Logic

Chen-Yu Wei

Wumpus World

Performance

Gold +1000, death -1000, -1 per step, -10 for using the arrow

Environment

Perceive stench if adjacent to wumpus

Perceive breeze if adjacent to pit

Perceive glitter if in the square of gold

Can grab gold if in the square of gold

Can shoot and kill wumpus if you're facing it (shooting uses up the only arrow)

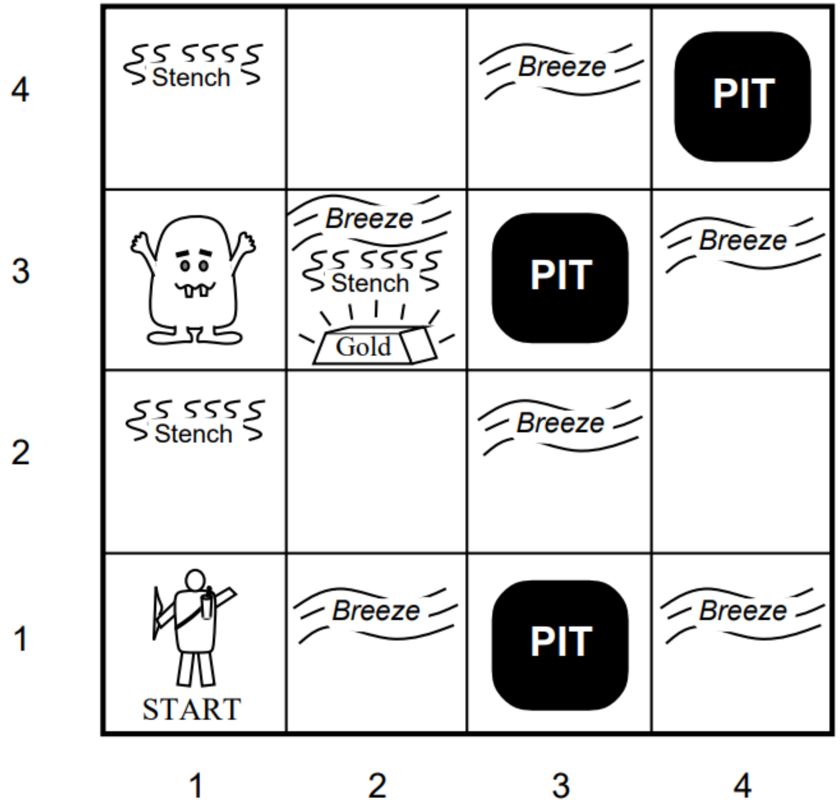
Die if entering a square with pit or living wumpus

Actions

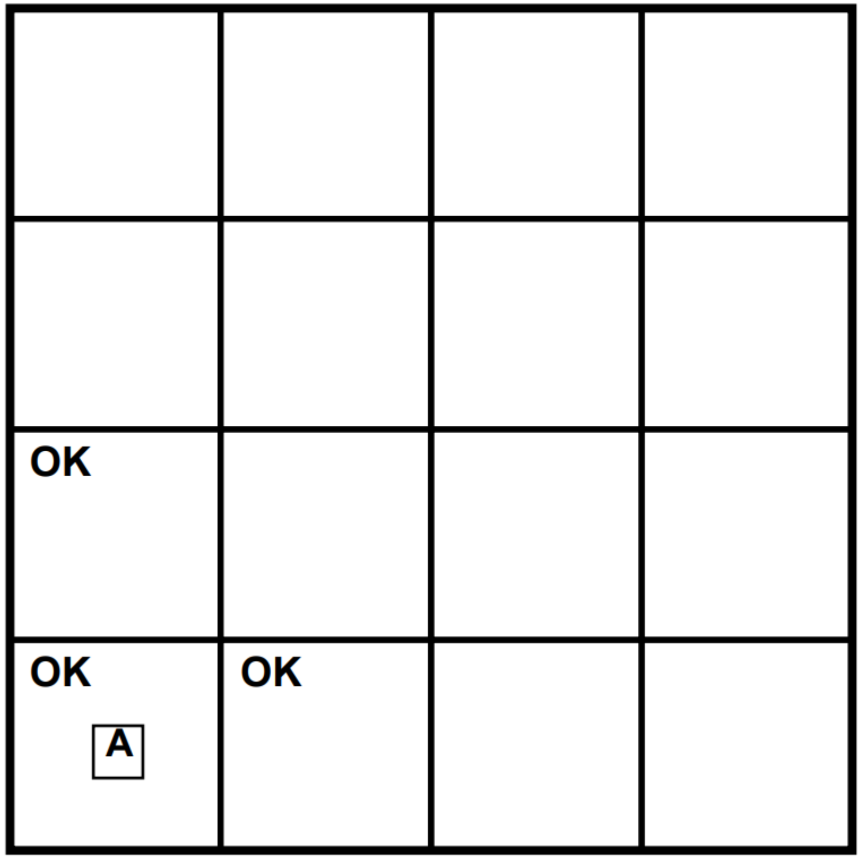
Left turn, right turn, forward, grab, shoot

