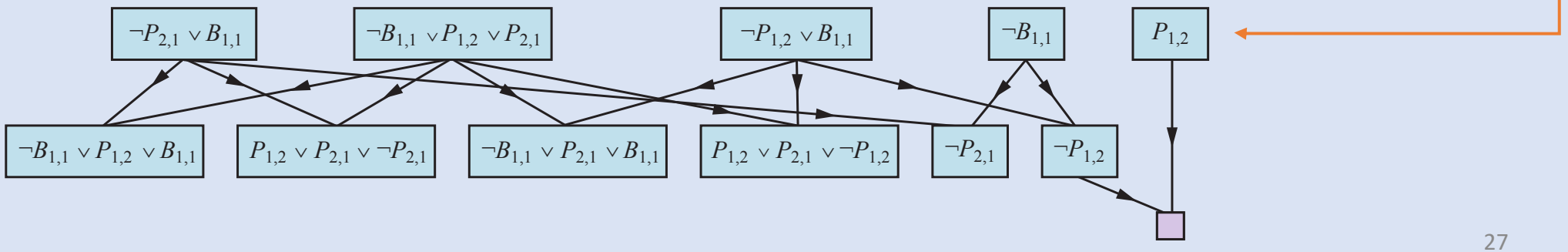
Resolution procedure applied to wumpus world.

- When the agent is in [1,1], there is no breeze, so there can be no pits in neighboring squares. The relevant knowledge base is

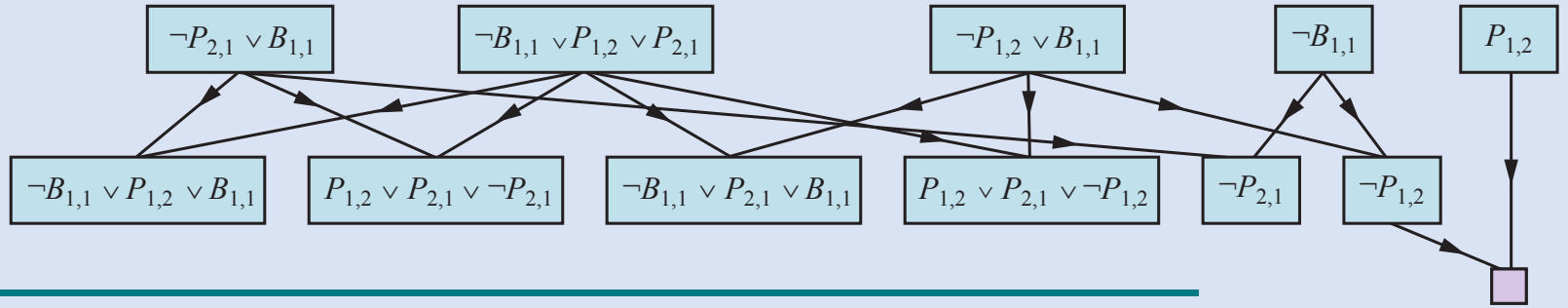
$$KB = R_2 \wedge R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$$

- We wish to prove α, which is, say, $\neg P_{1,2}$.
- When we convert ($KB \wedge \neg\alpha$) into CNF, we obtain the clauses on top of the Fig.
- The second row shows clauses obtained by resolving pairs in the first row.
- Then, when $P_{1,2}$ is resolved with $\neg P_{1,2}$, we obtain the empty clause.

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



A resolution algorithm



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*

Figure 7.13 A simple resolution algorithm for propositional logic. PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

Example

Propositional Resolution

- Resolution rule:

$$\frac{\alpha \vee \beta \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can't apply any more
- Resolution refutation is sound and complete
  - If we derive a contradiction, then the conclusion follows from the axioms
  - If we can't apply any more, then the conclusion cannot be proved from the axioms.

Propositional Resolution Example

Prove R

1	P v Q
2	$P \rightarrow R$
3	$Q \rightarrow R$

Step	Formula	Derivation

Lecture 7 • 9

So let's just do a proof. Let's say I'm given “P or Q”, “P implies R” and “Q implies R”. I would like to conclude R from these three axioms. I'll use the word "axiom" just to mean things that are given to me right at the moment.

Propositional Resolution Example

Prove R

1	P v Q
2	$P \rightarrow R$
3	$Q \rightarrow R$

Step	Formula	Derivation
1	P v Q	Given

We start by converting this first sentence into conjunctive normal form. We don't actually have to do anything. It's already in the right form.

Propositional Resolution Example

Prove R

1	P v Q
2	$P \rightarrow R$
3	$Q \rightarrow R$

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given

Lecture 7 • 11

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Now, “P implies R” turns into “not P or R”.

Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Similarly, “Q implies R” turns into “not Q or R

Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion

Now we want to add one more thing to our list of given statements. What's it going to be?

Not R. Right? We're going to assert the negation of the thing we're trying to prove. We'd like to prove that R follows from these things. But what we're going to do instead is say not R, and now we're trying to prove false. And if we manage to prove false, then we will have a proof that R is entailed by the assumptions.

Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion

Lecture 7 • 14

Now, we'll draw a blue line just to divide the assumptions from the proof steps.

And now, we look for opportunities to apply the resolution rule. You can do it in any order you like (though some orders of application will result in much shorter proofs than others).

Propositional Resolution Example

Prove R

1	P v Q
2	P $\rightarrow$ R
3	Q $\rightarrow$ R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2

We can apply resolution to lines 1 and 2, and get “Q or R” by resolving away P.

Propositional Resolution Example

Prove R

1	P v Q
2	P $\rightarrow$ R
3	Q $\rightarrow$ R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4

And we can take lines 2 and 4, resolve away R, and get “not P.”

Propositional Resolution Example

Prove R

1	P v Q
2	P $\rightarrow$ R
3	Q $\rightarrow$ R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4

Similarly, we can take lines 3 and 4, resolve away R, and get “not Q”.

Propositional Resolution Example

Prove R

1	P v Q
2	P $\rightarrow$ R
3	Q $\rightarrow$ R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7

By resolving away Q in lines 5 and 7, we get R.

Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	•	4,8

And finally, resolving away R in lines 4 and 8, we get the empty clause, which is false. We'll often draw this little black box to indicate that we've reached the desired contradiction.

Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

false v R
¬ R v false

~~false v false~~

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	•	4,8

How did I do this last resolution? Let's see how the resolution rule is applied to lines 4 and 8. The way to look at it is that R is really "false or R", and that "not R" is really "not R or false". (Of course, the order of the disjuncts is irrelevant, because disjunction is commutative). So, now we resolve away R, getting "false or false", which is false.

Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

false v R

¬ R v false

~~false v false~~

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
7	¬ Q	3,4
8	R	5,7
9	•	4,8

One of these steps is unnecessary. Which one? Line 6. It's a perfectly good proof step, but it doesn't contribute to the final conclusion, so we could have omitted it.

Horn clauses and definite clauses

- Definite clause: a disjunction of literals of which exactly one is positive.
  - The clause ($\neg L_{1,1} \vee \neg Breeze \vee B_{1,1}$) is a definite clause, whereas ($\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$) is not, because it has two positive clauses.
- Horn clause: a disjunction of literals of which at most one is positive.
- Inference with Horn clauses can be done through the forward-chaining and backward-chaining algorithms.
- The FC algorithm PL-FC-ENTAILS?(KB, q) determines if a single proposition symbol q—the query—is entailed by a KB of definite clauses.
  - It begins from known facts (positive literals) in the knowledge base.
  - If all the premises of an implication are known, then its conclusion is added to the set of known facts.

Forward and backward chaining

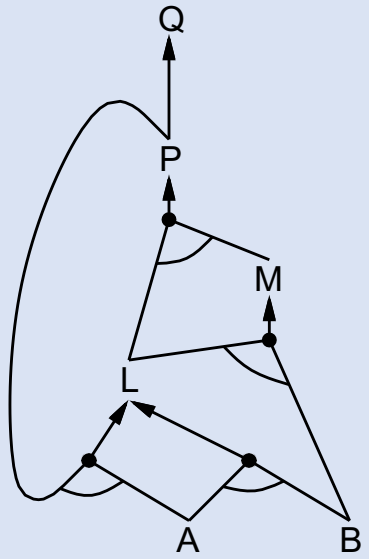
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s]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

Figure 7.15 The forward-chaining algorithm for propositional logic. The *queue* keeps track of symbols known to be true but not yet "processed." The *count* table keeps track of how many premises of each implication are not yet proven. Whenever a new symbol p from the agenda is processed, the count is reduced by one for each implication in whose premise p appears (easily identified in constant time with appropriate indexing.) If a count reaches zero, all the premises of the implication are known, so its conclusion can be added to the agenda. Finally, we need to keep track of which symbols have been processed; a symbol that is already in the set of inferred symbols need not be added to the agenda again. This avoids redundant work and prevents loops caused by implications such as $P \Rightarrow Q$ and $Q \Rightarrow P$.

Forward chaining

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

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



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

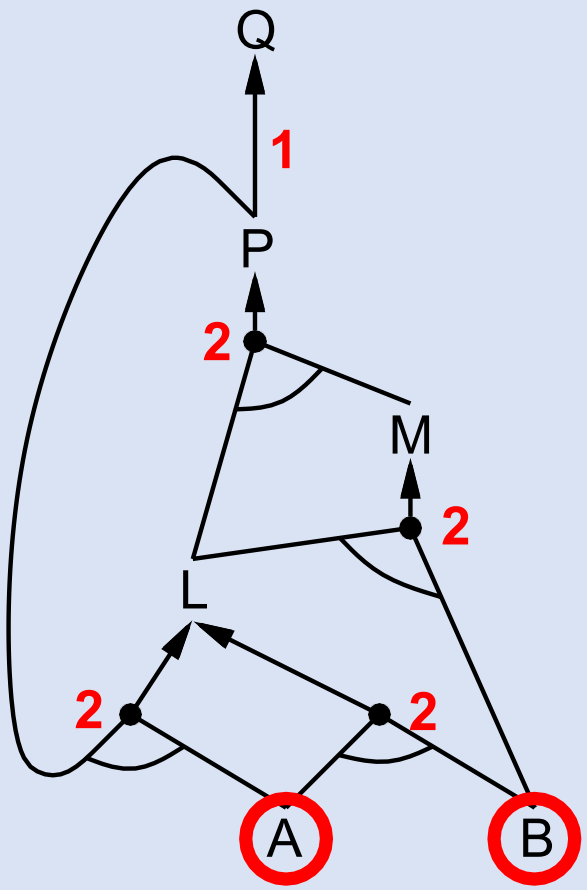
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

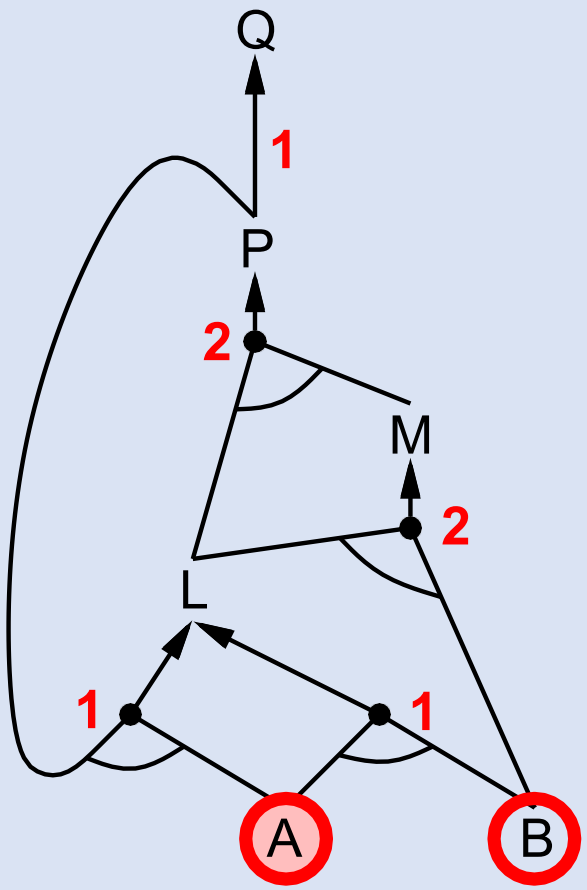
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

