# Introduction to Artificial Intelligence
# Logical Agents

Jianmin Li

Department of Computer Science and Technology
Tsinghua University

Spring, 2024

# Outline

- Knowledge-based agents
- Logic in general - models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving

# Knowledge based agent

| | | |
|---|---|---|
| Inference engine | ← | domain−independent algorithms |
| Knowledge base | ← | domain−specific content |

# Knowledge bases

- Knowledge base
  - a set of sentences in a formal language
- Tell
  - add new sentences about what the agent needs to know
- Ask
  - query what is known or what is to be done
- Inference
  - deriving new sentences from old
  - answers should follow from the KB
  - both Tell and Ask may involve inference

# A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action
   static: 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
```

# A simple knowledge-based agent

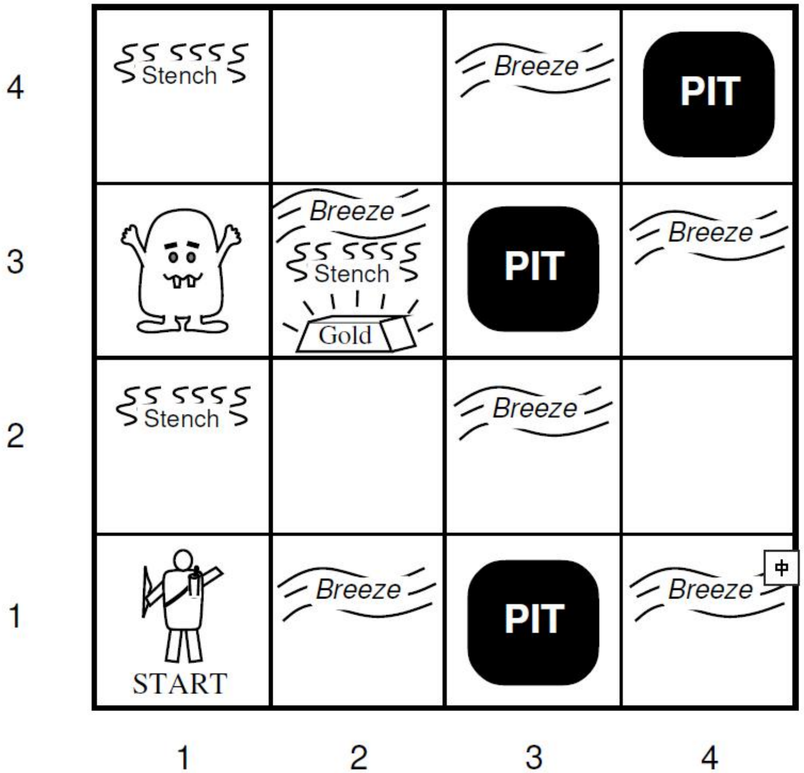
- The agent must be able to:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

# Knowledge based Agents

- Declarative approach to building an agent
  - TELL sentences one by one
- Procedural approach
  - encodes desired behaviors directly as program code
- Knowledge level
  - what they know, regardless of how implemented
- Implementation level
  - data structures in KB and algorithms that manipulate them

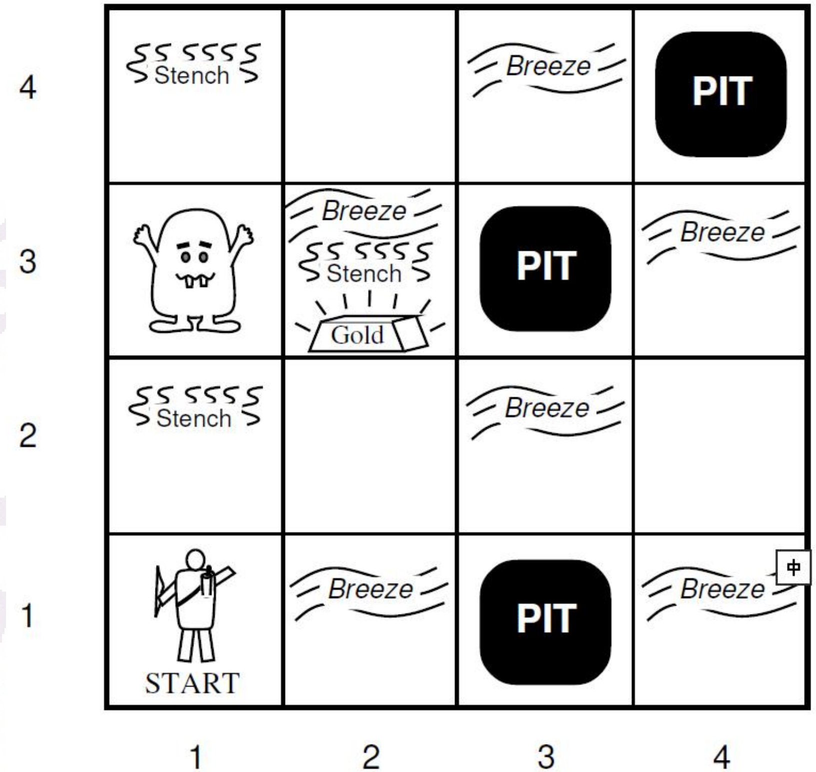
