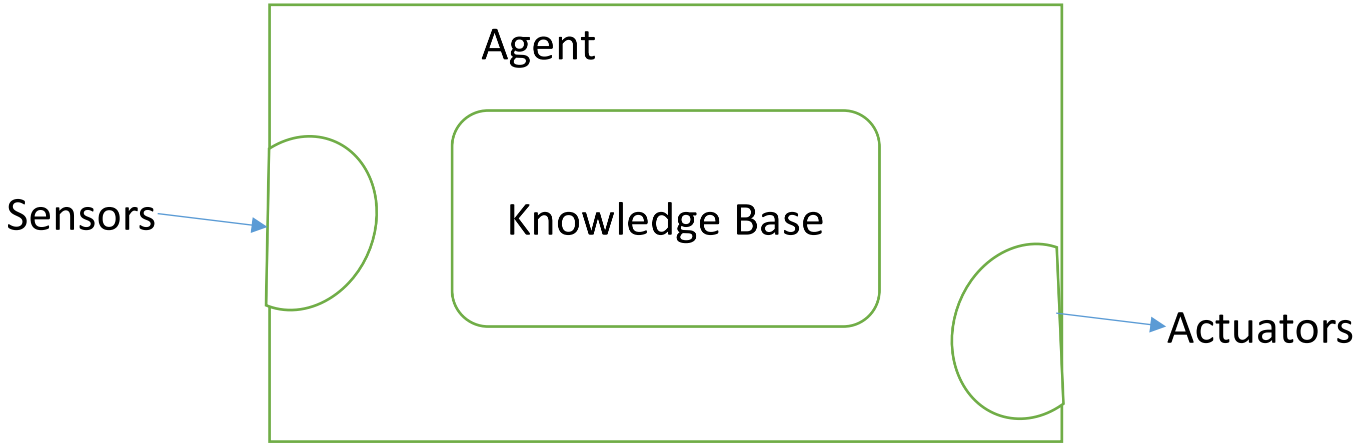
Logical Agents/ Knowledge based Agents



Agent

Sensors

Knowledge Base

Actuators

Knowledge base: Knowledge Representation and Reasoning system

KBA can accept new tasks in the form of explicitly described goals

can achieve competence quickly by being told or learning knowledge about the environment

can adapt to changes in the environment by updating the relevant knowledge

KBA adds percept to its KB, asks KB for the best action, and tells KB it has taken a specific action

Logical Agents/ Knowledge based Agents

Agents that can form **representations** of a complex world

Use a process of **inference** to derive new representations about the world

And use these new representations to **deduce** what to do

Internal representation of Knowledge, Reasoning

KB - A set of sentences,

Each sentence

expressed in knowledge representation language

represents some assertion about the world

TELL - add new sentences to KB

ASK - query what is known

Tell, Ask may involve - inference – deriving new sentences from old

when one asks a question of the KB, the answer should follow from what has been told(Tell'ed) to KB previously

(Axiom - when the sentence is taken as given, without being derived from other sentences)

Logical Agents/ Knowledge based Agents

```
function KB-AGENT(percept ) returns an action
        persistent: KB, a knowledge base
                t , a counter, initially 0, indicating time
        TELL(KB,MAKE-PERCEPT-SENTENCE(percept , t ))
        action ←ASK(KB,MAKE-ACTION-QUERY(t ))
        TELL(KB,MAKE-ACTION-SENTENCE(action, t ))
        T←t + 1
        return action
```

A generic knowledge-based agent.

Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

Building KB

Building KB/ agent designer

TELL what it needs to know. TELL sentences one by one until agent knows how to operate in an environment

declarative approach to system building

add mechanisms to allow KB to learn for itself from a series of percepts

Declarative Approach to system building

(procedural approach – encode desired behaviours directly as program code)

(can combine./ compile declarative knowledge to procedural code)

Logic

Concepts of logical representation and reasoning

Syntax (notion of syntax)

X + Y = 4   is a well formed sentence (in arithmetic)

X4Y+=   is not a well formed sentence

KB consists of sentences.

Sentences are formed according to the syntax of the representation language

(specifies all the sentences that are well formed)

Semantics - Meaning of sentences

defines truth of each sentence with respect to each possible world

sentence X + Y = 4 is true in a world where X = 2, Y = 2   (Semantics in arithmetic)

false in a world where X = 1, Y = 1

In Standard Logics, every sentence must be true or false in each possible world – no in-between

Logic - Possible World, Models

Possible world – real environments that the agent might or might not be in

Model – mathematical abstraction – models fix truth or false hood of every relevant sentence

Possible world:

Ex: X men, Y women sitting and playing bridge

sentence: $X + Y = 4$ - true when there are 4 people in total

Possible worlds/models: all possible assignments to the variables X, Y

Each such assignment fixes the truth of any sentence of arithmetic whose variables are X, Y

If a sentence α is true in model m, we say M satisfies α or m is a model of α

$M(\alpha)$: set of all models of α

Logical Reasoning

Logical reasoning - relation of logical entailment between sentences

idea – A sentence follows logically from another sentence

sentences α, β

$\alpha \models \beta$ sentence α entails β , if and only if , in every model in which α is true, β is also true

$M(\alpha) \subseteq M(\beta)$ α is a stronger assertion than β

Relation of entailment:

if $X = 0$, then $X \times Y = 0$

sentence $X = 0$ entails sentence $X \times Y = 0$

Wumpus World

Environment, rules

- Cave consisting of rooms connected by passage ways (4 x 4 grid)
- Wumpus (W), a beast somewhere in the cave. Eats anyone who enters it's room
- W can be shot by agent(A). A has only one arrow
- Some rooms contain bottom less pits(P). P traps anyone who wanders into these rooms (except for W, which is too big to fall in)
- Possibility of finding a heap of Gold(G), G glitters in the room
- Agent always starts in [1,1] facing to right
- locations of W, G are chosen randomly with uniform distribution (from the squares other than [1,1]
- Each square, other than [1,1] can be a pit with probability 0.2

Performance measures:

+1000 for climbing out of cave with Gold

- 1000 for falling into a Pit or being eaten by W

- 1 for each action taken

-10 for using up the arrow

game ends either when the agent dies or when the agent climbs out

Wumpus World

Actuators

Agent can move forward, turn left by 90^o or turn right by 90^o

agent dies if he enters a room with live Wumpus or a Pit. (safe to enter a room with dead W)

agent does not move if he bumps into a wall

action grab to pick up gold, if it is in the same room as agent

Action shoot can be used to fire an arrow in a straight line in the direction agent is facing

arrow continues until it hits a wall or W (hence kills W)

Agent has only one arrow

action climb to climb out of cave, but only from cave [1,1]

Wumpus World

Sensors : 5 sensors, each of which gives a single bit of information

in the square containing W, and directly adjacent (Horizontal, Vertical) squares, agent will perceive stench (S)

in the squares adjacent to a P, agent will perceive a breeze (B)

in the square where gold is, agent perceives glitter (G)

when the agent walks into a wall, A perceives bump (Bm)

When W is killed, it emits a scream(Sm) that is perceived anywhere in the cave

Percepts – a list of 5 symbols

[Stench, Breeze, Glitter, Bump, Scream]

Wumpus World

Discrete, Static, Single Agent, Sequential,

Partially Observable (locations of pits, agents location, W's status of health).

transition model is unknown as Agent does not know which forward actions are fatal

Wumpus World

Challenge – initial ignorance of the environment.

- Overcoming ignorance requires logical reasoning
- it is possible to retrieve gold safely
- Agent may chose to go home empty or risk death to find gold
- 21% of environments are utterly unfair, as gold is in a pit or surrounded by pits

Knowledge based Wumpus Agent

Informal KB Language:

A – Agent , B - Breeze, G – Glitter, OK – Safe square, P – Pit,

S – Stench, V - Visited, W - Wumpus

Initial knowledge base: Agent knows

rules of environment

knows it is in [1,1] and [1,1] is safe

state: A, OK

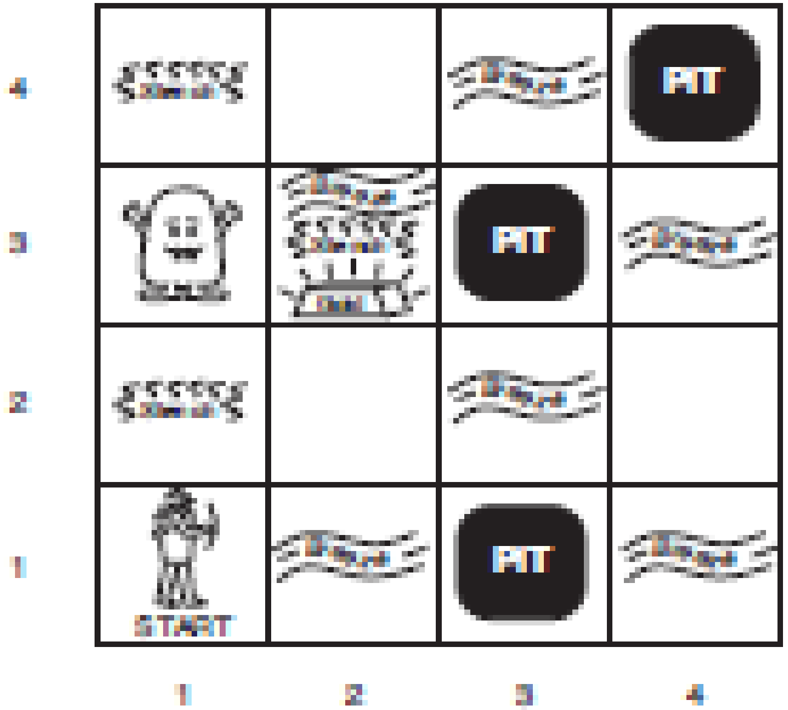
First percept : [none, none, none, none, none]

