Knowledge-Based Systems

Announcements

- Review sessions
- CS 4701 – focus on AI

Schedule

- Search
- Machine learning
- Knowledge based systems
- Discovery

History of AI
1943 – 1969 The Beginnings

1943 McCulloch and Pitts show networks of neurons can compute and learn any function

1950 Shannon and Turing wrote chess programs

1951 Minsky and Edmonds build the first neural network computer (SNARC)

1956 Dartmouth Conference – Newell and Simon brought a reasoning program "The Logic Theorist" which proved theorems.

1952 Samuel's checkers player

1958 McCarthy designed LISP, helped invent time-sharing and created Advice Taker (a domain independent reasoning system)

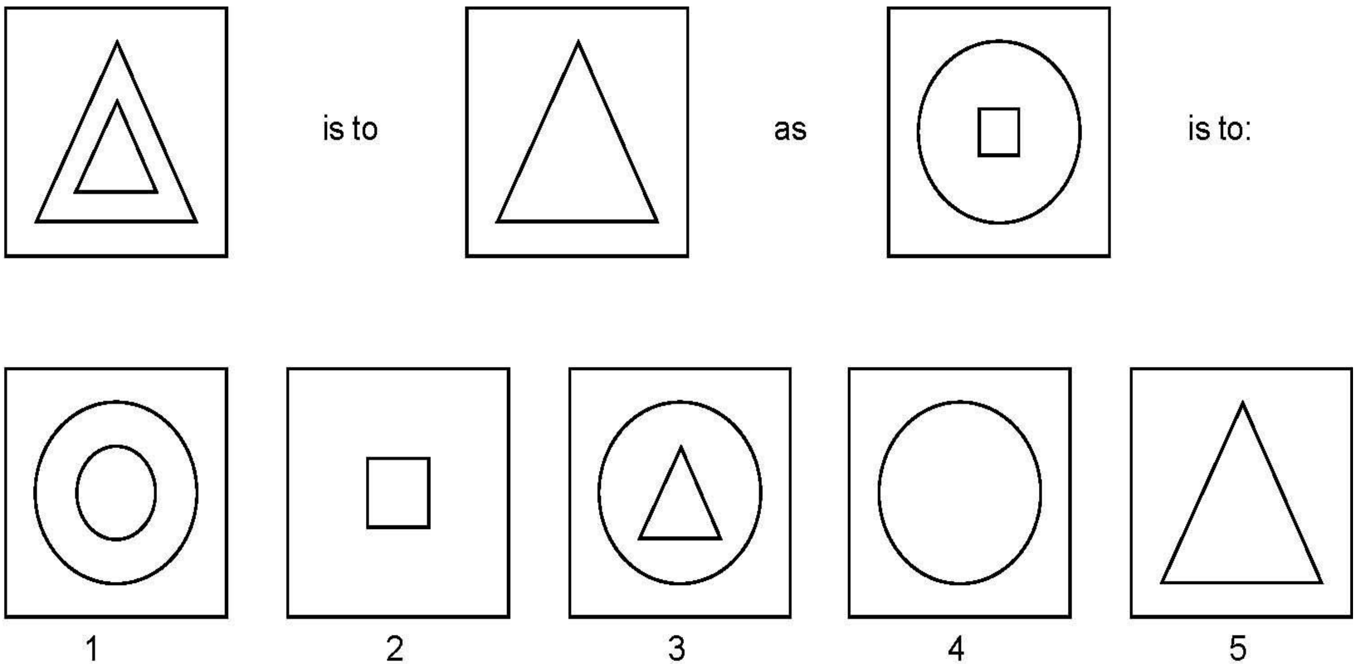
1960's Microworlds – solving limited problems: SAINT (1963), ANALOGY (1968), STUDENT (1967), blocksworld invented.