# Wumpus World PEAS description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Actuators
  - Left turn, Right turn, Forward, Grab, Release, Shoot
- Sensors
  - Breeze, Glitter, Smell



# Wumpus World PEAS description

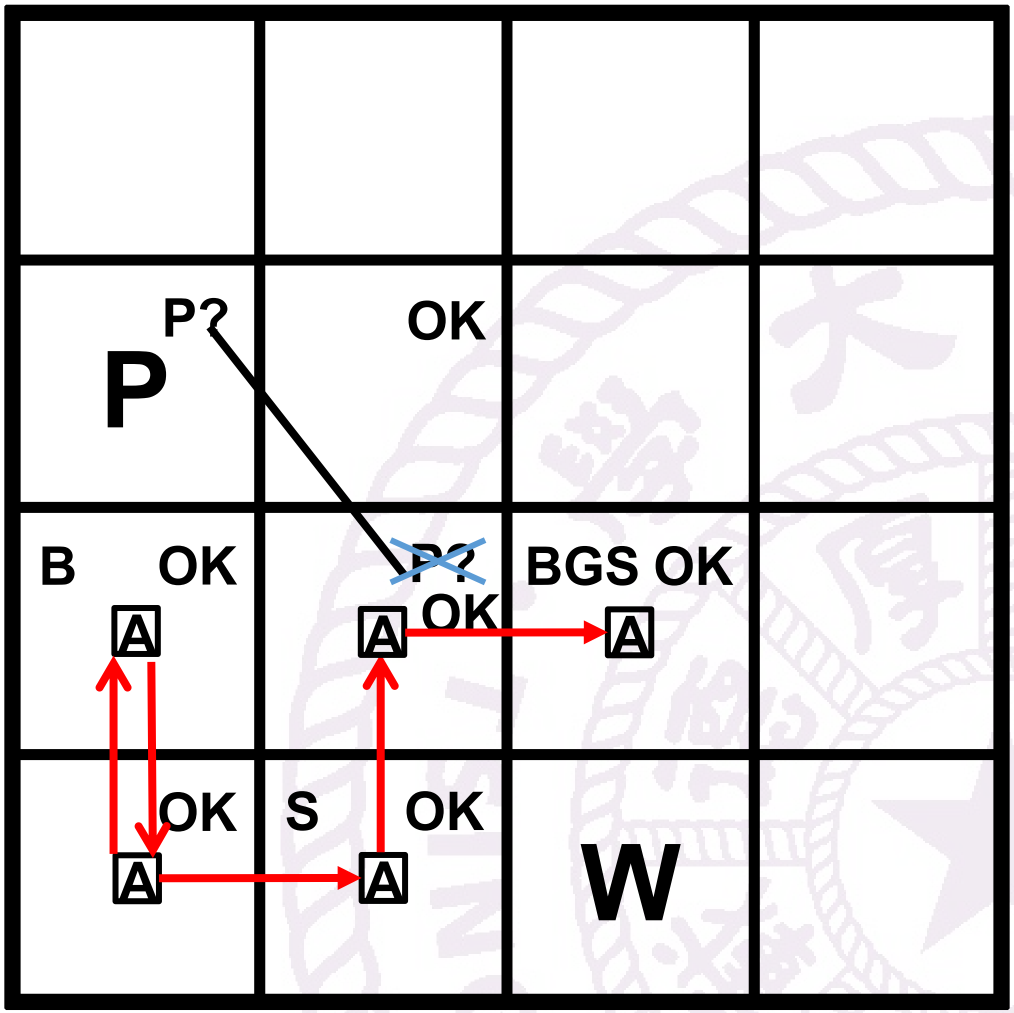
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square



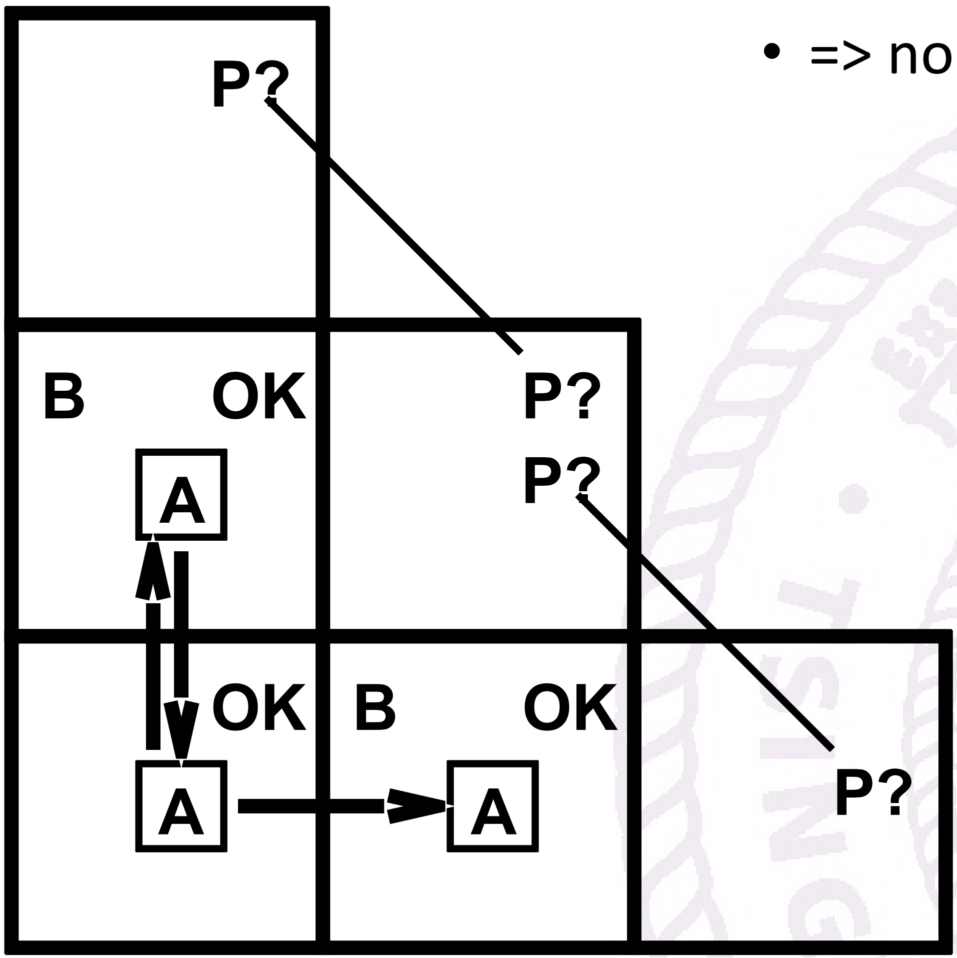
# Wumpus world characterization

- Observable
  - No - only local perception
- Deterministic
  - Yes - outcomes exactly specified
- Static
  - Yes - Wumpus and Pits do not move
- Discrete
  - Yes
- Single-agent
  - Yes - Wumpus is essentially a natural feature

# Exploring a wumpus world

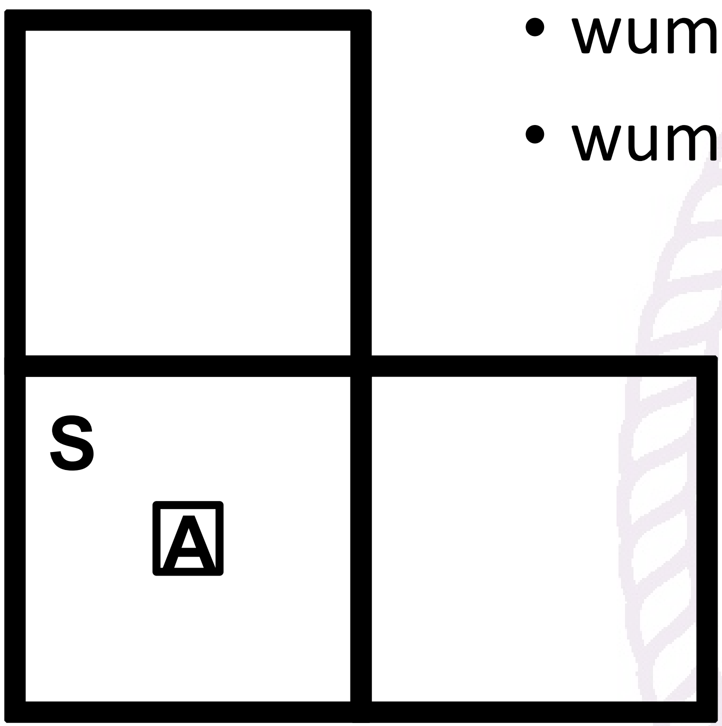


# Other tight spots




- Breeze in (1,2) and (2,1)
  - => no safe actions

# Other tight spots




- Smell in (1,1) => cannot move
- Can use a strategy of coercion:
  - shoot straight ahead
- wumpus was there => dead => safe
- wumpus wasn't there => safe

# Logic in general

- Logics are formal languages for representing information
- Syntax defines the sentences in the language
  - what is a well-formed sentence
- Semantics define the “meaning” of sentences
  - define truth of a sentence in a world
- E.g., the language of arithmetic
  - x +2 >= y is a sentence; x2+y > is not a sentence
  - x +2 >= y is true iff the number x +2 is no less than the number y
  - x +2 >= y is true in a world where x = 7, y = 1
  - x +2 >= y is false in a world where x = 0, y = 6

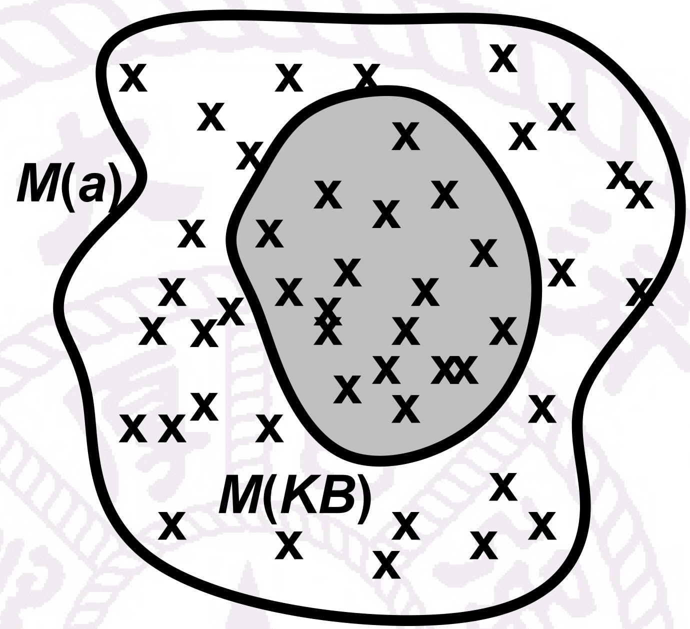
