Artificial Intelligence

Knowledge
Propositional Logic - Wumpus World

Nguyễn Văn Diêu

2025

Outline I

Outline II

Knowledge-Based Agent

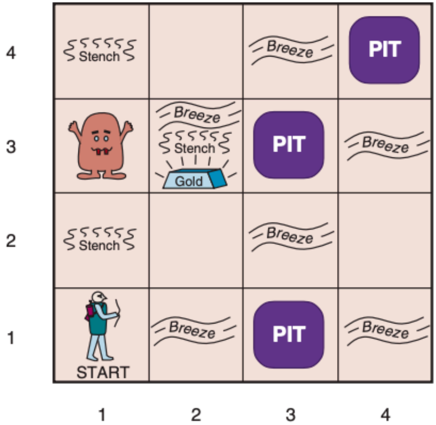
```
function KB-AGENT(percept) returns an action
   persistent: KB, a knowledge base
               t, a counter, initially 0, indicating time

   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
```

Each time the agent program is called, it does three things:

1. **TELL** the knowledge base what it perceives
2. **ASK** the knowledge base what action it should perform. Outcomes of possible action sequences
3. **TELL** the knowledge base which action was chosen. Returns the action so that it can be executed

Wumpus world game



Move around a square board looking for **Gold** while avoiding **Pits** and the **Wumpus**.

Performance Measure

- $+1000$ reward points if the agent comes out of the cave with the gold.
- -1000 points penalty for being eaten by the Wumpus or falling into the pit.
- -1 for each action, and -10 for using an arrow.
- The game ends if either agent dies or came out of the cave.

Environmen

- A 4*4 grid of rooms.
- The agent initially in room square [1, 1], facing toward the right.
- Location of Wumpus and gold are chosen randomly except the first square [1,1].
- Each square of the cave can be a pit with probability 0.2 except the first square.

Actuators

- Left turn.
- Right turn.
- Move forward.
- Grab.
- Release.
- Shoot.

Sensors

- **Stench** if the room adjacent to the **Wumpus**.
- **Breeze** if the room directly adjacent to the **Pit**.
- **Glitter** in the room where the **Gold** is present.
- **Bump** if walks into a **Wall**.
- **Scream** when the Wumpus is **shot** anywhere in the cave.
- There are five element percepts list.

  e.g. if agent perceives Stench, Breeze, but no Glitter, no Bump, and no Scream then it can be represented as:

  [Stench, Breeze, None, None, None].

Wumpus world, first step

(a) (b)

(a) The initial situation, after percept:

[None, None, None, None, None].

(b) After moving to [2,1], perceiving:

[None, Breeze, None, None, None].

Wumpus world, two later stages

(a) (b)

(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].

Logic in general

- **Sentences**: A technical term. It describe the logic.
- **Knowledge bases**: A set of sentences.
- **Syntax**: Syntax of the representation language.
- **Semantics**: Meaning of sentence.
- **Truth**: The semantics of sentence with respect to each possible world.
- **Standard logics**: **true** or **false** - there is no “in between.”

Entailment

Entailment means that one thing follows from another:

$$\alpha \models \beta$$

Sentence α **entails** the sentence β **if and only if**, in every model in which α is **true**, β is also **true**.

e.g., $x + y = 4$ entails $4 = x + y$

Model

- **Model**: Mathematical abstractions. It is Truth value (true or false) for every relevant sentence
- **Satisfaction**: If sentence α is true in model m, saying m satisfies α or sometimes m is a **model** of α
- $M(\alpha)$: Set of all **models** of α

$$\alpha \models \beta \text{ if and only if } M(\alpha) \subseteq M(\beta)$$

Inference

Inference is an algorithm i can derive α from KB, we write:

$$KB \vdash_i \alpha$$

Pronounce: "α is derived from KB by i" or "i derives α from KB"

Soundness: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Propositional Logic