Sensors

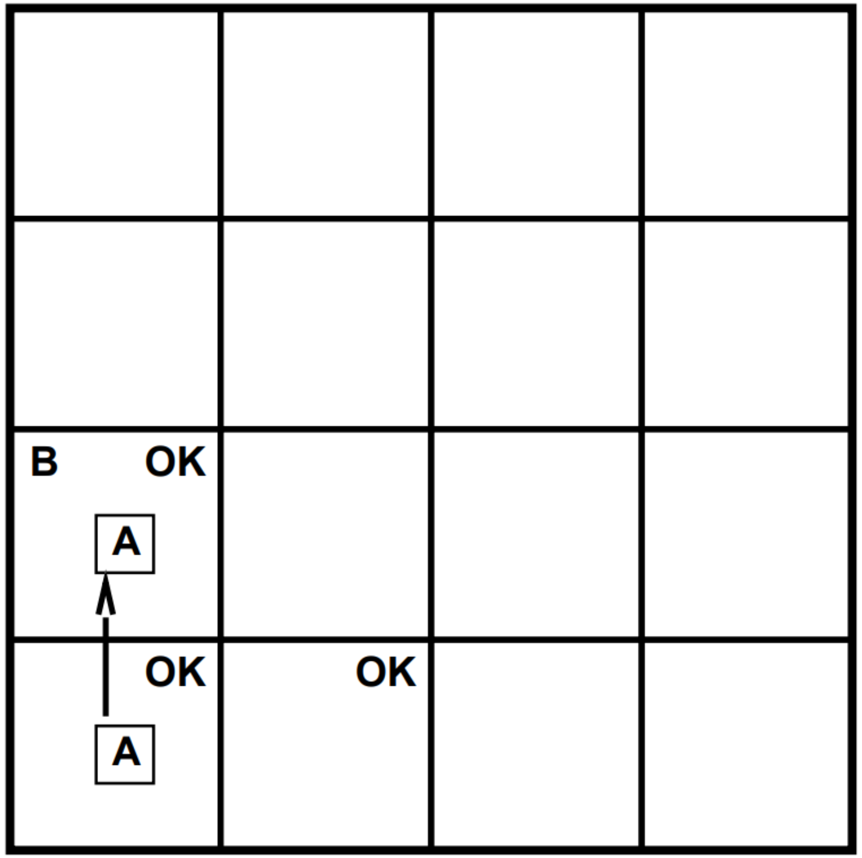
Breeze, glitter, smell



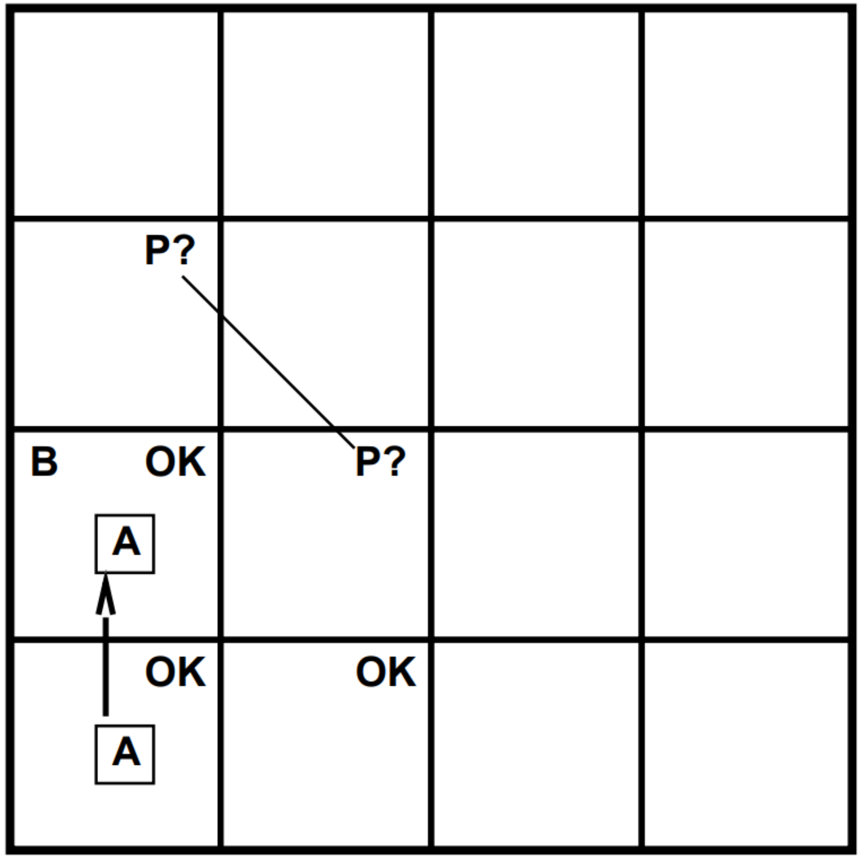
Exploring a wumpus world



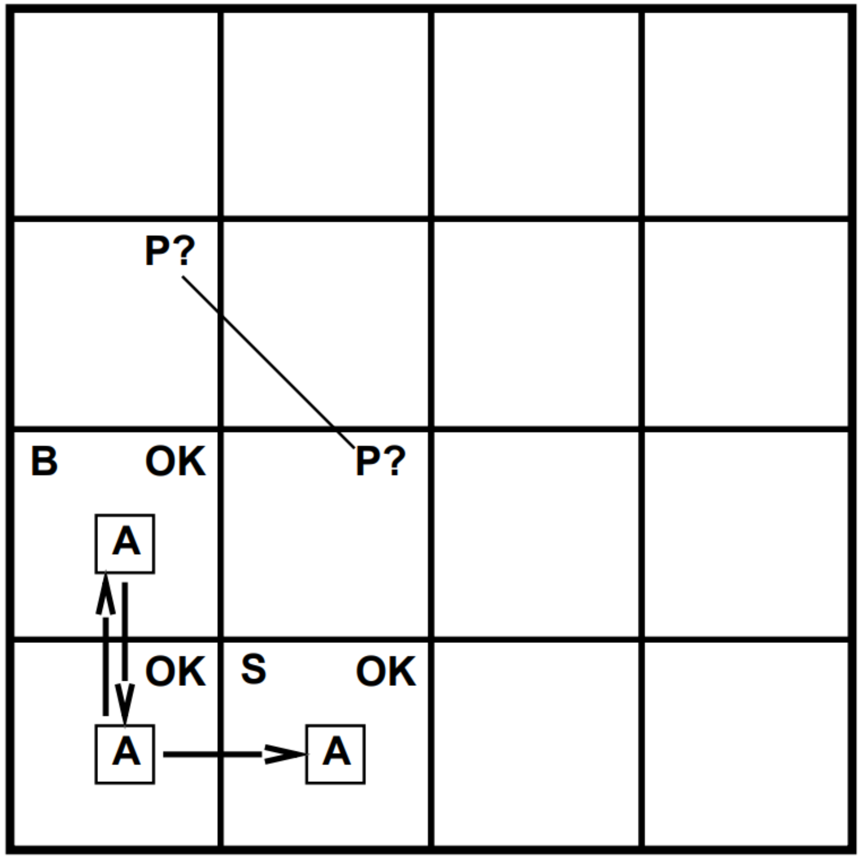
Exploring a wumpus world



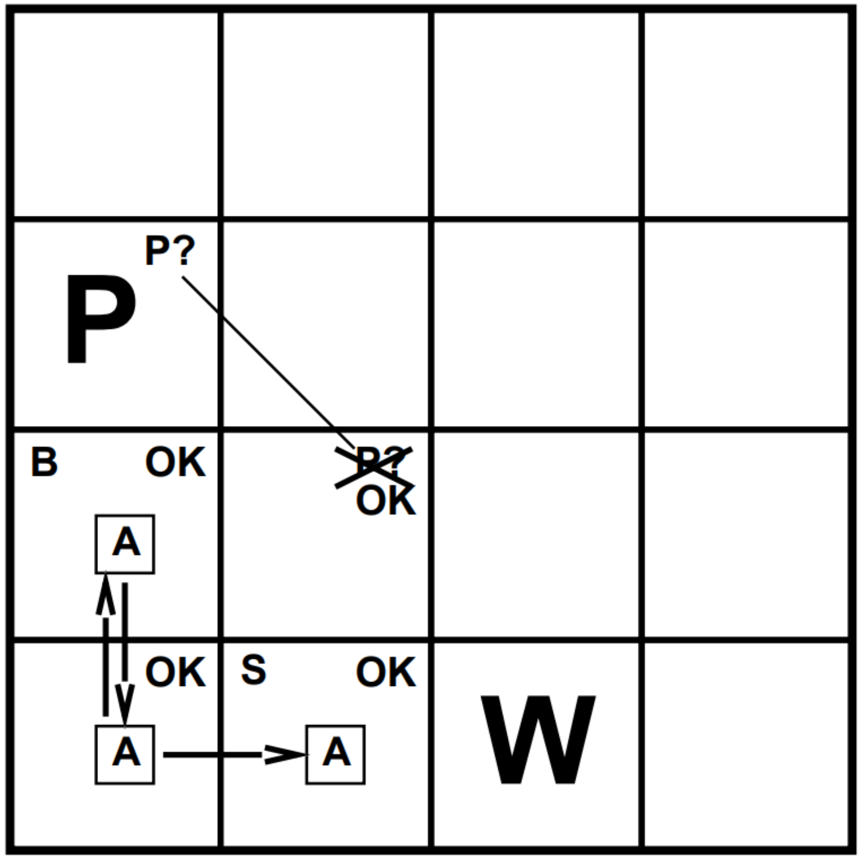
