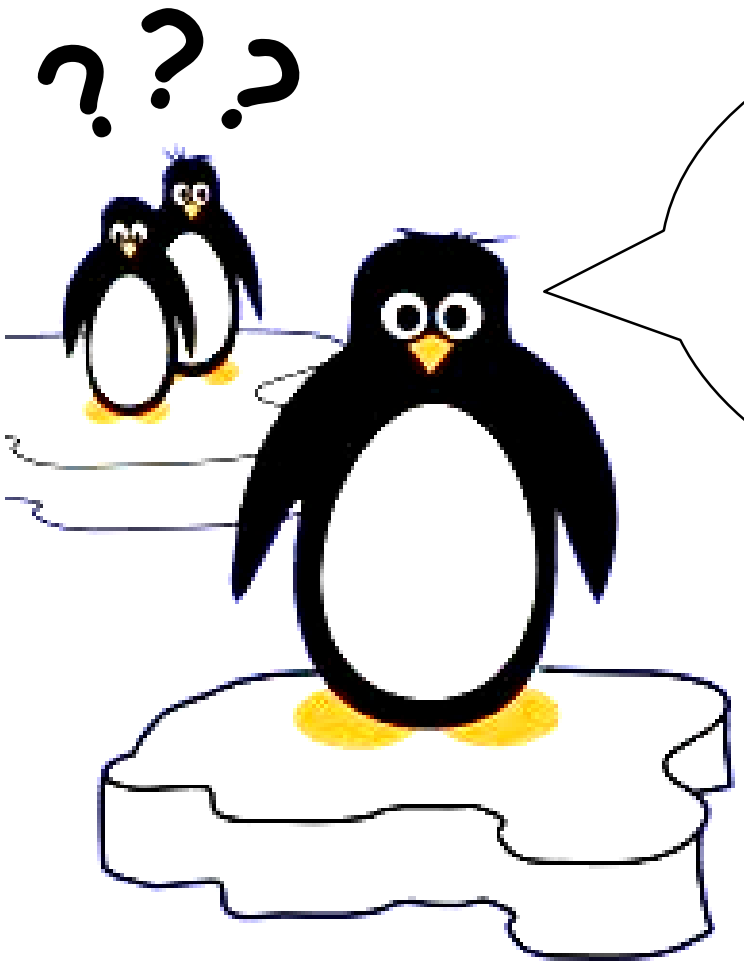
Logical Agents

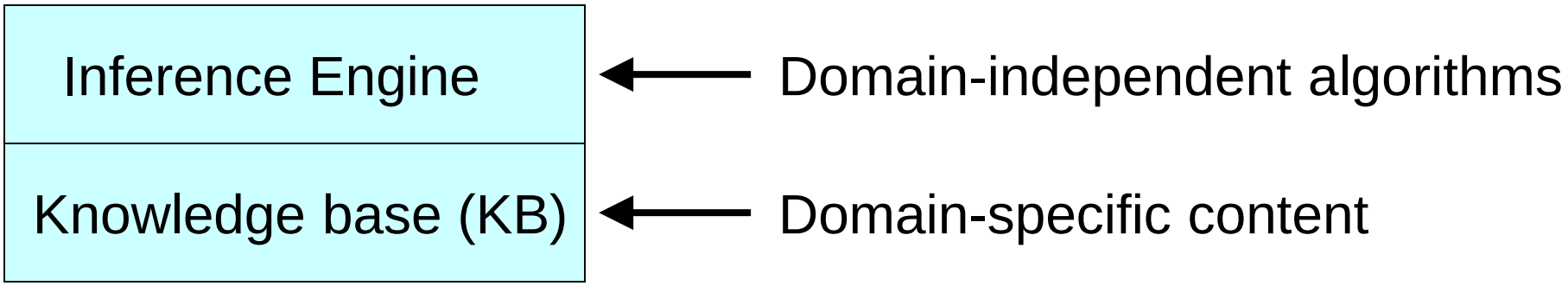


Penguins are black and white. Some old TV shows are black and white. Therefore, some penguins are old TV shows.

What's wrong with this reasoning?