- **Syntax** of propositional logic defines the allowable sentences.
- **Proposition symbol**: P, Q, R, $W_{1,3}$ and $FacingEast$ ...
- **Atomic sentences** consists of a single proposition symbol.
- **Complex sentences** are constructed from simpler sentences, using parentheses and operators called **logical connectives**.
- five **connectives** in common use:
  - $\neg$ (not). **negation** of.
  - $\wedge$ (and). **conjunction**.
  - $\vee$ (or). **disjunction**.
  - $\Rightarrow$ (implies). sometimes written $\supset$ or $\rightarrow$. **rules** or **if - then**
  - $\Leftrightarrow$ (if only if - iff). **biconditional**

BNF (Backus–Naur Form)

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\, \mathit{Sentence} \,) \\
& \mid & \neg\, \mathit{Sentence} \\
& \mid & \mathit{Sentence} \land \mathit{Sentence} \\
& \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
& \mid & \mathit{Sentence} \;\Rightarrow\; \mathit{Sentence} \\
& \mid & \mathit{Sentence} \;\Leftrightarrow\; \mathit{Sentence} \\
\\
\textsc{Operator Precedence} & : & \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

BNF grammar of sentences in propositional logic.

Semantics

The semantics defines the rules for determining the truth of a sentence.

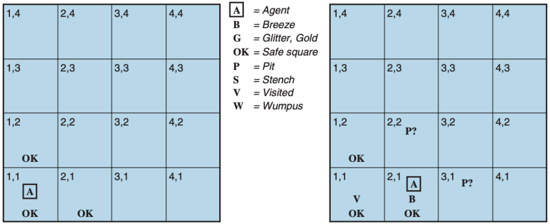
P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Truth tables for the five logical connectives.

In any model m:

- $\neg P$ is *true* iff P is *false* in m.
- $P \wedge Q$ is *true* iff both P and Q are *true* in m.
- $P \vee Q$ is *true* iff either P or Q is *true* in m.
- $P \Rightarrow Q$ is *true* unless P is *true* and Q is *false* in m.
- $P \Leftrightarrow Q$ is *true* iff P and Q are both *true* or both *false* in m.

KB for Wumpus World



(a) (b)

Symbols for each $[x, y]$ location:

- $P_{x,y}$ 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** $[x, y]$.

KB for Wumpus World

We concern 4 squares, it is represent knowledge base for Wumpus World: **[1,2], [2,1], [2,2]** and **[3,1]**.

R_i: Label sentence so that we can refer to them:

- There is no **Pit** in [1,1]:
  $R_1 : \neg P_{1,1}$
- A square is **Breezy** iff there is a **Pit** in a neighboring square:
  $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.
  $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$.
- **Breeze** percepts for the first two squares:
  $R_4 : \neg B_{1,1}$
  $R_5 : B_{2,1}$

KB for Wumpus World

7 symbol: $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, and $P_{3,1}$.
$2^7 = 128$ possible models.

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

Truth table for the knowledge base. KB is true if R_1 - R_5 are true, which occurs in just 3 of the 128 rows.
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]**.

Truth-table algorithm

```
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 ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, { })

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
      if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
      else return true        // when KB is false, always return true
   else
      P ← FIRST(symbols)
      rest ← REST(symbols)
      return (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})
              and
              TT-CHECK-ALL(KB, α, rest, model ∪ {P = false }))
```

TT: Truth-Table algorithm for deciding propositional entailment.
PL-TRUE?: Propositional Logic True?

Propositional Theorem Proving

Logical equivalence: α and β are logically equivalent if they are *true* in the same set of models: $\alpha \equiv \beta$
$\equiv$ is used to make claims about sentences, while
$\Leftrightarrow$ is used as part of a sentence.

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

Validity, **Tautology**:

A sentence is valid if it is *true* **in all models**.

Deduction theorem:

α and β, $\alpha \models \beta$ iff **sentence** $(\alpha \Rightarrow \beta)$ is valid.

Satisfiability: if the sentence is true in, or satisfied by, some model.
Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence.
It is **SAT** problem (Boolean satisfiability problem), NP-complete.

