CSCI 3230

Fundamentals of Artificial Intelligence

Chapter 7

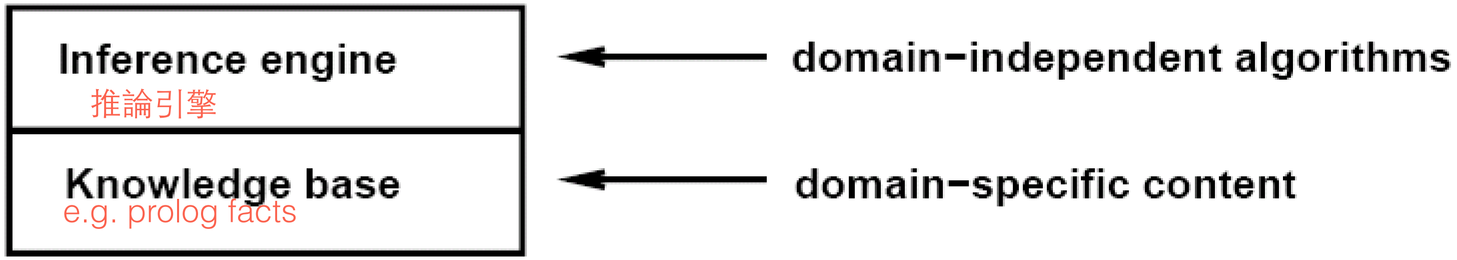
Logical Agents

Fall 2013

Outline

- Knowledge-based agents
- Wumpus World
- Logic in general – models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
    - Forward chaining
    - Backward chaining
    - Resolution

Knowledge bases



Knowledge base = set of sentences in a formal language

陳述的
Declarative approach to building an agent (or other system): Tell it what it needs to know

Then it can infer what to do – answers should follow from the KB

We can describe a knowledge-based agent at 3 levels:

- The knowledge level or epistemological 認識論的 level is the most abstract; we can describe the agent by saying what it knows.
- The logical level is the level at which the knowledge is encoded into sentences.
- The implementation level runs on the agent architecture.

A simple knowledge-based agent

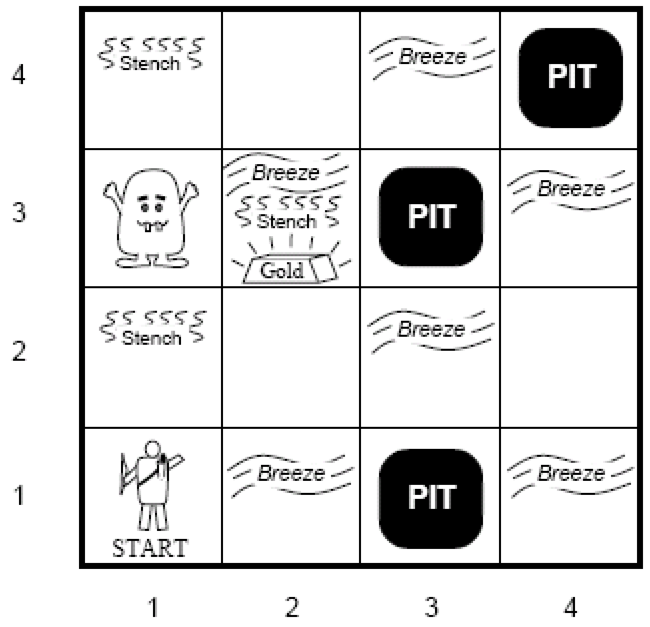
```
function KB-Agent (percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time
    Tell (KB, Make-Percept-Sentence (percept, t))
    action ← Ask (KB, Make-Action-Query (t))
    Tell (KB, Make-Action-Sentence( action , t))
    t ← t + 1
    return action
```

The agent must be able to:

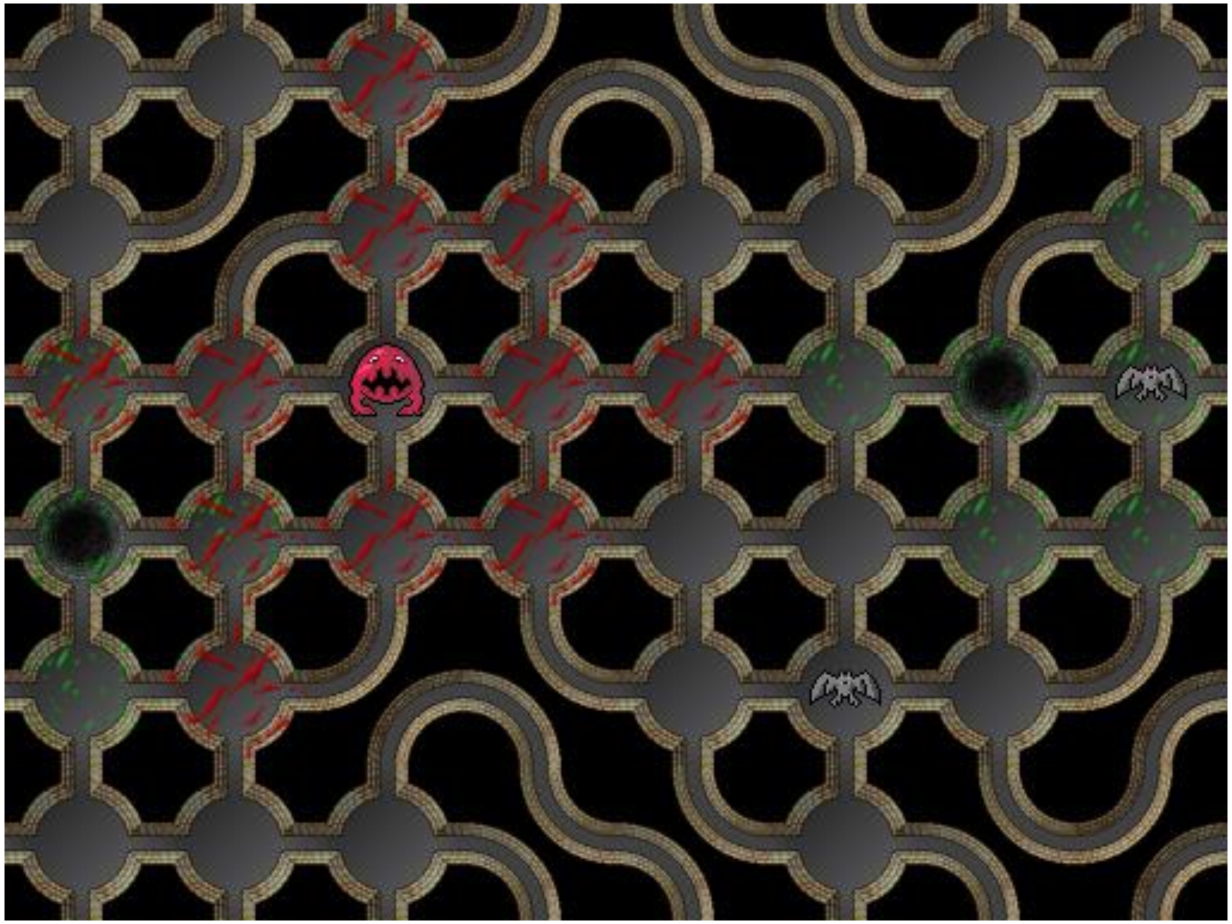
- Represent states, actions, etc.
- 吸收 Incorporate new percepts
- Update internal representations of the world.
- Deduce hidden properties of the world
- Deduce appropriate actions

Wumpus World PEAS description

- Performance measure (Goal: to pick up the gold & return to start)
  - Gold + 1000, death – 1000
  - −1 per step, −10 for using the arrow
- Environment:
  - Square adjacent to wumpus are smelly
  - Square 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
- Sensors: Breeze, Glitter, Smell
- Actuators:
  - Left turn, Right turn, Forward, Grab, Release, shoot



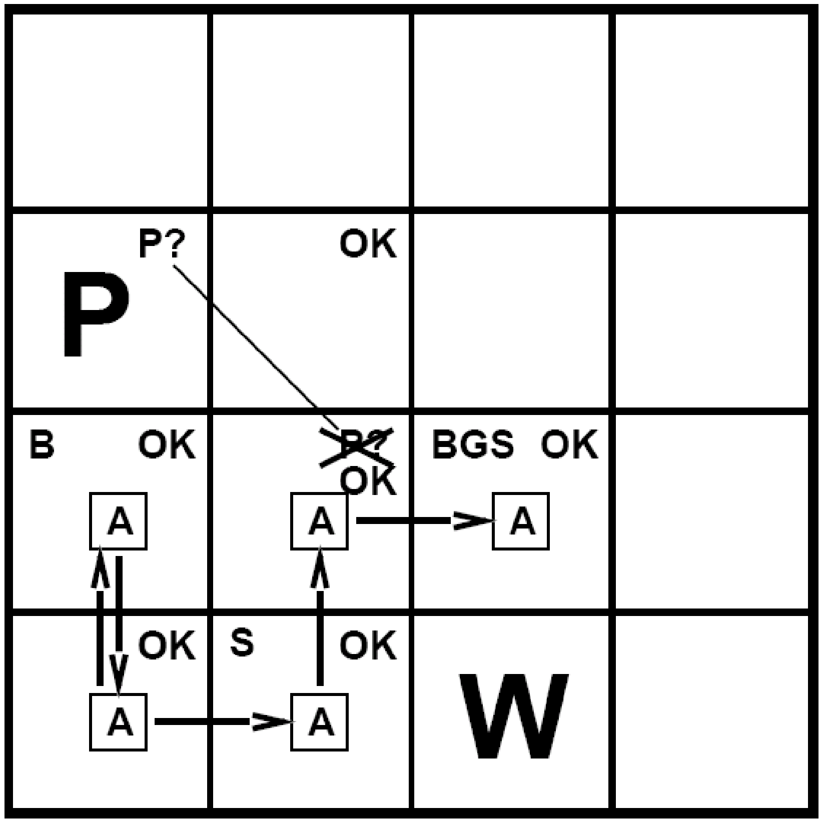
Demo



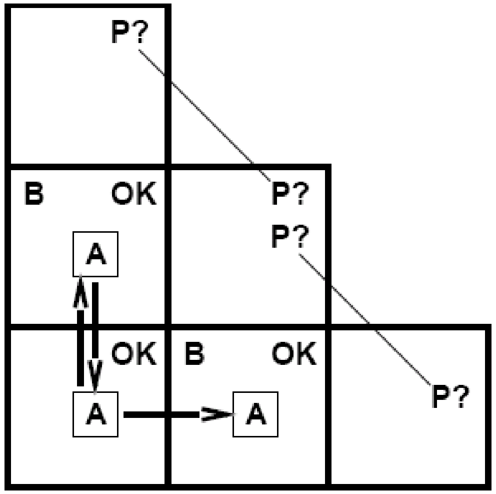
Wumpus world characterization

Observable	No, only local perception
Deterministic	Yes, outcomes exactly specified
Episodic	No, sequential at the level of actions
Static	Yes, wumpus and pits do not move
Discrete	Yes
Single-agent	Yes, wumpus is essentially a natural feature