# Entailment

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

# Models

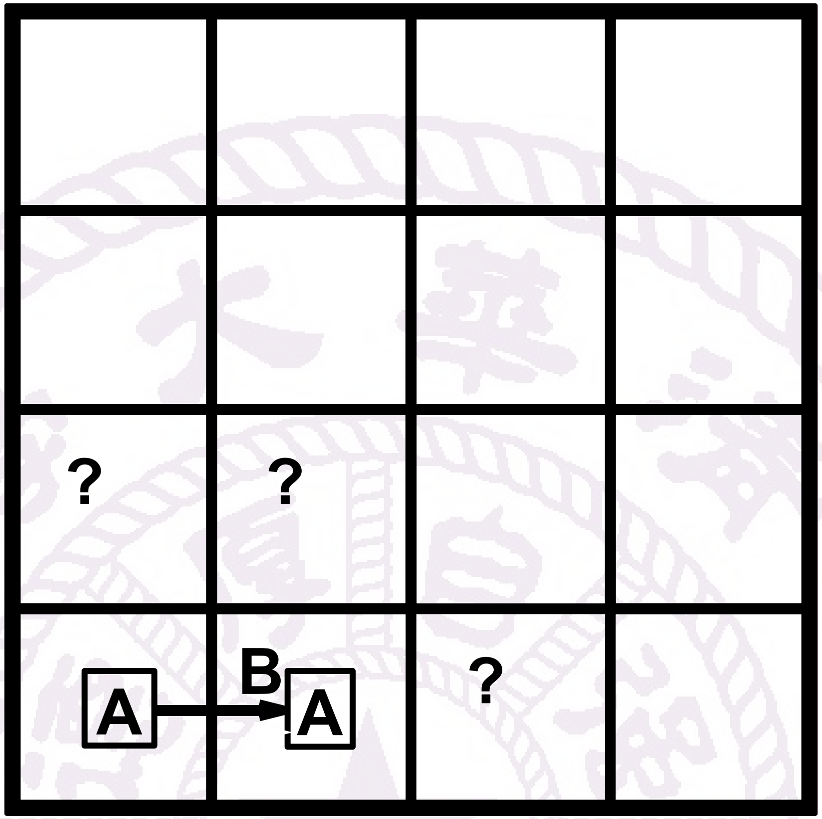
- formally structured worlds with respect to which truth can be evaluated
- *m* satisfies sentence *a* if *a* is true in *m*
- $M(a)$ is the set of all models of *a*
- $KB \vDash a$ if and only if

$$M(KB) \subseteq M(a)$$



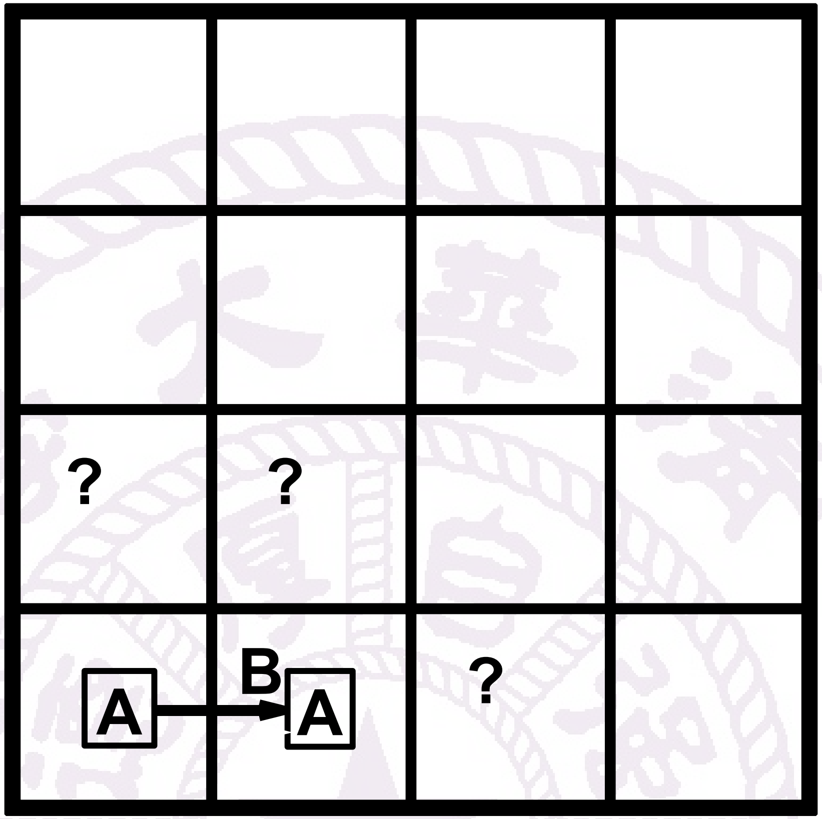
# Entailment in the wumpus world

- Situation after detecting nothing in [1,1]
  - moving right, breeze in [2,1]
- Consider possible models for ?s assuming only pits



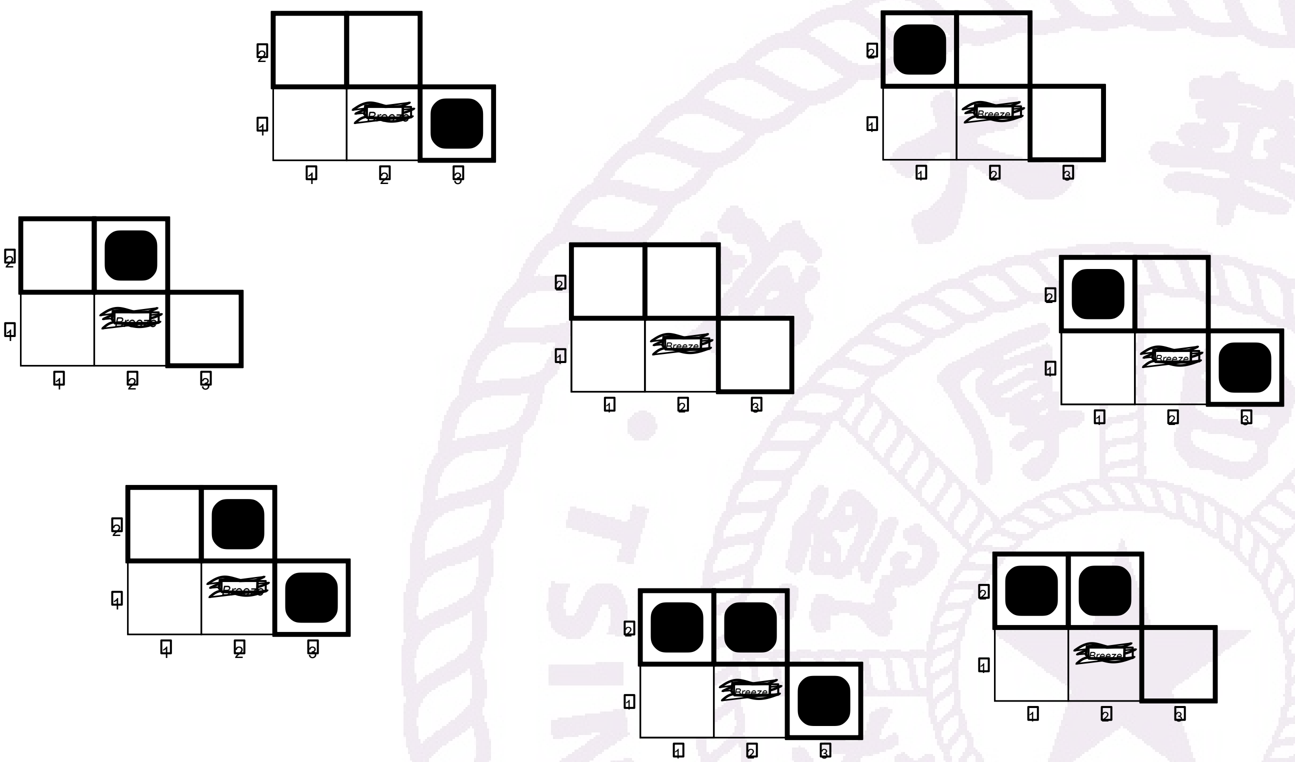
# 问题1

- 假设带？的房间中只可能有坑（pit），存在多少种模型？
  A. 0
  B. 1
  C. 3
  D. 8



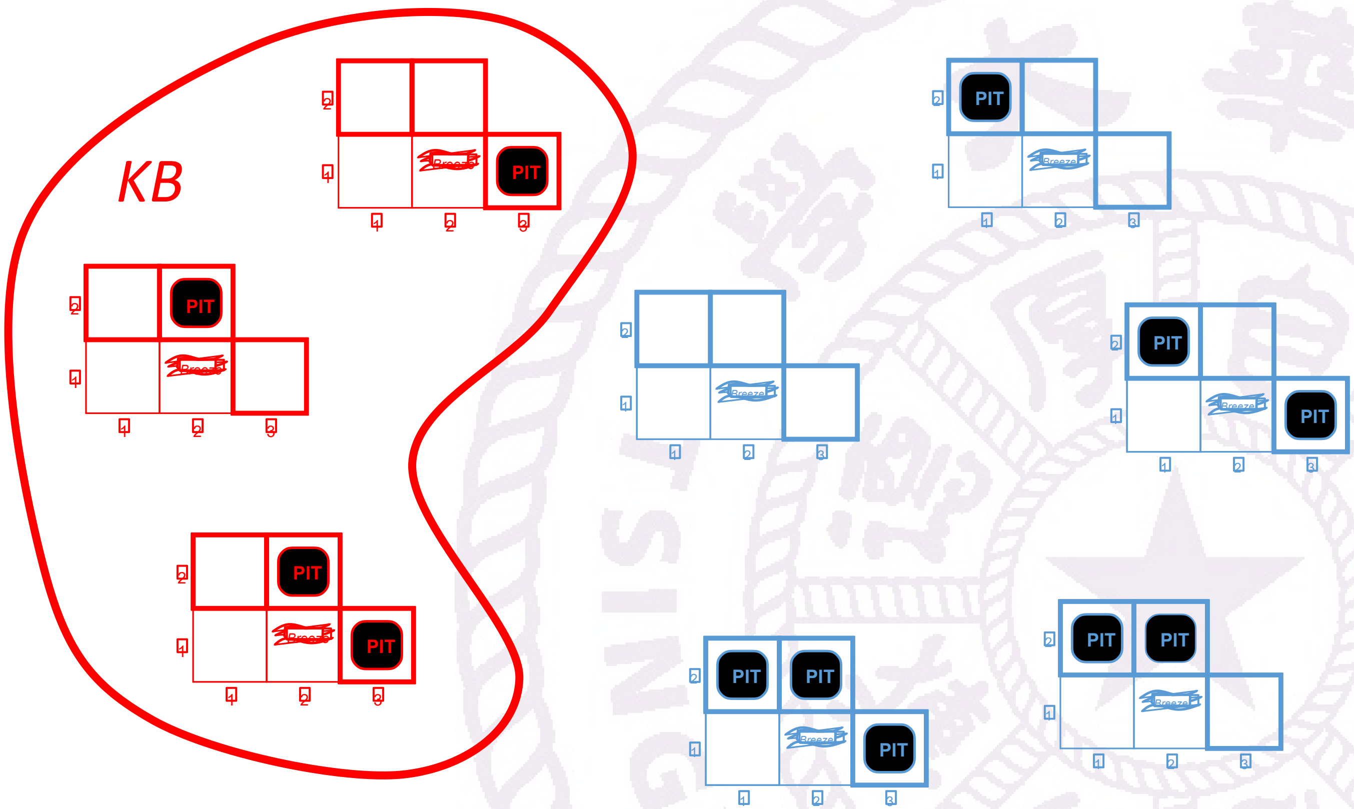
# Wumpus models

- 3 Boolean choices => 8 possible models



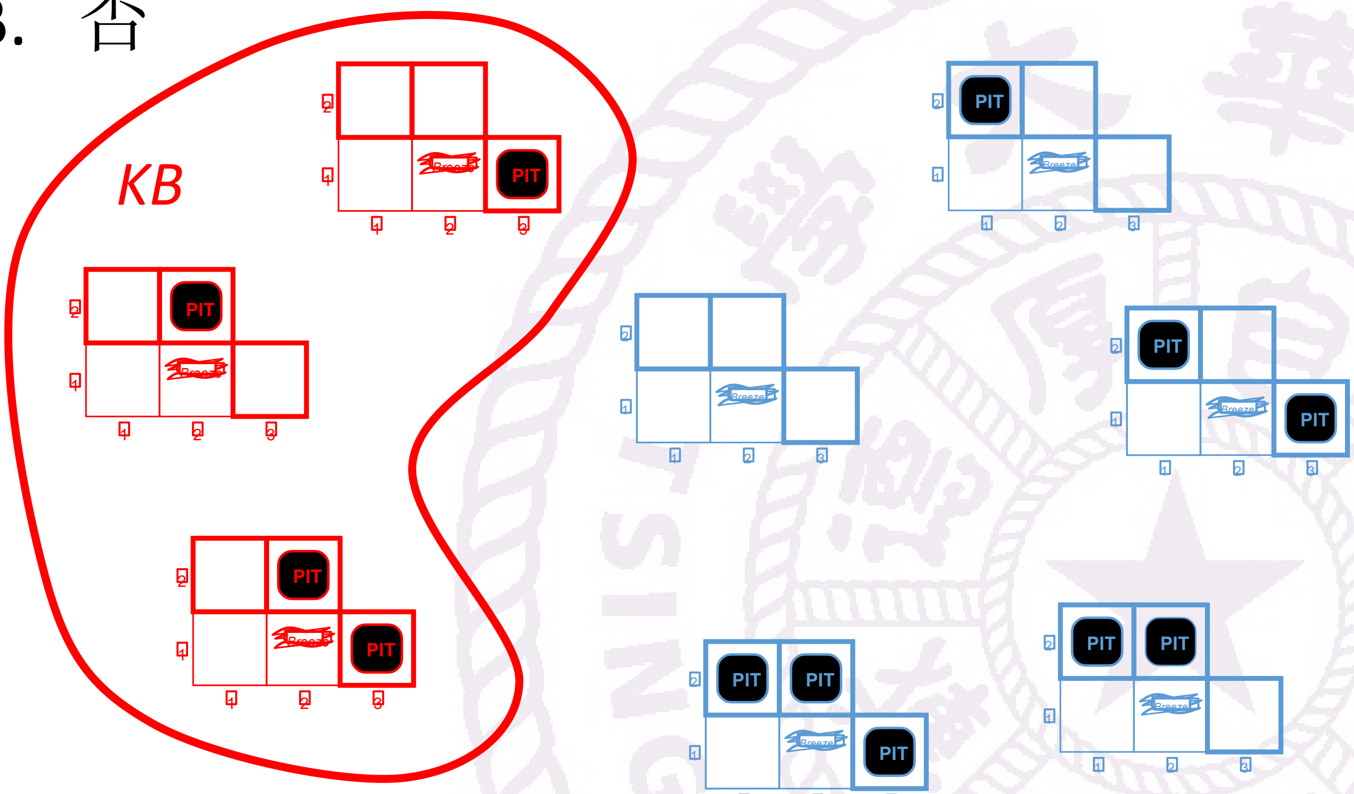
# Wumpus models

- *KB* = wumpus-world rules + observations
- nothing in [1,1], breeze in [2,1]



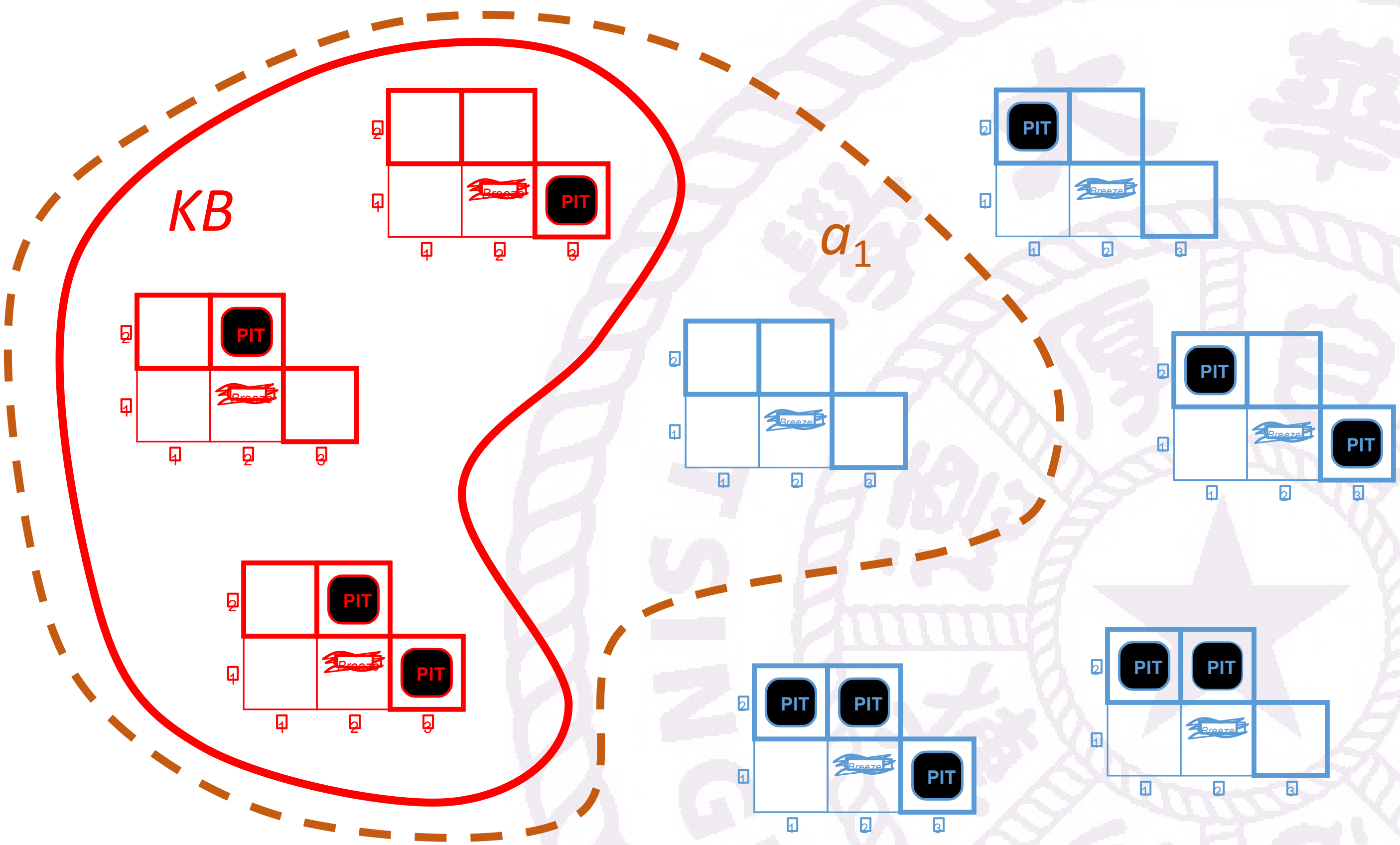
# 问题2

- $a_1$ = “[1,2] is safe”， $KB \vDash a_1$？
  A. 是
  B. 否



# Wumpus models

- $a_1$ = “[1,2] is safe”
- $KB \vDash a_1$

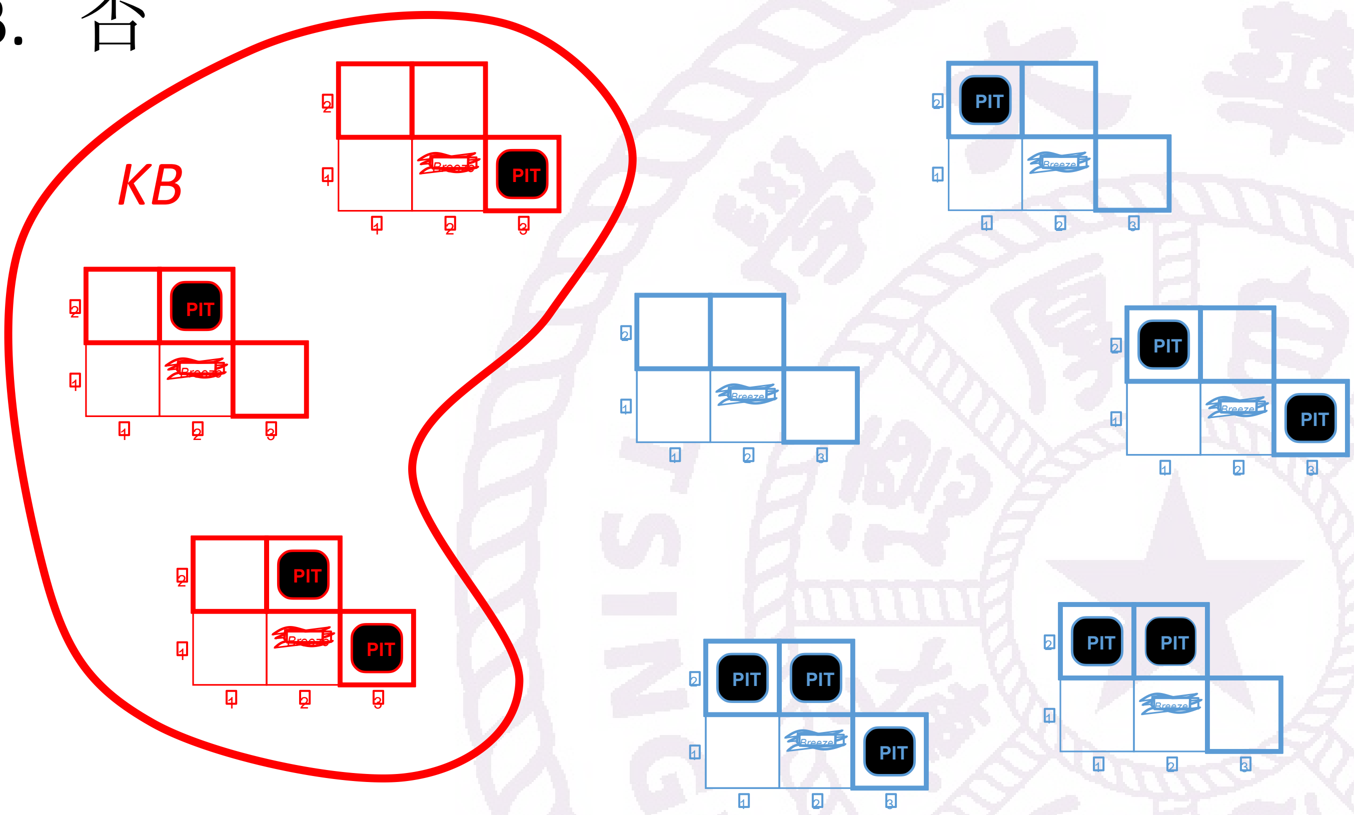


# 问题3

- $a_2$ = “[2,2] is safe”， $KB \vDash a_2$?

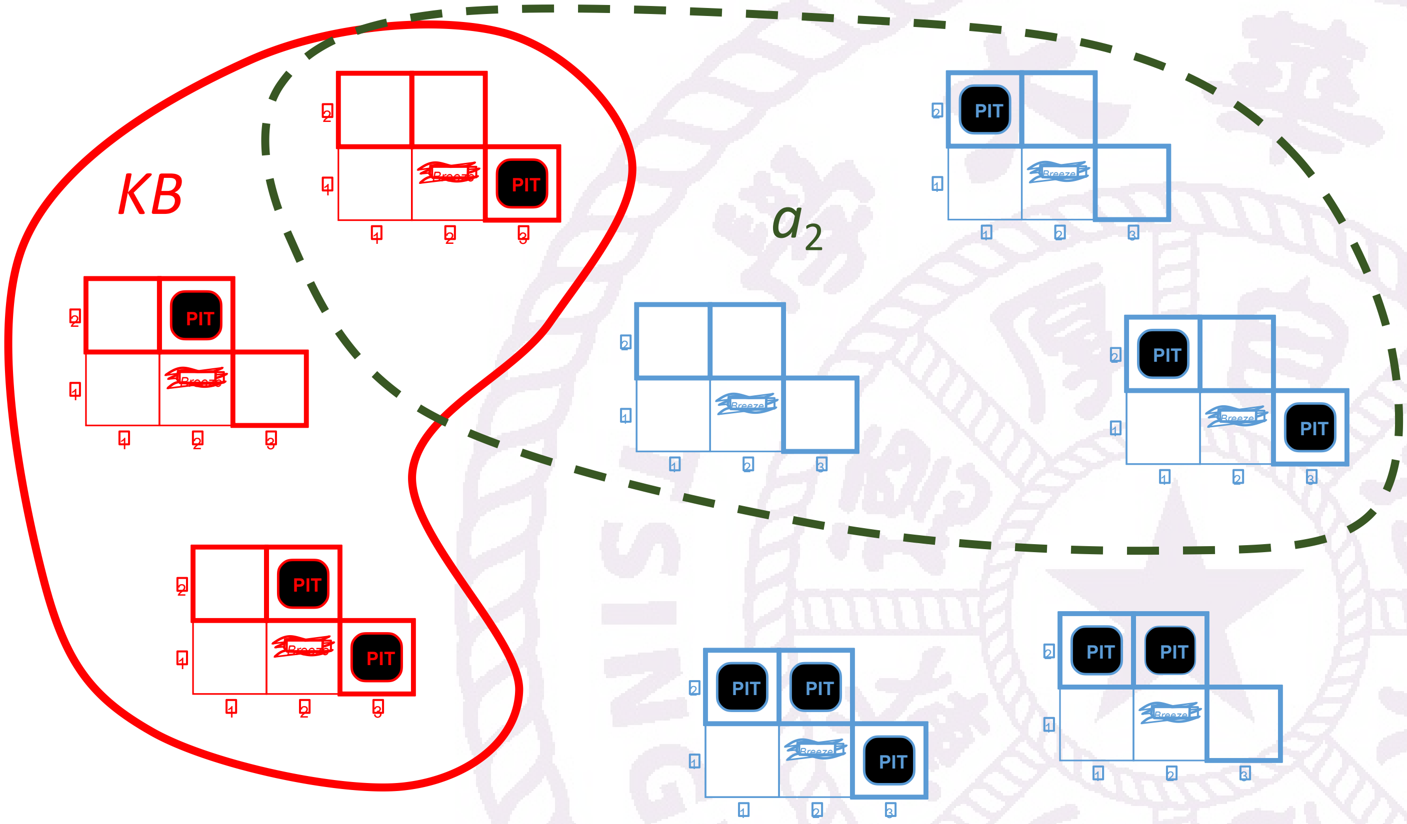
  A. 是

  B. 否



# Wumpus models

- $a_2$ = "[2,2] is safe"
- $KB \nvDash a_2$



# Inference

- Consequences of *KB* are a haystack; *a* is a needle.
- Entailment = needle in haystack; inference = finding it