Reasoning/conclusion from Percepts and KB:

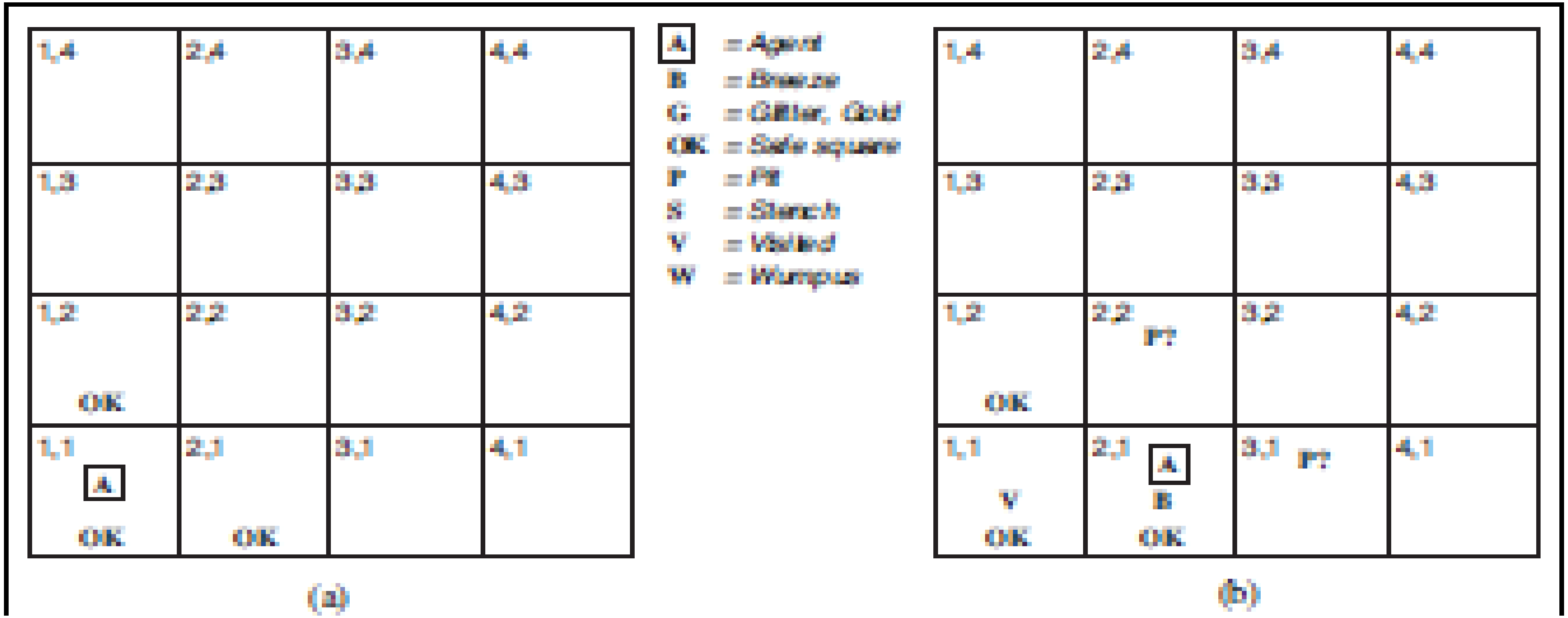
neighbouring squares [1,2], [2,1] are free from danger.

Next move?

cautious agent moves into square it knows is OK.



Wumpus world



The first step taken by the agent in the wumpus world

(a) The initial situation, after percept [None, None, None, None, None]

(b) After one move, with percept [None, Breeze, None, None, None]

Reasoning

A moves to [2,1].

In [2,1] A senses Breeze.

there must be a pit in a neighbouring square

[1,1] can not be a pit – KB – rules of game

[2,2] or [3,1] a pit? Or both?

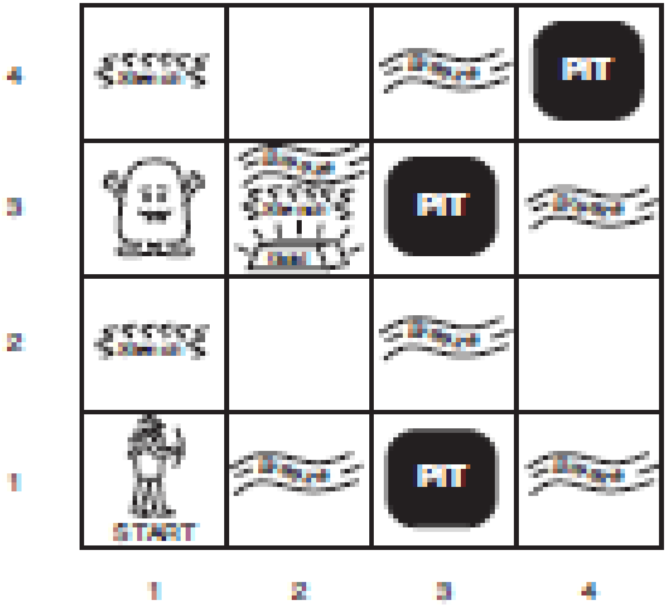
Prudent agent

only one known square that is OK. [1,2]

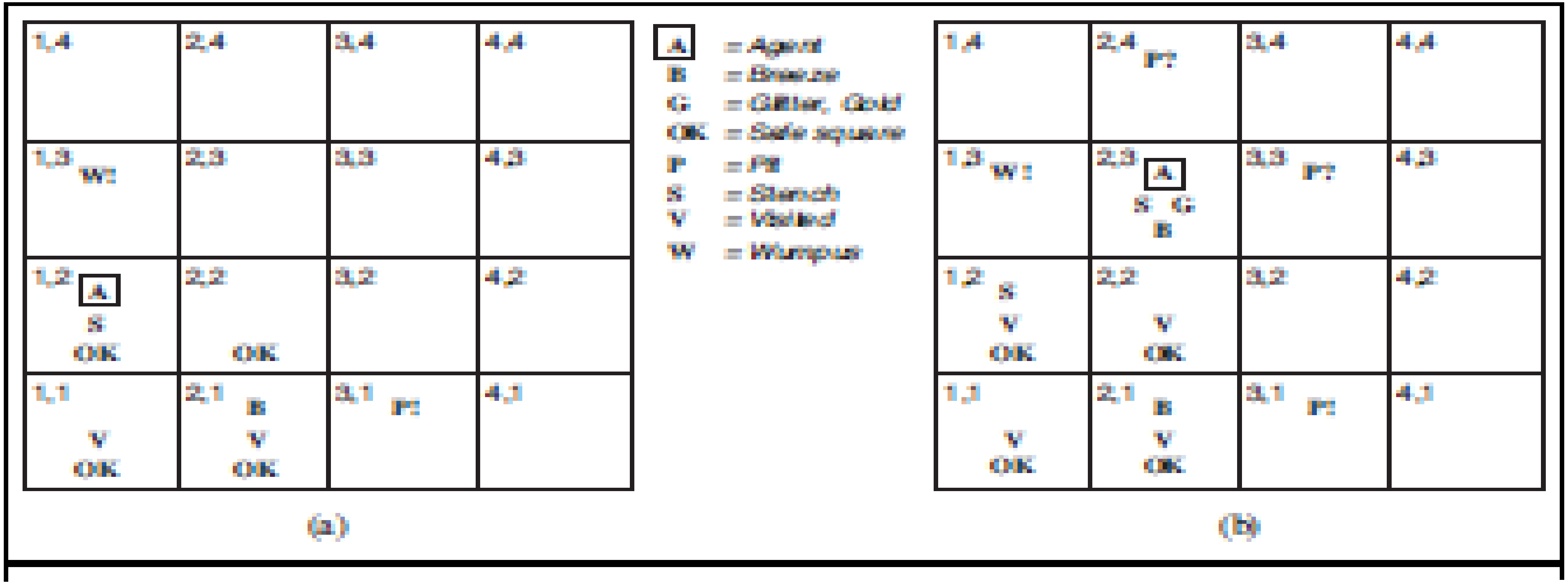
Action?