1962 Perceptron Convergence Theorem is proved.

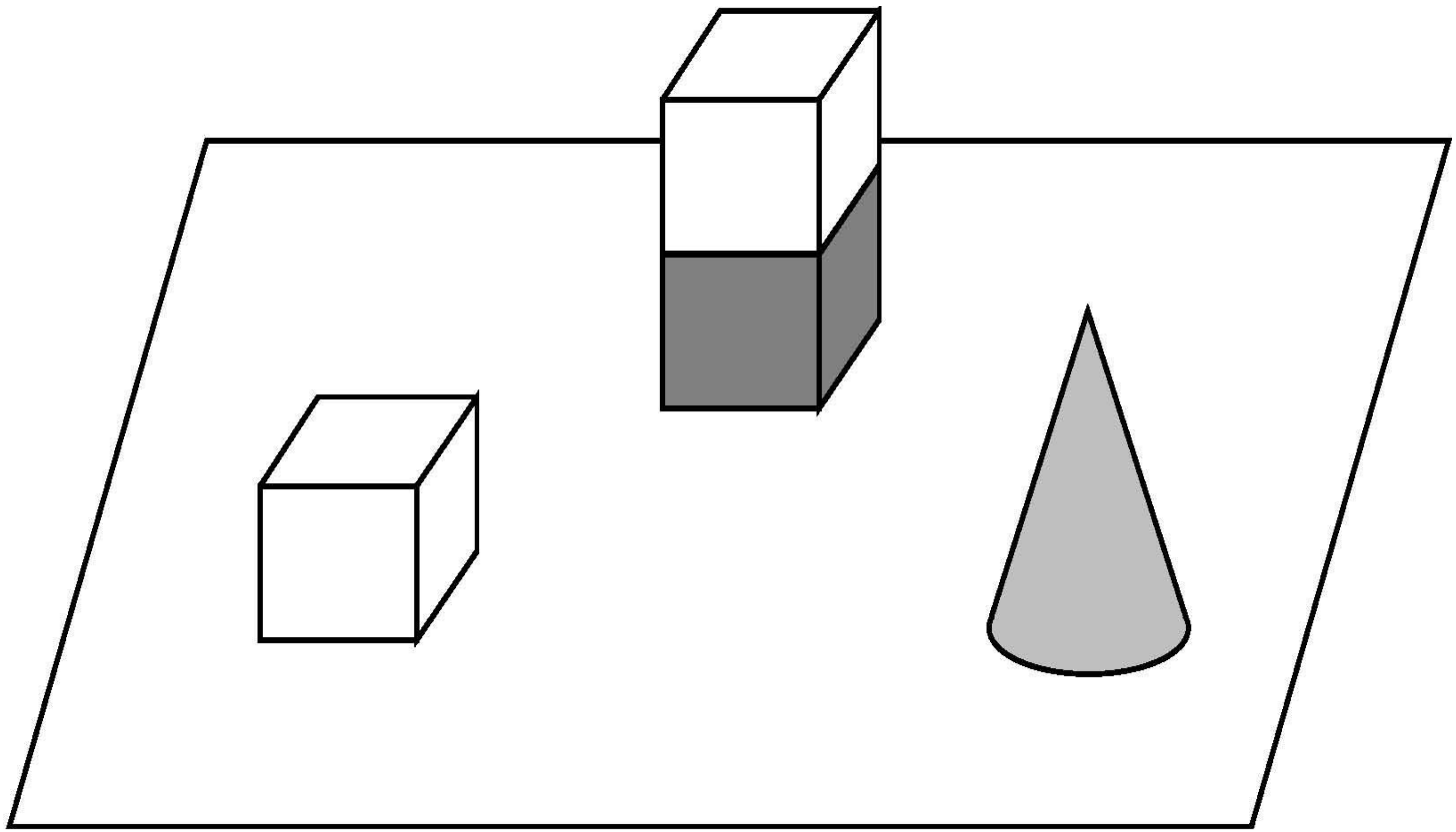
1952 Samuel’s checkers player o TV

Arthur Samuel (1901-1990)

Example ANALOGY Problem



Blocksworld



History of AI
1966 – 1974 Recognizing Lack of Knowledge

- Herb Simon (1957): Computer chess program will be world chess champion within 10 years.
- Intractable problems, lack of computing power (Lighthill Report, 1973)
- Machine translation
- Limitations in knowledge representation (Minsky and Papert, 1969)

➔ Knowledge-poor programs

Knowledge Representation

- Human intelligence relies on a lot of background knowledge
  - the more you know, the easier many tasks become
  - ”knowledge is power”
  - E.g. SEND + MORE = MONEY puzzle.
- Natural language understanding
  - Time flies like an arrow.
  - Fruit flies like a banana.
  - John saw the diamond through the window and coveted **it**
  - John threw the brick through the window and broke **it**
  - The spirit is willing but the flesh is weak. (English)
  - The vodka is good but the meat is rotten. (Russian)
- Or: Plan a trip to L.A.

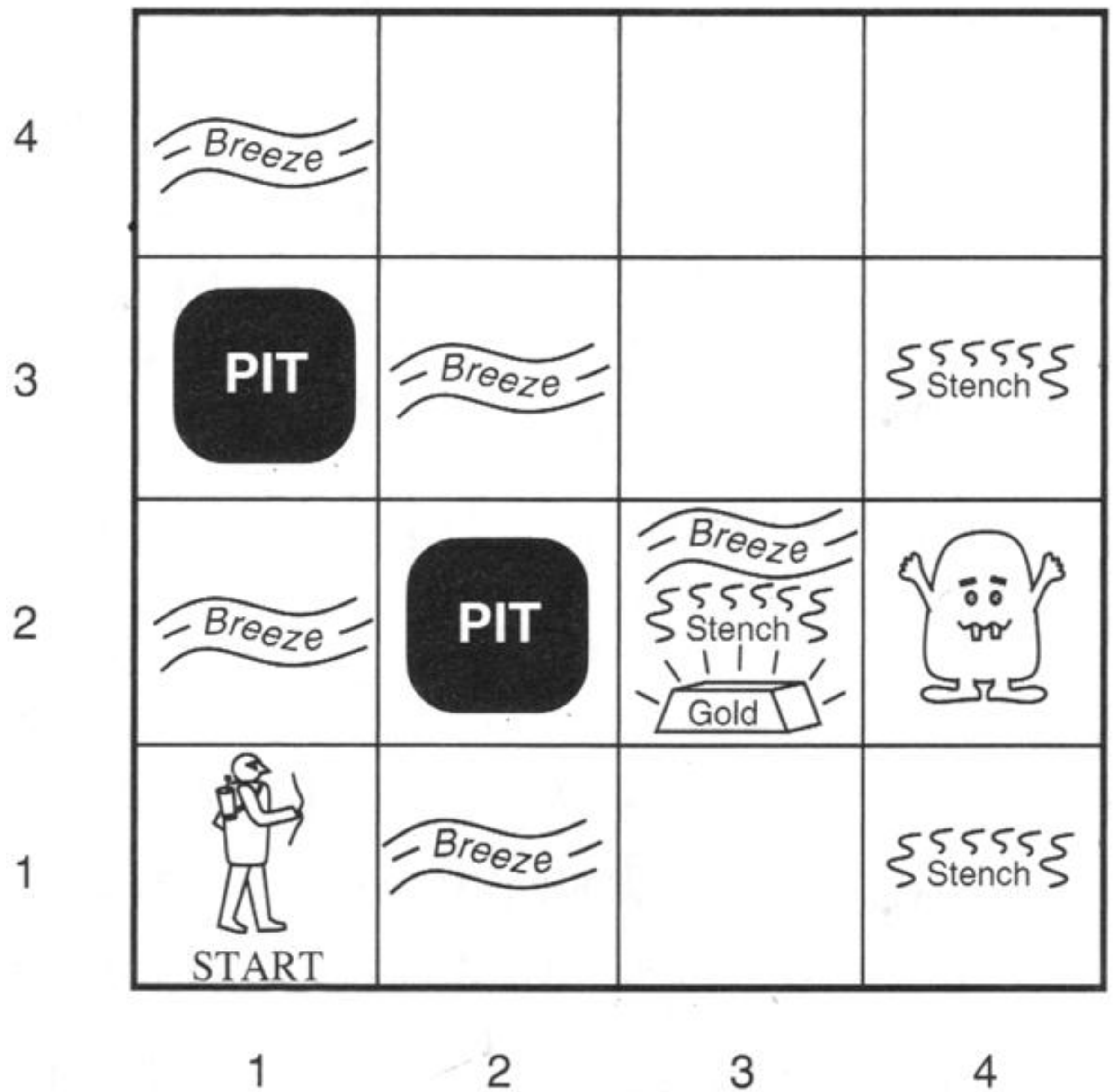
Domain knowledge

- How did we encode domain knowledge so far?
  - For search problems?
  - For learning problems?