Logical Agents

- A logical agent consists of the following two components:



- Logical agents are not limited to specific tasks; they can utilize the knowledge to solve different problems.
- The KB can be updated to add new information or adapt to environment changes.

The Knowledge Base

- The knowledge in a KB is encoded by a set of **sentences** in a knowledge representation language.
- **Axioms**: sentences that are taken as given without any derivation.
- **Inference**: The process of deriving new sentences from existing ones.
- Background knowledge: The initial content of a KB.

Knowledge-Based Agents

```
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 ← t + 1
return action
```

- **Tell** the *KB* what is perceived.
- **Ask** the *KB* to get suitable actions. Extensive reasoning may be done in this step.
- **Tell** the *KB* that the action is taken (so that the *KB* can update the state).

Logic in General

Here we use arithmetic as an example …

- **Syntax** (how a sentence is formed):
  - $x+y=4$ is a sentence; $=xy4+$ is not
- **Semantics** (meaning of sentence):
  - $x+y=4$ is true if x and y add to 4
- **Model** (a model is a possible way the world is)
  - e.g., $(x=1, y=2)$, $(x=2, y=2)$, $(x=3, y=1)$, etc., are all models of a world specified by x and y.
- We say a model satisfies a sentence α or is a model of α if α is true for that model.
  - e.g., the sentence $x+y=4$ is satisfied by the model $(x=2, y=2)$ but not by the model $(x=1, y=2)$.
  - $M(\alpha)$ is the set of all the models of α.

Entailment

- **Entailment** is a relation between sentences based on semantics (True or False).
- $\alpha \models \beta$ (α entails β) means that whenever α is true, then β is also true.
- $\alpha \models \beta$ if and only if $M(\alpha) \subseteq M(\beta)$
- Example: $x=0$ entails $xy=0$. The former is stronger (or more limiting) on the possible models.

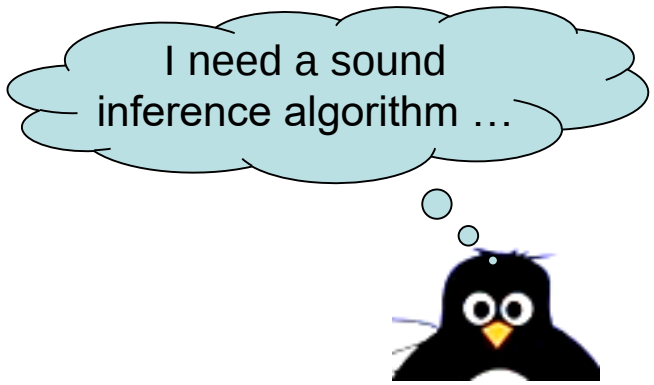
Entailment and Inference

- **Inference** is the process of deriving a new sentence from a set of known sentences.
- $KB \vdash_i \alpha$ means that α can be derived from KB using an inference algorithm i.
- **Sound** inference algorithm:

$$KB \vdash_i \alpha \Rightarrow KB \models \alpha$$

- **Complete** inference algorithm:

$$KB \models \alpha \Rightarrow KB \vdash_i \alpha$$



Propositional Logic: Syntax

- A very simple logic.
- The syntax defines allowable sentences.
- Atomic sentence: a propositional symbol that is either true or false

Sentence	→ *AtomicSentence* | *ComplexSentence*	
AtomicSentence	→ *True* | *False* | *P* | *Q* | *R* | …	
ComplexSentence	→ (*Sentence*) | [*Sentence*]	
	| ¬*Sentence*	negation
	| *Sentence* ∧ *Sentence*	conjunction
	| *Sentence* ∨ *Sentence*	disjunction
	| *Sentence* ⇒ *Sentence*	implication
	| *Sentence* ⇔ *Sentence*	biconditional

antecedent / premise ⇒ consequent / conclusion

Propositional Logic: Semantics

- The semantics is about how to determine whether a sentence is true or false (its truth value).
- A model in propositional logic is specified by the truth values for all the propositional symbols.
- The rules of determining truth values can be given by a truth table.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