- $KB \vdash_i a$ = sentence *a* can be derived from *KB* by procedure *i*
- Soundness: *i* is sound if
  - whenever $KB \vdash_i a$, it is also true that $KB \vDash a$
- Completeness: *i* is complete if
  - whenever $KB \vDash a$, it is also true that $KB \vdash_i a$

# Propositional logic: Syntax

- Propositional logic is the simplest logic
- The proposition symbols $P_1$, $P_2$, etc are atomic sentences
- If $S$ is a sentence, $\neg S$ is a sentence (negation)
- If $S_1$ and $S_2$ are sentences
  - $S_1 \wedge S_2$ is a sentence (conjunction)
  - $S_1 \vee S_2$ is a sentence (disjunction)
  - $S_1 \Rightarrow S_2$ is a sentence (implication)
  - $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

# Propositional logic: Syntax

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\, \mathit{Sentence} \,) \mid [\, \mathit{Sentence} \,] \\
 & \mid & \neg\, \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
 & & \\
\text{OPERATOR PRECEDENCE} & : & \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

# Propositional logic: Semantics

- Each model specifies true/false for each proposition symbol
- e.g.

| $P_{12}$ | $P_{22}$ | $P_{31}$ |
|---|---|---|
| false | false | true |

  - With these symbols, 8 possible models, can be enumerated automatically

# Propositional logic: Semantics

- Rules for evaluating truth with respect to a model *m*:

| | | | | | | |
|---|---|---|---|---|---|---|
| $\neg S$ | is true iff | $S$ | is false | | | |