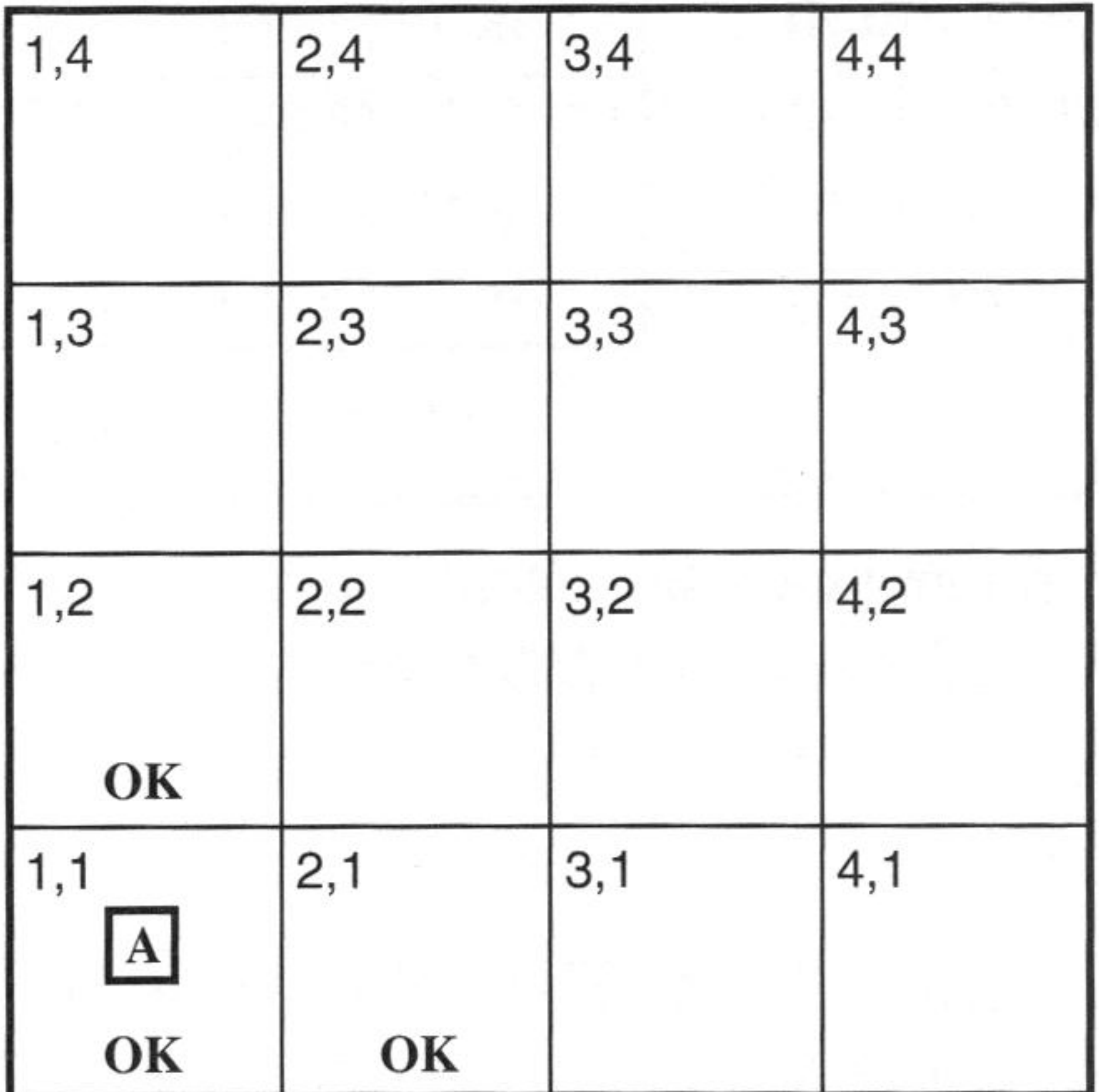
Knowledge-Based Systems/Agents

- Key components:
  - Knowledge base: a set of sentences expressed in some knowledge representation language
  - Inference/reasoning mechanisms to query what is known and to derive new information or make decisions.
- Natural candidate:
  - logical language (propositional/first-order)
  - combined with a logical inference mechanism
- How close to human thought?
  - In any case, appears reasonable strategy for machines.