Theorem Proving

Validity and satisfiability are connected:
α is valid iff $\neg\alpha$ is unsatisfiable.
α is satisfiable iff $\neg\alpha$ is not valid.

$\alpha \models \beta$ iff the sentence $(\alpha \land \neg\beta)$ **is unsatisfiable**.

Proof by **contradiction**:

Proving β from α by checking the unsatisfiability of $(\alpha \land \neg\beta)$.

Assumes a sentence β to be false and shows that this leads to a **contradiction** with known axioms α. It is meant the sentence $(\alpha \land \neg\beta)$ is unsatisfiable

Logical equivalences

Two kind of logical operator:

- **Entail, equivalences:** $\models$, $\equiv$
- **Connectives:** $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

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

Inference and proofs

Modus Ponens Give $\alpha \Rightarrow \beta$ and α, then β.

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

e.g. Give $(WumpusAhead \wedge WumpusAlive) \Rightarrow Shoot$
and $(WumpusAhead \wedge WumpusAlive)$, then $Shoot$.

And-Elimination from a $\wedge$ conjunction, any of the conjuncts can be inferred:

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

e.g. Give $(WumpusAhead \wedge WumpusAlive)$, then $WumpusAlive$.
All of the logical equivalences can be used as inference rules.

Inference and proofs

e.g.

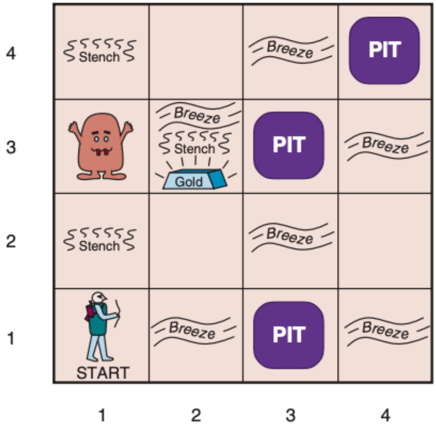
$$\frac{\alpha \Rightarrow \beta}{\neg\alpha \vee \beta}, \quad \frac{\neg\alpha \vee \beta}{\alpha \Rightarrow \beta}, \quad \frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

monotonicity: increase as information.

Any sentences α and β,

if $KB \models \alpha$ **then** $KB \wedge \beta \models \alpha$

Wumpus world



Wumpus world

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

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

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

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

1. "$\Leftrightarrow$" to R_2:
   $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. And-Elimination to R_6:
   $R_7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
3. "$\Rightarrow$" to R_7:
   $R_8 : (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1}))$
4. Modus Ponens with R_8 and the percept R_4 $(\neg B_{1,1})$:
   $R_9 : \neg (P_{1,2} \vee P_{2,1})$
5. De Morgan's rule, giving the conclusion:
   $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$: **Neither [1,2] nor [2,1] contains a Pit.**

Inference and proofs

Searching algorithms can be used to find a sequence of the steps.

Proof problem as follows:

- **INITIAL STATE**: the initial knowledge base.
- **ACTIONS**: the set of actions consists of all the inference rules applied to all the sentences that match the top half of the inference rule.
- **RESULT**: the result of an action is to add the sentence in the bottom half of the inference rule.
- **GOAL**: the goal is a state that contains the sentence we are trying to prove.

Proof by truth-table algorithm: It would be overwhelmed by the exponential explosion of models

clause disjunction of literal

Unit resolution rule

ℓ: literal

ℓ_i and m: Complementary literals (one is the negation of the other)

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \quad m}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k}$$

Full resolution rule

ℓ_i and m_j: Complementary literals

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

another represent:

$$\frac{\{p, r\}, \{\neg p, k\}}{\{r, k\}}$$

e.g.

- agent returns from [2,1] to [1,1] and then goes to [1,2]. Add the following facts to the KB:

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

- absence of pits in [2,2] and [1,3]

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

- 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]

$$R_{15} : P_{1,1} \lor P_{2,2} \lor P_{3,1}$$

e.g.