Exploring a wumpus world



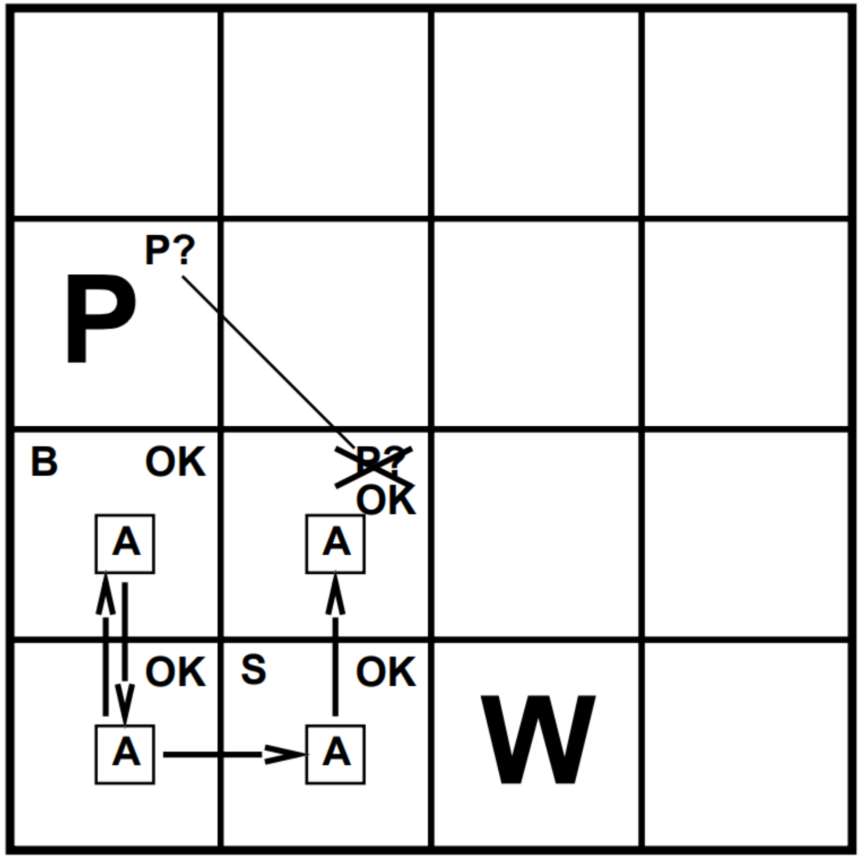
Exploring a wumpus world



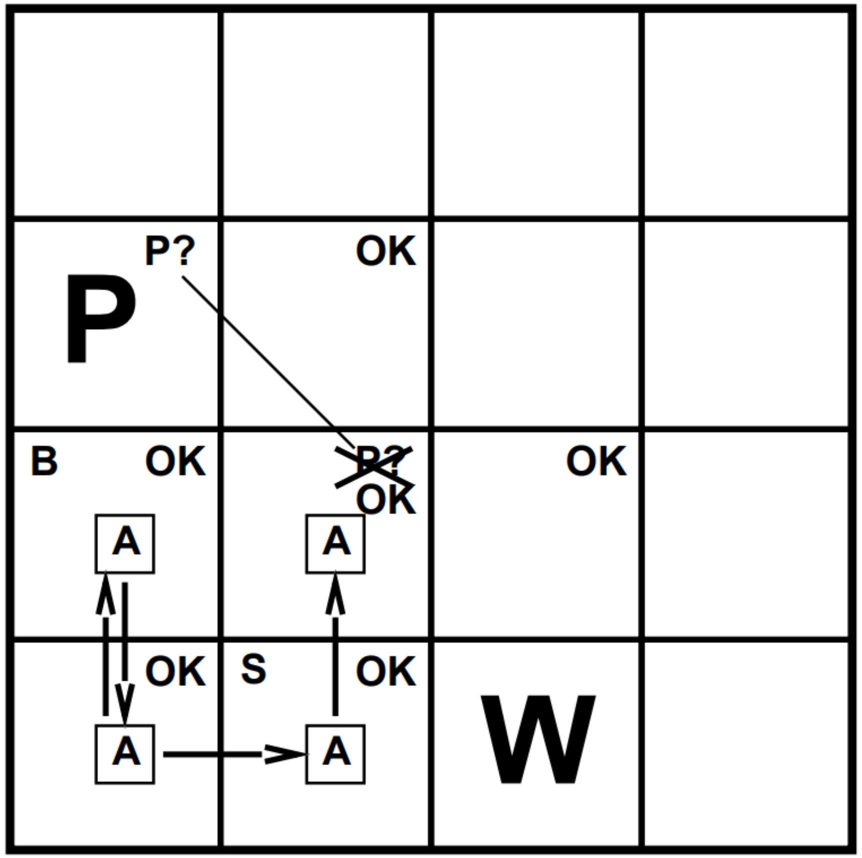
Exploring a wumpus world



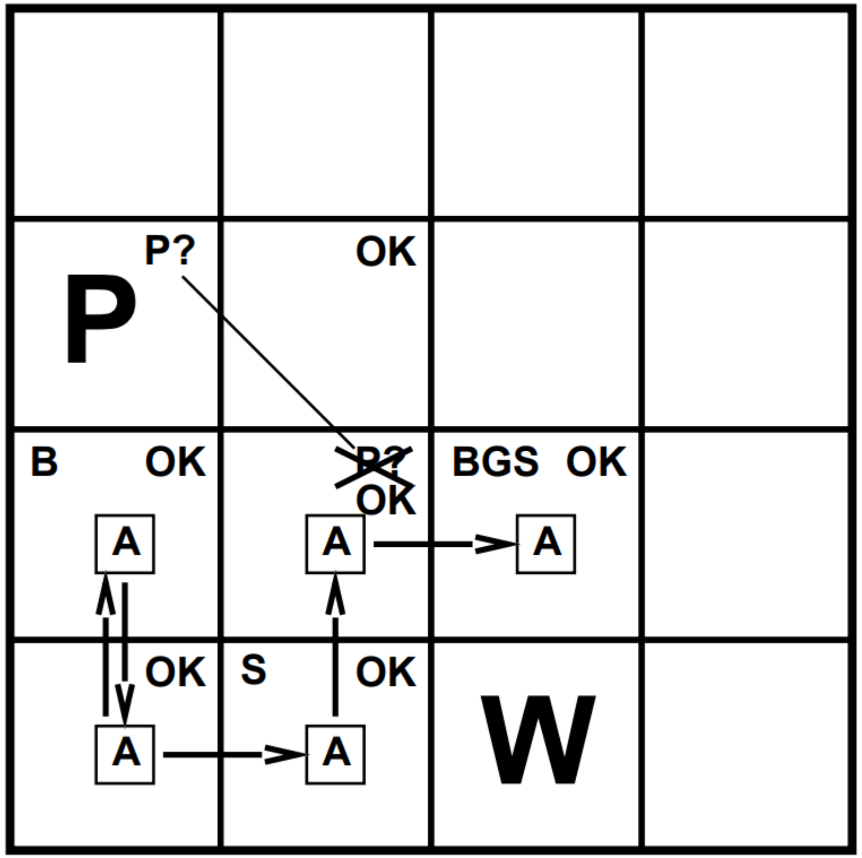
Exploring a wumpus world



Exploring a wumpus world



