पी.डी.पी.एम.
भारतीय सूचना प्रौद्योगिकी, अभिकल्पन
एवं विनिर्माण संस्थान, जबलपुर

PDPM
Indian Institute of Information Technology,
Design and Manufacturing, Jabalpur

CS 3011: Artificial Intelligence

Knowledge-Based Agents and Logic

Instructors: Dr. Durgesh Singh

CSE Discipline, PDPM IIITDM, Jabalpur -482005

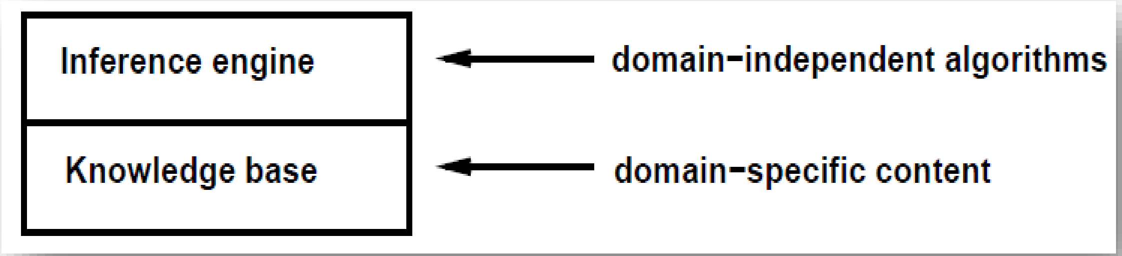
Knowledge-based agents

- Knowledge-based agents use a process of reasoning over an internal representation of knowledge to decide what actions to take.
- The central component of a knowledge-based agent is its knowledge base(KB).
- KB contains a set of sentences in a formal languages and represents some assertion about the world.
  - Each sentence is expressed in a language called a knowledge representation (KR) language.

Knowledge Representation Languages

- Propositional logic
- Predicate Calculus
- Bayesian networks influence diagrams ontologies
- Frame systems certainty factors
- Semantic networks,
- Concept description languages,
- fuzzy logic, description logic.

Knowledge Base Agent…



Knowledge-based agents…

- There must be a way to add new sentences (facts) to the knowledge base and a way to query what is known.
  - The standard names for these operations are TELL and ASK, respectively.

    TELL: "what it needs to know"

    ASK : "what to do next"
  - Both operations may involve inference—that is, deriving new sentences from old.