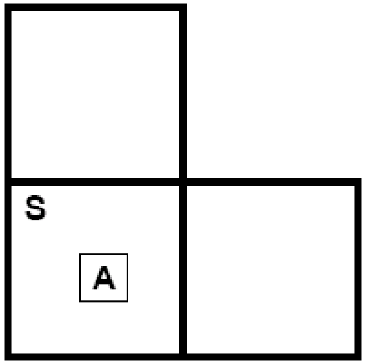
Exploring a wumpus world



Other tight spots



- Breeze in (1,2) and (2,1)
  ⇒ no safe actions
- Assuming pits uniformly distributed, (2,2) has pit w/ prob
  0.89 = 1– (1/3)*(1/3) vs.
  2/3 for (1,3) and (3,1)



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 such that conclusions can be drawn inference

Syntax語法 defines the sentences in the language. grammar

Semantics語義 define the “meaning” of sentences.

i.e. define truth of sentences in a world.

E.g. the language of arithmetic

- $x + 2 \geq y$ is sentence; $x2 + y >$ is not a sentence syntax error
- $x + 2 \geq y$ is true iff the number $x + 2$ is not less than the number y
- $x + 2 \geq y$ is true in a world where $x = 7$, $y = 1$
- $x + 2 \geq y$ is false in a world where $x = 0$, $y = 6$

Entailment

限定繼承

- Entailment means that one thing follows from another:

  KB $\models \alpha$

- Knowledge base KB entails sentence α if and only if (iff) α is true in all worlds where KB is true
- E.g. the KB containing "HKU won" and "CUHK won" entails "Either HKU won or CUHK won"
- E.g. x + y = 4 entails 4 = x + y
- Entailment is a relationship between sentences (i.e. syntax) that is based on semantics
- Note: brains process syntax (of some sort) function based on grammar

Models

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say *m* is a model of a sentence α if α is true in *m*

Model: a world state represented
by a logic sentence (with truth values)
– a Truth Table row

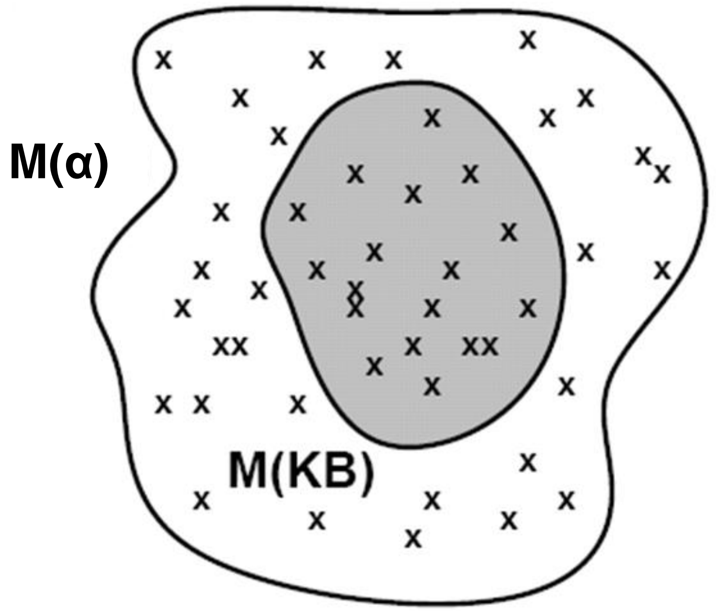
M(α) is the set of all models of α

Then *KB* ⊨ α iff *M(KB)* ⊆ *M*(α)

E.g. World: CUHK, HKU won or not?
4 possible models (TT, TF, FT, FF)

KB = CUHK won and HKU won (TT)
α = CUHK won *M*(α)=(TT, TF)

X: a model of a true sentence

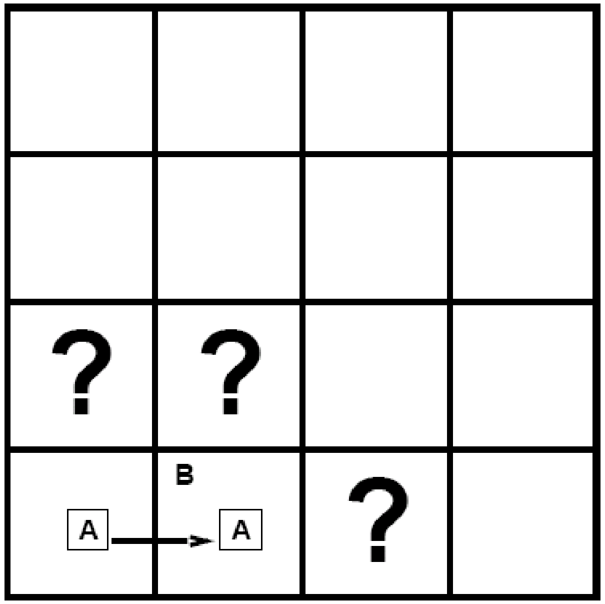


Entailment in the wumpus world

Situation after detecting nothing in [1,1] moving right, breeze in [2,1]