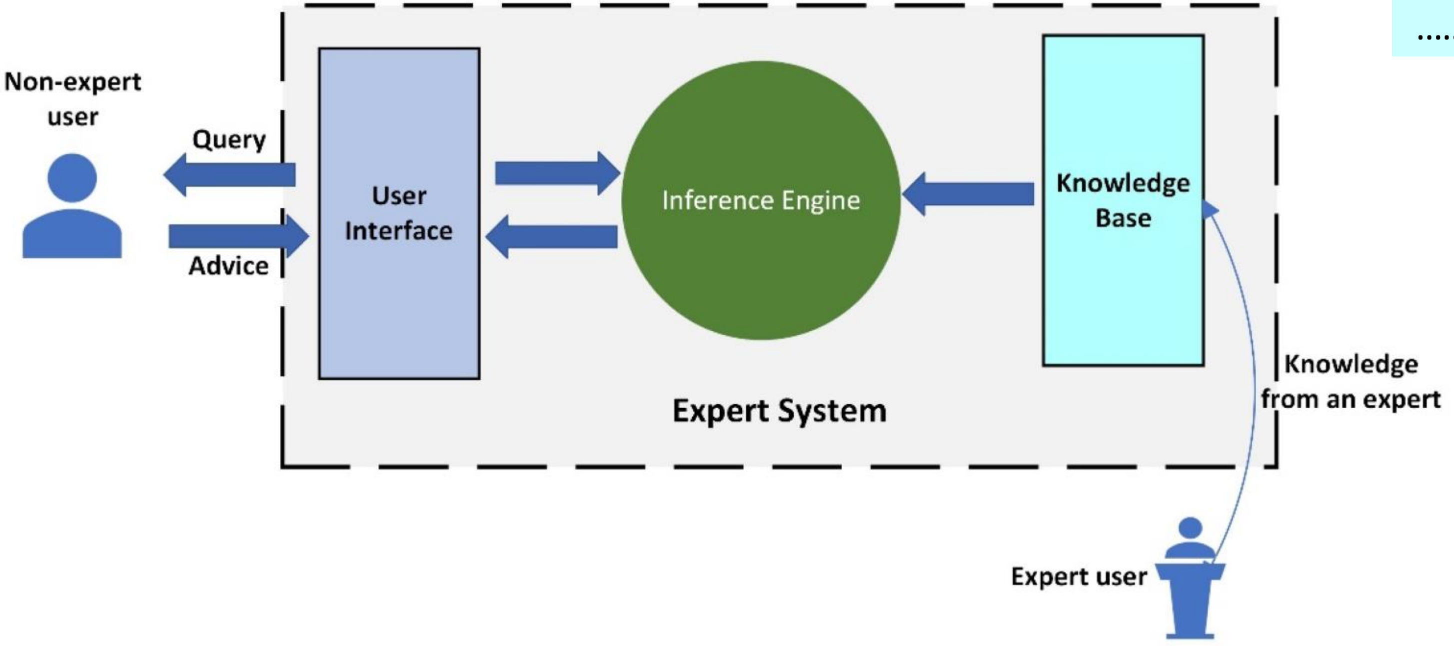
Exploring a wumpus world



Systems with Logical Reasoning

- Knowledge base
  - Consists of some prior knowledge
- Inference engine
  - Derive new knowledge or make some claims
- User Interaction
  - **Tell** information
  - **Ask** question