A Knowledge-based Agent

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

Knowledge Base Agent Program

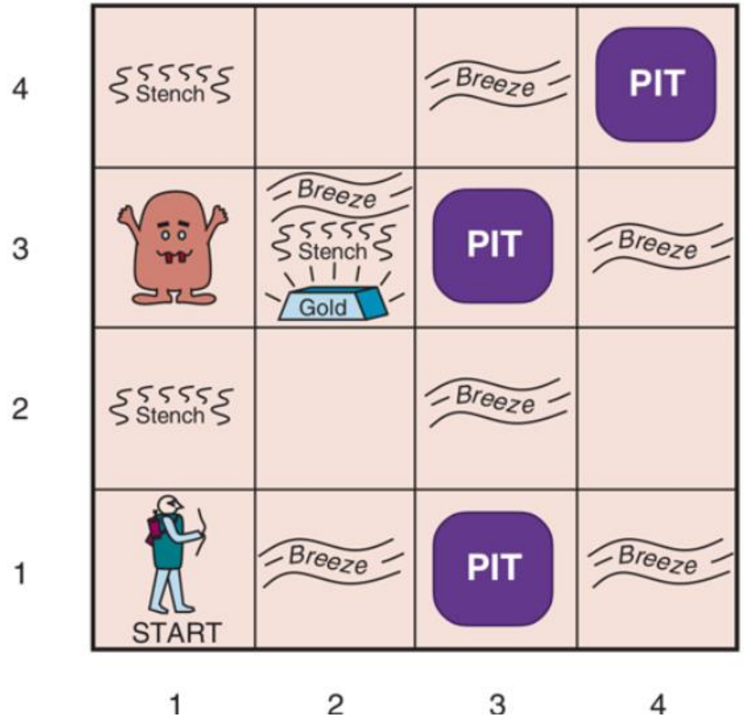
```
function KB-AGENT(percept) returns an action
   persistent: KB, a knowledge base
               t, a counter, initially 0, indicating time

   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.

- **MAKE-PERCEPT-SENTENCE()** constructs a sentence asserting that the agent perceived the given percept at the given time.
- **MAKE-ACTION-QUERY()** constructs a sentence that asks what action should be done at the current time.
- **MAKE-ACTION-SENTENCE()** constructs a sentence asserting that the chosen action was executed.

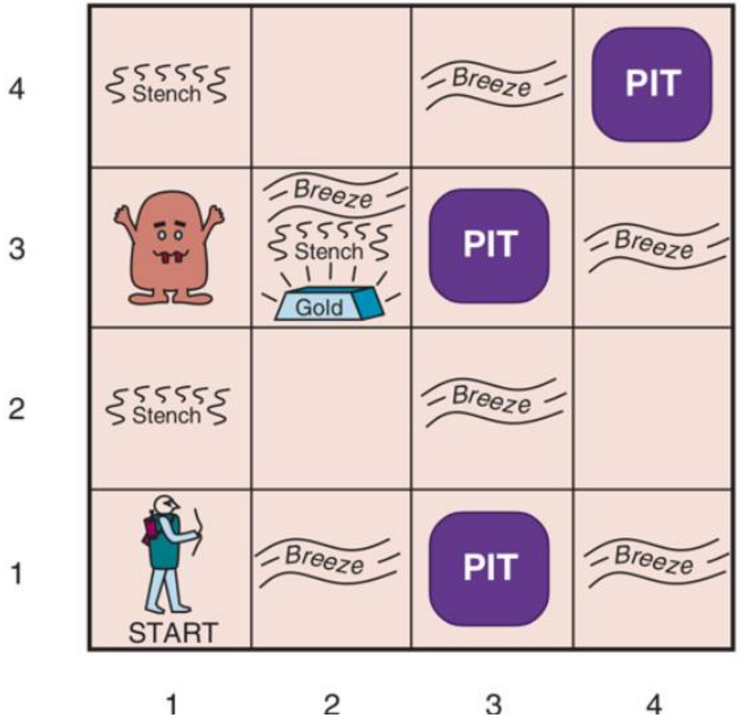
Example: Wumpus World

- 4 X 4 grid of rooms
- Squares adjacent to Wumpus are smelly and squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills Wumpus if you are facing it
- Wumpus emits a horrible scream when it is killed that can be heard anywhere
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square



Example: Wumpus World

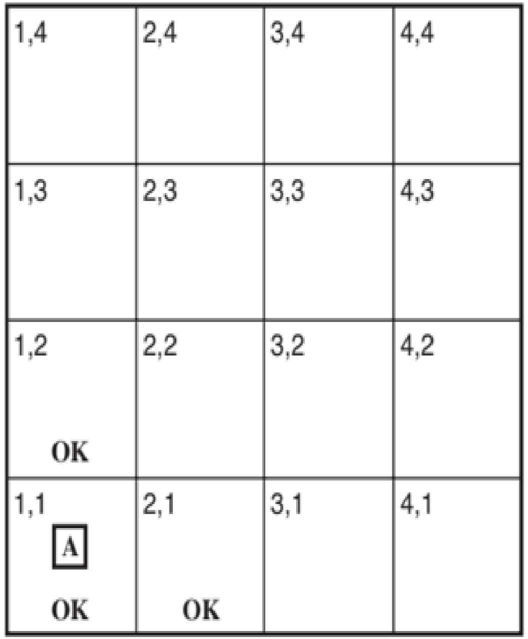
- Performance measure: Gold +1000, Death (eaten or falling in a pit) -1000, -1 per step, -10 for using the arrow
- Environment:
  - ✓ 4 X 4 grid of rooms
  - ✓ Agent starts in square [1,1] facing to the right
  - ✓ Locations of the gold, and Wumpus are chosen randomly with a uniform distribution from all squares except [1,1]
  - ✓ Each square other than the start can be a pit with probability of 0.2
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot
- Sensors: Breeze, Glitter, Smell.



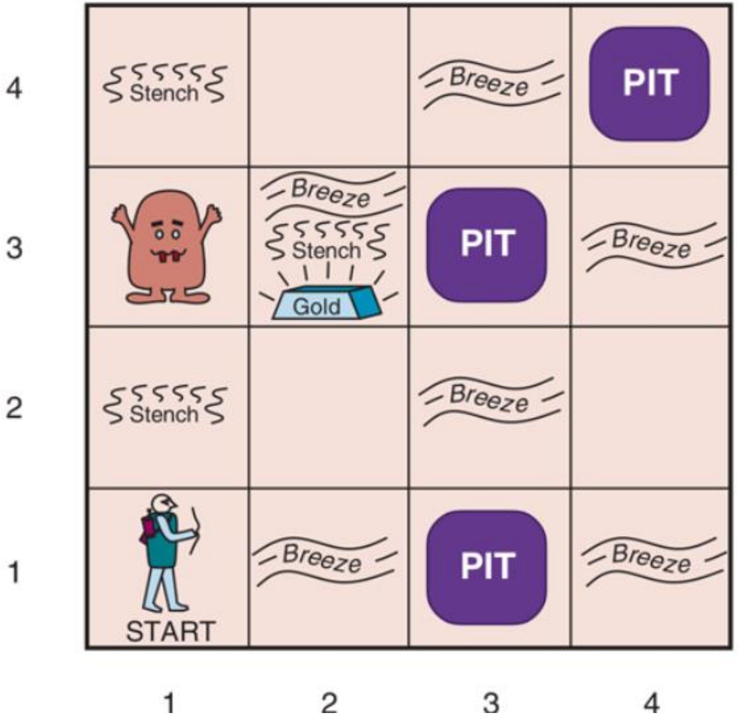
Wumpus World Characterization

- 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