turn around and go back to [1,1]

And then proceed to [1,2]



Wumpus World



1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(a)

A	= Agent
B	= Breeze
G	= Glitter, Gold
OK	= Safe square
P	= Pit
S	= Stench
V	= Visited
W	= Wumpus

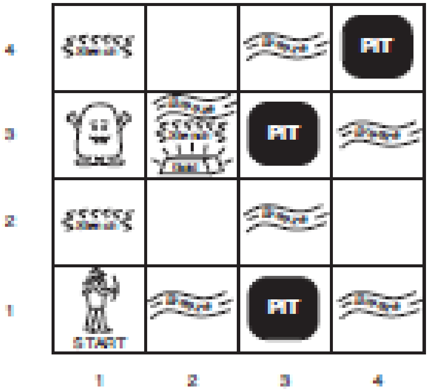
1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(b)

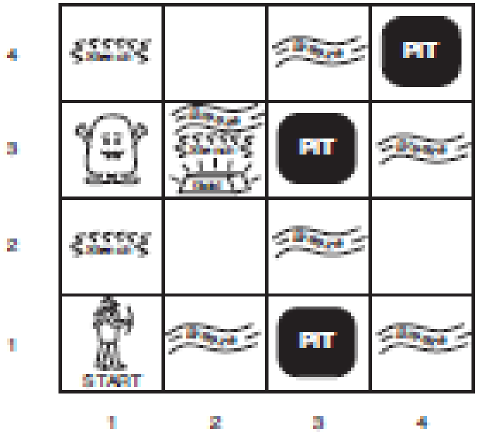
Two later stages in the progress of the agent.

(a) After the third move, with percept [Stench, None, None, None, None].

(b) After the fifth move, with percept [Stench, Breeze, Glitter, None, None].



Reasoning



A moves to [1,2].

In [1,2] A senses Stench. Percepts – [stench, none, none, none, none]

there must be Wumpus in a neighbouring square

[1,1] can not have Wumpus – KB – rules of game

Wumpus can not be in [2,2] (or, the agent would have detected stench when it was in [2,1]

Agent's inference

I1 - Wumpus is in [1,3] W! in [1,3]

I2 - lack of breeze in [1,2] inference - there is no pit in [2,2]

(agent has already inferred from action1 that there must be a pit in either [2,2] or [3,1]

I3 – I2 and action 1 inference -> pit must be in [3,1] -

I2, I3 - difficult inference – agent combines knowledge gained at different times in different places and relies on lack of percept to make one crucial step

I4 – there is neither a pit nor a Wumpus in [2,2]

Action? move to [2,2] . Percepts [none, none, none, none, none]

Reasoning

Next

Agent moves to [2,3]. Percepts – [stench, breeze, glitter, none, none]

Action?

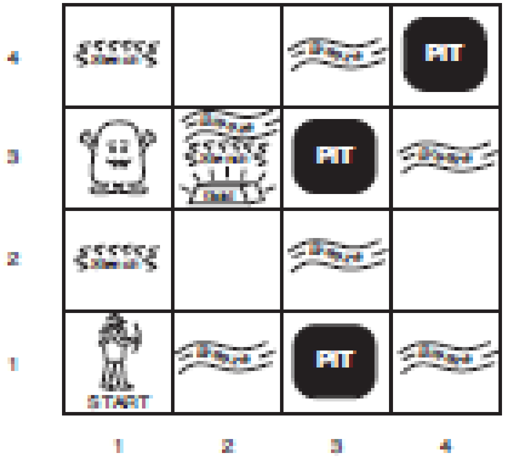
grab gold

return home

In each case agent draws a conclusion from available information

Conclusion is guaranteed to be correct if the available information is correct

- fundamental property of logical reasoning



Logic – inference algorithm: model checking

Agent has detected

nothing in [1,1], breeze in [2,1]

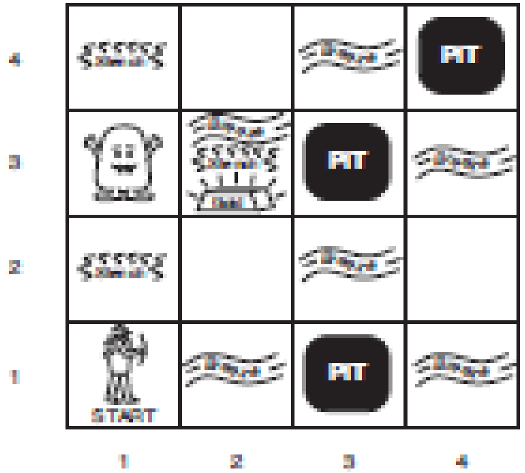
Knowledge base: These percepts combined with Agents knowledge of rules of the wumpus world constitute KB

A set of sentences

Agent is interested in weather the adjacent squares [1,2], [2,2], [3,1] contain pits

each of these might or might not contain a pit.

Possible models? $2^3 = 8$



Sentences: (possible conclusions)

α1: there is no pit in [1,2]

α2: there is no pit in [2,2]

KB is false in models that contradict what the agent knows

KB is false in any model in which [1,2] contains a pit, as there is no breeze in [1,1]

KB is true in just 3 models

Wumpus World

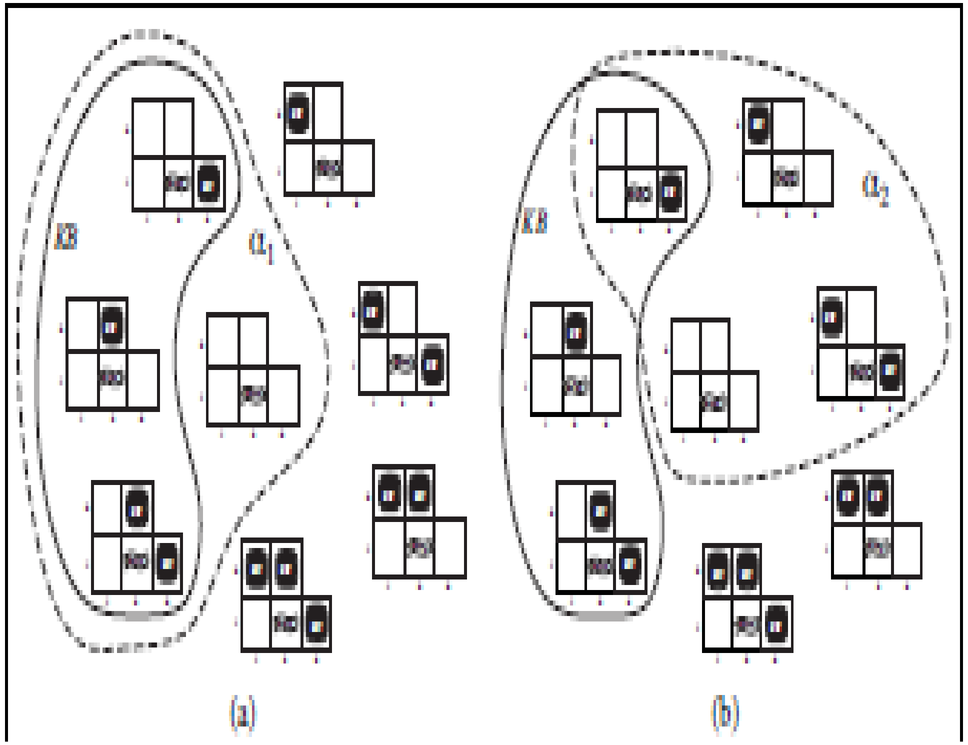
Possible models for the presence of pits in squares [1,2], [2,2], and [3,1].

The KB corresponding to the observations of nothing in [1,1]

and a breeze in [2,1] is shown by the solid line.

(a) Dotted line shows models of α1 (no pit in [1,2]).

(b) Dotted line shows models of α2 (no pit in [2,2]).