- a pit in one of [1,1], [2,2], and [3,1]? it's not in [2,2], then it's in [1,1] or [3,1].
- apply resolution rule: $\neg P_{2,2}$ in R_{13} with $P_{2,2}$ in R_{15}:

  $R_{16} : P_{1,1} \vee P_{3,1}$
- a pit in [1,1] or [3,1]? it's not in [1,1], then it's in [3,1].
- apply resolution rule: $\neg P_{1,1}$ in R_1 with $P_{1,1}$ in R_{16}:

  $R_{17} : P_{3,1}$: a pit in [3,1]

Conjunctive normal form

Every sentence of propositional logic is logically equivalent to a conjunction of clauses.

Grammar for conjunctive normal form, Horn clauses, and definite clauses:

$$
\begin{array}{rcl}
\mathit{CNFSentence} & \rightarrow & \mathit{Clause}_1 \land \cdots \land \mathit{Clause}_n \\
\mathit{Clause} & \rightarrow & \mathit{Literal}_1 \lor \cdots \lor \mathit{Literal}_m \\
\mathit{Fact} & \rightarrow & \mathit{Symbol} \\
\mathit{Literal} & \rightarrow & \mathit{Symbol} \mid \neg \mathit{Symbol} \\
\mathit{Symbol} & \rightarrow & P \mid Q \mid R \mid \ldots \\
\mathit{HornClauseForm} & \rightarrow & \mathit{DefiniteClauseForm} \mid \mathit{GoalClauseForm} \\
\mathit{DefiniteClauseForm} & \rightarrow & \mathit{Fact} \mid (\mathit{Symbol}_1 \land \cdots \land \mathit{Symbol}_l) \Rightarrow \mathit{Symbol} \\
\mathit{GoalClauseForm} & \rightarrow & (\mathit{Symbol}_1 \land \cdots \land \mathit{Symbol}_l) \Rightarrow \mathit{False}
\end{array}
$$

Converting to CNF

e.g. $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ converting to **CNF**

The steps are as follows:

1. 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. 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 using:
   $\neg(\neg\alpha) \equiv \alpha$ (double-negation)
   $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ (De Morgan)
   $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ (De Morgan)
   the e.g. result: $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4. Apply the distributivity law $\wedge$ and $\vee$
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$
   Sentence is now in **CNF**, as a conjunction of three clauses

Resolution algorithm

- Using the principle of proof by contradiction
- To show $KB \models \alpha$, we show that $(KB \land \neg\alpha)$ is **unsatisfiable**

Algorithm $(\textit{show } KB \models \alpha)$

1. Convert $(KB \land \neg\alpha)$ into **CNF**
2. Apply resolution rule to the resulting clauses
   - Each pair that contains complementary literals is resolved to produce a new clause
   - added to the set if it is not already present
3. Loop step 2 until one of two things happens:
   - **No new clauses that can be added**: $KB \not\models \alpha$ or,
   - 2 clauses resolve to yield the **empty clause**: $KB \models \alpha$.

Resolution algorithm

PL-Resolve returns the set of all possible clauses obtained by resolving its two inputs.

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 ← the set of clauses in the CNF representation of $KB \land \neg\alpha$
  new ← { }
  while *true* **do**
    for each pair of clauses C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

e.g.

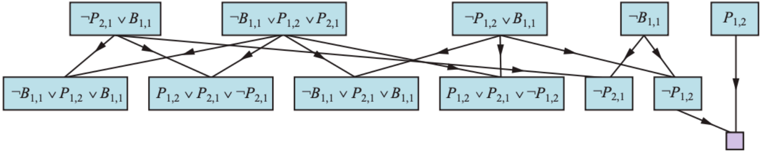
e.g. Agent is in [1,1], no breeze, so no pits in neighboring squares:

$$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) \qquad R_4 : \neg B_{1,1}$$

- $KB = R_2 \wedge R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- Prove $\alpha = \neg P_{1,2}$
- Convert $(KB \wedge \neg\alpha)$ into **CNF**

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

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



Query $\neg P_{1,2}$, PL-RESOLUTION yield the **empty clause**.
So that the query is proven.

e.g. Resolution alg.