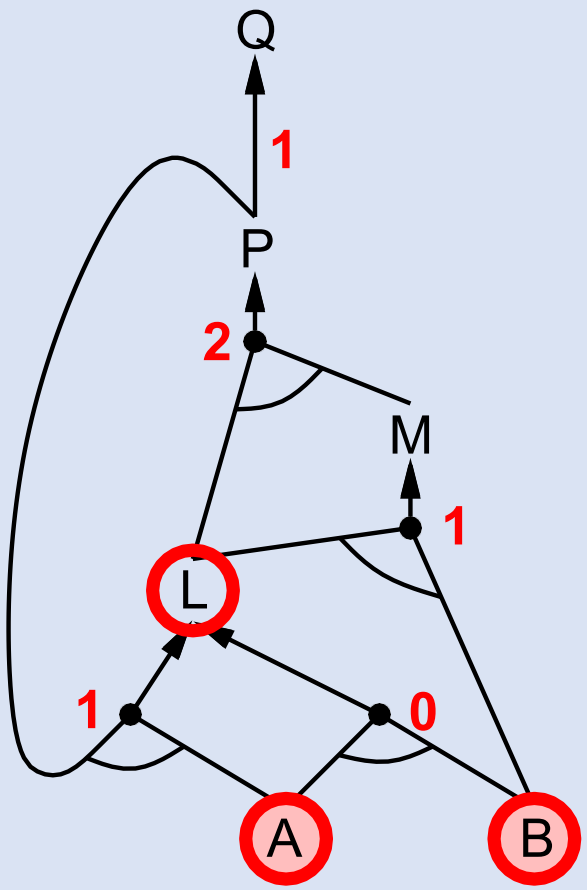
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

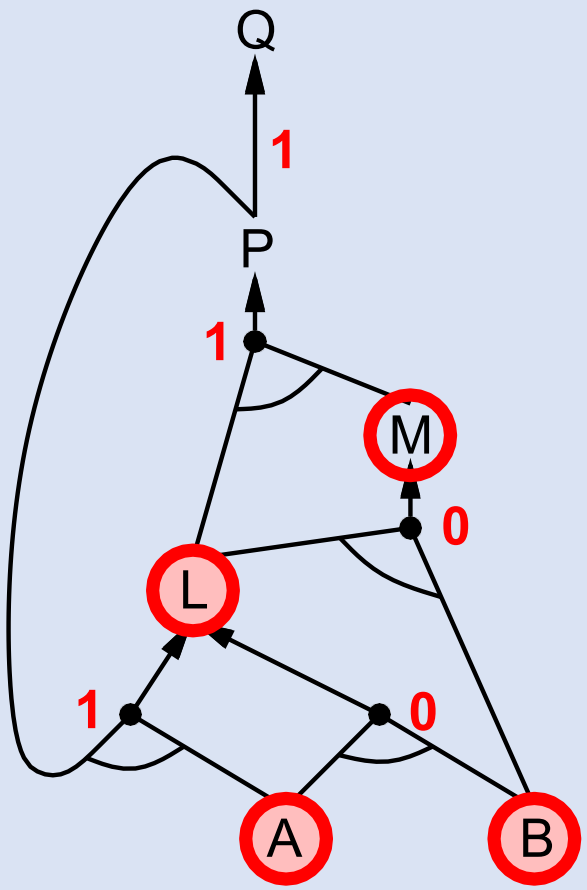
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

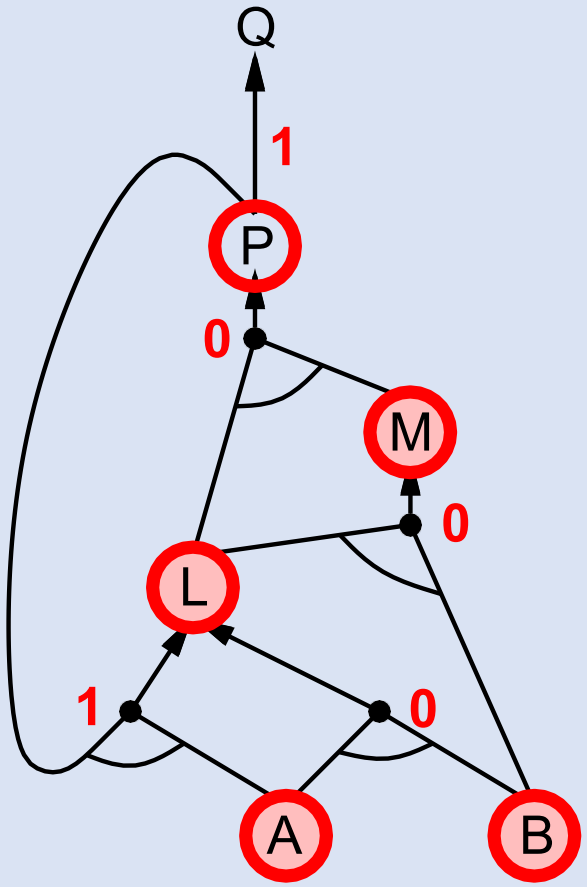
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

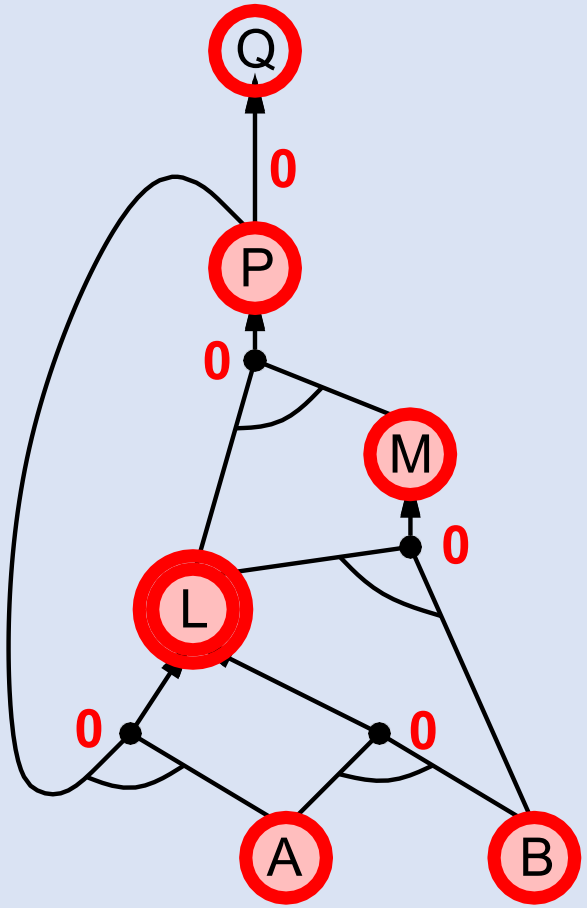
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