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

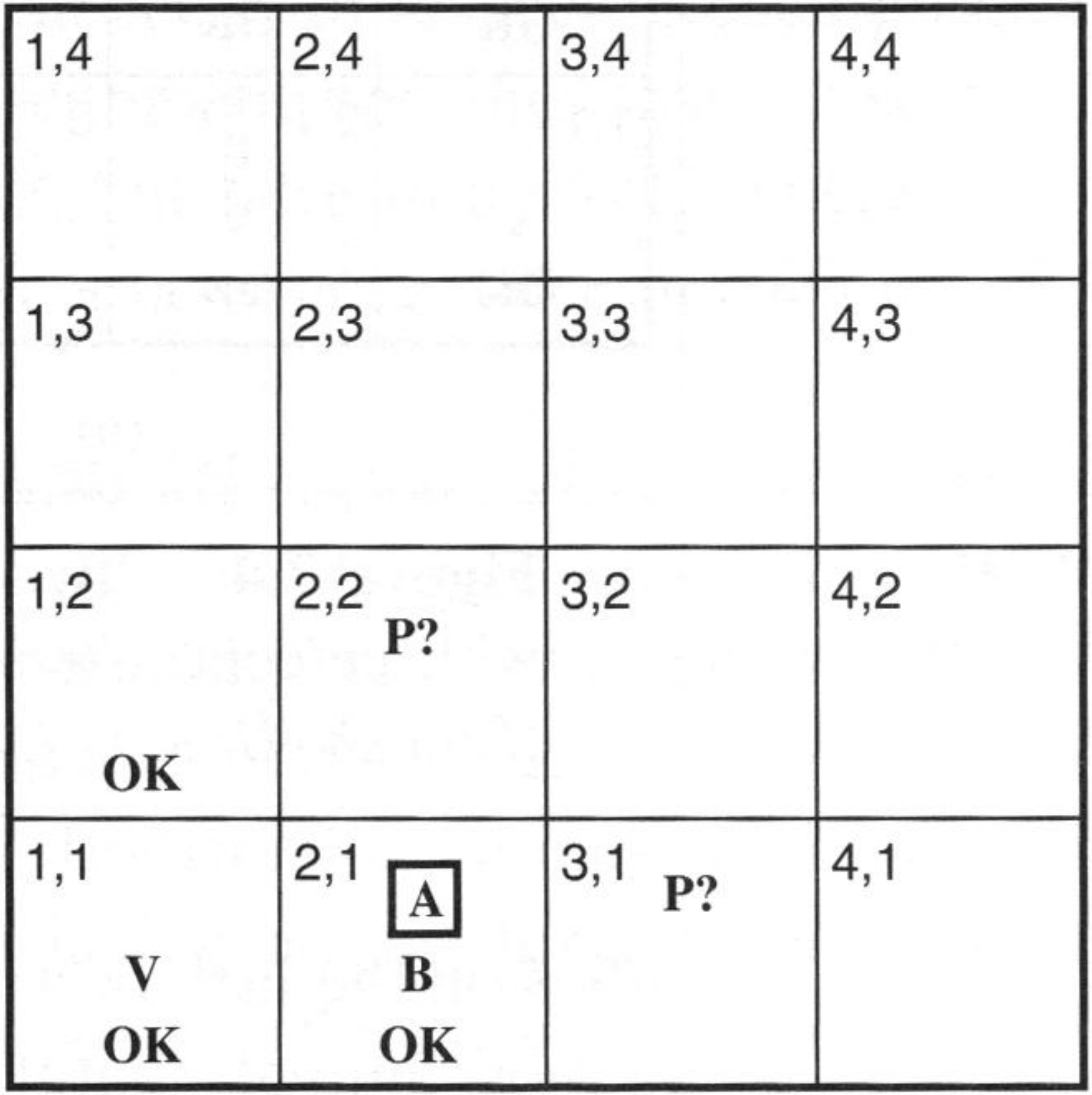


$\boxed{A}$ = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

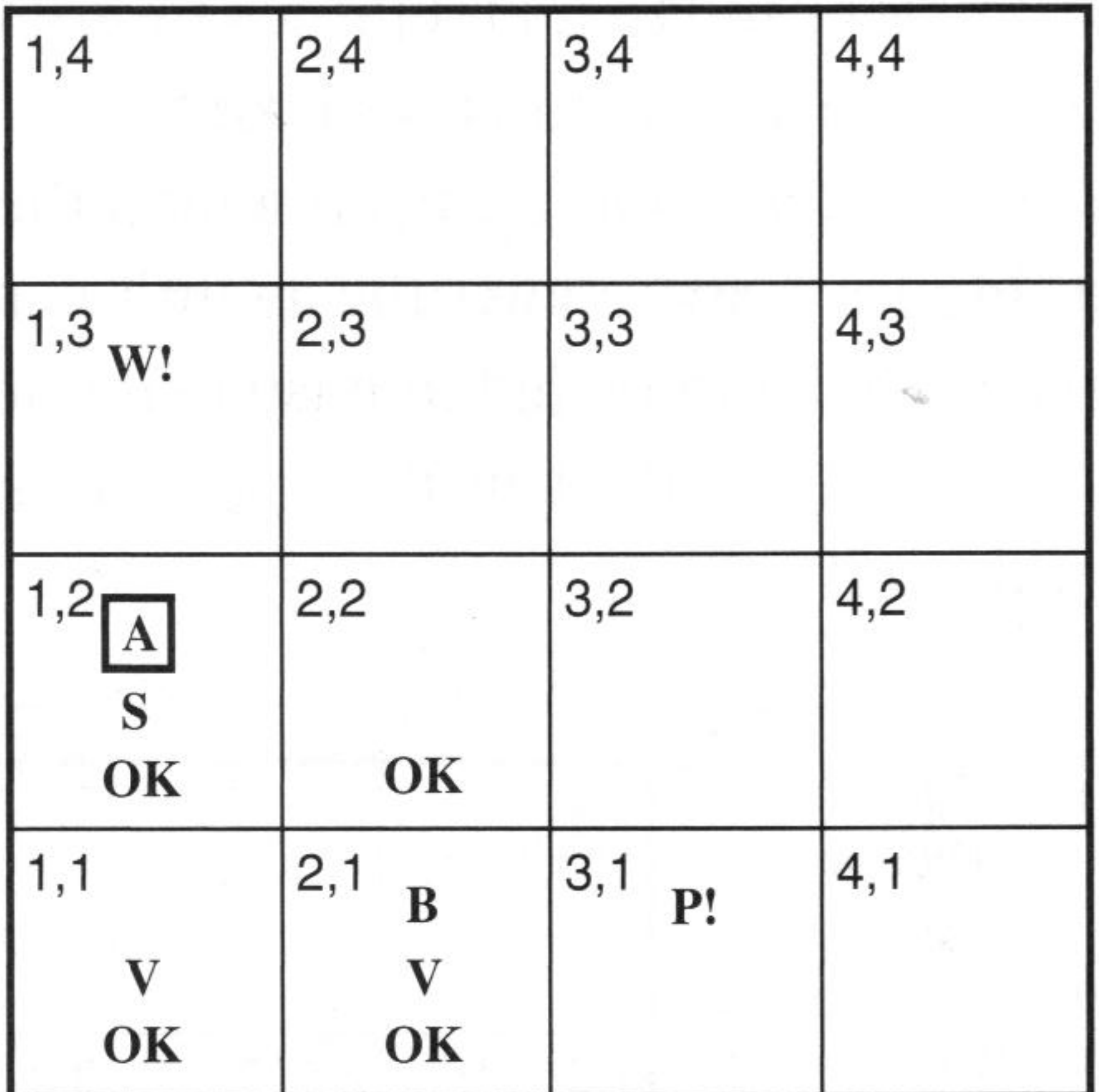


1

$\boxed{A}$ = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*



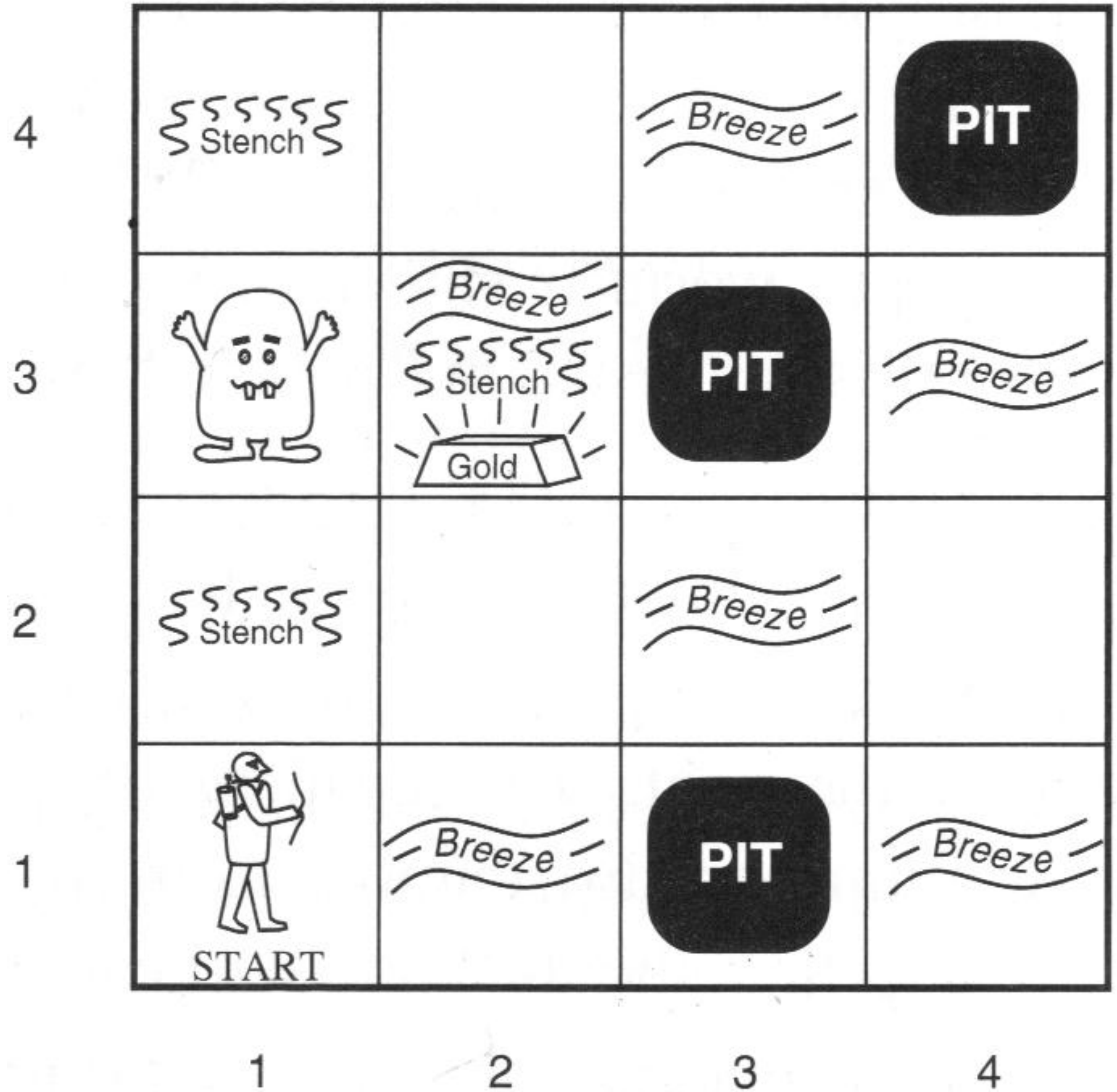
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*



- $\boxed{A}$ = *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 $\boxed{A}$ **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

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



Example: Autonomous Car

State: k-tuple

(PersonInFrontOfCar, Policeman, Policecar, Slippery, YellowLight, RedLight)

Actions:

Brake, Accelerate, TurnLeft, etc.

Knowledge-base describing when the car should brake:

$(PersonInFrontOfCar \Rightarrow Brake)$

$(((YellowLight \wedge Policeman) \wedge (\neg Slippery)) \Rightarrow Brake)$

$(Policecar \Rightarrow Policeman)$

$(Snow \Rightarrow Slippery)$

$(Slippery \Rightarrow \neg Dry)$

$(RedLight \Rightarrow Brake)$

Does (Policecar, YellowLight, Snow) imply Brake?

A=Yes B= No

What the computer "sees":

State: k-tuple

(x1, x2, x3, x4, x5, x6, x7)

Actions:

x8, x9, x10, etc.

Knowledge-base describing when x:

$(x1 \Rightarrow x8)$

$(((x5 \wedge x2) \wedge (\neg x4)) \Rightarrow x8)$

$(x3 \Rightarrow x2)$

$(x7 \Rightarrow x4)$

$(x4 \Rightarrow \neg x11)$

$(x6 \Rightarrow x8)$

Does (x3, x5, x7) imply x8?