In every model KB is true, α1 is true

Hence KB ⊨ α1 : there is no pit in [1,2]

In some models in which KB is true, α2 is false

Hence KB |≠ α2 :

agent can not conclude that there is no pit in [2,2]

Entailment applied to derive conclusions – carry out logical inferences

Model checking/inference algorithm – enumerates all possible models to check that α is true in all models in which KB is true. M(KB) ⊆ M(α)

Entailment, inference

Set of all consequences – haystack

α - needle

Entailment – needle in haystack

Inference - finding α in KB / needle in haystack

$KB \vdash_i \alpha$: If an inference algorithm, i can derive α from KB

Sound/truth preserving whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

i that derives only entailed sentences is sound or truth preserving

Model checking is sound procedure

Completeness whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

i is complete if it can derive any sentence that is entailed

If KB is true in the real world, then any sentence a that is derived from KB by a sound inference procedure is also true in the real world

Inference

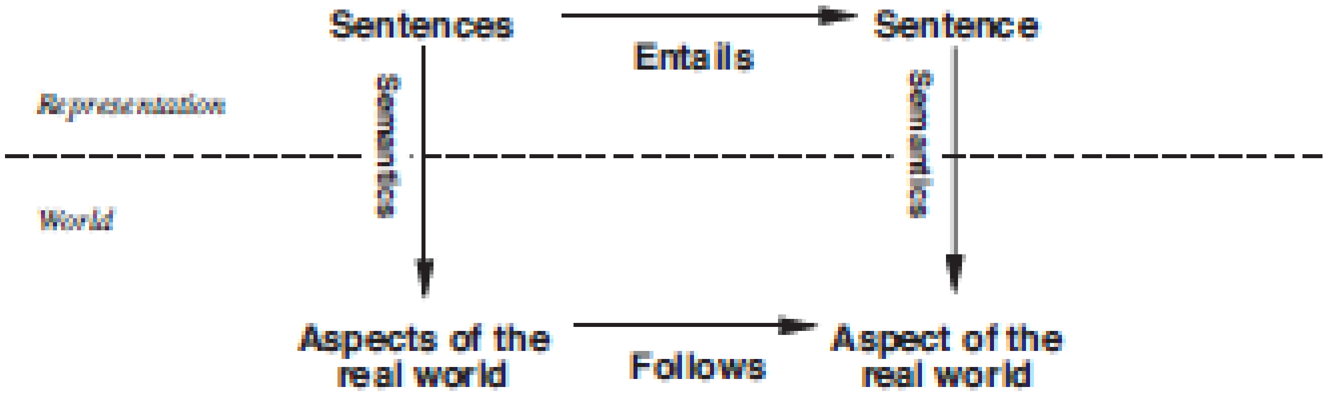
Inference process works on syntax

Grounding

Connection between logical reasoning and the real environment in which the agent exists.

Agents sensors create connection between KB and real world

Agents knowledge is also built from learning – sentence construction process from perceptual learning



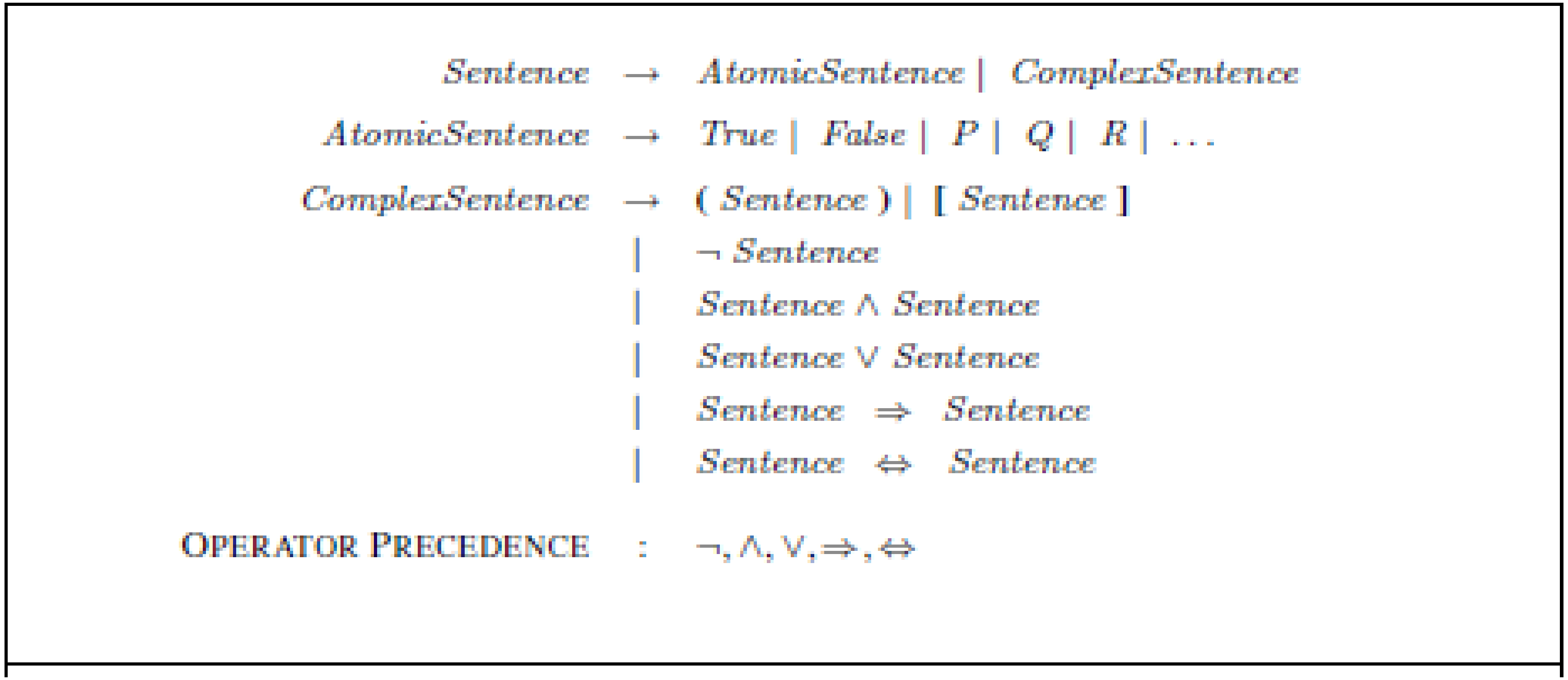
Propositional Logic

Syntax, Semantics, Entailment, Logical Inference

Syntax: defines allowable sentences – atomic sentences, complex sentences, connectives

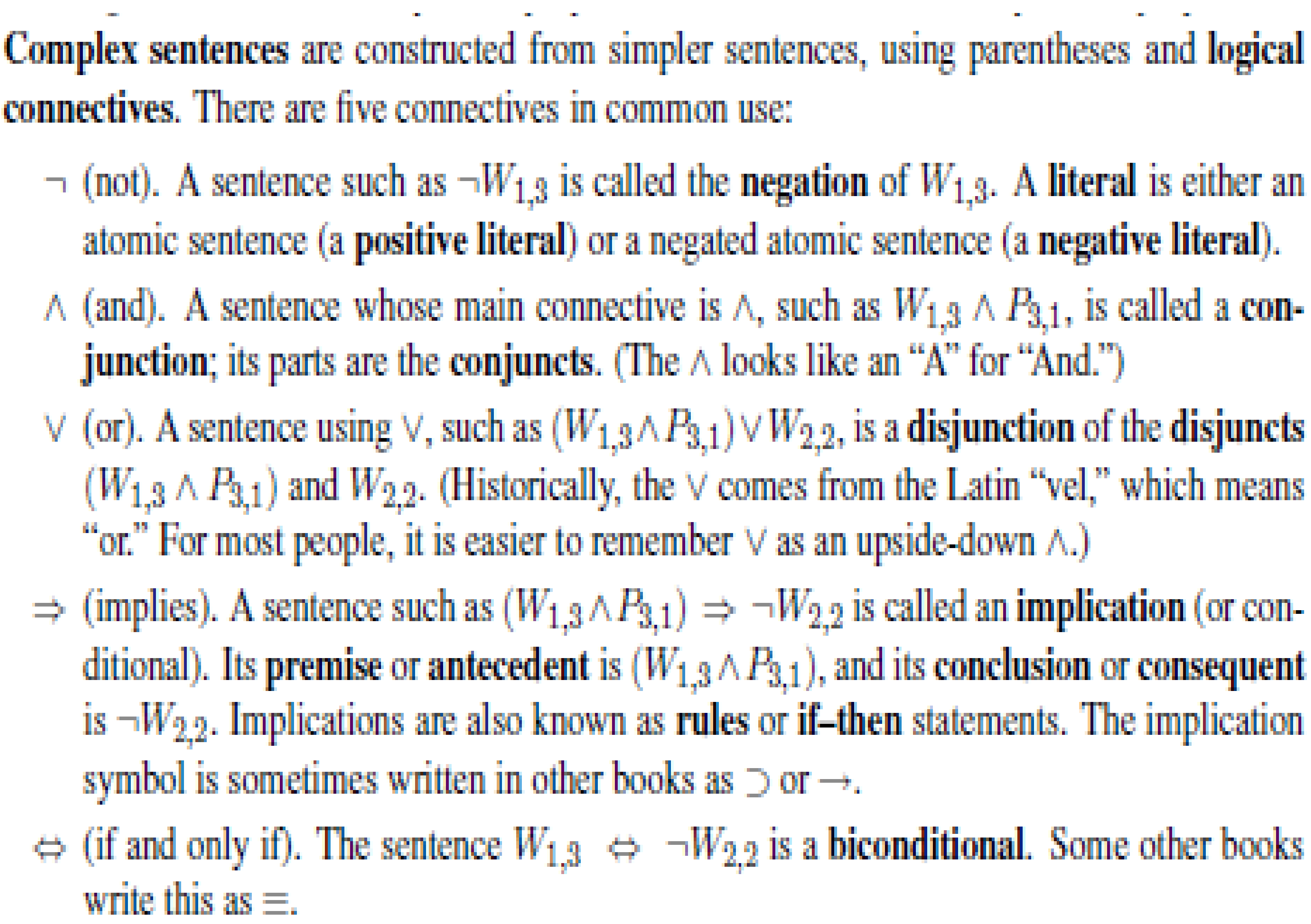
Ex: P, Q, R, $W_{1,3}$, North ; $\neg W_{1,3}$, $W_{1,3} \wedge P_{3,1}$, $(W_{1,3} \wedge P_{3,1})\ V\ W_{2,2}$

A BNF (Backus–Naur Form) grammar of sentences in propositional logic, along with operator precedence, from highest to lowest.