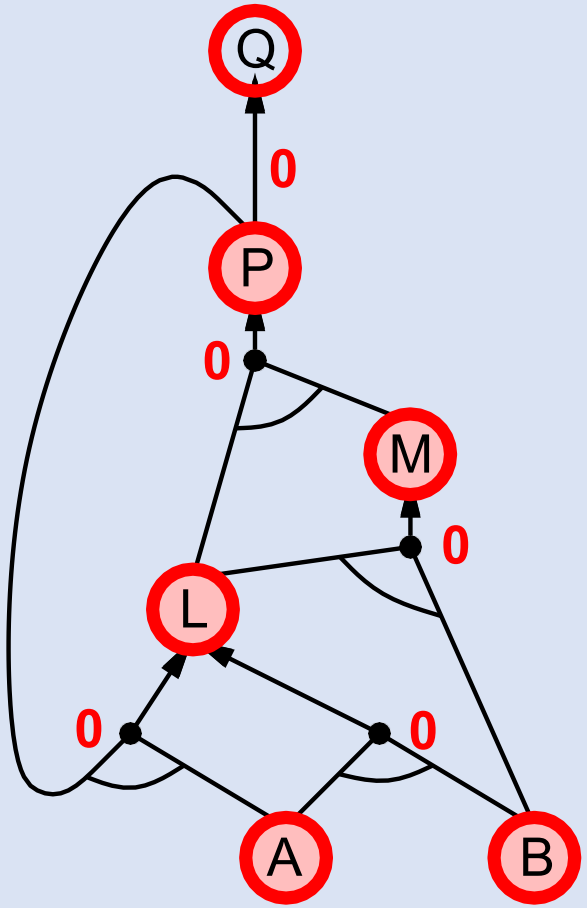
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

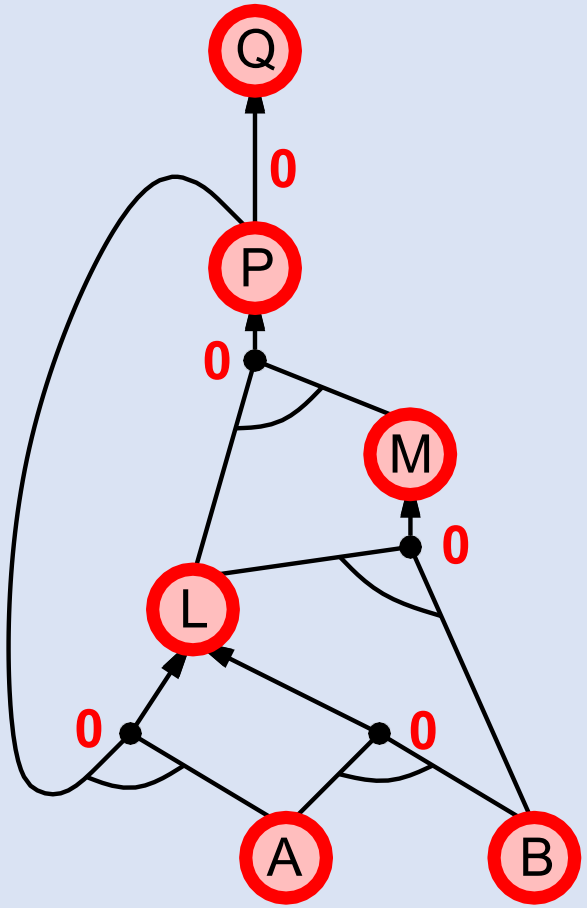
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



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

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

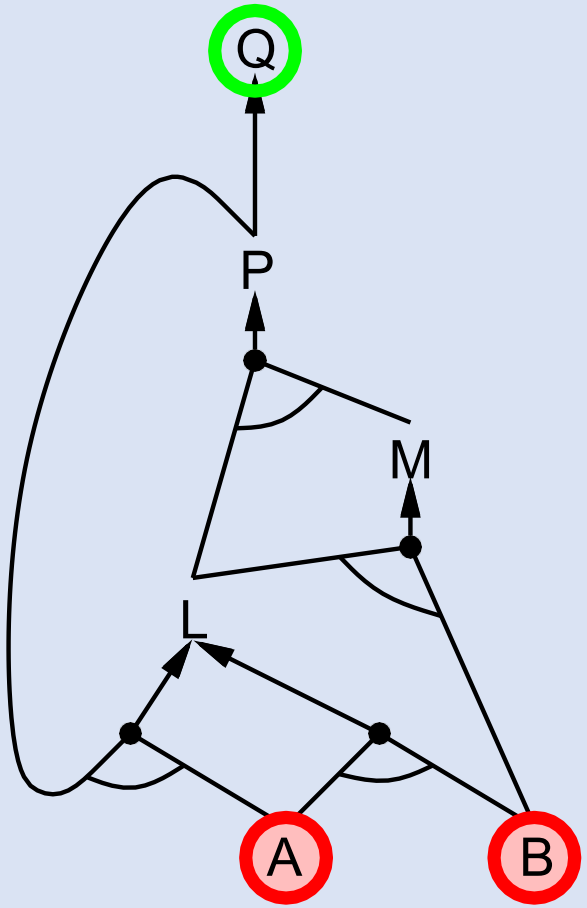
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