A=Yes B= No

Logic as a Knowledge Representation

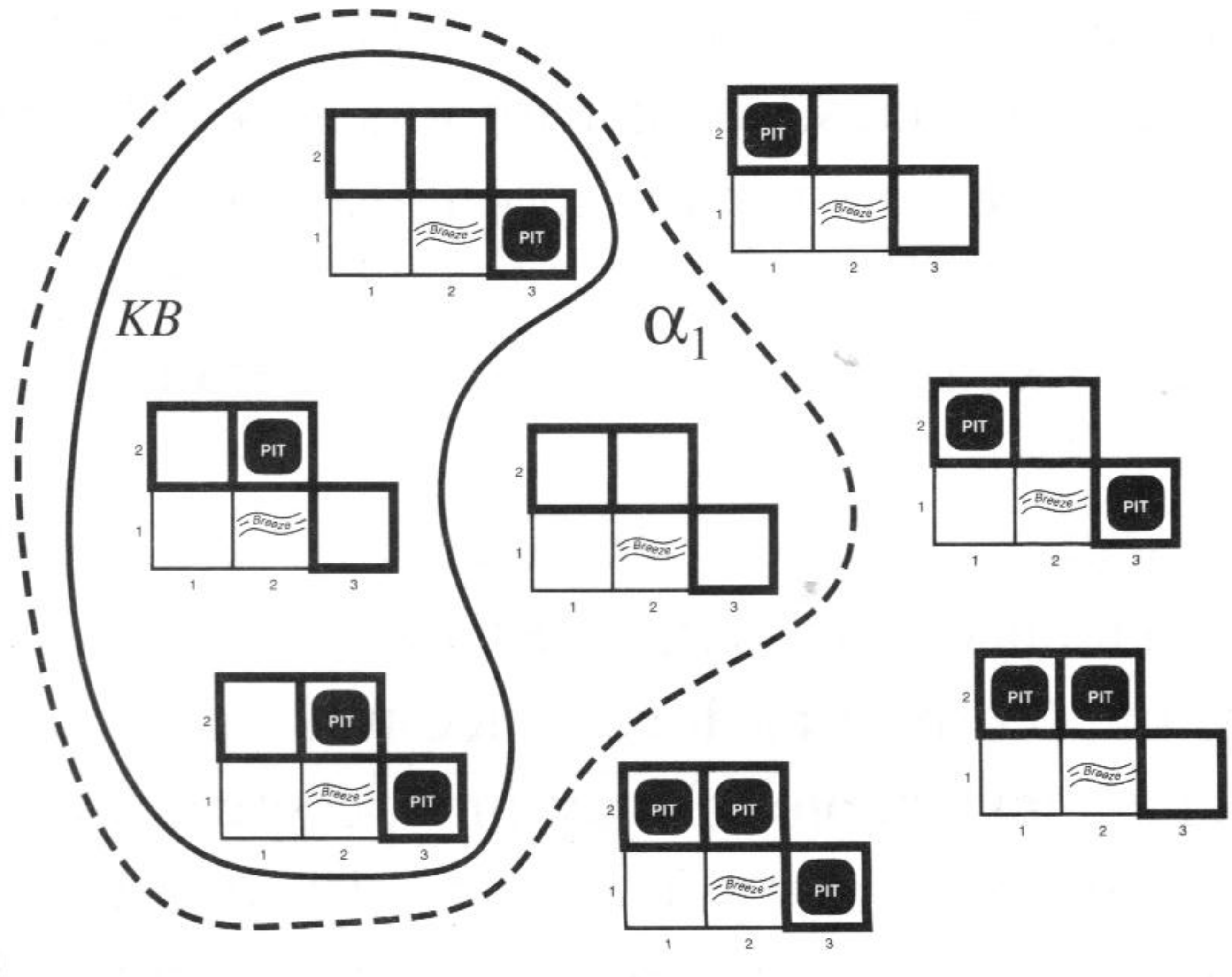
- Components of a Formal Logic:
  - Variables and operators, syntax
  - semantics (link to the world, truth in worlds)
  - logical reasoning: entailment $\alpha \models \beta$
    - if, in every **model** in which α is true, β is also true.
  - inference algorithm **derives**
    - KB $\Rightarrow$ α, i.e., α is derived from KB.

(x+y=4) entails that

A) x=2 y=2 B) 2x+2y=8 C) Neither D) Both

Models

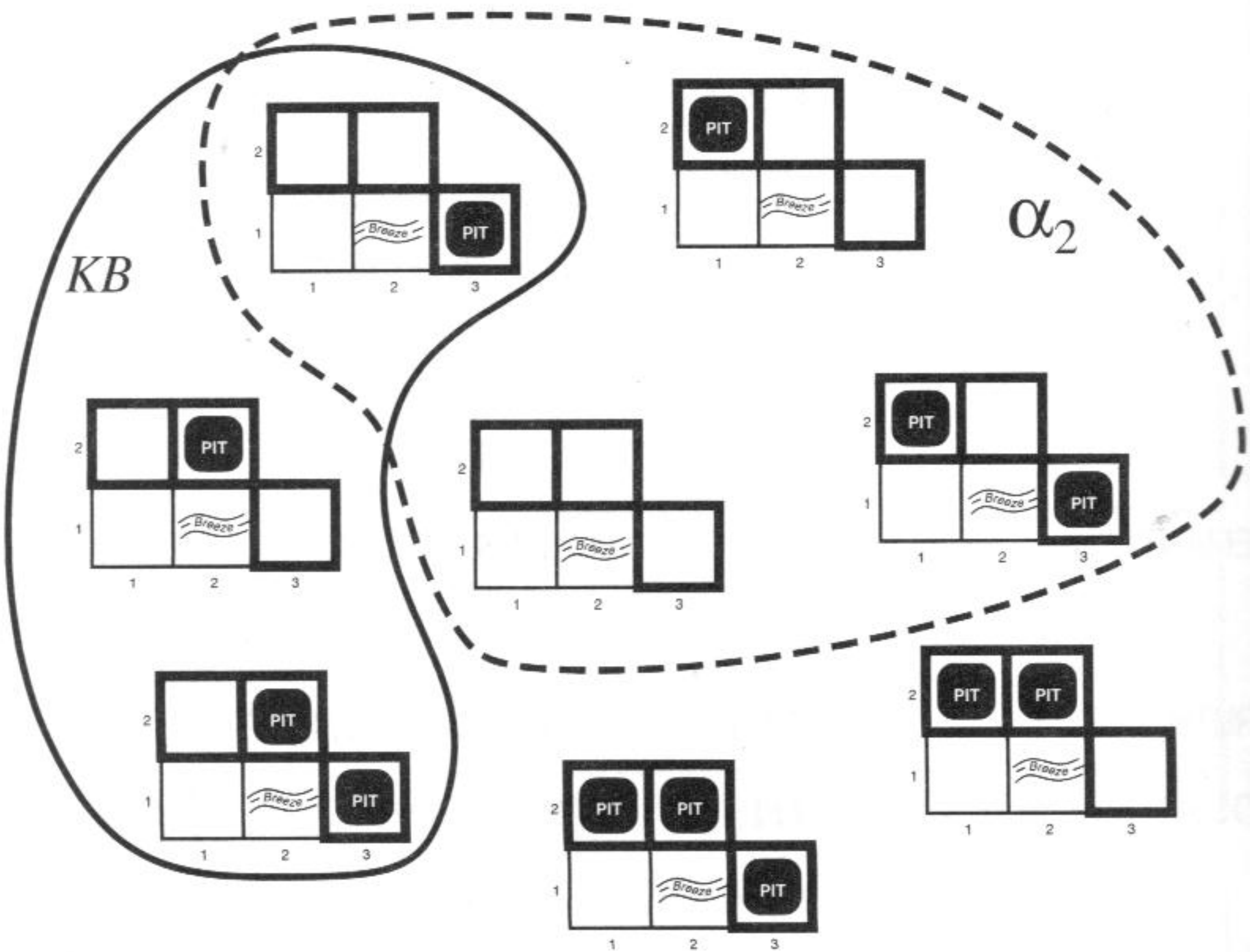
- Model is an instantiation of all variables
- All models = all possible assignments
- Sentence α is true in model m, then m is a model of α
- M(α) refers to the set of all models that satisfy α
- $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$
    - β iff M(α) is contained in M(β)



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

Dashed = M(α_1) where $\alpha_1 = \neg P_{1,2}$ (no pit in [1,2])

Solid = M(KB) with observation of $\neg B_{1,1} \wedge B_{2,1}$ (no breeze in [1,1] and breeze in [2,1])



Possible models for the presence of pits in [1,2] [2,2] [3,1]
Dashed = $M(\alpha_2)$ where $\alpha_2 = \neg P_{2,2}$ (no pit in [2,2])
Solid = M(KB) with observation of $\neg B_{1,1} \wedge B_{2,1}$ (no breeze in [1,1] and breeze in [2,1])

Soundness and Completeness

Soundness:

An inference algorithm that derives only entailed sentences is called *sound* or *truth-preserving*.