Consider the following six statements, all of which to be true:

1. If you go swimming you will get wet.
2. If it is raining and you are outside then you will get wet.
3. If it is warm and there is no rain then it is a pleasant day.
4. You are not wet.
5. You are outside.
6. It is a warm day.

Determine following statements must be true:

a. You are not swimming.
b. It is not raining.
c. It is a pleasant day.

e.g. Resolution alg.

In propositional logic:

1. $swimming \Rightarrow wet$
2. $(rain \wedge outside) \Rightarrow wet$
3. $(warm \wedge \neg rain) \Rightarrow pleasant$
4. $\neg wet$
5. $outside$
6. $warm$

Statements:

a. $\neg swimming$

b. $\neg rain$

c. $pleasant$

e.g. Resolution alg.

a) $\{1, 2, 3, 4, 5, 6\} \models \neg swimming$

Contradition method and convert to CNF:

1. $\neg swimming \vee wet$
2. $\neg rain \vee \neg outside \vee wet$
3. $\neg warm \vee rain \vee pleasant$
4. $\neg wet$
5. $outside$
6. $warm$
7. $swimming$

exercises:

b. $\neg rain$

PL-Resolution:

8. $(1) \wedge (4) : \neg swimming$
9. $(7) \wedge (8) : \blacksquare$

$\neg swimming$

e.g. The Power of False

$P \wedge \neg P \models Z$

We know: $P \wedge \neg P \equiv false$

$P \wedge \neg P \wedge \neg Z$

PL-Resolution:

$P \wedge \neg P : \blacksquare$

Z

Horn clauses and definite clauses

- **Definite clause**: **disjunction** of literals of which **exactly one is positive**.

  e.g. $(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1})$ is a definite clause,

  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$ is not, because it has two positive clauses
- **Horn clause**: **disjunction** of literals of which **at most one is positive**.

  all definite clauses are Horn clauses.

  KB containing only definite clauses are interesting:

  1. Every definite clause can be written as an implication.

  e.g. $(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1})$ can be written as the implication
  $(L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$

Horn clauses and definite clauses

- **Horn clause**

  premise is call **body**

  conclusion is call **head**

  sentence consisting of a single positive literal is call **fact**

  e.g. $L_{1,1}$

  fact $L_{1,1}$ can be written in implication form: $True \Rightarrow L_{1,1}$

  2. Inference with Horn clauses can be done through the **forward-chaining** and **Backward-chaining** algorithms

  3. Entailment with Horn clauses in linear time with the size of knowledge base

Horn clauses and definite clauses

- **Goal clauses**: clauses with no positive literals.
- **Modus Ponens** (for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, ..., \alpha_n \Rightarrow \beta, \quad \alpha_1, ..., \alpha_n}{\beta}$$

FW and BW Chaining

Inference Engine

Apply **rules** of **KB** to **infer** new information.

Modus Ponens

e.g.

A	It is raining
$A \Rightarrow B$	if it is rainning i will carry an umbrella
B	I will carry an umbrellaq (new knowledge)

- **Forward Chaining**: Start with atomic sentences in the KB and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.
- **Backward Chaining**: Starts with the goal and works backward, chaining through rules to find known facts that support the goal.

FW and BW Chaining

e.g.

Forward Chaining

A	He exercises regularly.
$A \Rightarrow B$	if he is exercising regularly, he is fit.
B	He is fit.

Backward Chaining

B	He is fit.
$A \Rightarrow B$	if he is exercising regularly, he is fit.
A	He exercises regularly.