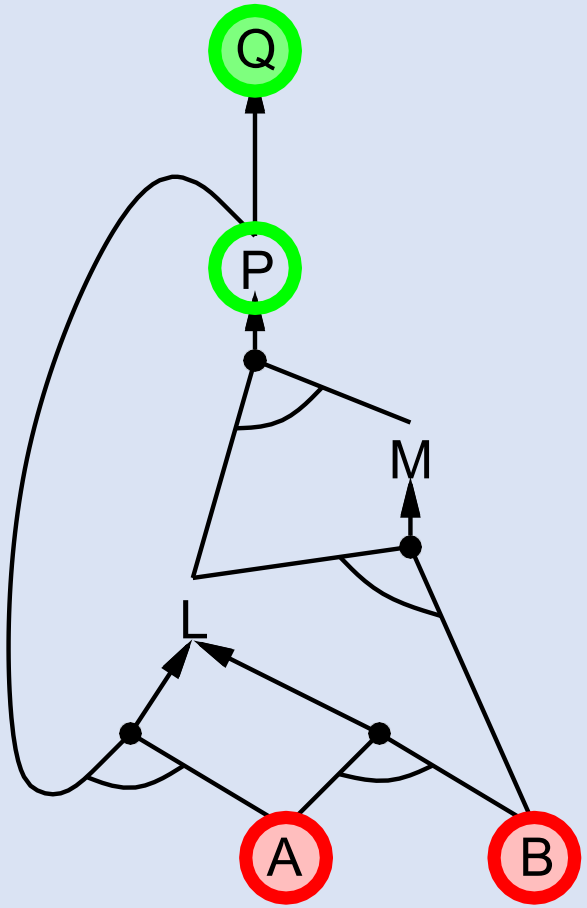
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

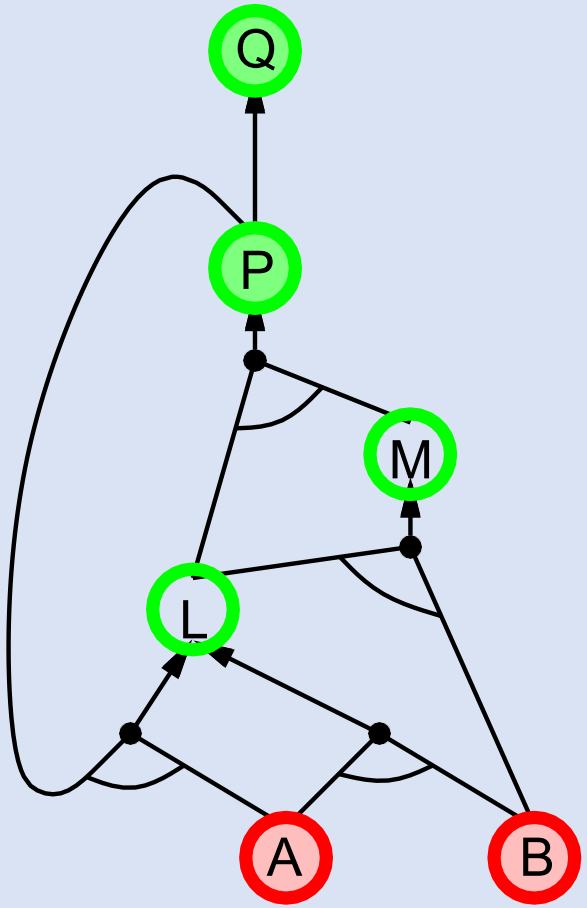
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

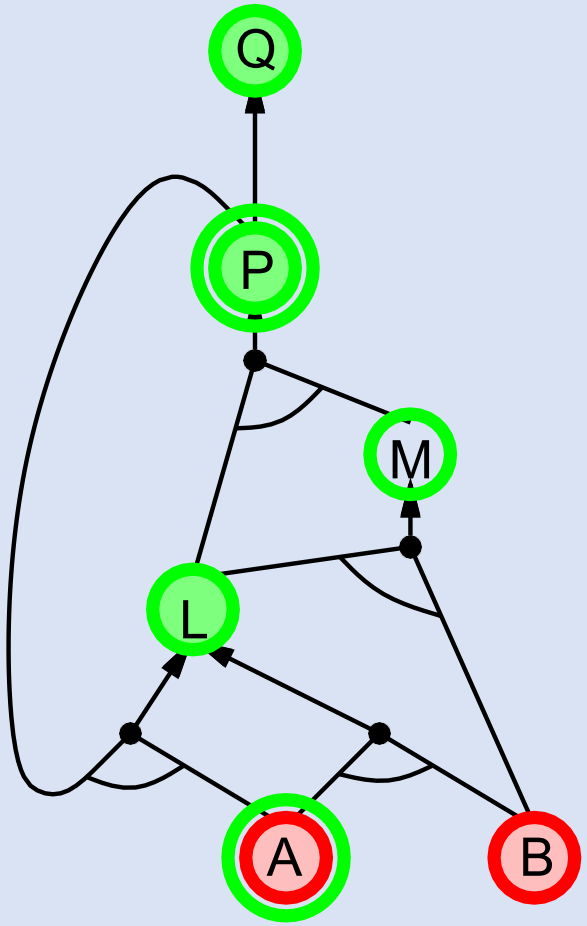
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

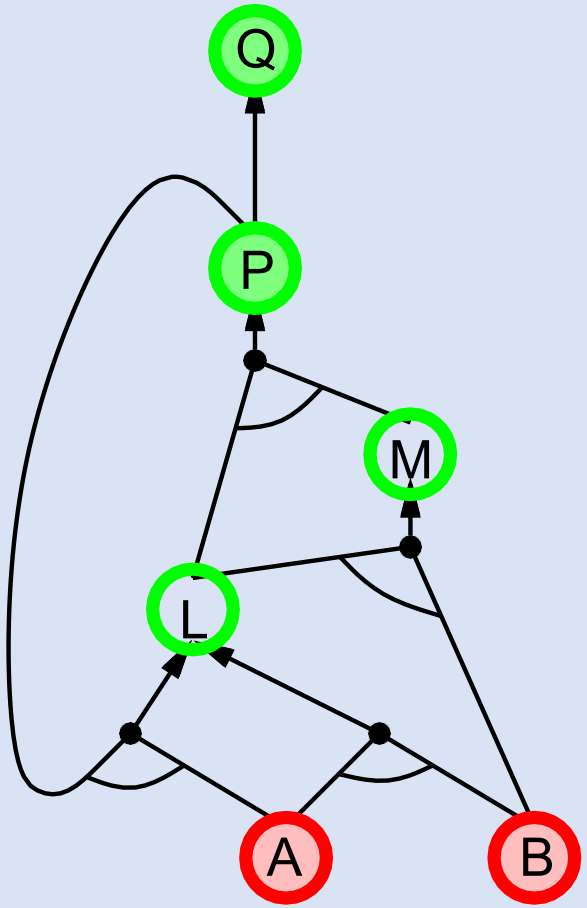
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