Example: Wumpus World

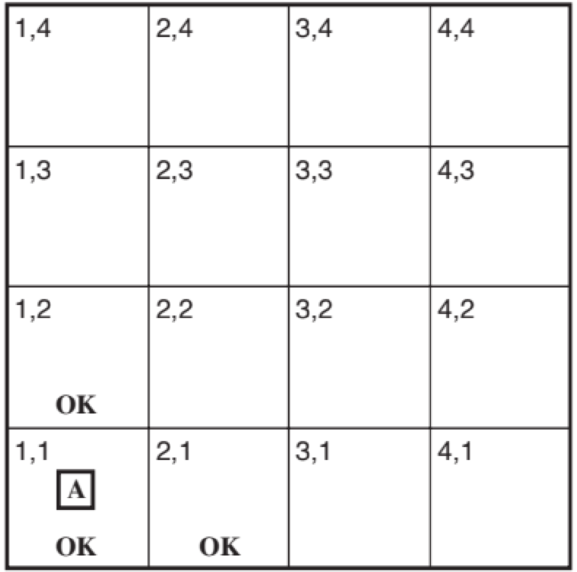


- **A** = *Agent*
- **B** = *Breeze*
- **G** = *Glitter, Gold*
- **OK** = *Safe square*
- **P** = *Pit*
- **S** = *Stench*
- **V** = *Visited*
- **W** = *Wumpus*



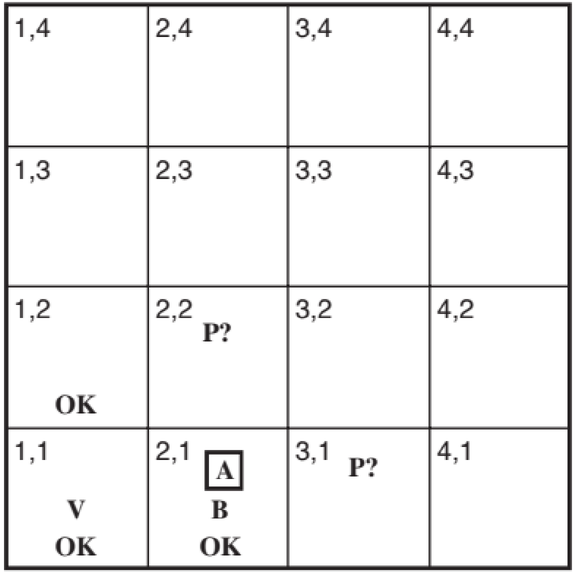
Exploring Wumpus World

Agent's first steps:



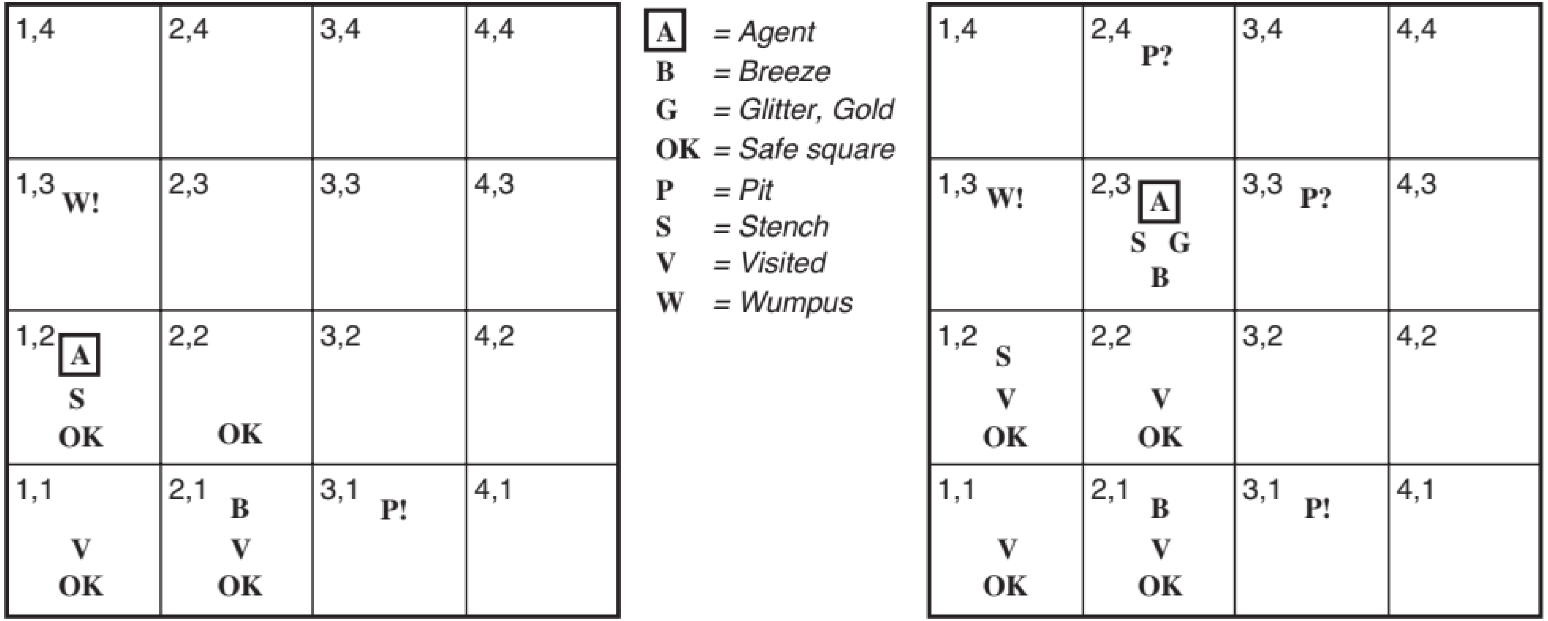
(a)

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



(b)

Exploring Wumpus World



(c) (d)

Logic In General

- Logics are formal languages for representing information, such that conclusions can be drawn
- Syntax defines the structures of sentences in the language
- Semantics define the “meaning" of sentences, i.e., define truth of a sentence in each possible world or model.
- E.g., the language of arithmetic
- $x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence
- $x + 2 \geq y$ is true iff the number $x + 2$ is no 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 $x=0$; $y=6$

Propositional Logic

- Propositional logic is also called Boolean logic
- An assertion is a statement.
- A proposition is an assertion which is either true or false but not both.
- The followings are propositions
  - 4 is a prime number
  - 3+3=6
  - The moon is made of cheese
- The followings are not propositions
  - $X+Y > 4$
  - $X=3$

Propositional Variables

- A Propositional variable denotes an arbitrary proposition with unspecified truth value P, Q, R, ….

Example

P: Anil is honest

Q: Anil is intelligent

Propositional logic: Syntactic Elements

- **Logical constants**: true, false
- **Propositional symbols**: P, Q,... (**atomic sentences**)
- Wrapping **parentheses**: (…)
- Sentences are combined by **connectives**:

$\neg$	not	[negation]
$\wedge$	and	[conjunction]