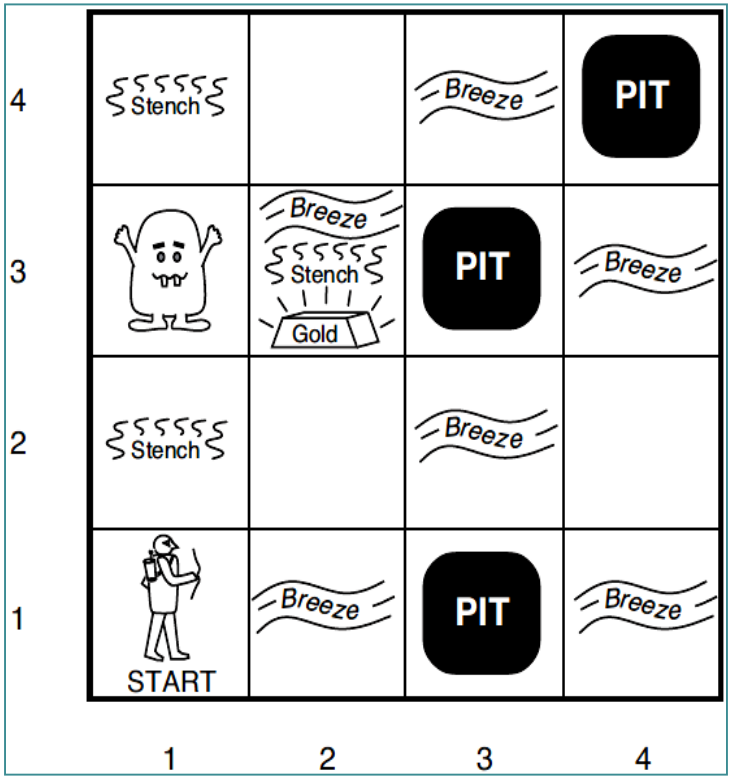
Toy Problem: The Wumpus World

Performance measure
- gold +1000, death -1000
- -1 per step, -10 for using the arrow

Environment
- 4x4 grid of rooms
- start position: (1,1) facing right
- one wumpus, one gold piece, and unknown number of pits randomly located (not at (1,1))

Actuators
- Left turn, Right turn, Forward
- Grab: pick up the gold
- Shoot (there is only one arrow)

Sensors
- Stench: rooms adjacent to the wumpus
- Breeze: rooms adjacent to the pits
- Glitter: room containing gold
- Bump: attempting to move facing a wall
- Scream: when the wumpus is killed



Given the map (which is unknown to the agent), can the agent find a safe sequence of actions to retrieve the gold?

KB of the Wumpus World

- For simplicity, let's consider only pits and breezes, and the agent has only been at (1,1) and (2,1).
- From the rules of the wumpus world:

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

R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

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

- What we know from the percepts:

R_4: $\neg B_{1,1}$

R_5: $B_{2,1}$

Inference by Model Enumeration

A truth table for all the relevant symbols and KB rules.

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

- The KB entails a sentence α iff $M(KB) \subseteq M(\alpha)$.
- Using the truth table, we can prove that the KB entails $\neg P_{1,2}$.
- However, we can say nothing about $P_{2,2}$ or $\neg P_{2,2}$.

Logical Theorem Proving

- Proof by model enumeration is impractical for most problems.
- Logical theorem proving: A sequence of applications of (sound) inference rules to generate new sentences from existing ones in the KB.
- Two well-known inference rules:
  - Modus Ponens: $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$
  - And-Elimination: $\dfrac{\alpha \wedge \beta}{\alpha}$

$$\frac{\text{given}}{\text{inferred}}$$

- Logical equivalences (next slide) can also be used as inference rules.