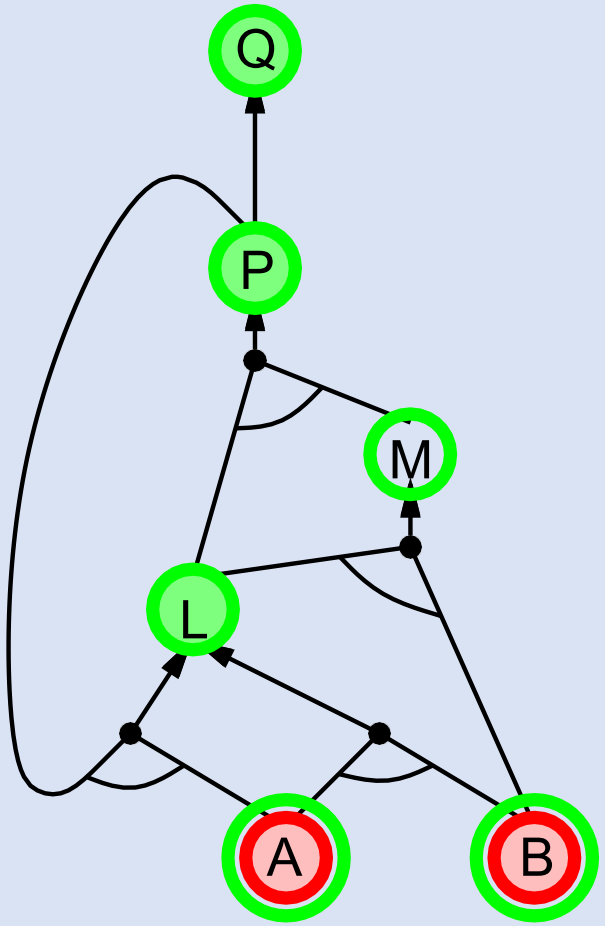
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

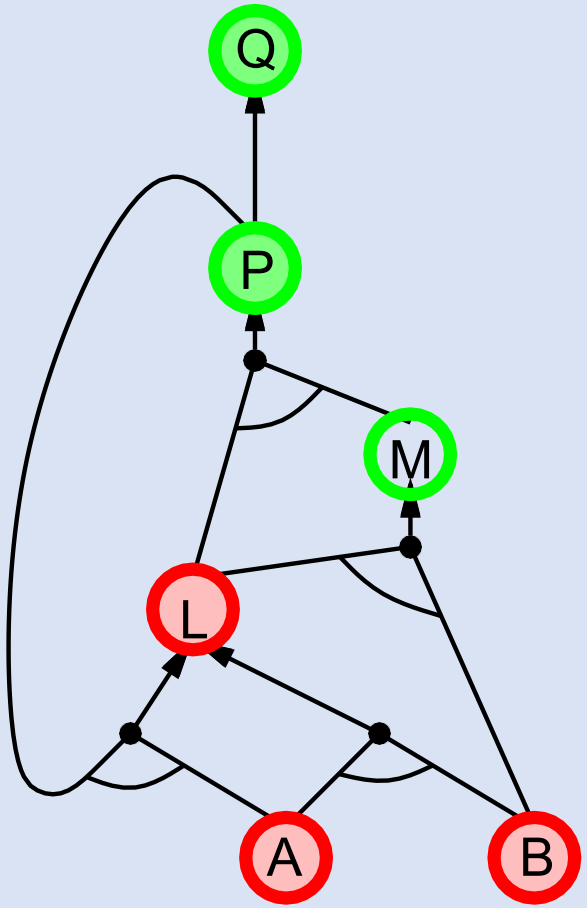
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

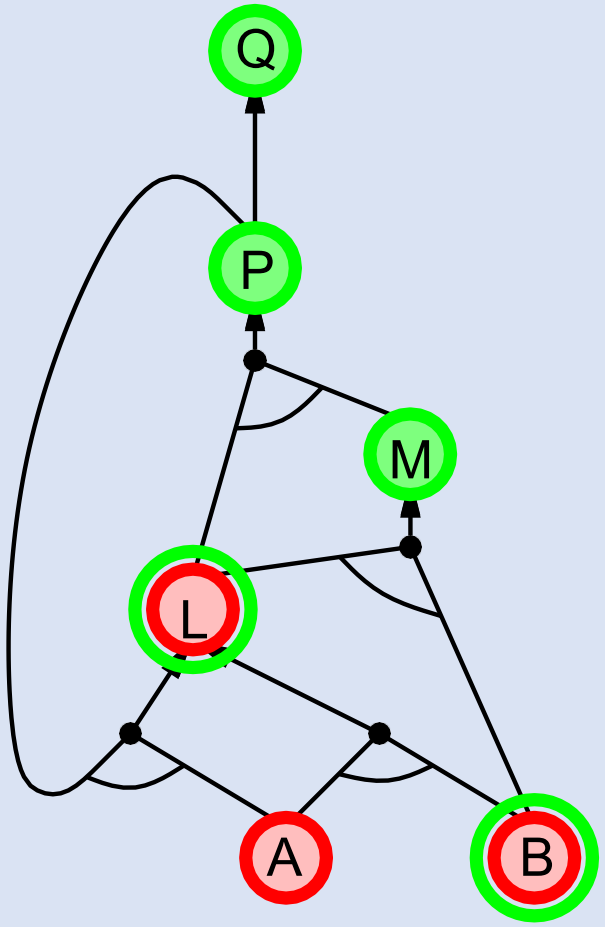
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