Example: Expert System



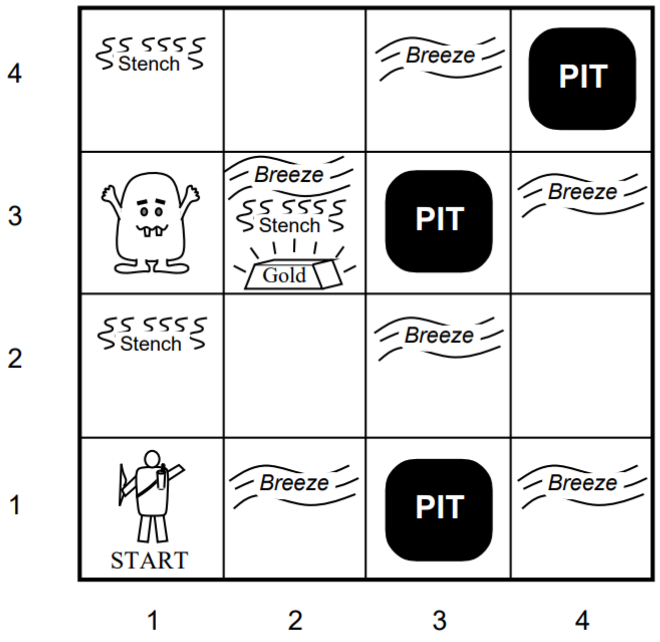
Knowledge base

If **has_hair**, then **mammal**.
If **mammal** and **has_hooves**, then **ungulate**.
If **has_feathers**, then **bird**.
If **mammal** and **carnivore** and **has_dark_spots**, then **cheetah**.
If **mammal** and **carnivore** and **has_black_stripes**, then **tiger**.
If **bird** and **does_not_fly** and **has_long_neck**, then **ostrich**.
……

User interaction

```
File  Edit  Settings  Run  Debug  Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.6)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- go.
Does the animal have hair? yes.

Does the animal eat meat? |: no.

Does the animal have pointed teeth? |: no.

Does the animal have hooves? |: yes.

Does the animal have a long neck? |: yes.

Does the animal have long legs? |: yes.

I guess that the animal is: giraffe
true.

?-
```

Example: wumpus world



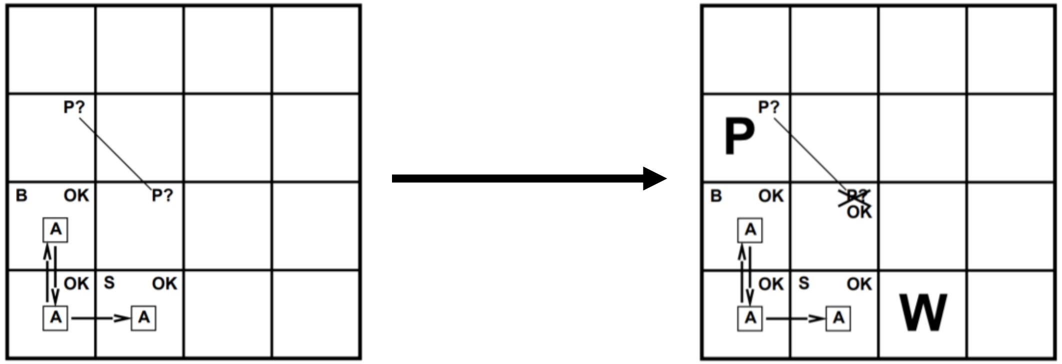
Knowledge base

- Perceive stench if adjacent to wumpus
- Perceive breeze if adjacent to pit
- Perceive glitter if in the square of gold
- …

User interaction

- Tell the logic system whether stench, breeze, glitter is perceived
- Ask for the next action

Inference Engine



Ingredients of Propositional Logic

Sentence

Knowledge base consists of "sentences"

Inference algorithm derives new "sentences" and add them to the knowledge base

Example:

KB = { "Rain→Wet", "Rain" }

Inference algorithm derives a new sentence "Wet" based on KB

Now KB becomes

KB = {"Rain→Wet", "Rain", "Wet" }

Ingredients of Logic – Syntax

Define what are valid sentences.

E.g., syntax in **python**:

“ for x in range(10): ” Valid

“ for x range(10): ” Invalid (the python interpreter cannot understand)

E.g. syntax in **math**:

“ x + y = 5” Valid

“ x 5 = y + ” Invalid

Ingredients of Logic – Syntax

Syntax in **propositional logic**:

- A proposition symbols X is a sentence
  (a propositional symbol is a Boolean variable)
- If α is a sentence then $\neg\alpha$ is a sentence
- If α and β are sentences then $\alpha \wedge \beta$ is a sentence
- If α and β are sentences then $\alpha \vee \beta$ is a sentence
- If α and β are sentences then $\alpha \Rightarrow \beta$ is a sentence
- If α and β are sentences then $\alpha \Leftrightarrow \beta$ is a sentence

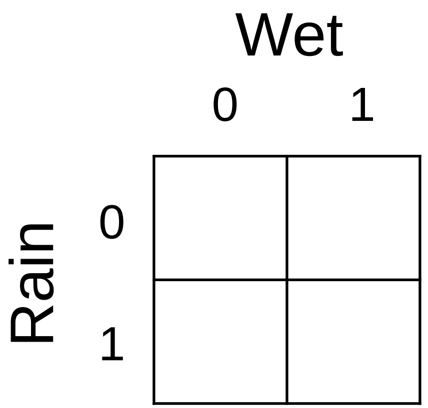
The $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ symbols have no meaning here. Their meanings are specified by the "semantics" of logic (discussed next).

Ingredients of Logic – Semantics

Let’s first define “models”. A model is a configuration of the world.

In propositional logic, a model is an **assignment of truth values** to propositional symbols.

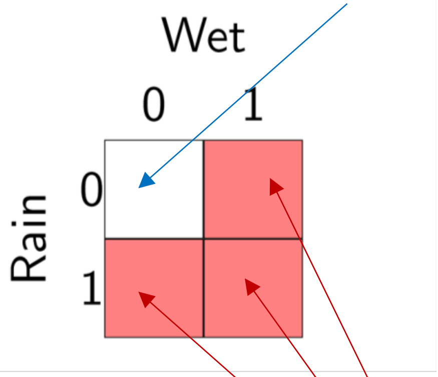
E.g., There are four possible models in the raining example:



Ingredients of Logic – Semantics

$$f = \mathsf{Rain} \vee \mathsf{Wet}$$

models where the sentence f is false



P	Q	(P v Q)
T	T	T
T	F	T
F	T	T
F	F	F

models where the sentence f is true

Ingredients of Logic – Semantics

P	~P
T	F
F	T

P	Q	(P ^ Q)
T	T	T
T	F	F
F	T	F
F	F	F

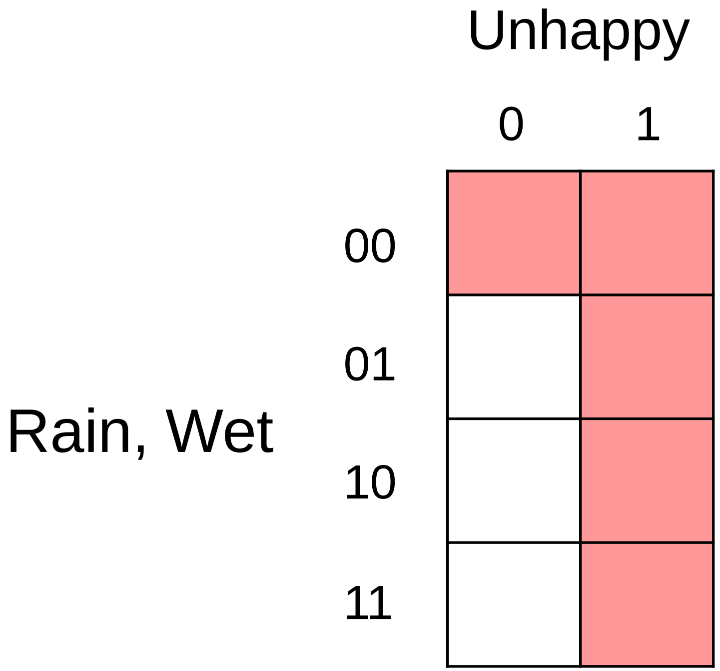
P	Q	(P v Q)
T	T	T
T	F	T
F	T	T
F	F	F

P	Q	(P =>Q)
T	T	T
T	F	F
F	T	T
F	F	T

P	Q	(P ⇔Q)
T	T	T
T	F	F
F	T	F
F	F	T

Ingredients of Logic – Semantics

f: (Rain ∨ Wet) ⇒ Unhappy

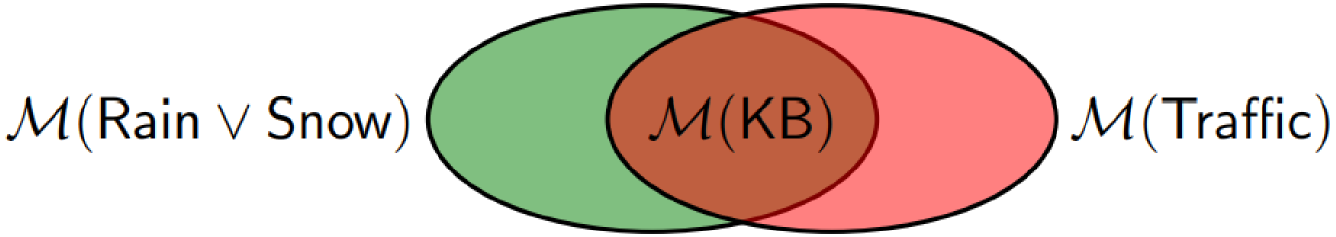


$\mathcal{M}(f)$: the set of models where sentence f is true.

Ingredients of Logic – Knowledge Base

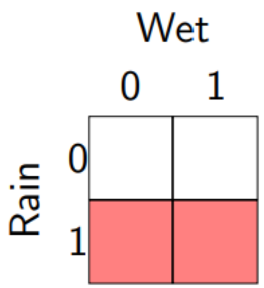
Knowledge base = a collection of sentences

Let $\mathrm{KB} = \{\mathrm{Rain} \vee \mathrm{Snow}, \mathrm{Traffic}\}$.

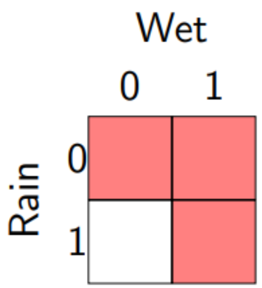


Ingredients of Logic – Knowledge Base

$\mathcal{M}(\text{Rain})$

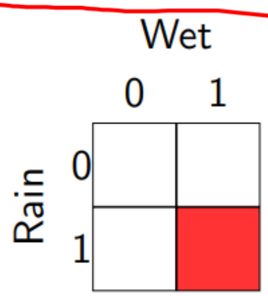


$\mathcal{M}(\text{Rain} \rightarrow \text{Wet})$



KB

$\mathcal{M}(\{\text{Rain}, \text{Rain} \rightarrow \text{Wet}\})$



$\alpha = \text{Wet}$

Adding more formulas to the knowledge base:

$$\text{KB} \longrightarrow \text{KB} \cup \{f\}$$

Shrinks the set of models:

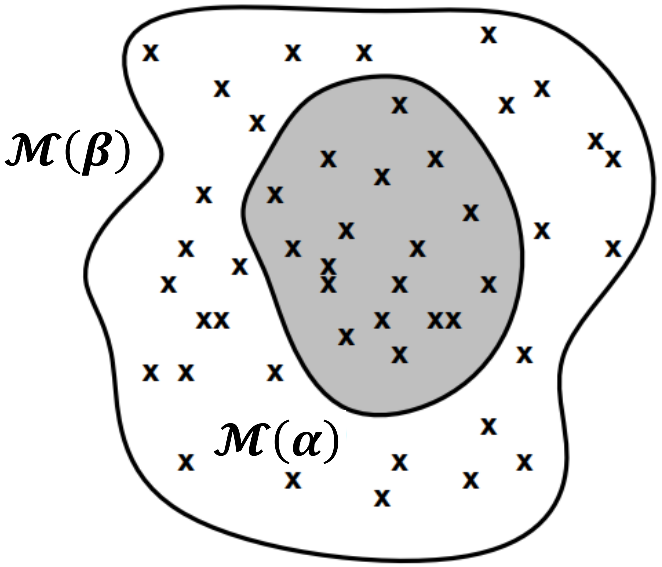
$$\mathcal{M}(\text{KB}) \longrightarrow \mathcal{M}(\text{KB}) \cap \mathcal{M}(f)$$

Recap: Propositional Logic

- **Sentence:** propositional symbols, or their negations ($\neg$), or their combinations through $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$.
- **Models:** An assignment of truth values to propositional symbols.
- **Knowledge base:** a set of sentences
- $\mathcal{M}(f)$: the set of models where sentence f is true.