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

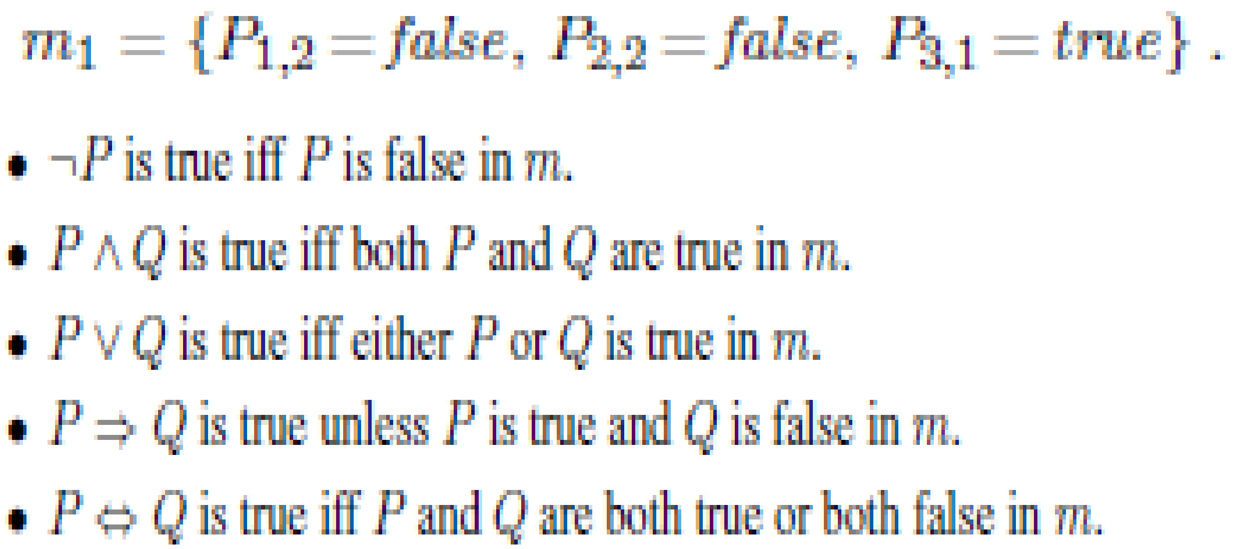
PL



Complex sentences are constructed from simpler sentences, using parentheses and **logical connectives**. There are five connectives in common use:

- $\neg$ (not). A sentence such as $\neg W_{1,3}$ is called the **negation** of $W_{1,3}$. A **literal** is either an atomic sentence (a **positive literal**) or a negated atomic sentence (a **negative literal**).
- $\wedge$ (and). A sentence whose main connective is $\wedge$, such as $W_{1,3} \wedge P_{3,1}$, is called a **conjunction**; its parts are the **conjuncts**. (The $\wedge$ looks like an "A" for "And.")
- $\vee$ (or). A sentence using $\vee$, such as $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$, is a **disjunction** of the **disjuncts** $(W_{1,3} \wedge P_{3,1})$ and $W_{2,2}$. (Historically, the $\vee$ comes from the Latin "vel," which means "or." For most people, it is easier to remember $\vee$ as an upside-down $\wedge$.)
- $\Rightarrow$ (implies). A sentence such as $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an **implication** (or conditional). Its **premise** or **antecedent** is $(W_{1,3} \wedge P_{3,1})$, and its **conclusion** or **consequent** is $\neg W_{2,2}$. Implications are also known as **rules** or **if–then** statements. The implication symbol is sometimes written in other books as $\supset$ or $\rightarrow$.
- $\Leftrightarrow$ (if and only if). The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a **biconditional**. Some other books write this as $\equiv$.

PL - Semantics



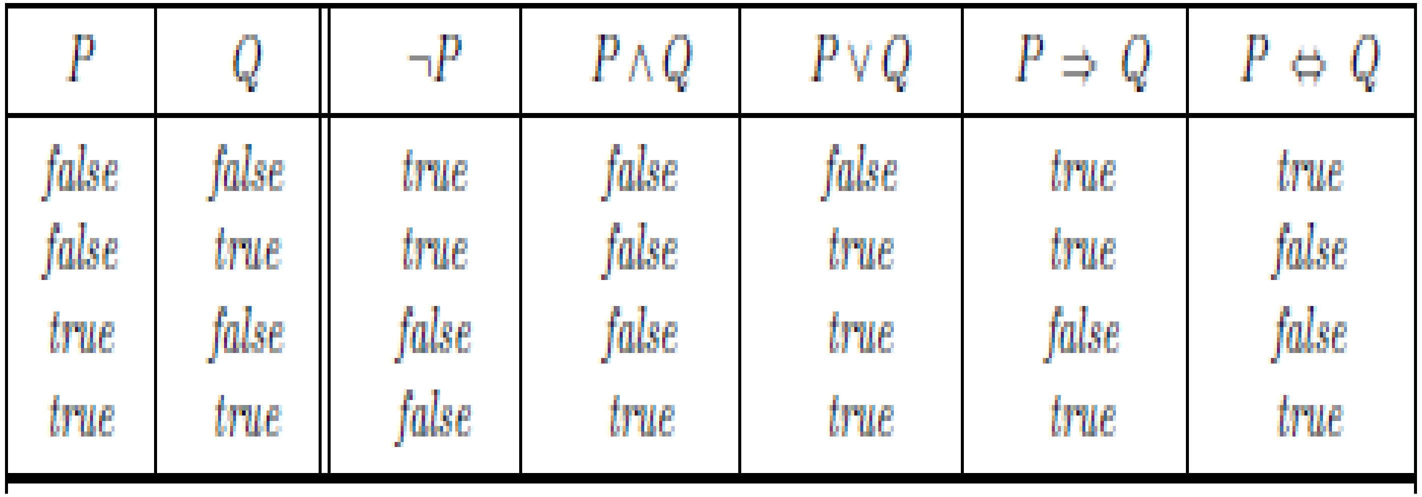
$$m_1 = \{P_{1,2} = false,\ P_{2,2} = false,\ P_{3,1} = true\}\ .$$

- $\neg P$ 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.

P => Q : if P is true, then Q is true. Otherwise no claim about Q. This sentence is false when P is true and Q is false
P < = > Q is true whenever both P => Q and Q => P are true. P if and only if Q

Ex: A square is breezy **if and only if** a neighbouring square has a pit
$B_{2,1}$ < = > ($P_{2,2}$ V $P_{3,1}$)

Truth Table 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*

Validity and Inference

Given the model m1, is the sentence $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$ valid in m1?

m1 = $\{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}$

Simple recurive process can evaluate the sentence

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

Wumpus World

KB

Symbols for each [x,y] location

Px,y, Wx,y, Bx,y, Sx,y

Prove sentence, α: there is no pit in $P_{1,2}$: $\neg P_{1,2}$ goal: KB $|= \alpha$

R1: $\neg P_{1,1}$: there is no pit in $P_{1,1}$

R2: $B_{1,1} <=> (P_{1,2} \vee P_{2,1})$: A square is breezy if and only if there is a pit in a neighbouring square

R3: $B_{2,1} <=> (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R1, R2, R3 true in all Wumpus worlds