| $S_1 \wedge S_2$ | is true iff | $S_1$ | is true | and | $S_2$ | is true |
| $S_1 \vee S_2$ | is true iff | $S_1$ | is true | or | $S_2$ | is true |
| $S_1 \Rightarrow S_2$ | is true iff | $S_1$ | is false | or | $S_2$ | is true |
| | is false iff | $S_1$ | is true | and | $S_2$ | is false |
| $S_1 \Leftrightarrow S_2$ | is true iff | $S_1 \Rightarrow S_2$ | is true | and | $S_1 \Rightarrow S_2$ | is true |

# Truth tables for connectives

| $P$ | $Q$ | $\neg P$ | $P \wedge Q$ | $P \vee Q$ | $P \Rightarrow Q$ | $P \Leftrightarrow Q$ |
|---|---|---|---|---|---|---|
| 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 connectives

- Simple recursive process evaluates an arbitrary sentence

| $P_{12}$ | $P_{22}$ | $P_{31}$ |
|---|---|---|
| false | false | true |

- $\neg P_{12} \wedge (P_{22} \vee P_{31})$ = true $\wedge$ (false $\vee$ true)

  = true $\wedge$ true

  = true

# Wumpus world sentences

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

$$\neg P_{11}$$

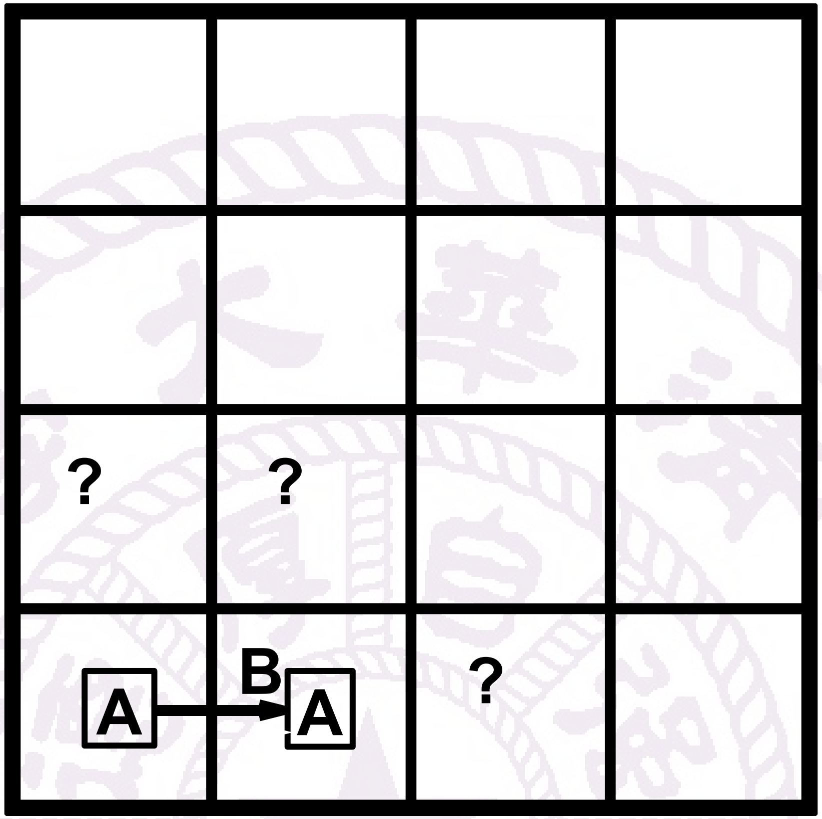
$$\neg B_{11}$$

$$B_{21}$$

- "Pits cause breezes in adjacent squares"
- "A square is breezy if and only if there is an adjacent pit"

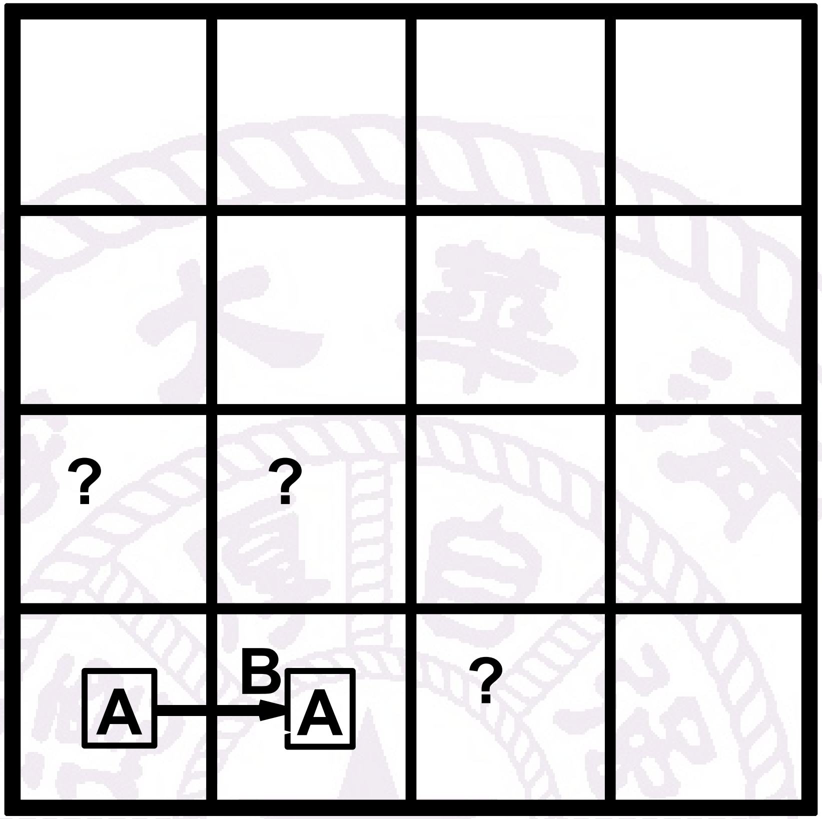
$$B_{11} \Leftrightarrow (P_{12} \vee P_{21})$$

$$B_{21} \Leftrightarrow (P_{11} \vee P_{22} \vee P_{31})$$



# Knowledge base

- $R_1$: $\neg P_{11}$
- $R_2$: $\neg B_{11}$
- $R_3$: $B_{21}$
- $R_4$: $B_{11} \Leftrightarrow (P_{12} \vee P_{21})$