Logical Equivalences

Two sentences are logically equivalent iff they are true in the same set of models:

$$\alpha \equiv \beta \text{ iff } \alpha \models \beta \text{ and } \beta \models \alpha$$

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

Validity and Satisfiability

- A sentence is **valid** iff it is true in all models.
  - Examples: $P \vee \neg P$, $P \Rightarrow P$
  - Connected to inference via the **deduction theorem**:

    $\alpha \models \beta$ iff $(\alpha \Rightarrow \beta)$ is valid
- A sentence is **satisfiable** iff it is true in some models.
  - Examples: P, $P \wedge Q$
  - Connected to inference via **proof by contradiction**:

    $\alpha \models \beta$ iff $(\alpha \wedge \neg\beta)$ is unsatisfiable (false in all models)

Example of Inference

Here we can try to apply inference rules to the wumpus world KB to prove $\neg P_{1,2}$, i.e., there is no pit in (1,2):

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$	R_2
$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$	biconditional elimination
$((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) \wedge (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1}))$	commutativitiy
$(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$	and-elimination
$\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})$	contraposition
$\neg (P_{1,2} \vee P_{2,1})$	modus ponens + R_4
$\neg P_{1,2} \wedge \neg P_{2,1}$	de Morgan's
$\neg P_{1,2}$	and-elimination

Proof by Search

Q: How can a logical agent find the proof?

A search algorithm can be used to find the proof:

- States: The KB, which grows during the search.
- Initial state: The initial KB.
- Actions: Possible applications of inference rules applied to the sentences in the KB; the sentences need to match the top half of the inference rule.
- Result: Each action generates a new sentence from the bottom half of the inference rule. The new sentence is added to the KB.
- Goal: A state containing the sentence we want to prove.
- Solution: The path.

Inference by Resolution

The resolution rule is **sound and complete** for propositional logic:

$$\frac{l_1 \vee \ldots \vee l_k, \;\; m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

(m is the negation of l_i)

(general form)

$$\frac{l_1 \vee \ldots \vee l_k, \;\; m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

(m_j is the negation of l_i)

(easier to understand)

$$\frac{\alpha \vee \beta, \;\; \neg\beta}{\alpha} \quad \text{and} \quad \frac{\alpha \vee \beta, \;\; \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Conjunctive Normal Form (CNF)

The application of the resolution rule requires the KB to be in **conjunctive normal form** (CNF):

- CNF: The whole KB becomes the conjunction (AND) of a set of clauses.
- **clauses**: disjunctions (OR) of literals, or individual literals
- **literals**: atomic sentences or their negations

Converting a general sentence to CNF:

- biconditional: use biconditional elimination
- implication: use implication elimination
- negation of complex sentences: use De Morgan's Laws
- distribute $\vee$ over $\wedge$ when possible

Example of CNF

Now let us try to convert R_2 of the wumpus world KB to CNF:

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

$$\left((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}\right) \wedge \left(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})\right)$$

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

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

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

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

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

$$\boxed{\neg P_{1,2} \vee B_{1,1}} \qquad \boxed{\neg P_{2,1} \vee B_{1,1}}$$

We end up with 3 CNF clauses.