Forward 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
```

The forward-chaining algorithm for propositional logic

- **Forward chaining is sound and complete for *Horn KB***

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Queue
0		A B

Inferred	
A	false
B	false
L	false
M	false
P	false
Q	false

Count	
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	2
$B \wedge L \Rightarrow M$	2
$A \wedge P \Rightarrow L$	2
$A \wedge B \Rightarrow L$	2

e.g. Forward chaining

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$. A, B

No.	Reached	Queue
0		A B
1	A	B

Inferred	
A	**true**
B	false
L	false
M	false
P	false
Q	false

Count	
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	2
$B \wedge L \Rightarrow M$	2
$A \wedge P \Rightarrow L$	**1**
$A \wedge B \Rightarrow L$	**1**

e.g. Forward chaining

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Queue
0		A B
1	A	B
2	B	**L**

Inferred	
A	true
B	**true**
L	false
M	false
P	false
Q	false

Count	
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	2
$B \wedge L \Rightarrow M$	**1**
$A \wedge P \Rightarrow L$	1
$A \wedge B \Rightarrow L$	**0**

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L. \quad A, \quad B$

No.	Reached	Queue
0		A B
1	A	B
2	B	L
3	L	**M**

Inferred	
A	true
B	true
L	**true**
M	false
P	false
Q	false

Count	
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	**1**
$B \wedge L \Rightarrow M$	**0**
$A \wedge P \Rightarrow L$	1
$A \wedge B \Rightarrow L$	0

e.g. Forward chaining

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$. A, B

No.	Reached	Queue
0		A B
1	A	B
2	B	L
3	L	M
4	M	**P**

Inferred	
A	true
B	true
L	true
M	**true**
P	false
Q	false

Count	
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	**0**
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	1
$A \wedge B \Rightarrow L$	0

e.g. Forward chaining

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Queue
0		A B
1	A	B
2	B	L
3	L	M
4	M	P
5	P	**L Q**

Inferred	
A	true
B	true
L	true
M	true
P	**true**
Q	false

Count	
$P \Rightarrow Q$	**0**
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	**0**
$A \wedge B \Rightarrow L$	0

e.g. Forward chaining

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Queue
0		A B
1	A	B
2	B	L
3	L	M
4	M	P
5	P	L Q
6	L	Q

Inferred	
A	true
B	true
L	true
M	true
P	true
Q	false

Count	
$P \Rightarrow Q$	0
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	0
$A \wedge B \Rightarrow L$	0

e.g. Forward chaining

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L, \quad A \wedge B \Rightarrow L, \quad A, \quad B$

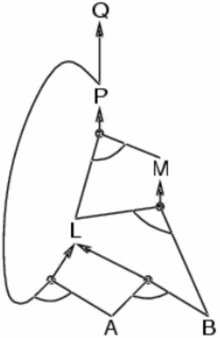
No.	Reached	Queue
0		A B
1	A	B
2	B	L
3	L	M
4	M	P
5	P	L Q
6	L	Q
7	**Q**	

Inferred	
A	true
B	true
L	true
M	true
P	true
Q	false

Count	
$P \Rightarrow Q$	0
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	0
$A \wedge B \Rightarrow L$	0

e.g. Forward chaining

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



Backward chaining

Idea:

Check whether a particular fact Q is *true*

Backward chaining

Given a fact Q to be "proven",

1. See if Q is already in the **KB**. If so, return *true*.
2. Find all implications, I, whose conclusion "matches" Q.
3. Recursively establish the premises of all i in I via backward chaining.

e.g. Backward chaining

Goal Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Goal stack
0		Q

Inferred	
A	false
B	false
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$

Inferred	