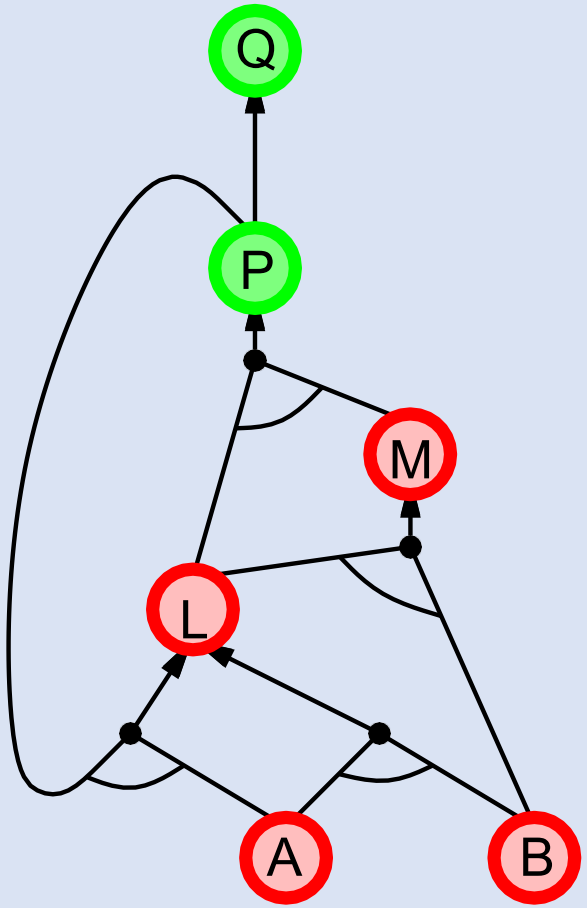
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

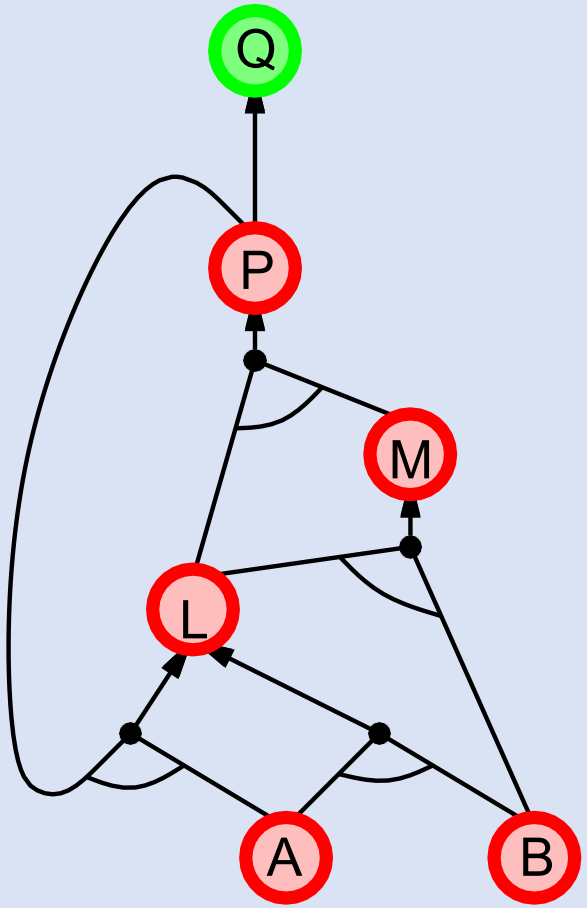
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

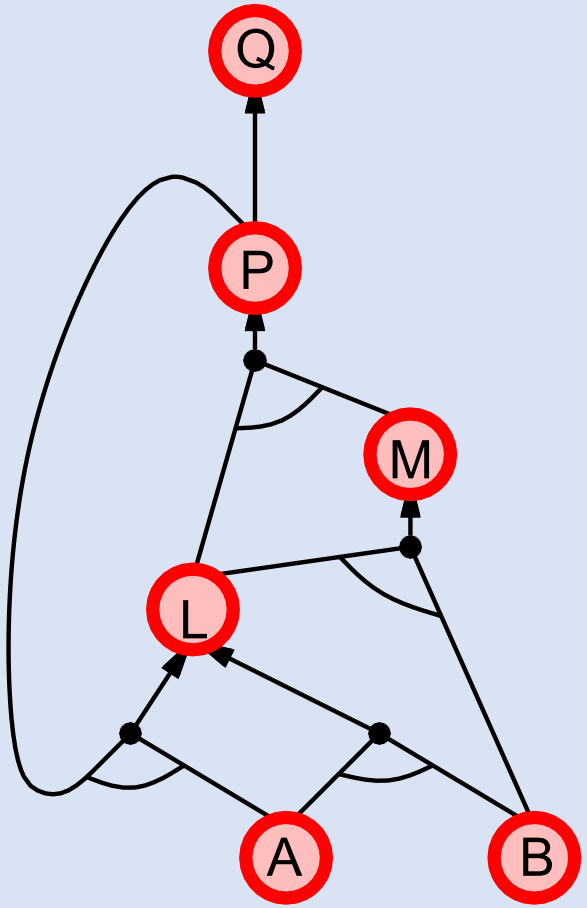
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



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

Agents Based on Propositional Logic

function KB-AGENT(*percept*) **returns** an *action*
  persistent: *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*))
  $t \leftarrow t + 1$
  return *action*

Figure 7.1 A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

Agents Based on Propositional Logic

- The *KB* is composed of axioms - general knowledge about how the world works - and percept sentences obtained from the agent's experience in a particular world.