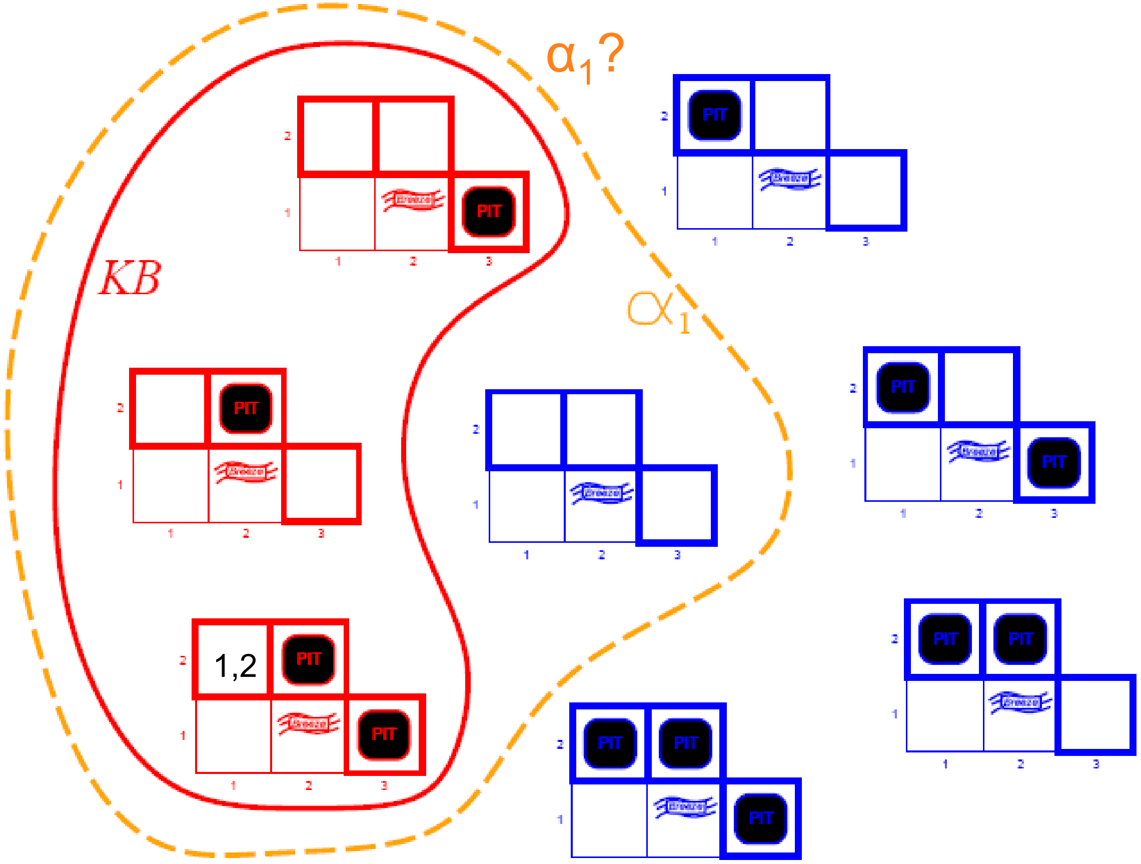
Consider possible models for ?s assuming only pits

3 Boolean choices ⇒ 8 possible models



Wumpus models

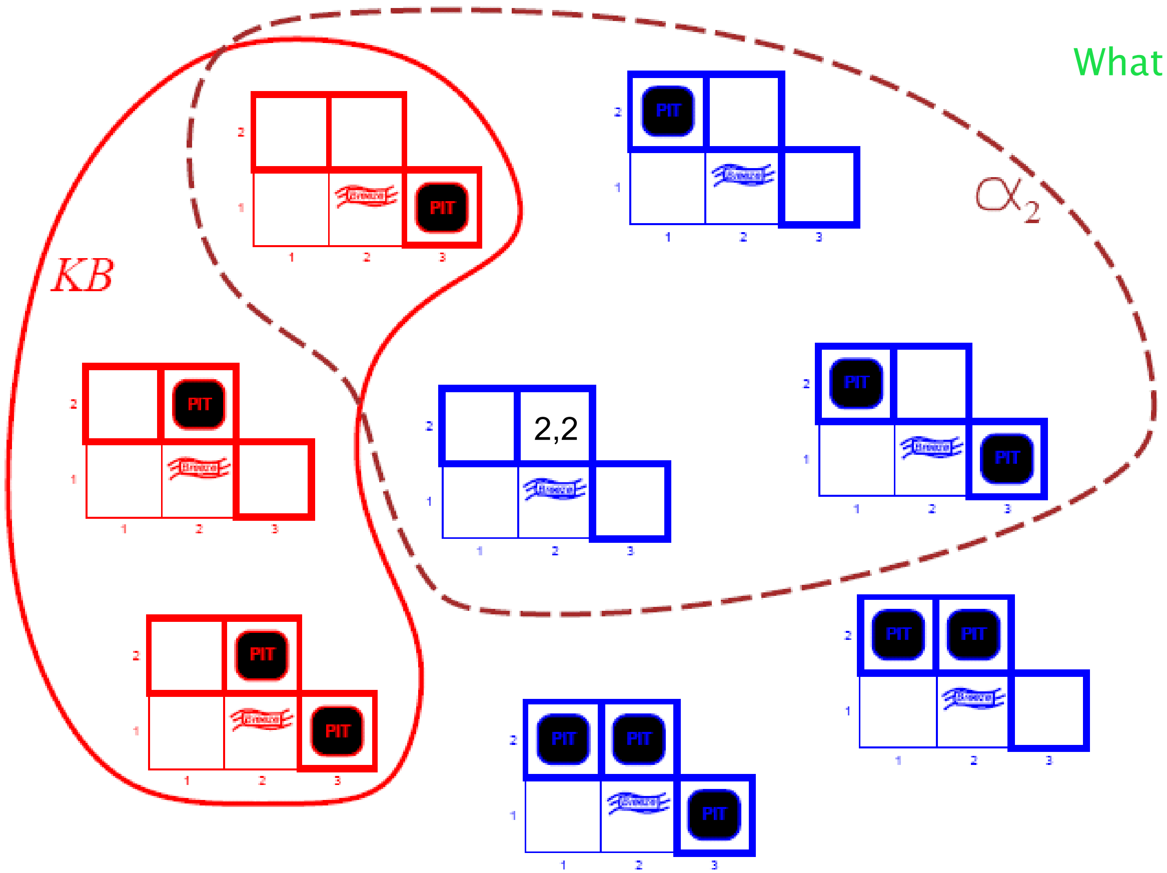
8 possible models



KB = wumpus-world rules + observations

α_1 = "[1,2] is safe", KB ⊨ α_1, proved by model checking enumeration

Wumpus models



What if α_2 covers *KB*?