$KB \Rightarrow \alpha$ implies $KB \models \alpha$

Completeness:

An inference algorithm is *complete* if it can derive any sentence that is entailed.

$KB \models \alpha$ implies $KB \Rightarrow \alpha$

Why soundness and completeness important?

→ Allow computer to ignore semantics and “just push symbols”!

IS STRICT IMPLICATION THE SAME AS ENTAILMENT?

By Austin E. Duncan-Jones

1 Entailment and the modal functions

In this note I shall try to show how strict implication, as developed by C. I. Lewis (in *symbolic logic*, Lewis and Langford), differs from entailment, and to suggest certain requirements which a calculus of entailment would have to satisfy.[1]

Consider the two expressions (1) $xRy.yRz$ and (2) xRz. Suppose we let x y and z be classes and R the relation of inclusion; or suppose we let x y and z be propositions, and R the relation of material implication, or strict implication, or entailment: then most people would admit that (2) is materially implied by (1), that (2) follows from (1), and perhaps that (2) is

[1] The general point of view which I am adopting is much the same as that of Everett J. Nelson in *intensional relations*, *Mind* 1930. I had already worked out this paper in outline before I discovered the existence of Nelson's paper: otherwise the note would have been based directly on Nelson. I differ from Nelson on one or two points to be mentioned below.

AE Duncan-Jones - 1935

Entailment vs. Implication

- *Entailment* ($KB \models \alpha$) and implication ($KB \Rightarrow \alpha$) can be treated equivalently if the inference process is sound and complete.

Propositional Logic: Syntax

- Propositional Symbols
  - A, B, C, …
- Connectives
  - $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
- Sentences
  - Atomic Sentence: True, False, Propositional Symbol
  - Complex Sentence:
    - ($\neg$Sentence)
    - (Sentence V Sentence)
    - (Sentence $\wedge$ Sentence)
    - (Sentence $\Rightarrow$ Sentence)
    - (Sentence $\Leftrightarrow$ Sentence)
- A KB is a conjunction (ANDs) of many sentences

Example: Autonomous Car

Propositional Symbols

PersonInFrontOfCar, Policeman, .. Brake, Accelerate, TurnLeft

Rules:

$(\ \text{PersonInFrontOfCar} \Rightarrow \text{Brake}\ )$
$\wedge\ (((\ \text{YellowLight} \wedge \text{Policeman}\ ) \wedge (\neg \text{Slippery}\ )) \Rightarrow \text{Brake}\ )$
$\wedge\ (\ \text{Policecar} \Rightarrow \text{Policeman}\ )$
$\wedge\ (\ \text{Snow} \Rightarrow \text{Slippery}\ )$
$\wedge\ (\ \text{Slippery} \Rightarrow \neg \text{Dry}\ )$
$\wedge\ (\ \text{RedLight} \Rightarrow \text{Brake}\ )$

Initial KB

from sensors:

YellowLight
$\wedge\ \neg \text{RedLight}$
$\wedge\ \neg \text{Snow}$
$\wedge\ \text{Dry}$
$\wedge\ \text{Policecar}$
$\wedge\ \neg \text{PersonInFrontOfCar}$

Added to KB

Propositional Logic: Semantics

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*

Models

- Model (i.e. possible world):
  - Assignment of truth values to symbols
  - Example: m={P=True , Q=False}
    - Note: Often called "assignment" instead of "model", and "model" is used for an assignment that evaluates to true.
- Validity:
  - A sentence α is valid, if it is true in every model.
- Satisfiability:
  - A sentence α is satisfiable, if it is true in at least one model.
- Entailment:
  - $\alpha \models \beta$ if and only if, in every model in which α is true, β is also true.

Stay at home

Sick	StayAtHome
true	true
false	false
false	true

Does *Sick* entail *StayAtHome*?
A=Yes B=No

Puzzling aspects of Propositional Logic

- Non causality
  - (5 is odd $\Rightarrow$ Tokyo is the capital of Japan)
    - True, because whenever 5 is odd, Tokyo is the capital of Japan. Nothing to do with causality
- Statement always true when antecedent is false
  - (5 is even $\Rightarrow$ Sam is smart)
    - True, because 5 is never even, so no models where this statement is incorrect, regardless of whether Sam is smart or not
- $A \Rightarrow B$
  - read: B is true whenever A is true

Propositional Logic: Semantics

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*

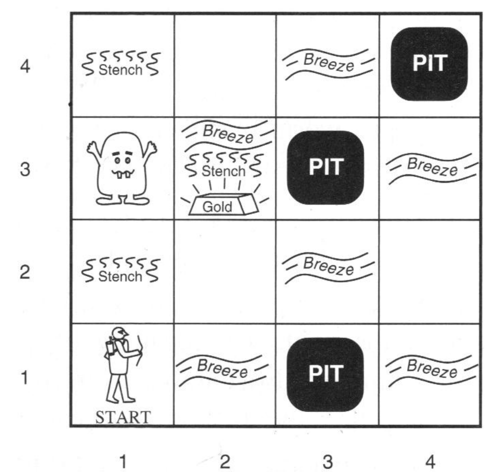
Models

$\neg(P \vee Q) \Leftrightarrow (\neg P) \wedge (\neg Q)$

A) True B) False

Creating a KB

- Variables
  - $P_{i,j}$ is true if there is a pit at position (i,j)
  - $B_{i,j}$ is true if there is a breeze at position (i,j)
- Knowledge
  - R1: $\neg P_{1,1}$ *There is no pit in [1,1]*
  - R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ *Square is breezy iff next to pit*
  - R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- Perceptions
  - R4: $\neg B_{1,1}$ *There is no breeze in [1,1]*
  - R5: $B_{2,1}$ *There is breeze in [2,1]*



Model Checking

- Idea:
  - To test whether $\alpha \models \beta$, enumerate all models and check truth of α and β.
  - α entails β if no model exists in which α is true and β is false (i.e. $(\alpha \wedge \neg\beta)$ is unsatisfiable)
- Proof by Contradiction:

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

Example of model checking

P	Q	$\neg$P	Q→P	$\neg$P $\wedge$ (Q → P)	($\neg$P $\wedge$ (Q → P)) $\wedge$Q
T	T	F	T	F	F
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	F

Models

- $\alpha \models \beta$ iff the sentence ($\alpha \wedge \neg\beta$) is unsatisfiable
- Prove that (-P and (Q → P)) → $\neg$Q
  - By showing that [(-P and (Q → P)) $\wedge$ Q] is not satisfiable
- Possible English translation:
  - P="The street is wet"
  - Q="It is raining"
  - Does "The street not wet" ($\neg$P) and "it is raining →street is wet " (Q → P) imply that "It is not raining? ($\neg$Q)?
- Test if [(-P and (Q → P)) $\wedge$ Q] is satisfiable.
  - It is not satisfiable (always false), therefore (-P and (Q → P)) entails $\neg$ Q

Model Chekcing

- Variables: One for each propositional symbol
- Domains: {true, false}
- Objective Function: $(\alpha \land \neg\beta)$
- Which search algorithm works best?

Doesn't scale well…

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

Inference: Reasoning with Propositional Logic

Modus Ponens:

Latin for *"the way that affirms by affirming"* $(\alpha \Rightarrow \beta) \land \alpha \equiv \beta$

Know:	$\alpha \Rightarrow \beta$	If raining, then soggy courts.
and	α	It is raining.
Then:	β	Soggy Courts.

Modus Tollens:

Latin for "*the way that denies by denying*" $(\alpha \Rightarrow \beta) \land \neg\beta \equiv \neg\alpha$

Know:	$\alpha \Rightarrow \beta$	If raining, then soggy courts.
And	$\neg\beta$	No soggy courts.
Then:	$\neg\alpha$	It is not raining.

And-Elimination:

Know:	$\alpha \land \beta$	It is raining and soggy courts.
Then:	α	It is raining.

Example: Forward Chaining

Knowledge-base describing when the car should brake?

$(\ PersonInFrontOfCar \Rightarrow Brake\ )$

$\wedge\ (((\ YellowLight \wedge Policeman\ ) \wedge (\neg Slippery\ )) \Rightarrow Brake\ )$

$\wedge\ (\ Policecar \Rightarrow Policeman\ )$

$\wedge\ (\ Snow \Rightarrow Slippery\ )$

$\wedge\ (\ Slippery \Rightarrow \neg Dry\ )$

$\wedge\ (\ RedLight \Rightarrow Brake\ )$

$\wedge\ (\ Winter \Rightarrow Snow\ )$

Observation from sensors:

$YellowLight \wedge \neg RedLight \wedge \neg Snow \wedge Dry \wedge Policecar \wedge \neg PersonInFrontOfCar$

What can we infer?

- Policecar ∧ (Policecar ⇒ Policeman): Modus Ponens: Policeman
- Dry ∧ (Slippery ⇒ ¬Dry): Modus Tollens: ¬Slippery
- YellowLight ∧ Policeman ∧ ¬Slippery ∧ (((YellowLight ∧ Policeman) ∧ (¬Slippery)) ⇒ Brake): Modus Ponens: Brake
- YellowLight ∧ ¬RedLight: And Elimination: YellowLight

Inferring (¬Winter) from (¬Snow ∧ (Winter ⇒ Snow)) is

A) Modus Ponens B) Modus Tollens C) And elimination

Other rules

$$
\begin{aligned}
(\alpha \land \beta) &\equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) &\equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) &\equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) &\equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) &\equiv (\neg\alpha \lor \neg\beta) \quad \text{De Morgan} \\
\neg(\alpha \lor \beta) &\equiv (\neg\alpha \land \neg\beta) \quad \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) &\equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) &\equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{aligned}
$$

Inference Strategy: Forward Chaining

Idea:

- Infer everything that can be inferred.
- Notation: In implication $\alpha \Rightarrow \beta$, we say that
  - 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, holds
   a) Add the consequent in *i* to the KB

Continue until no more facts can be inferred.

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.

➔ Avoids inferring unrelated facts.

Example: Backward Chaining

Knowledge-base describing when the car should brake:

$(\text{PersonInFrontOfCar} \Rightarrow \text{Brake})$

$\wedge (((\text{YellowLight} \wedge \text{Policeman}) \wedge (\neg\text{Slippery})) \Rightarrow \text{Brake})$

$\wedge (\text{Policecar} \Rightarrow \text{Policeman})$

$\wedge (\text{Snow} \Rightarrow \text{Slippery})$

$\wedge (\text{Slippery} \Rightarrow \neg\text{Dry})$

$\wedge (\text{RedLight} \Rightarrow \text{Brake})$

$\wedge (\text{Winter} \Rightarrow \text{Snow})$

Observation from sensors:

$\text{YellowLight} \wedge \neg\text{RedLight} \wedge \neg\text{Snow} \wedge \text{Dry} \wedge \text{Policecar} \wedge \neg\text{PersonInFrontOfCar}$

Should the agent brake (i.e. can “brake” be inferred)?

- Goal: Brake
  - Modus Ponens (brake): PersonInFrontOfCar
    - Failure: PersonInFrontOfCar → Backtracking
- Goal: Brake
  - Modus Ponens (brake): YellowLight $\wedge$ Policeman $\wedge$ $\neg$Slippery
  - Known (YellowLight): Policeman $\wedge$ $\neg$Slippery
  - Modus Ponens (Policeman): Policecar $\wedge$ $\neg$Slippery
  - Known (Policecar): $\neg$ Slippery
  - Modus Tollens ($\neg$Slippery): Dry
  - Known (Dry)

Conjunctive Normal Form

- Convert expressions into the form
  - $(l_{1,1} \vee \ldots \vee l_{1,k}) \wedge \ldots \wedge (l_{n,1} \vee \ldots \vee l_{n,k})$
  - *Conjunction of disjunctions*
  - *k*-CNF (k literals)
- Every expression can be transformed into 3-CNF

Conjunctive Normal Form

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

- Original R_2 (From Wumpus)
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- Biconditional elimination
  - $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Implication elimination
  - $(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
- De Morgan
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
- Distributivity of $\wedge$
  - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Conjunctive Normal Form

- Algorithms exist for 3-CNF
  - E.g. 3-SAT