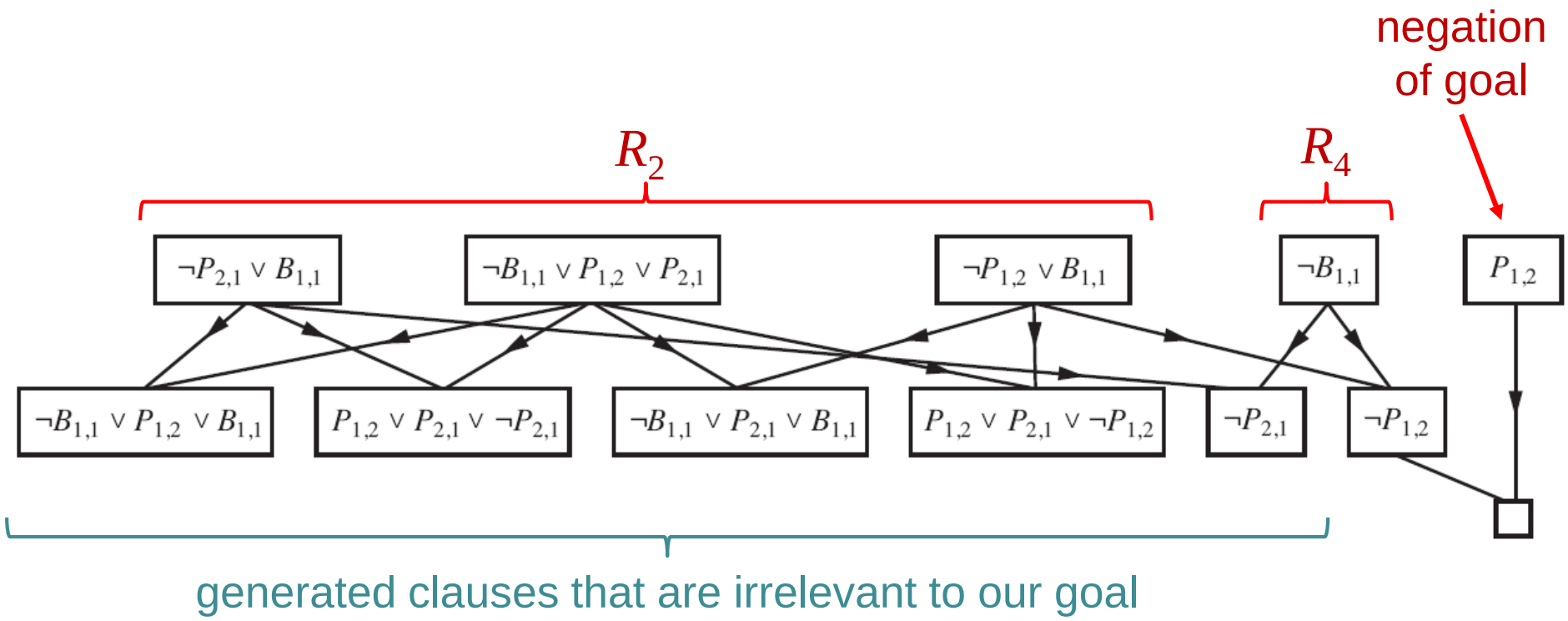
Resolution Algorithm

To prove that the KB entails a sentence α:

- Proof by contradiction: We prove that $(KB \wedge \neg\alpha)$ is **unsatisfiable** (always false).
- We repeatedly combine clause pairs with complementary literals to generate new clauses.
  - If we end up with an empty clause (false), KB entails α.
  - If no new clauses can be added, KB does not entail α.

Example Proof by Resolution

Here we try to see if the wumpus world KB entails $\neg P_{1,2}$.



Horn Clauses

If we limit the types of clauses, we can have more efficient algorithms than the general resolution algorithm.

- Definite clauses: Clauses with exactly one positive literal.
- Goal clauses: Clauses with no positive literal.
- Horn clauses: definite clauses or goal clauses (at most one positive literal).
- Horn clauses are closed under resolution.
- Inference with Horn clauses can be done through forward-chaining or backward-chaining, which have time complexity that is linear in KB size.
- Definite clauses are converted to implications:

$$\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_k \vee Q \quad \equiv \quad (P_1 \wedge P_2 \wedge \ldots \wedge P_k) \Rightarrow Q$$

Forward Chaining (FC)

Data-driven: starting with the known facts

The algorithm uses the following (q: the symbol being querried):

- *agenda*: initially the set of symbols known to be *true*
- *inferred*[p]: initially all *false*
- *count*[c]: the number of symbols in clause c's premise
- In each iteration, a symbol p is popped out of *agenda*. If *inferred*[p] is *false*:
  - If $p = q$, return *true*
  - If p is in c's premise, then decrement *count*[c] by one
  - If *count*[c] becomes zero, add c's conclusion to *agenda*
  - set *inferred*[p] to *true*
- return *false* (q is never reached)