KB = wumpus-world rules + observations

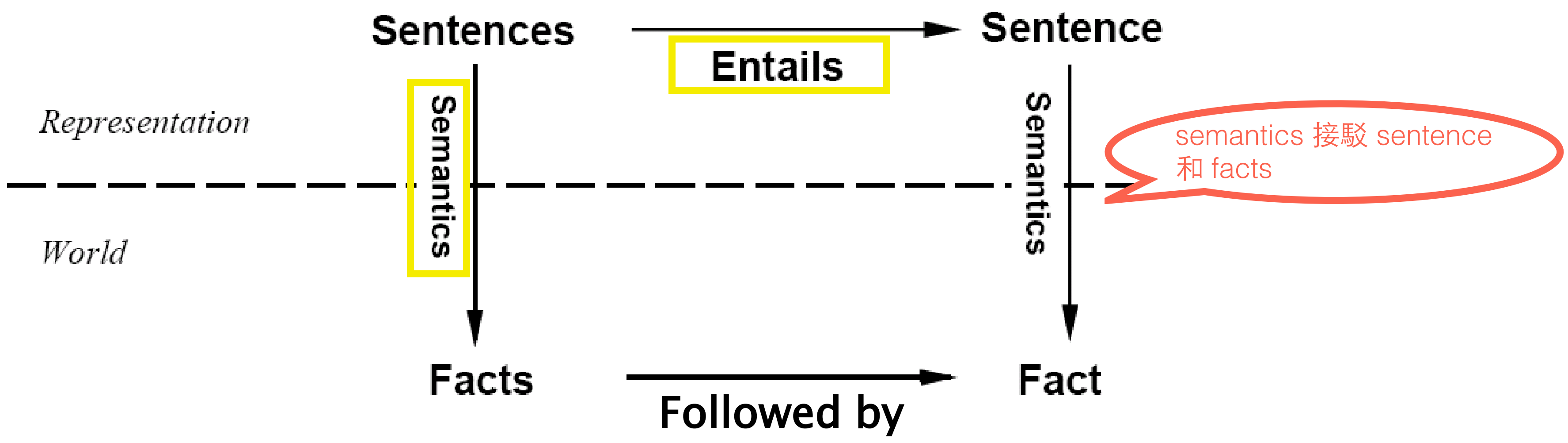
α_2 = "[2,2] is safe", KB $\not\models$ α_2 ??

Logic

- The syntax of a language describes the possible configurations that can constitute sentences.
- The semantics determines the facts in the world to which the sentences refer.

Provided the syntax and semantics are defined precisely, we can call the language a logic. From the syntax and semantics, we can derive an inference mechanism for an agent that uses the language.

Logic



mapping

- The combination between sentences and facts is provided by the semantics of the language.
- The property of one fact following from some other facts is mirrored by the property of one sentence being entailed by some other sentences.
- Logical inference generates new sentences that are entailed by existing sentences. ⇔ reasoning in real world

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from *KB* by procedure *i*

Consequences of *KB* are a haystack; α is a needle.

Entailment = needle in haystack; inference = finding it

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

a logical system has the soundness property if and only if its inference rules prove only formulas that are valid with respect to its semantics

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

In logic, semantic completeness is the converse of soundness for formal systems.

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

Propositional logic: Syntax

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

Many rule-based Expert systems!

Propositional logic: Semantics

Each model specifies true/ false for each proposition symbol

Pit in 1,2

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
true true false

(With these symbols, 8 possible models, can be enumerated automatically.)

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
i.e.,	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	$S_2 \Rightarrow S_1$	is true

Simple recursive process evaluates an arbitrary sentence, e.g.

$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = true \wedge (false \vee true) = true \wedge true = true$

Truth tables for connectives

P	Q	¬P	P ∧ Q	P ∨ Q	P ⇒ Q	P ⇔ 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

rule base systems

F⇒ F ??

Wumpus world sentences

KB:

Let $P_{i,j}$ be true if there is a pit in [i, j]

Let $B_{i,j}$ be true if there is a breeze in [i, j]

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

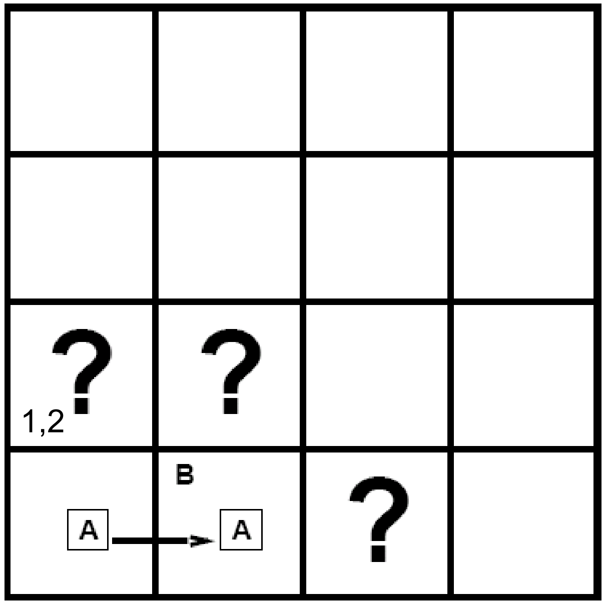
"Pits cause breeze in adjacent squares"

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

$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

"A square is breezy iff there is an adjacent pit"

Question: $\alpha_1 = \neg P_{1,2}$; $KB \models \alpha_1$?



Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	false	false	false	false	false	false	false	true
false	false	false	false	false	false	true	false	true
…	…	…	…	…	…	…	…	…
false	true	false	false	false	false	false	false	true
false	true	false	false	false	false	true	true	true
false	true	false	false	false	true	false	true	true
false	true	false	false	false	true	true	true	true
false	true	false	false	true	false	false	false	true
…	…	…	…	…	…	…	…	…
true	true	true	true	true	true	true	false	false

$\alpha_1 = \neg P_{1,2}$

KB is true when all the 5 sentences in KB is true

KB $\vDash \alpha_1$ why?

$\neg P_{1,1}$; $\neg B_{1,1}$; $B_{2,1}$; $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$; $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

iff *M(KB)* $\subseteq$ *M(α)*

Goto p.32

Inference by enumeration

Build the complete truth table by recursive calls forming a depth-first tree(?)

Depth-first enumeration of all models is sound and complete

```
function TT-Entails? ($KB$, $\alpha$) returns true or false
    symbols ← a list of proposition symbols in $KB$ and $\alpha$
    return TT-Check-All ($KB$, $\alpha$, symbols, [ ] )
```

```
function TT-Check-All($KB$, $\alpha$, symbols, model) returns true or false
    if Empty?(symbols) then  // row completed in TT
        if PL-True?($KB$, model) then return PL-True?($\alpha$, model) // $T \Rightarrow T/F \Leftrightarrow T/F$
        else return true // $F \Rightarrow T/F \Leftrightarrow T$
    else do
        $P$ ← First(symbols); rest ← Rest(symbols)
        return TT-Check-All($KB$, $\alpha$, rest, Extend($P$, true, model)) and
                    TT-Check-All($KB$, $\alpha$, rest, Extend($P$, false, model))
```

$O(2^n)$ for n symbols; problem is co-NP-complete

S

T F

T F T F

T F T F T F T F

TTT	FTT
TTF	FTF
TFT	FFT
TFF	FFF

Logical Equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$ 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 $\vee$
$\neg(\neg \alpha)$	$\equiv$	α 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 $\wedge$

all models some models

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:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ 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:

$KB \models \alpha$ iff $(KB \land \neg \alpha)$ is unsatisfiable; $(\neg(\neg KB \lor \alpha) \equiv KB \land \neg \alpha)$

$(\neg(KB \Rightarrow \alpha))$

i.e. prove α by *reductio ad absurdum* (reduction to an absurd thing)

- prove by refutation or by contradiction

Proof methods

Proof methods divide into (roughly) two kinds:

- 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 algorithm
  - Typically require translation of sentences into a normal form
- 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

Forward and backward chaining

Horn Form (restricted)

KB = conjunction of Horn clauses

Horn clause =

- Proposition symbol; or
- (conjunction of symbols) ⇒ symbol

E.g. $C \wedge (\underline{B \Rightarrow A}) \wedge (\underline{C \wedge D \Rightarrow B})$

肯定前件

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

$$\frac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

inference rule

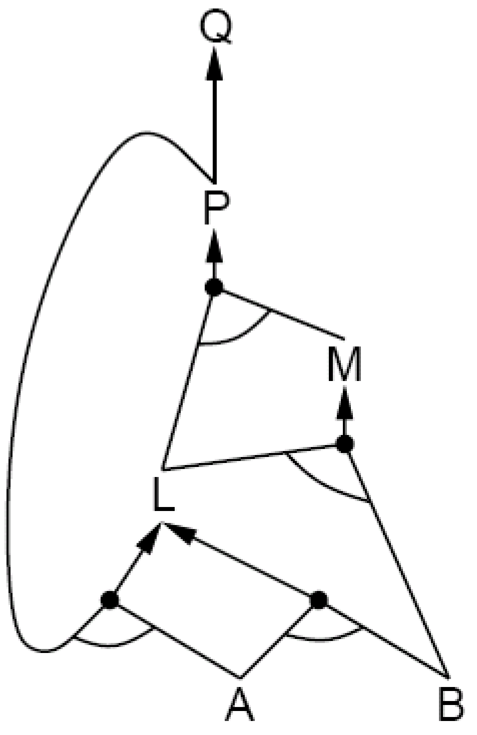
Can be used with forward chaining or backward chaining.
There algorithms are very natural and run in linear time

Forward chaining

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

Rules, implications
Facts
Antecedent-consequence
-conclusion
Premises: elements of Ante.

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



Where to check Q first??