- $R_5$: $B_{21} \Leftrightarrow (P_{11} \vee P_{22} \vee P_{31})$



# Truth tables for inference

| $B_{1,1}$ | $B_{2,1}$ | $P_{1,1}$ | $P_{1,2}$ | $P_{2,1}$ | $P_{2,2}$ | $P_{3,1}$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $KB$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *false* | *false* | *false* | *false* | *false* | *false* | *false* | *true* | *true* | *true* | *true* | *false* | *false* |
| *false* | *false* | *false* | *false* | *false* | *false* | *true* | *true* | *true* | *false* | *true* | *false* | *false* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *false* | *true* | *false* | *false* | *false* | *false* | *false* | *true* | *true* | *false* | *true* | *true* | *false* |
| *false* | *true* | *false* | *false* | *false* | *false* | *true* | *true* | *true* | *true* | *true* | *true* | *true* |
| *false* | *true* | *false* | *false* | *false* | *true* | *false* | *true* | *true* | *true* | *true* | *true* | *true* |
| *false* | *true* | *false* | *false* | *false* | *true* | *true* | *true* | *true* | *true* | *true* | *true* | *true* |
| *false* | *true* | *false* | *false* | *true* | *false* | *false* | *true* | *false* | *false* | *true* | *true* | *false* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *true* | *true* | *true* | *true* | *true* | *true* | *true* | *false* | *true* | *true* | *false* | *true* | *false* |

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

# Inference by enumeration