$\vee$	or	[disjunction]
$\Rightarrow$	implies	[implication / conditional]
$\Leftrightarrow$	is equivalent	[biconditional]

- **Literal**: atomic sentence or negated atomic sentence

  P, $\neg$ P

How to form Propositional Sentences?

- Each symbol is a sentence
- If P is a sentence and Q is a sentence, then
  - (P) is a sentence
  - $P \wedge Q$ is a sentence
  - $P \vee Q$ is a sentence
  - $\neg P$ is a sentence
  - $P \Rightarrow Q$ is a sentence

Note: Propositional Sentences are also called well formed formulae(wff)

Syntax -Summary

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

Semantics

- Defines the rules for determining the truth of a sentence with respect to a particular model.
- In propositional logic, a model simply sets the truth value (true or false) for every proposition symbol.

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

Semantics

- The semantics for propositional logic must specify how to compute the truth value of *any* sentence, given a model.
  - This is done recursively from the atomic sentences

Semantics for Atomic sentences

- *True* is true in every model and *False* is false in every model.
- The truth value of every other proposition symbol must be specified directly in the model. For example, in the model m_1 given earlier, $P_{1,2}$ is false.

Semantics for Complex sentences

- For complex sentences, we have five rules, which hold for any subsentences P and Q (atomic or complex) in any model m, then
  - $\neg P$ 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.

Negation (NOT)

- Unary Operator, Symbol: $\neg$

P	$\neg P$
true (T)	false (F)
false (F)	true (T)

Conjunction (AND)

- Binary Operator, Symbol: $\land$

P	Q	$P \land Q$