$R_1: \quad \neg P_{1,1}.$
$R_2: \quad B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}).$
$R_3: \quad B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}).$
$R_4: \quad \neg B_{1,1}.$
$R_5: \quad B_{2,1}.$
$R_6: \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}).$
$R_7: \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}).$
$R_8: \quad (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})).$
$R_9: \quad \neg (P_{1,2} \vee P_{2,1}).$
$R_{10}: \quad \neg P_{1,2} \wedge \neg P_{2,1}.$
$R_{11}: \quad \neg B_{1,2}.$
$R_{12}: \quad B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3}).$
$R_{13}: \quad \neg P_{2,2}.$
$R_{14}: \quad \neg P_{1,3}.$
$R_{15}: \quad P_{1,1} \vee P_{2,2} \vee P_{3,1}.$
$R_{16}: \quad P_{1,1} \vee P_{3,1}.$
$R_{17}: \quad P_{3,1}.$

For each square, it knows that the square is breezy if and only if a neighboring square has a pit; and a square is smelly if and only if a neighboring square has a wumpus. Thus, we include a large collection of sentences of the following form:

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

Agents Based on Propositional Logic

The agent also knows that there is exactly one wumpus. This is expressed in two parts. First, we have to say that there is *at least one* wumpus:

$$W_{1,1} \vee W_{1,2} \vee \cdots \vee W_{4,3} \vee W_{4,4} .$$

Then we have to say that there is *at most one* wumpus. For each pair of locations, we add a sentence saying that at least one of them must be wumpus-free:

$$\neg W_{1,1} \vee \neg W_{1,2}$$
$$\neg W_{1,1} \vee \neg W_{1,3}$$
$$\cdots$$
$$\neg W_{4,3} \vee \neg W_{4,4} .$$

Agents Based on Propositional Logic

- Let's consider the agent's percepts:
  - We are using $S_{1,1}$ to mean there is a stench in [1,1];
  - Can we use a single proposition, *Stench* to mean that the agent perceives a stench?
  - We can't: if there was no stench at the previous time step, then ¬*Stench* would already be asserted, and the new assertion would simply result in a contradiction.
- If the time step is 4,
  - we add $Stench^4$ to the knowledge base,
  - rather than Stench
  - The same goes for the breeze, bump, glitter, and scream percepts.

Agents Based on Propositional Logic

- The idea of associating propositions with time steps extends to any aspect of the world that changes over time.
- Initial KB includes $L^0_{1,1}$ - the agent is in square [1,1] at time 0
- FacingEast^0, HaveArrow^0, and WumpusAlive^0.
- We use the noun fluent to refer to an aspect of the world that changes.

We can connect stench and breeze percepts directly to the properties of the squares where they are experienced as follows.[11] For any time step t and any square $[x, y]$, we assert

$$L^t_{x,y} \Rightarrow (Breeze^t \Leftrightarrow B_{x,y})$$
$$L^t_{x,y} \Rightarrow (Stench^t \Leftrightarrow S_{x,y}) .$$

Agents Based on Propositional Logic

- The percept for a given time step happens first,
- Followed by the action for that time step,
- Followed by a transition to the next time step.

To describe how the world changes, we can try writing **effect axioms** that specify the outcome of an action at the next time step. For example, if the agent is at location $[1,1]$ facing east at time 0 and goes *Forward*, the result is that the agent is in square $[2,1]$ and no longer is in $[1,1]$:

$$L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1}) . \quad (7.1)$$

We would need one such sentence for each possible time step, for each of the 16 squares, and each of the four orientations. We would also need similar sentences for the other actions: *Grab*, *Shoot*, *Climb*, *TurnLeft*, and *TurnRight*.

Agents Based on Propositional Logic

- Given the effect axiom in $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$, combined with the initial assertions about the state at time 0
- *ASK(KB, $L^1_{2,1}$)=true*.
- If we *ASK(KB,HaveArrow^1)*
  - the answer is false, the agent cannot prove; nor can it prove it doesn't have it!
- The information has been lost because the effect axiom fails to state what remains unchanged as the result of an action.
- This is known as frame problem and frame axioms are needed.

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

Agents Based on Propositional Logic

- In a world with *m* different actions and *n* fluents, the set of frame axioms will be of size *O(mn) :* the representational frame problem.
- The representational frame problem is significant because the real world has very many fluents.
- Fortunately for us humans, each action typically changes no more than some small number *k* of those fluents—the world exhibits locality.
- The solution to the problem involves changing one's focus from writing axioms about actions to writing axioms about fluents.

Agents Based on Propositional Logic

- successor-state axioms

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t) .$$

$$HaveArrow^{t+1} \Leftrightarrow (HaveArrow^t \wedge \neg Shoot^t) .$$

For the agent's location, the successor-state axioms are more elaborate. For example, $L^{t+1}_{1,1}$ is true if either (a) the agent moved *Forward* from $[1,2]$ when facing south, or from $[2,1]$ when facing west; or (b) $L^t_{1,1}$ was already true and the action did not cause movement (either because the action was not *Forward* or because the action bumped into a wall). Written out in propositional logic, this becomes

$$\begin{aligned} L^{t+1}_{1,1} \Leftrightarrow\ & (L^t_{1,1} \wedge (\neg Forward^t \vee Bump^{t+1})) \\ \vee\ & (L^t_{1,2} \wedge (FacingSouth^t \wedge Forward^t)) \\ \vee\ & (L^t_{2,1} \wedge (FacingWest^t \wedge Forward^t)) . \end{aligned} \tag{7.3}$$

Agents Based on Propositional Logic

$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 \;;\; Forward^0$$
$$\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 \;;\; TurnRight^1$$
$$\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 \;;\; TurnRight^2$$
$$\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 \;;\; Forward^3$$
$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 \;;\; TurnRight^4$$
$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 \;;\; Forward^5$$
$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

- *ASK(KB, $L^6_{1,2}$ = true)*, so the agent knows where it is.
- *ASK(KB, $W_{1,3}$) = true* and *ASK(KB, $P_{3,1}$) = true*, so the agent has found the wumpus and one of the pits.
- whether a square is OK (free of a pit or live wumpus) to move into:

$$OK^t_{x,y} \Leftrightarrow \neg P_{x,y} \wedge \neg (W_{x,y} \wedge WumpusAlive^t) .$$

- *ASK(KB, $OK^6_{2,2}$) = true*, so the square *[2,2] is OK* to move into.

Final words about propositional logic

- There is no complete solution within logic;
- System designers have to use good judgment in deciding how detailed they want to be in specifying their model, and what details they want to leave out.
- We will see in probability theory allows us to summarize all the exceptions without explicitly naming them.

The End!