A	false
B	false
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$ ✔

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$
2	$P \Rightarrow Q$	$L \wedge M \Rightarrow P$

Inferred	
A	false
B	false
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$ ✔

$L \wedge M \Rightarrow P$ ✔

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$
2	$P \Rightarrow Q$	$L \wedge M \Rightarrow P$
3	$L \wedge M \Rightarrow P$	$A \wedge P \Rightarrow L, \quad B \wedge L \Rightarrow M$

Inferred	
A	false
B	false
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$ ✔

$L \wedge M \Rightarrow P$ ✔

$B \wedge L \Rightarrow M$ ✔

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L, \quad A, \quad B$

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$
2	$P \Rightarrow Q$	$L \wedge M \Rightarrow P$
3	$L \wedge M \Rightarrow P$	$A \wedge P \Rightarrow L, \quad B \wedge L \Rightarrow M$
4	$B \wedge L \Rightarrow M$	$A \wedge P \Rightarrow L, \quad B, \quad A \wedge B \Rightarrow L$

Inferred	
A	false
B	false
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$ ✔

$L \wedge M \Rightarrow P$ ✔

$B \wedge L \Rightarrow M$ ✔

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$ ✔, A, B

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$
2	$P \Rightarrow Q$	$L \wedge M \Rightarrow P$
3	$L \wedge M \Rightarrow P$	$A \wedge P \Rightarrow L$, $A \wedge B \Rightarrow L$
4	$B \wedge L \Rightarrow M$	$A \wedge P \Rightarrow L$ B, $A \wedge B \Rightarrow L$
5	$A \wedge B \Rightarrow L$	$A \wedge P \Rightarrow L$, B, A

Inferred	
A	false
B	false
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$ ✔

$L \wedge M \Rightarrow P$ ✔

$B \wedge L \Rightarrow M$ ✔

$A \wedge P \Rightarrow L$, $A \wedge B \Rightarrow L$ ✔, A ✔, B

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$
2	$P \Rightarrow Q$	$L \wedge M \Rightarrow P$
3	$L \wedge M \Rightarrow P$	$A \wedge P \Rightarrow L,\ A \wedge B \Rightarrow L$
4	$B \wedge L \Rightarrow M$	$A \wedge P \Rightarrow L$
5	$A \wedge B \Rightarrow L$	$A \wedge P \Rightarrow L,\ B,\ A$
6	A	$A \wedge P \Rightarrow L,\ B$

Inferred	
A	**true**
B	false
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$ ✔

$L \wedge M \Rightarrow P$ ✔

$B \wedge L \Rightarrow M$ ✔

$A \wedge P \Rightarrow L, \quad A \wedge B \Rightarrow L$ ✔, A ✔, B ✔

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$
2	$P \Rightarrow Q$	$L \wedge M \Rightarrow P$
3	$L \wedge M \Rightarrow P$	$A \wedge P \Rightarrow L, \quad A \wedge B \Rightarrow L$
4	$B \wedge L \Rightarrow M$	$A \wedge P \Rightarrow L$
5	$A \wedge B \Rightarrow L$	$A \wedge P \Rightarrow L, \quad B, \quad A$
6	A	$A \wedge P \Rightarrow L, \quad B$
7	B	$A \wedge P \Rightarrow L$

Inferred	
A	true
B	**true**
L	false
M	false
P	false
Q	false

e.g. Backward chaining

Goal Q ✔

$P \Rightarrow Q$ ✔

$L \wedge M \Rightarrow P$ ✔

$B \wedge L \Rightarrow M$ ✔, $A \wedge P \Rightarrow L$ ✔, $A \wedge B \Rightarrow L$ ✔, A ✔, B ✔

No.	Reached	Goal stack
0		Q
1	Q	$P \Rightarrow Q$
2	$P \Rightarrow Q$	$L \wedge M \Rightarrow P$
3	$L \wedge M \Rightarrow P$	$A \wedge P \Rightarrow L,\ A \wedge B \Rightarrow L$
4	$B \wedge L \Rightarrow M$	$A \wedge P \Rightarrow L$
5	$A \wedge B \Rightarrow L$	$A \wedge P \Rightarrow L,\ B,\ A$
6	A	$A \wedge P \Rightarrow L,\ B$
7	B	$A \wedge P \Rightarrow L$
8	$A \wedge P \Rightarrow L$	$\{\ \}$

Inferred	
A	true
B	true
L	**true**
M	**true**
P	**true**
Q	**true**