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction (OR)

- Binary Operator, Symbol: $\vee$

P	Q	P∨Q
T	T	T
T	F	T
F	T	T
F	F	F

Implication (if - then)

- Binary Operator, Symbol: $\rightarrow$

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

Implication (if - then)

- This can be read several ways: **$P \rightarrow Q$**

If P, then Q

P only if Q

P is a sufficient condition for Q

Q is a necessary condition for P

Q if P

Q follows from P

Q provided P

Implication (if - then)

- $Q \rightarrow P$ is called converse
- $\neg Q \rightarrow \neg P$ is called the contrapositive

Biconditional (if and only if)

- Binary Operator, Symbol: $\leftrightarrow$

P	Q	$P \leftrightarrow Q$
T	T	T
T	F	F
F	T	F
F	F	T

Biconditional (if and only if)

- P is equivalent to Q
- **P if and only if Q**
- **P is a necessary and sufficient condition for Q**

Statements and Operators

- Statements and operators can be combined in any way to form new statements.

P	Q	$\neg P$	$\neg Q$	$(\neg P)\vee(\neg Q)$
T	T	F	F	F
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

Statements and Operations

- Statements and operators can be combined in any way to form new statements.

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

Equivalent Statements

P	Q	$\neg(P \wedge Q)$	$(\neg P) \vee (\neg Q)$	$\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q)$
T	T	F	F	T
T	F	T	T	T
F	T	T	T	T
F	F	T	T	T

- The statements $\neg(P \wedge Q)$ and $(\neg P) \vee (\neg Q)$ are logically equivalent, since $\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q)$ is always true.

Logical Identities

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

Example

- Simplify the expression

$$[(A \rightarrow B) \vee (A \rightarrow D)] \rightarrow (B \vee D)$$