Breeze percepts for the 2 squares visited in the specific world, the agent is in

R4: $\neg B_{1,1}$

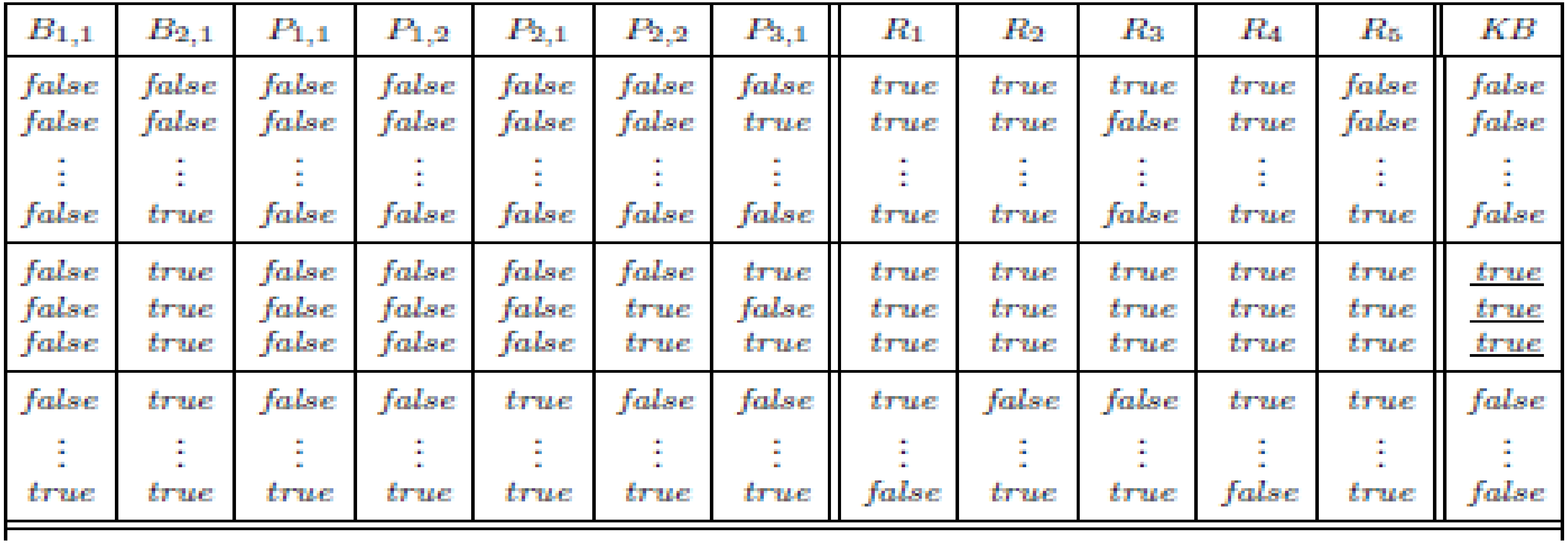
R5: $B_{2,1}$

Inference procedure

Model checking approach:

Propositional symbols: $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, and $P_{3,1}$

Models are assignments of true , false to every propositional symbols. Possible models = 2^7



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

KB is true if R1 through R5 are true, which occurs in 3 of the 128 rows. In all these three rows, $P_{1,2}$ is false.

there might or might not be a pit in $P_{2,2}$

Truth table enumeration algorithm for deciding propositional entailment

```
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 do
                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 }))
```

Complexity

If KB and α contain n symbols in all,

Time complexity: O (2^n)

Space complexity: O(n) , enumeration is depth-first

Propositional Theorem Proving

Entailment by Model checking has exponential time complexity

Logical equivalence

two sentences a and b are logically equivalent if they are true in the same set of models

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

$M(\alpha) \subseteq M(\beta)$ and $M(\beta) \subseteq M(\alpha)$ or $M(\alpha) = M(\beta)$

Validity

A sentence is valid if it is true in all models

$P \vee \neg P$ is true in all models. Proposition True is true in all models

every valid sentence is logically equivalent to True

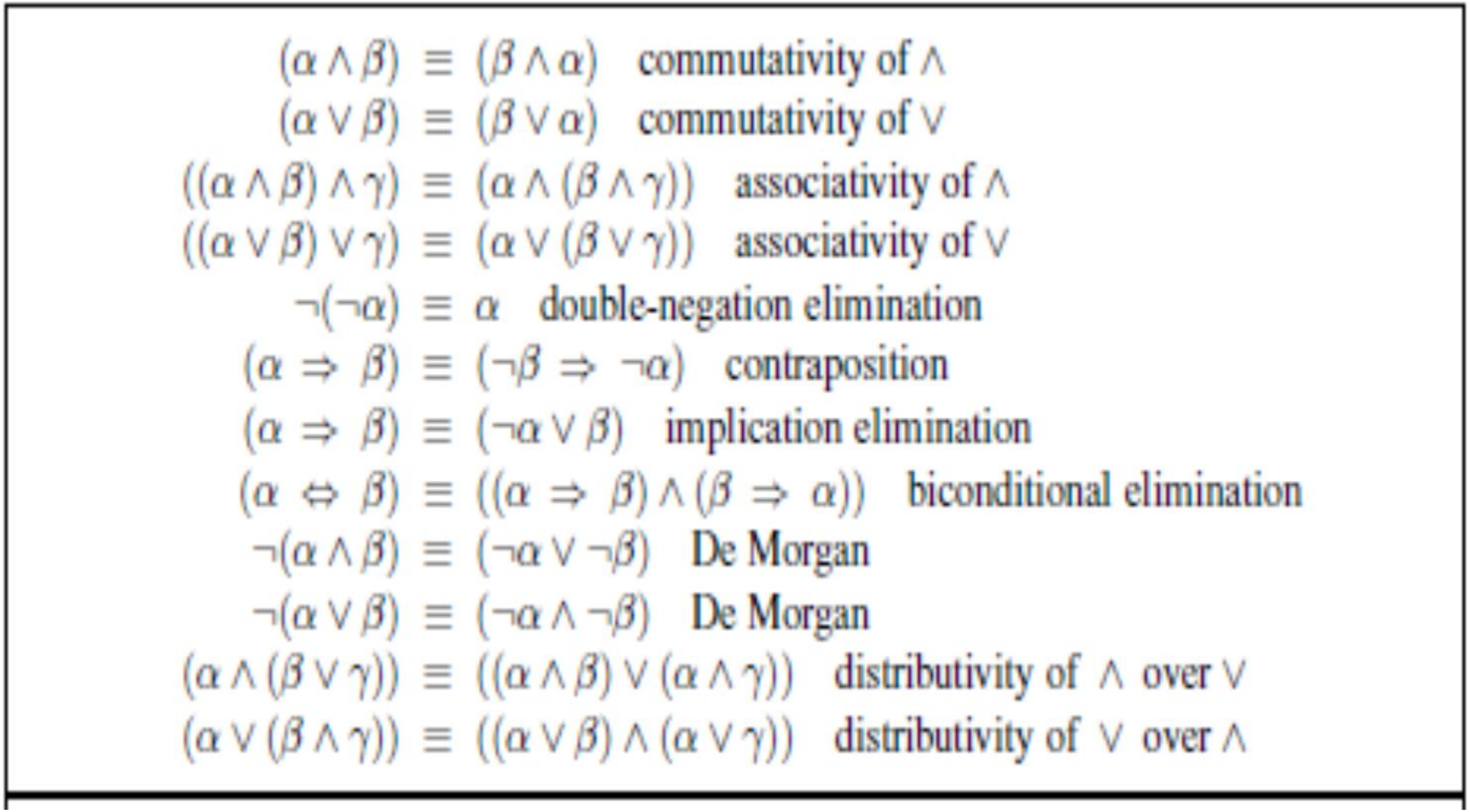
Tautologies

Valid sentences

Deduction theorem

for any sentences α and β, $\alpha \models \beta$ if and only if the sentence ($\alpha \Rightarrow \beta$) is valid

Propositional Theorem Proving



$(\alpha \land \beta) \equiv (\beta \land \alpha)$ commutativity of ∧

$(\alpha \lor \beta) \equiv (\beta \lor \alpha)$ commutativity of ∨

$((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))$ associativity of ∧

$((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))$ associativity of ∨

$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta)$ implication elimination

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$ biconditional elimination

$\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$ De Morgan

$\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$ De Morgan

$(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))$ distributivity of ∧ over ∨

$(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$ distributivity of ∨ over ∧

Deduction theorem

Deduction theorem

for any sentences α and β , α |= β if and only if the sentence (α => β) is valid

Inference algorithm

check for validity of **(α => β) or prove that (α => β)** is equivalent to True for deciding on **α |= β**

every valid implication sentence describes a legitimate inference

Satisfiability

A sentence is satisfiable if it is true in, or satisfied by, some model

(ex: wumpus world: $R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5$) is satisfiable as there are three models in which it is true

check for satisfiability by enumerating possible models until one is found that satisfies the sentence

SAT Problem

determining satisfiability of sentences in PL

NP complete

Validity vs. Satisfiability

α is valid if $\neg\, \alpha$ is unsatisfiable

α is satisfiable iff $\neg\, \alpha$ is not valid

$\alpha \models \beta$ if and only if the sentence ($\alpha \wedge \neg\beta$) is unsatisfiable

Inference algorithm

check for validity of **($\alpha \Rightarrow \beta$) or prove that ($\alpha \Rightarrow \beta$)** is equivalent to True for deciding on **$\alpha \models \beta$**

if the sentence ($\alpha \wedge \neg\beta$) is unsatisfiable

Proof by refutation or proof by contradiction or reduction to an absurd thing

proving **β** from a by checking the unsatisfiablity of **($\alpha \wedge \neg\beta$)**

Inference rules

Inference rules that can applied to derive proof

Modus ponens (mode that affirms)

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

Whenever any sentences of the form $\alpha \Rightarrow \beta$ **and** α are given, then the sentence β can be derived.