Forward chaining algorithm

```
function PL-FC-Entails? (KB, q) return true or false //q=query
    local variables:
            count, a table, indexed by clause, initially the number of premises //clause=rule
            inferred, a table, indexed by symbol, each entry initially false
            agenda , a list of symbols, initially the symbols known to be true //fact base

    while agenda is not empty do
        p ← Pop(agenda)
        unless inferred[p] do  //check for repeating inferred symbols
            inferred[p] ← true
            for each Horn clause c in whose premise p appears do //matched premise
                decrement count[c]  //cth rule
                if count[c] = 0 then do //cth rule fired
                    if Head[c] = q then return true
                    Push (Head[c], agenda)
    return false
```

Forward chaining algorithm

- Forward-chaining(FC) for proposition logic(PL)
- *agenda*: symbols know to be true but not yet "processed"
- *count*: how many premises of each implication are yet unknown
- Whenever a new symbol p from the agenda is processed, the count is reduced by 1 for each implication in whose premise p appears.
- If a count = 0, all the premises of the implication are known, so fire and its conclusion is added to the agenda
- Keep track of symbols processed; an inferred symbol need not to be added to the agenda if it has been processed previously. This avoids redundant work and prevents infinite loops such $P \Rightarrow Q$ and $Q \Rightarrow P$

Forward chaining Example

Agenda

A

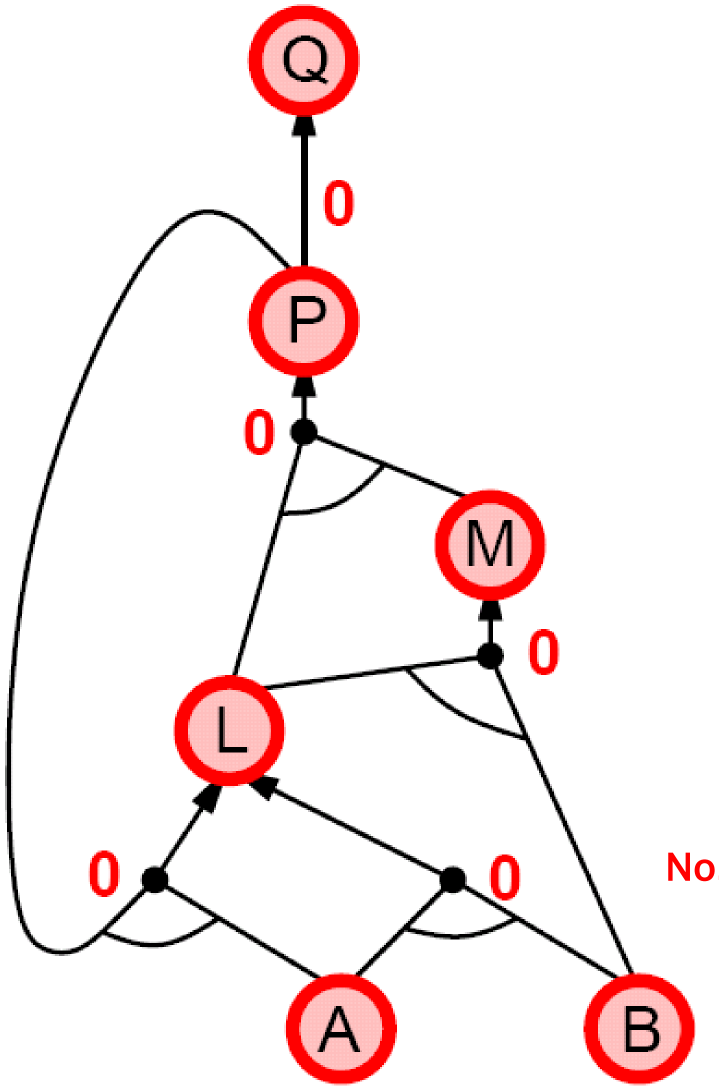
B

L

M

P

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

Nos.: counts of unknown nos. of premises of rules

Backward chaining

Idea: work backwards from query *q*:

- to prove *q* by BC,

1. check if *q* is known already, or
2. prove by BC all premises of some rule concluding *q*

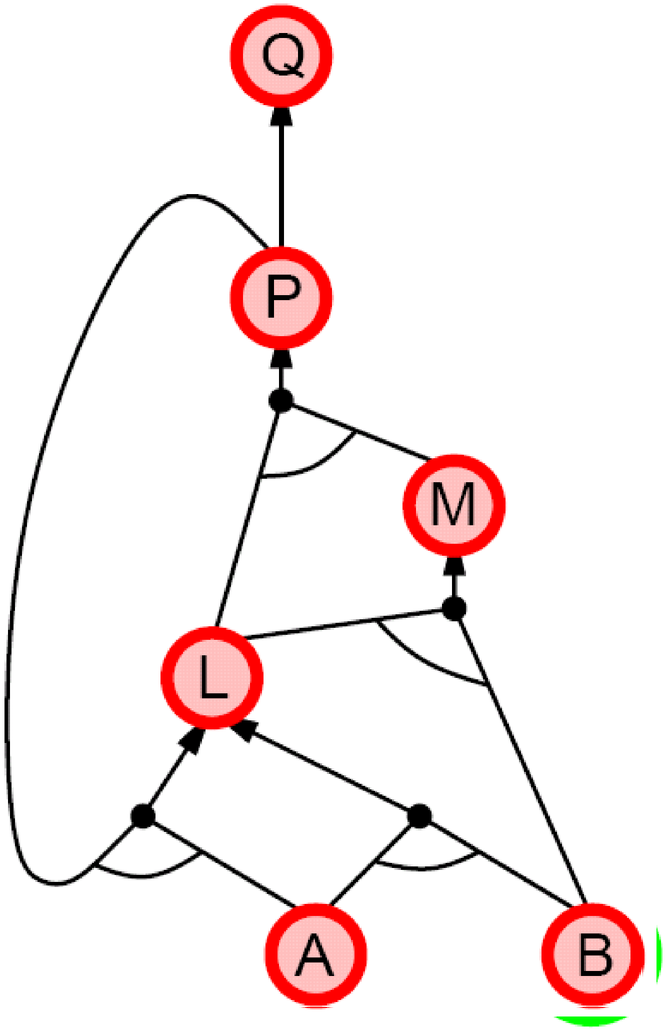
Avoid loops: check if new sub-goal is already on the goal stack