```
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
   else do
      P ← FIRST(symbols); rest ← REST(symbols)
      return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

# Inference by enumeration

- Depth-first enumeration of all models
- Sound and Complete
- $O(2^n)$ for $n$ symbols

# Proof methods

- Proof methods divide into (roughly) two kinds:
  - Model checking
    - truth table enumeration (always exponential in *n*)
    - improved backtracking, e.g., Davis - Putnam - Logemann - Loveland
    - heuristic search in model space (sound but incomplete), e.g., min-conflicts-like hill-climbing algorithms
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
    - Can use inference rules as operators in a standard search alg.
    - Typically require translation of sentences into a normal form

# Logical equivalence

- Two sentences are logically equivalent iff true in same models:
  - $a \equiv \beta$ if and only if $a \vDash \beta$ and $\beta \vDash a$

# Logical equivalence

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

# Validity and satisfiability

- A sentence is valid if it is true in all models
  - e.g. True, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$
- Validity is connected to inference via the Deduction Theorem:
  - $\alpha \vDash \beta$ if and only if $\alpha \Longrightarrow \beta$ is valid
- A sentence is satisfiable if it is true in some model
  - e.g. $A \lor B$, $C$
- A sentence is unsatisfiable if it is true in no models
  - e.g. $A \land \neg A$
- Satisfiability is connected to inference via the following:
  - $\alpha \vDash \beta$ if and only if $\alpha \land \neg \beta$ is unsatisfiable
  - i.e., prove $\beta$ by *reductio ad absurdum*

# Forward and backward chaining

- Horn Form (restricted)
  - KB = conjunction of Horn clauses
  - Horn clause
    - a disjunction of literals of which at most one is positive
    - proposition symbol; or
    - (conjunction of symbols) $\Rightarrow$ symbol
  - Definite clause
    - a disjunction of literals of which exactly one is positive
  - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

# Forward and backward chaining

- Modus Ponens (for Horn Form): complete for Horn *KBs*

$$\frac{a_1, \ldots, a_n, a_1 \wedge \ldots \wedge a_n \Rightarrow \beta}{\beta}$$

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

# Forward chaining

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

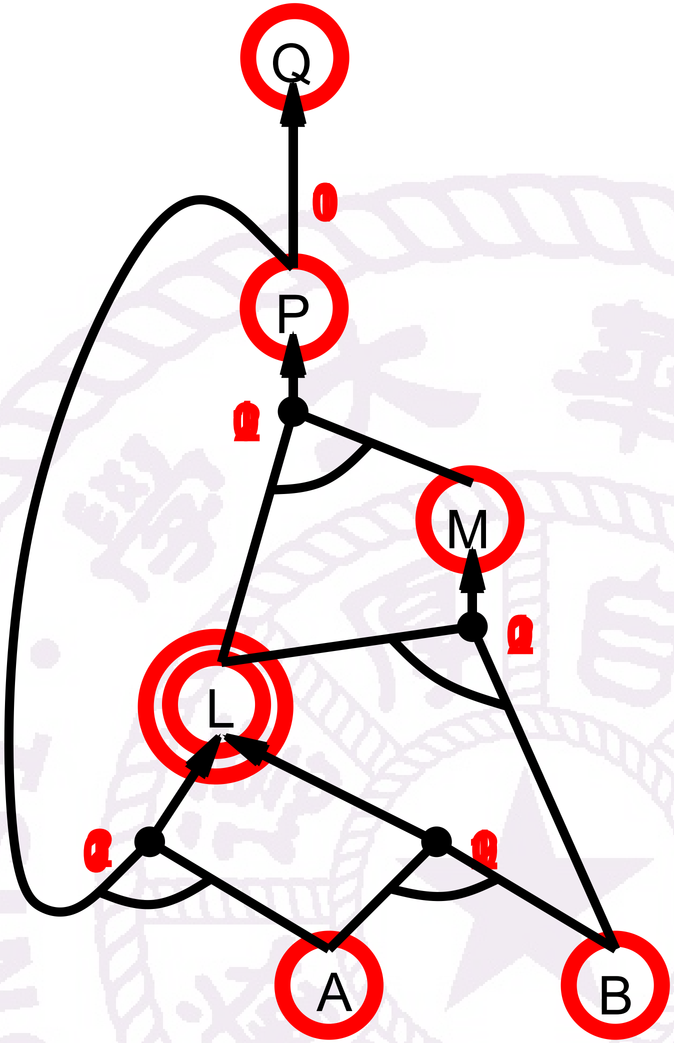
# Forward chaining algorithm

```
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 the number of symbols in c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  agenda ← a queue of symbols, initially symbols known to be true in KB

  while agenda is not empty do
     p ← POP(agenda)
     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 agenda
  return false
```

# Forward chaining example

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



# Proof of completeness

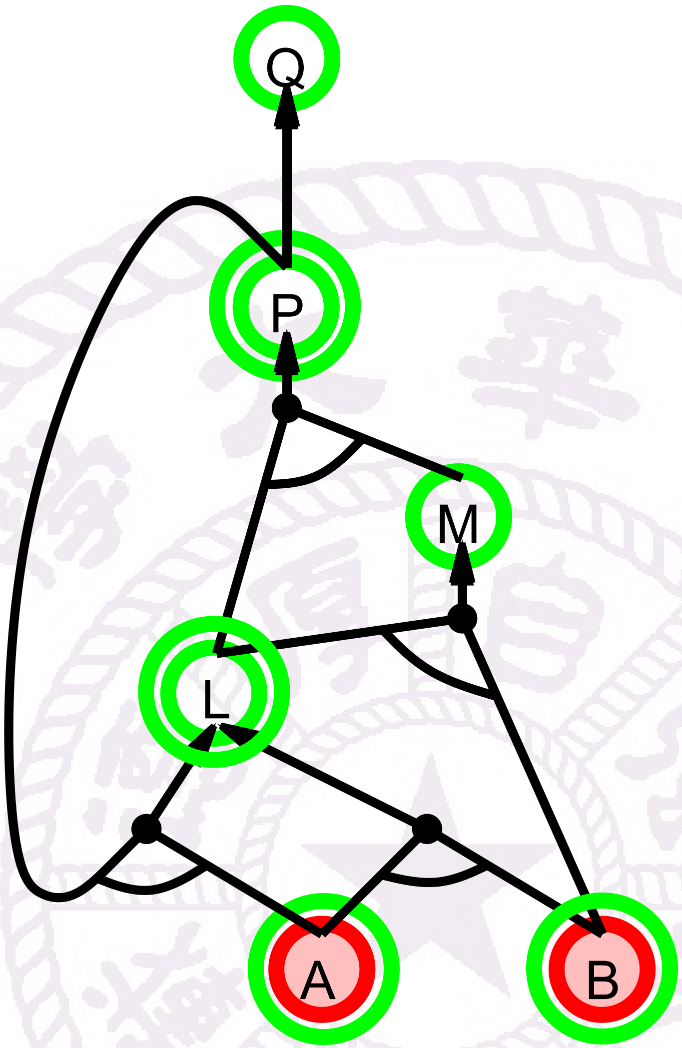
- FC derives every atomic sentence that is entailed by *KB*
  - FC reaches a fixed point where no new atomic sentences are derived
  - Consider the final state of infered table as a model *m*, assigning true/false to symbols
  - Every definite clause in the original *KB* is true in m
    - Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_n \Rightarrow \beta$ is false in m
    - Then $a_1 \wedge \ldots \wedge a_n$ is true in *m* and *b* is false in *m*
    - Therefore the algorithm has not reached a fixed point!
  - Hence *m* is a model of *KB*
  - If $KB \vDash q$, *q* is true in every model of *KB*, including *m*