Entailment

- Sentence α **entails** sentence β means that (in high level) sentence β follows logically from sentence α
- Denoted as $\alpha \vDash \beta$
- $\alpha \vDash \beta$ if and only if $\mathcal{M}(\alpha) \subset \mathcal{M}(\beta)$
- **Example:** Rain ∧ Snow ⊨ Snow



Inference Algorithms

- Given KB and α, the algorithm tries to derive sentence α.
- If an algorithm $\mathcal{A}$ is able to derive α from KB, we write KB $\vdash_{\mathcal{A}} \alpha$
  - This is different from KB $\vDash \alpha$,
- Soundness (correctness)
  - The algorithm can only derive α when α is entailed by KB.
  - In other words: If KB $\vdash_{\mathcal{A}} \alpha$, then KB $\vDash \alpha$
- Completeness
  - For any α that KB entails, the algorithm is able to derive α.
  - If other words: If KB $\vDash \alpha$, then if KB $\vdash_{\mathcal{A}} \alpha$

A (Simple) Inference Algorithm: Model Checking

function TT-ENTAILS?(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  $symbols \leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL($KB, \alpha, symbols, \{\ \}$)

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true*   // when KB is false, always return true
  else
    $P \leftarrow$ FIRST($symbols$)
    $rest \leftarrow$ REST($symbols$)
    return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)
        and
        TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))

A (Simple) Inference Algorithm: Model Checking

Model Checking (KB, α):

Let $\mathcal{M}$ be the set of all possible models
($|\mathcal{M}| = 2^N$ if there are N propositional symbols in KB $\cup \{\alpha\}$)

For $m \in \mathcal{M}$:

If KB is True in m and α is False in m : **return** False

return True

Theorem Proving

Idea: Instead of checking all models, will just perform manipulations on the sentence level.

Inference Rules

$\alpha \to \beta \equiv \neg \alpha \vee \beta$

- Modus Ponens (Latin for *mode the affirms*)

$\neg(\alpha_1 \wedge \cdots \wedge \alpha_k)$

$\equiv (\neg \alpha_1) \vee (\quad) \cdots \vee (\neg \alpha_k)$

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_k, \quad (\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_k) \Rightarrow \beta}{\beta}$$

← premises

← conclusion

$\parallel$

or $$\frac{\alpha_1, \alpha_2, \ldots, \alpha_k, \quad (\neg\alpha_1 \vee \neg\alpha_2 \vee \cdots \vee \neg\alpha_k \vee \beta)}{\beta}$$

- And Eliminations

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_k}{\alpha_i}$$

Standard Logical Equivalence

(can be applied in any steps in the inference algorithm)

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

Inference Rules

Example: KB = {Rain ⇒ Wet, Wet ⇒ Unhappy, Rain}, α = Unhappy.

Applying Modus Ponens on KB
(i.e., try to **match** sentences in KB with premises α and β)

$$\frac{\text{Rain, Rain} \Rightarrow \text{Wet}}{\text{Wet}}$$

Modus Ponens:

$$\frac{\alpha_1, \dots, \alpha_k, \quad (\alpha_1 \wedge \cdots \wedge \alpha_k) \Rightarrow \beta}{\beta}$$

KB = {Rain ⇒ Wet, Wet ⇒ Unhappy, Rain, Wet}
Applying Modus Ponens on KB

$$\frac{\text{Wet}, \quad \text{Wet} \Rightarrow \text{Unhappy}}{\text{Unhappy}}$$

Forward Inference

Input: KB, α, $\mathfrak{I} =$ a set of inference rule

If $\alpha \in$ KB: **return** True

Repeat:

Choose a set of sentences $\alpha_1, \dots, \alpha_k \in$ KB such that

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_k}{\beta}$$

matches a rule in $\mathfrak{I}$, and $\beta \notin$ KB.

If $\beta = \alpha$: **return** True

If such $(\alpha_1, \alpha_2, \dots, \alpha_k, \beta)$ does not exist: **return** False

Add β to KB.

Forward Inference

- Forward inference is a search problem
  - What are the states, actions, successor function, and goal test?
  - Algorithms introduced for search problems can be applied here.

- Is the forward inference algorithm sound?
  - Yes, as long as all inference rules you use are sound

- Is forward inference complete?

Forward Inference

Example:

KB = {Rain ⇒ Wet,   Rain v Shine,   Wet v Shine ⇒ Happy}

α = Happy

Use Forward Inference algorithm with $\mathfrak{I} =$ {Modus Ponens}

- Can KB entail α?
- Can the algorithm derive α from KB?

$$\frac{P,\ P \Rightarrow Q}{Q}$$

$$\frac{P,\ \neg P \lor Q}{Q}$$

Forward Inference with Modus Ponens is **sound** but **not complete**

A Sound and Complete Algorithm?

Fact 1. If KB only consists of **Horn clauses,**

then Forward Inference with **Modus Ponens** is sound and complete.

Fact 2. In general, Forward Inference with **Resolution** is sound and complete.

Horn Clauses + Modus Ponens is Complete

Horn clause: sentence that have the following forms

$$X_1 \land X_2 \land \cdots \land X_{k-1} \Rightarrow X_k \quad \text{or} \quad X_1 \land X_2 \land \cdots \land X_k \Rightarrow \text{False}$$

$$\equiv \qquad\qquad \equiv$$

$$\neg X_1 \lor \neg X_2 \lor \cdots \lor \neg X_{k-1} \lor X_k \qquad \neg X_1 \lor \neg X_2 \lor \cdots \lor \neg X_k$$

Disjunction with only one positive symbol
(Definite clause)

Disjunction with no positive symbol
(Goal clause)

Horn Clauses + Modus Ponens is Complete

KB

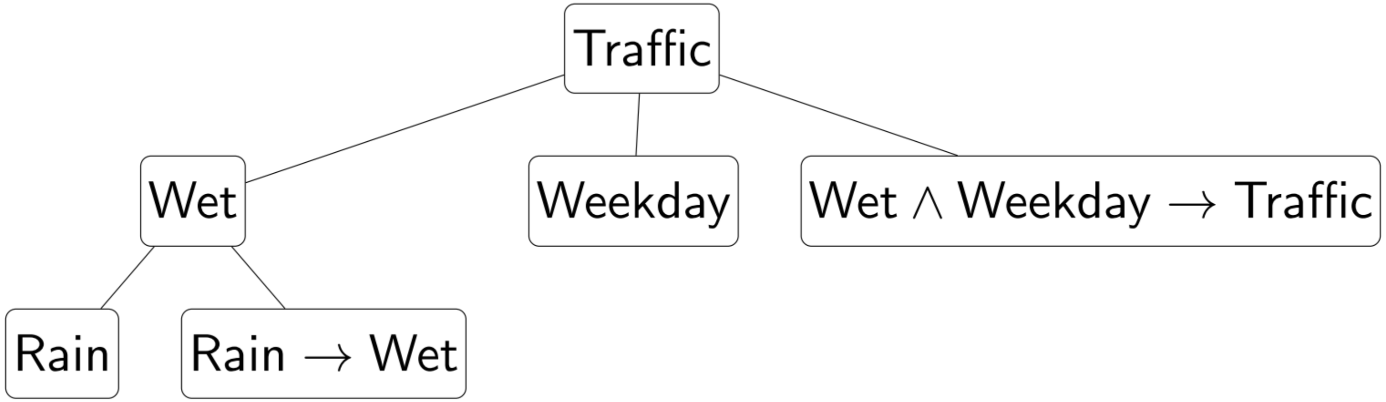
Rain

Weekday

Rain $\to$ Wet

Wet $\land$ Weekday $\to$ Traffic

Traffic $\land$ Careless $\to$ Accident



Intuition: The inference procedure of horn clauses is *direct*, in the sense that there is no branching.