Ex:

$$\frac{\neg B_{1,1} \Rightarrow (\neg P_{1,2} \wedge \neg P_{2,1}) \;, \qquad \neg B_{1,1}}{(\neg P_{1,2} \wedge \neg P_{2,1})}$$

And- elimination

$$\frac{\alpha \wedge \beta \;, \qquad \alpha}{\beta}$$

From a conjunction any of the conjuncts can be derived

Modus ponens and And – elimination are sound and can be used to generate sound inferences without the need for enumerating models

Inference and proofs

Inference rule:

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \quad \text{and} \quad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Propositional Theorem Proving

R1: $\neg P_{1,1}$: there is no pit in $P_{1,1}$

R2: $B_{1,1} < = > (P_{1,2}\ V\ P_{2,1})$

R3: $B_{2,1} < = > (P_{1,1}\ V\ P_{2,2}\ V\ P_{3,1})$ — R1, R2, R3 true in all Wumpus worlds

Breeze percepts for the 2 squares visited in the specific world, the agent is in

R4: $\neg B_{1,1}$

R5: $B_{2,1}$

Prove $\boldsymbol{\neg P_{1,2}}$

R6: $(B_{1,1} = > (P_{1,2}\ V\ P_{2,1}))\wedge ((P_{1,2}\ V\ P_{2,1}) = > B_{1,1})$ — Bidirectional elimination of R2

R7: $(P_{1,2}\ V\ P_{2,1}) = > B_{1,1}$ — And elimination to R6

R8: $\neg B_{1,1} => \neg (P_{1,2}\ V\ P_{2,1})$ — logical equivalence for contrapositives

R9: $\neg (P_{1,2}\ V\ P_{2,1})$ — R8, R4 an modus ponens

R10: $\neg P_{1,2} \wedge \neg P_{2,1}$ — Demorgon’s rule

Neither [1,2] nor [2,1] contains a pit.

Proof Problem

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.

Searching for proofs is an alternative to enumerating models.

Proof can ignore irrelevant propositions, hence efficient

Monotonicity

Monotonicity

set of entailed can only increase as is added to KB

for any sentences a and b,

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

Monotonicity means that inference rules can be applied whenever

suitable premises are found in the knowledge base—

the conclusion of the rule must follow regardless of what else is in the knowledge base.

Proof by Resolution

Inference rules are sound. Completeness?

If the available inference rules are inadequate, then the goal is not reachable

Inference rule – **resolution** yields a complete inference algorithm when coupled with a complete search algorithm

Ex: Wumpus world: the agent returns from [2,1] to [1,1] and then goes to [1,2], where it perceives stench and no breeze

R11: $\neg B_{1,2}$: there is no breeze in [1,2]

R12: $B_{1,2} <=> (P_{1,1} \vee P_{2,2} \vee P_{1,3})$

R13: $\neg P_{2,2}$: $\neg P_{1,1}$ is already known. Derive R13, R14 using theorem proving from R12

R14: $\neg P_{1,3}$

R:15: $P_{1,1} \vee P_{2,2} \vee P_{3,1}$: apply bi-conditional elimination to R3, followed by modus ponens with R5

R13: $\neg P_{2,2}$ resolves with $P_{2,2}$ in R15 giving resolvant

R16: $P_{1,1} \vee P_{3,1}$: R1 : **$\neg P_{1,1}$ resolves with** $P_{1,1}$ in R16

R17: $P_{3,1}$

Unit resolution inference rule

each *l* is a literal, *l*i and *m* are complementary

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k},$$

Unit resolution inference rule takes a clause - a disjunction of literals

and a literal as input

and produces a new clause

Full resolution rule: *li* and *mj* are complementary

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

Ex:

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

Factoring: removal of multiple copies of literals: by resolving (A V B) with (A V ¬ B) , we get (A V A), reduced to A

Resolution rule forms basis for Sound , complete inference procedures

A resolution-based theorem prover can, for any sentences α and β in propositional logic, decide whether α |= β.

Conjunctive Normal From

CNF

Resolution rule applies only to clauses (disjunction of literals)

PL – every sentence ≡ conjunction of classes

CNF

Conjunctive Normal Form - a PL sentence expressed as a conjunction of clauses

Procedure for converting to CNF the sentence B1,1 ⇔ (P1,2 ∨ P2,1)

1. Eliminate ⇔, replacing α ⇔ β with (α => β) ∧ (β => α)

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

2. Eliminate =>, replacing α => β with **¬α V β**

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

3. CNF requires ¬ to appear only in literals, move ¬ inwards using De morgan 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. distribute ∧ over V to remove nested ∧ and V operators applied to literals

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

Resolution Algorithm

to show that KB $|= \alpha$, we show that (KB $\wedge \neg\alpha$) is unsatisfiable. We do this by proving a contradiction.

First, (KB $\wedge \neg\alpha$) is converted into CNF

Then, the resolution rule is applied to the resulting clauses.

Each pair that contains complementary literals is resolved to produce a new clause,

which is added to the set if it is not already present.

The process continues until one of two things happens:

there are no new clauses that can be added, in which case KB does not entail α; or,

two clauses resolve to yield the *empty* clause, in which case KB entails α.

empty clause represents a contradiction

PL resolution always terminates as Resolution Closure of a set of clauses, RC(S) is finite as there are only finitely many distinct clauses that can be constructed out of the symbols that appear in S.

Ground Resolution Theorem: The completeness theorem for resolution in PL

Wumpus World

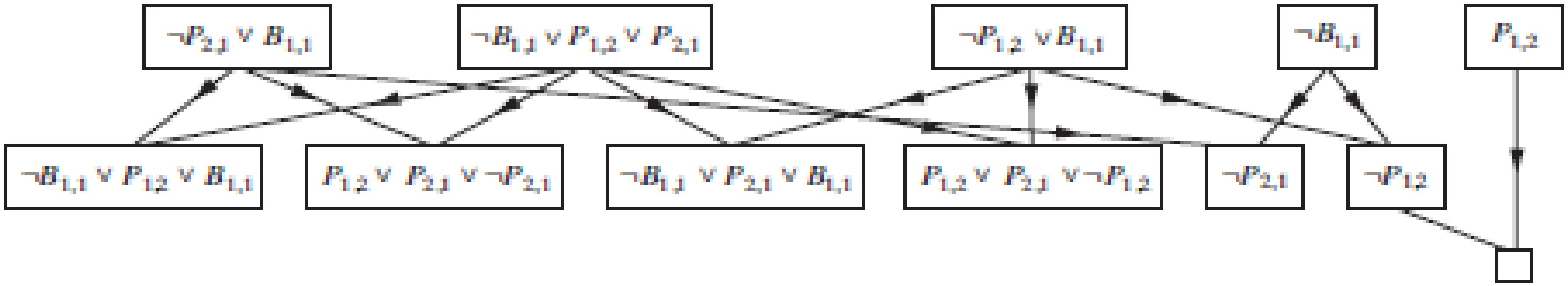
When the agent is in [1,1], there is no breeze, so there can be no pits in neighboring squares.

The relevant knowledge base is

$KB = R_2 \land R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$

Prove α : $\neg P_{1,2}$

Convert (KB ∧ ¬α) into CNF


$\neg P_{2,1} \lor B_{1,1}$
$\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}$
$\neg P_{1,2} \lor B_{1,1}$
$\neg B_{1,1}$
$P_{1,2}$
$\neg B_{1,1} \lor P_{1,2} \lor B_{1,1}$
$P_{1,2} \lor P_{2,1} \lor \neg P_{2,1}$
$\neg B_{1,1} \lor P_{2,1} \lor B_{1,1}$
$P_{1,2} \lor P_{2,1} \lor \neg P_{1,2}$
$\neg P_{2,1}$
$\neg P_{1,2}$
□

Partial application of PL-RESOLUTION to a simple inference in the wumpus world.
¬P1,2 is shown to follow from the first four clauses in the top row.