Avoid repeated work: check if new sub-goal

1. has already been proved true or
2. has already failed

Backward chaining example

	Goal
Fact	*Stack*
A	A
B	B
L	L
M	M
P	P
Q	Q



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

Forward vs. backward chaining

Forward chaining:

- FC is data-driven, cf. automatic, unconscious processing, e.g. object recognition, routine decisions
- May do lots of works that is irrelevant to the goal

Backward chaining:

- 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 "disjunctions of literals" (clauses)

E.g. $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

Cf. Horn Form (clauses)

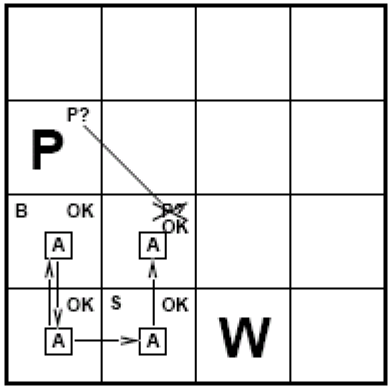
Resolution inference rule (for CNF): complete for propositional logic

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

where l_i and m_j are complementary literals. E.g.

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

Resolution is sound and complete for propositional logic



Conversion to CNF

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

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

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

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$.

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

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

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

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

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

Resolution algorithm

To proof KB entails α

Proof by contradiction, i.e. show KB ∧ ¬ α unsatisfiable

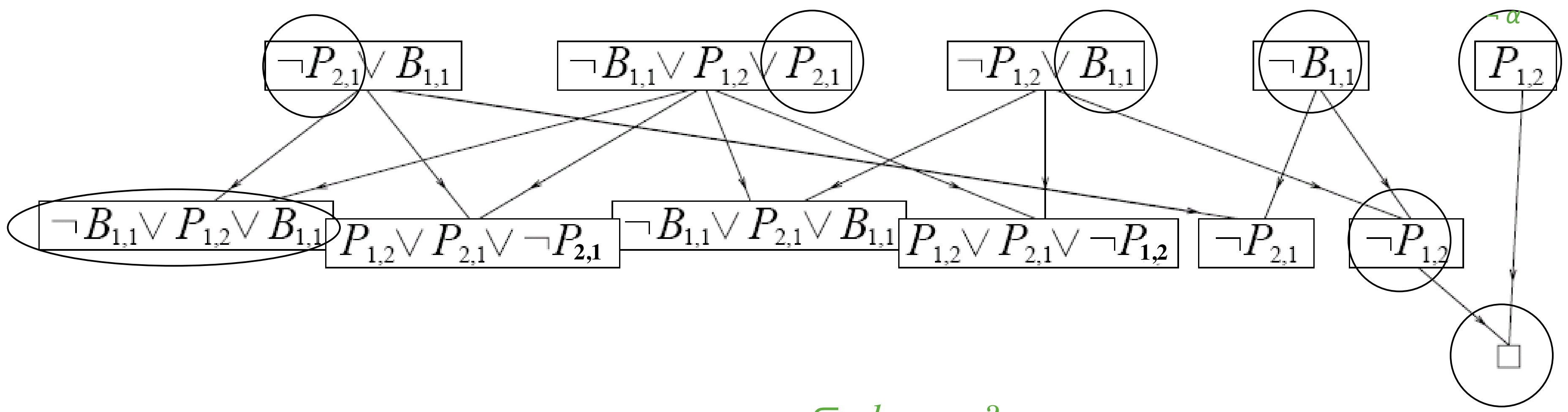
```
function PL-Resolution (KB, α ) returns true or false
    clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
    new ← {}
    loop do
      for each $C_i$ , $C_j$ in clauses do
        resolvents ← PL-Resolve ($C_i$ , $C_j$)
        if resolvents contains the empty clause then return true
        new ← new ∪ resolvents
     if new ⊆ clauses then return false //no new resolvent generated ⇒ proof failed
     clauses ← clauses ∪ new
```

1- 在知識庫中所有句子和要證明的句子(猜測(conjecture))的否定都合取連結。
2- 結果的句子變換成合取範式(處理成一組子句)。
3- 把歸結規則應用到包含互補的文字的所有可能的子句對，通過除去重複的文字來簡化結果的句子。如果句子包含互補的文字，則(作為重言式)丟棄它。如果沒有，並且它在子句的集合中仍然不存在，則增加上它，並考慮做進一步的歸結推理。
4- 如果在應用歸結規則之後推導出空子句，則全部的公式是不可滿足的(或者說矛盾的)，所以可以做出最初的猜測從這些公理中推出的結論。
5- 在另一方面，如果不能推導出空子句，並且不能應用歸結規則推導更多的新子句，則這個猜測不是最初的知識庫的定理

Resolution example

No breeze in 1,1. Prove NO Pit in 1,2.

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



$\neg\alpha$
$\neg P_{2,1} \vee B_{1,1}$
$\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$
$\neg P_{1,2} \vee B_{1,1}$
$\neg B_{1,1}$
$P_{1,2}$
$\neg B_{1,1} \vee P_{1,2} \vee B_{1,1}$
$P_{1,2} \vee P_{2,1} \vee \neg P_{2,1}$
$\neg B_{1,1} \vee P_{2,1} \vee B_{1,1}$
$P_{1,2} \vee P_{2,1} \vee \neg P_{1,2}$
$\neg P_{2,1}$
$\neg P_{1,2}$
$\square$

new $\subseteq$ *clauses ?*

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 with respect to 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
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Forward, backward chaining are linear-time, complete for Horn clauses
- Resolution is complete for propositional logic
- Propositional logic lacks expressive power