Horn clause: Rain ∧ Snow → Dark ∧ Traffic

R ∧ S → D

R ∧ S → T

Non- horn clause: Wet → $\underbrace{\text{Rain} \lor \text{Snow}}$

Has to branch into the cases ¬Rain, ¬Snow etc.

A pseudocode for Forward Inference with Modus Ponens (this algorithm is also called **Forward Chaining**). This pseudocode assumes that all sentences are definite clauses (but it's easy to extend it to handle goal clauses as well).

The time complexity is linear in the "**size of KB**", i.e., the sum of the lengths of all sentences in KB.

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional Horn clauses
          q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known in KB

  while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then do
                  if HEAD[c] = q then return true
                  PUSH(HEAD[c], agenda)
  return false
```

General Case: Resolution is Complete

Resolution

$$\frac{\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_k \vee p, \quad \neg p \vee \beta_1 \vee \beta_2 \vee \cdots \vee \beta_m}{\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_k \vee \beta_1 \vee \beta_2 \vee \cdots \vee \beta_m}$$

Example

$$\frac{\text{Rain} \vee \text{Shine}, \quad \neg\text{Rain} \vee \text{Wet}}{\text{Shine} \vee \text{Wet}}$$

Converting Sentences to CNF Before Applying Resolution

Conjunctive Normal Form (CNF)

Example: $(A \lor B \lor \neg C) \land (\neg B \lor D)$

Converting Sentences to CNF: Example

Initial formula:

$$(\text{Summer} \rightarrow \text{Snow}) \rightarrow \text{Bizzare}$$

Remove implication ($\rightarrow$):

$$\neg(\neg\text{Summer} \vee \text{Snow}) \vee \text{Bizzare}$$

Push negation ($\neg$) inwards (de Morgan):

$$(\neg\neg\text{Summer} \wedge \neg\text{Snow}) \vee \text{Bizzare}$$

Remove double negation:

$$(\text{Summer} \wedge \neg\text{Snow}) \vee \text{Bizzare}$$

Distribute $\vee$ over $\wedge$:

$$(\text{Summer} \vee \text{Bizzare}) \wedge (\neg\text{Snow} \vee \text{Bizzare})$$

Converting Sentences to CNF: General Rules

Conversion rules:

- Eliminate $\leftrightarrow$: $\frac{f \leftrightarrow g}{(f \rightarrow g) \wedge (g \rightarrow f)}$
- Eliminate $\rightarrow$: $\frac{f \rightarrow g}{\neg f \vee g}$
- Move $\neg$ inwards: $\frac{\neg(f \wedge g)}{\neg f \vee \neg g}$
- Move $\neg$ inwards: $\frac{\neg(f \vee g)}{\neg f \wedge \neg g}$
- Eliminate double negation: $\frac{\neg\neg f}{f}$
- Distribute $\vee$ over $\wedge$: $\frac{f \vee (g \wedge h)}{(f \vee g) \wedge (f \vee h)}$

Resolution-Based Inference Algorithm

Note that $KB \vDash \alpha$ is equivalent to $\mathcal{M}(KB \land \neg \alpha)$ = empty set

KB' ← KB U $\{\neg \alpha\}$

Convert all sentences in KB' to CNF

Repeatedly apply Resolution Rule until

1) False is derived → return $KB \vDash \alpha$
2) No new sentence can be derived → return $KB \nvDash \alpha$

Resolution-Based Inference 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 $KB \land \neg\alpha$
  $new \leftarrow \{\ \}$
  loop do
    for each 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*
    $clauses \leftarrow clauses \cup new$

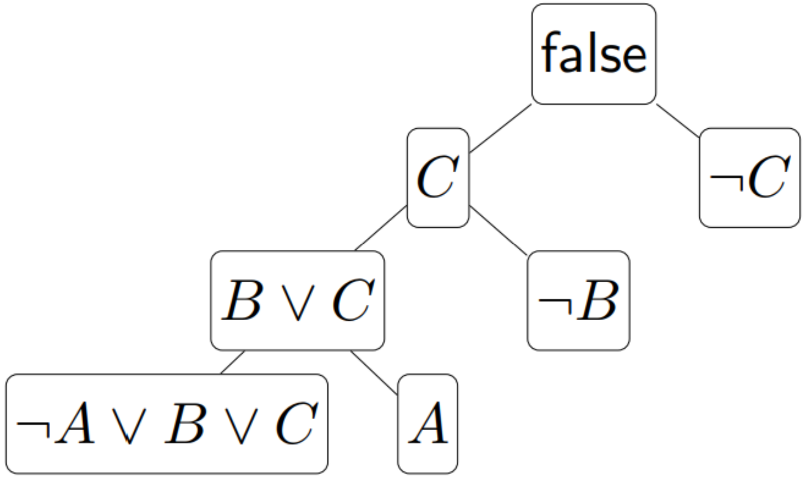
Resolution-Based Inference Algorithm

$$\mathsf{KB}' = \{A \rightarrow (B \vee C), A, \neg B, \neg C\}$$

Convert to CNF:

$$\mathsf{KB}' = \{\neg A \vee B \vee C, A, \neg B, \neg C\}$$

Repeatedly apply **resolution** rule:



Conclusion: KB **entails** f

Time Complexity

- Modus Ponens

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_k, \quad (\alpha_1 \land \alpha_2 \land \cdots \land \alpha_k) \Rightarrow \beta}{\beta}$$

  Each rule application adds sentence with **one** propositional symbol
  → **linear time**

- Resolution

$$\frac{\alpha_1 \lor \alpha_2 \lor \cdots \lor \alpha_k \lor p, \quad \neg p \lor \beta_1 \lor \beta_2 \lor \cdots \lor \beta_m}{\alpha_1 \lor \alpha_2 \lor \cdots \lor \alpha_k \lor \beta_1 \lor \beta_2 \lor \cdots \lor \beta_m}$$

  Each rule application adds sentence with **many** propositional symbol
  → **exponential time**

Recap

	Modus Ponens	Resolution
Sound?	Yes	Yes
Complete?	No	Yes
Complete for horn clauses?	Yes	Yes
Time complexity	linear	exponential