Resolution Algorithm for PL

```
function PL-RESOLUTION(KB,$\alpha$) returns true or false
        inputs: KB, the knowledge base, a sentence in propositional logic
              $\alpha$, 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 pair of clauses Ci, Cj in clauses do
                        resolvants $\leftarrow$PL-RESOLVE(Ci,Cj )
                        if resolvants contains the empty clause then return true
                        new $\leftarrow$new $\cup$ resolvants
                if new $\subseteq$ clauses then return false
                clauses $\leftarrow$clauses $\cup$new
```

Horn Clauses

Definite clause: a disjunction of literals of which exactly one is positive

$$(\neg P_{1,2} \lor \neg P_{2,1} \lor B_{11})$$

Horn Clause: disjunction of literals of which at most one is positive

Goal clauses: disjunction of literals with no positive literals

If you resolve two horn clauses, you get back a horn clause: Horn clauses are closed under resolution

$$(P_{1,2} \land P_{2,1}) => B_{1,1}$$

CNF grammer

$\text{CNFSentence} \rightarrow \text{Clause}_1 \wedge \cdot \cdot \cdot \wedge \text{Clause}_n$

$\text{Clause} \rightarrow \text{Literal}_1 \vee \cdot \cdot \cdot \vee \text{Literal}_m$

$\text{Literal} \rightarrow \text{Symbol} \mid \neg\text{Symbol}$

$\text{Symbol} \rightarrow P \mid Q \mid R \mid \ldots$

$\text{HornClauseForm} \rightarrow \text{DefiniteClauseForm} \mid \text{GoalClauseForm}$

$\text{DefiniteClauseForm} \rightarrow (\text{Symbol}_1 \wedge \cdot \cdot \cdot \wedge \text{Symbol}_l) \Rightarrow \text{Symbol}$

$\text{GoalClauseForm} \rightarrow (\text{Symbol}_1 \wedge \cdot \cdot \cdot \wedge \text{Symbol}_l) \Rightarrow \text{False}$

Forward chaining and backward chaining

KB with definite clauses – interesting as

1. Every definitive clause can be written as an implication

whose premise is a conjunction of positive literals and

whose conclusion is a single positive literal

(Premise is called body; conclusion is called head in Horn clauses)

ex: $(P_{1,2} \wedge P_{2,1}) = > B_{1,1}$

2. Inference with Horn clauses can be done with forward chaining and backward chaining algorithm
3. deciding entailment with Horn clauses can be done in time that is linear in the size of the KB

Forward chaining and backward chaining

Forward chaining algorithm determines if a single proposition symbol q – the query - is entailed by a KB of definitive clauses

Begins from known facts (positive literals) in KB.

If all premises are known then the conclusion is added to the set of known facts.

The process continues till q is added or until no further inferences can be made

Data driven reasoning

Backward chaining works backwards from the query q. If q is known to be True, no work is needed

Else the algorithm finds those implications in the KB whose conclusion is q.

If all the premises of one of those implications can be proved true (by backward chaining), then q is true.

Goal –directed reasoning

example

If a Unicorn is Mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Can you prove that the unicorn is mythical? Magical?, horned?

example

If a Unicorn is Mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Can you prove that the unicorn is mythical?   Magical?, horned?

Propositional logic statements of the given facts:

(1) $\text{Mythical} \rightarrow \neg\text{Mortal}$

(2) $\neg\text{Mythical} \rightarrow \text{Mortal} \wedge \text{Mammal}$

(3) $\neg\text{Mortal} \vee \text{Mammal} \rightarrow \text{Horned}$

(4) $\text{Horned} \rightarrow \text{Magical}$

example

If a Unicorn is Mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Can you prove that the unicorn is mythical? Magical?, horned?

Propositional logic statements of the given facts:

(1) Mythical → ¬Mortal

(2) ¬Mythical → Mortal ∧ Mammal

(3) ¬Mortal ∨ Mammal → Horned

(4) Horned → Magical

inference is done by direct application -

(5) ¬Mythical ∨ ¬Mortal	From (1)
(6a) (Mythical ∨ Mortal)	From (2)
(6b) (Mythical ∨ Mammal)	From (2)
(7) ¬Mortal ∨ Mammal	Resolution on (5) and (6b)
(8) Horned	Modus Ponens on (3) and (7)
(9) Magical	Modus Ponens on (4) and (8)

example

According to some political pundits, a person who is radical (R) is electable (E) if he/she is conservative (C), but otherwise is not electable.

a. Which of the following are correct representations of this assertion?

(i) $(R \wedge E) \Leftarrow> C$

(ii) $R \Rightarrow (E \Leftarrow> C)$

(iii) $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$

b. Which of the sentences in (a) can be expressed in Horn form?

example

Consider the following sentence:

$[(\text{Food} \Rightarrow \text{Party}) \vee (\text{Drinks} \Rightarrow \text{Party})] \Rightarrow [(\text{Food} \wedge \text{Drinks}) \Rightarrow \text{Party}]$.

a. Determine, using enumeration, whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.

b. Convert the left-hand and right-hand sides of the main implication into CNF, showing each step, and explain how the results confirm your answer to (a).

c. Prove your answer to (a) using resolution.

example

a. A simple truth table has eight rows, and shows that the sentence is true for all models and hence valid.

b. For the left-hand side we have:

(Food ⇒ Party) ∨ (Drinks ⇒ Party)

(¬Food ∨ Party) ∨ (¬Drinks ∨ Party)

(¬Food ∨ Party ∨ ¬Drinks ∨ Party)

(¬Food ∨ ¬Drinks ∨ Party)

and for the right-hand side we have

(Food ∧ Drinks) ⇒ Party

¬(Food ∧ Drinks) ∨ Party

(¬Food ∨ ¬Drinks) ∨ Party

(¬Food ∨ ¬Drinks ∨ Party)

The two sides are identical in CNF, and hence the original sentence is of the form P ⇒ P, which is valid for any P.

example

c. To prove that a sentence is valid, prove that its negation is unsatisfiable. I.e., negate it, convert to CNF, use resolution to prove a contradiction. We can use the above CNF result for the LHS.

$\neg[[(\text{Food} \Rightarrow \text{Party}) \vee (\text{Drinks} \Rightarrow \text{Party})] \Rightarrow [(\text{Food} \wedge \text{Drinks}) \Rightarrow \text{Party}]]$

$[(\text{Food} \Rightarrow \text{Party}) \vee (\text{Drinks} \Rightarrow \text{Party})] \wedge \neg[(\text{Food} \wedge \text{Drinks}) \Rightarrow \text{Party}]$

$(\neg\text{Food} \vee \neg\text{Drinks} \vee \text{Party}) \wedge \text{Food} \wedge \text{Drinks} \wedge \neg\text{Party}$

Each of the three unit clauses resolves in turn against the first clause, leaving an empty clause.

Example Forward Chaining

Knowledge-base describing when the car should brake?

(PersonInFrontOfCar ⇒ Brake)

∧ (((YellowLight ∧ Policeman) ∧ (¬Slippery)) ⇒ Brake)

∧ (Policecar ⇒ Policeman)

∧ (Snow ⇒ Slippery)

∧ (Slippery ⇒ ¬Dry)

∧ (RedLight ⇒ Brake)

∧ (Winter ⇒ Snow)

Observation from sensors:

YellowLight ∧ ¬RedLight ∧ ¬Snow ∧ Dry ∧ Policecar ∧ ¬PersonInFrontOfCar

What can we infer?

- And-elimination: Policecar
- Modus Ponens: Policeman
- And-elimination: Dry
- Modus Tollens: ¬Slippery
- And-elimination: YellowLight ∧ Policeman ∧ ¬Slippery
- Modus Ponens: Brake
- And-elimination: ¬Snow
- Modus Tollens: ¬Winter

Modus Tollens:

$$\frac{(p => q)\ ,\ \neg q}{\neg p}$$

Example Forward Chaining

Inference Strategy: Forward Chaining

Idea:

– Infer everything (?) that can be inferred.

– Notation: In implication $\alpha \Rightarrow \beta$, α (or its components) are called premises, β is called consequent/conclusion.

Forward Chaining:

Given a fact p to be added to the KB,

1. Find all implications I that have p as a premise

2. For each i in I, if the other premises in i are already known to hold

a) Add the consequent in i to the KB

Continue until no more facts can be inferred.

Example Backward Chaining

Inference Strategy: 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.

Ł Avoids inferring unrelated facts.

Back tracking

Goal: Brake

– Modus Ponens (brake): PersonInFrontOfCar

- Failure: PersonInFrontOfCar Backtracking
- Goal: Brake

– Modus Ponens (brake): YellowLight ^ Policeman ^ ¬Slippery

– Known (YellowLight): Policeman ^ ¬ Slippery

– Modus Ponens (Policeman): Policecar ^ ¬ Slippery

– Known (Policecar): ¬ Slippery

– Modus Tollens (¬ Slippery): Dry

– Known (Dry)

Example

1.

A propositional 2-CNF expression is a conjunction of clauses, each containing exactly 2 literals,

$(A \lor B) \land (\neg A \lor C) \land (\neg B \lor D) \land (\neg C \lor G) \land (\neg D \lor G)$

Prove using resolution that the above sentence entails G.

2.

Convert the following set of sentences to clausal form.

S1: $A \Leftrightarrow (B \lor E)$.

S2: $E \Rightarrow D$.

S3: $C \land F \Rightarrow \neg B$.

S4: $E \Rightarrow B$.

S5: $B \Rightarrow F$.

S6: $B \Rightarrow C$

Example

Set of Horn Clauses:

Forward Chaining Algorithm identical to AND-OR Graph Search

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B