# 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

# 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
  - 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*

# Resolution

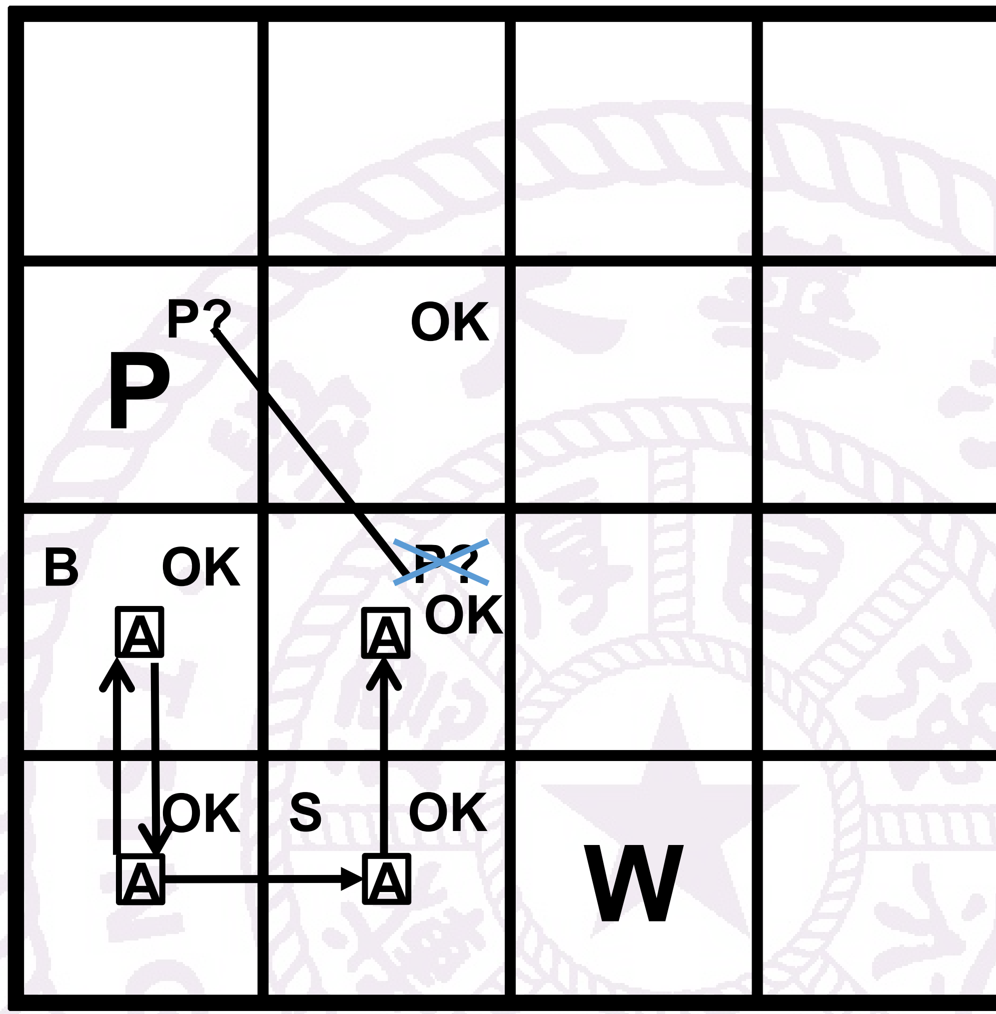
- Conjunctive Normal Form (CNF—universal)
  - conjunction of disjunction of literals
  - clauses
  - E.g., (A∨¬B) ∧ (B∨¬C∨¬D)
- Resolution inference rule (for CNF):

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

    - where $l_i$ and $m_j$ are complementary literals
- Sound and complete for propositional logic

# Resolution

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



# Conversion to CNF

$$B_{11} \Leftrightarrow (P_{12} \vee P_{21})$$

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

$$\big(B_{11} \Longrightarrow (P_{12} \vee P_{21})\big) \wedge \big((P_{12} \vee P_{21}) \Longrightarrow B_{11}\big)$$

- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$

$$(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg(P_{12} \vee P_{21}) \vee B_{11})$$

- Move $\neg$ inwards using de Morgan's rules and double-negation

$$(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge \big((\neg P_{12} \wedge \neg P_{21}) \vee B_{11}\big)$$

- Apply distributivity law ($\vee$ over $\wedge$) and flatten

$$(\neg B_{11} \vee P_{12} \vee P_{21}) \wedge (\neg P_{12} \vee B_{11}) \wedge (\neg P_{21} \vee B_{11})$$

# Resolution algorithm

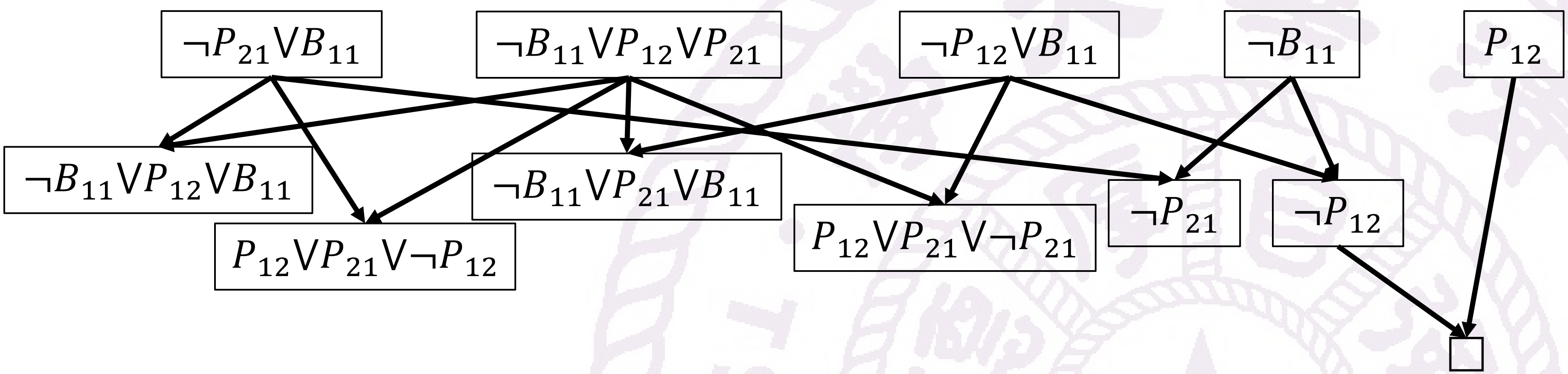
- Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

```
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 ∧ ¬α
    new ← { }
    loop do
        for each 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
```

# Resolution example

$$KB = \big(B_{11} \Leftrightarrow (P_{12} \lor P_{21})\big) \land \neg B_{11}$$
$$a = \neg P_{12}$$



# Summary

- Logical agents apply inference to a knowledge base
  - to derive new information and make decisions
- Basic concepts of logic:
  - syntax: formal structure of sentences
  - semantics: truth of sentences wrt models
  - entailment: necessary truth of one sentence given another
  - inference: deriving sentences from other sentences
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences
- Forward, backward chaining are linear-time, complete for Horn clauses
- Resolution is complete for propositional logic
- Propositional logic lacks expressive power

谢谢！