Example of Forward-Chaining

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

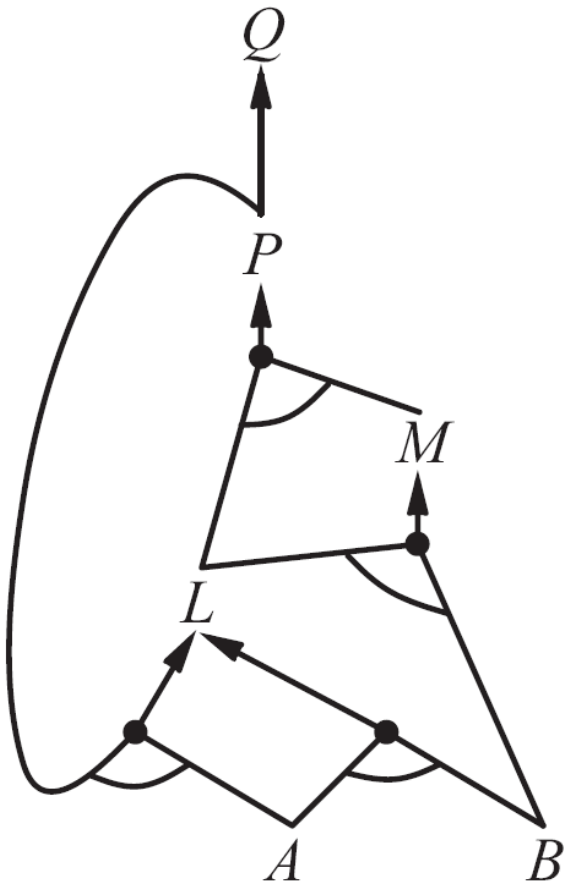
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge B \Rightarrow L$$

$$A$$

$$B$$



This is an AND-OR graph. Multiple links joined by an arc is conjunction (AND) and multiple links joined without an arc is disjunction (OR).

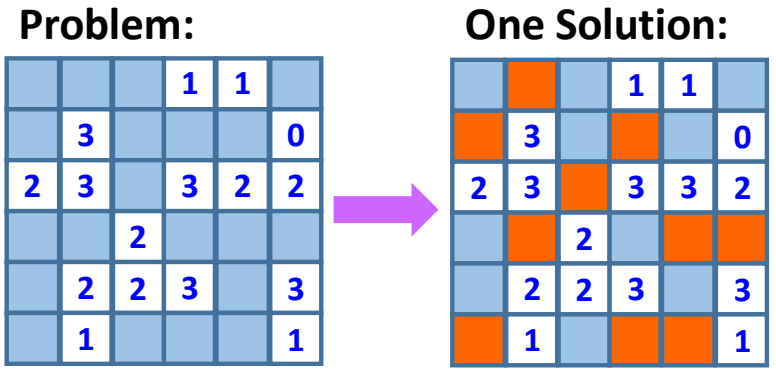
Backward-Chaining (BC)

- Goal-driven: starting with the goal (querried symbol q)
- Working backward: To prove q, find implications in the KB whose conclusion is q.
  - The problem becomes to prove the premises of one of such implications.
  - If the premises of one of such implications are all true, then q is true.
- In practice, BC is much faster than FC because it can skip irrelevant information.
- We will get to the BC algorithm when talking about first-order logic.

SAT Problems

- Given a set of logical variables and a sentence, the SAT (satisfiability) problem concerns finding a set of assignments so that the sentence evaluates to True.
- This is a special type of CSPs.
- Solving SAT problems corresponds to "model checking": Finding models that satisfy the KB.
- Two search approaches used for CSPs are discussed here:
  - Backtracking search.
  - Local search (hill climbing).

An example SAT problem based on Minesweeper:



SAT with Backtracking Search

- The representative algorithm is DPLL (Davis-Putnam-Logemann-Loveland).
- The KB/sentence consists of CNFs.
- Some specialized techniques/heuristics:
  - Unit clause heuristic / unit propagation: Use single-literal clauses (direct assignment) to simplify other clauses. (Similar to forward chaining.)
  - Pure symbol heuristic: Symbols that only appear in one way (as positive or negative literals) can be assigned directly.
  - Early termination: Stop when any clause becomes false.
  - Variable ordering (degree heuristic)
  - Efficient clause indexing (keeping track of clauses that can be affected by a particular assignment)

SAT with Local Search

- Any local search meta-heuristic can be applied.
- Local search for SAT is semi-decidable (unable to prove that no solution exists), but is empirically more efficient than complete search algorithms if a solution exists.
- Search starts from a complete set of random assignments.
- WalkSAT algorithm:
  - In each step, choose a clause that is currently false.
  - Do one of the following with a probability:
    - Flip one symbol in the clause to maximize the number of satisfied clauses (min-conflicts).
    - Flip one symbol in the clause randomly.