Knowledge and Reasoning
Sub: Logic

Informatics Engineering Study Program
School of Electrical Engineering and Informatics
Institute of Technology Bandung

Overview

- Review
- Intro to Knowledge Representation
- Knowledge Representation Manipulation
- Intro to Logic

Review: Simple Problem Solving Agent

- Agent design: formulate problem → search solution → execute

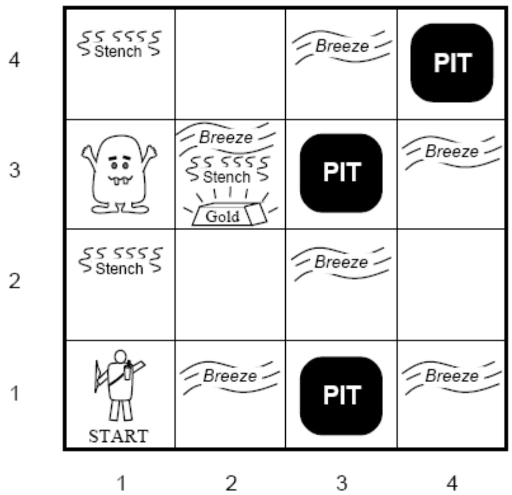
```
function SIMPLE-PROBLEM-SOLVING-AGENT( percept) returns an action
   static: seq, an action sequence, initially empty
           state, some description of the current world state
           goal, a goal, initially null
           problem, a problem formulation

   state ← UPDATE-STATE(state, percept)
   if seq is empty then
       goal ← FORMULATE-GOAL(state)
       problem ← FORMULATE-PROBLEM(state, goal)
       seq ← SEARCH( problem)
   action ← RECOMMENDATION(seq, state)
   seq ← REMAINDER(seq, state)
   return action
```


Simple Problem Solving Agent

- Problem Solving Agent handles finite states (i.s – goal)
- States in path finding problem: agent locations
  e.g.: in Arad, in Bucharest
- States in CSP: set variables X_i with values from domain D_i
  e.g.: {}, {WA=red, NT=green, Q=red, SA=blue, NSW=green, V=red, T=green}
- Informed search enables problem solving agents to perform well (with admissible heuristics)
  - This knowledge is very specific and inflexible
  - General knowledge and reasoning → knowledge –based agent

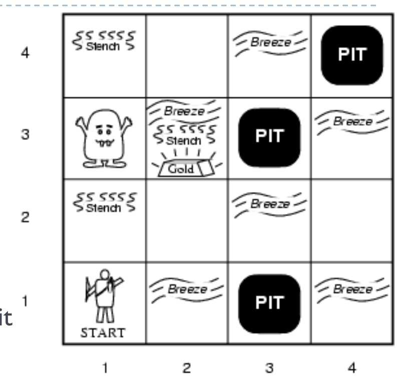
The Wumpus World



- Cave with rooms
- Wumpus eats anyone who enters its room
- Wumpus can be shot by an agent, but the agent has only one arrow
- Pit will trap anyone, except for the wumpus
- Agent can find gold heap

The Wumpus World: Task Environment

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
  - Sensors: Stench, Breeze, Glitter, Bump, Scream
  - Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

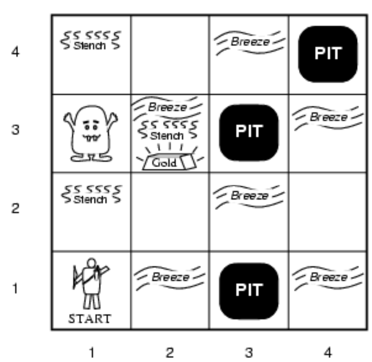


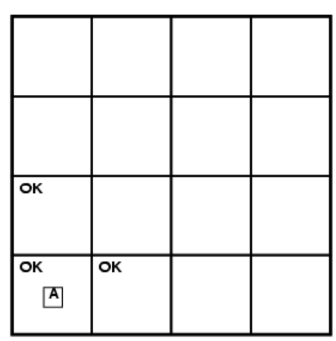
Wumpus world characterization

- Fully 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

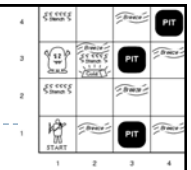
Exploring a wumpus world

[1,1] : OK (safe)
Percept [1,1] : [None, None, None, None, None]
No stench in [1,1] : No wumpus in [1,2] and [2,1]
No breeze in [1,1]: No pit in [1,2] and [2,1]
Action: forward

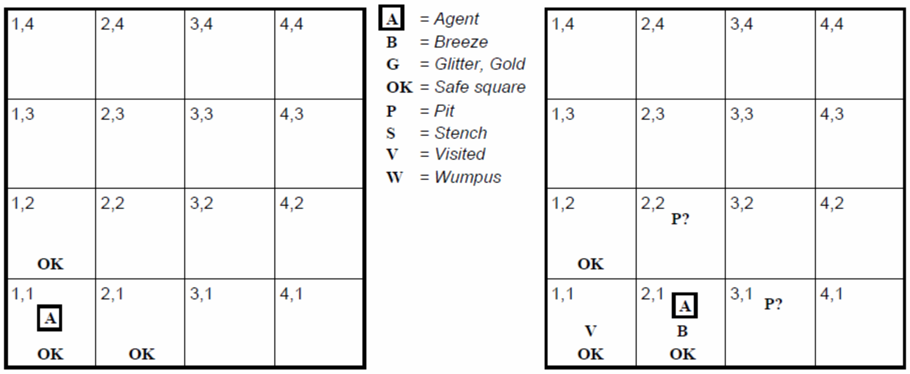




Exploring a wumpus world (2)



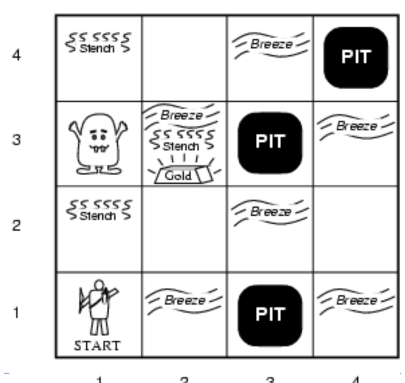
Percept [2, 1] : [None, Breeze, None, None, None]
No stench in [2,1] : No wumpus in [3,1] and [2,2]
Breeze in [2,1]: there must be a pit in [3,1] or [2,2]
Set action: go back to [1,1] and forward to [1,2]

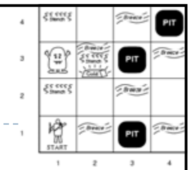


(a) (b)

Exploring a wumpus world (3)

Percept [1,2] : [Stench, None, None, None, None]
Stench in [1,2] : there must be a wumpus in [1,3] or [2,2] or [1,1]
No wumpus in [1,1] and No stench in [2,1] → wumpus in [1,3]
No breeze in [1,2]: No pit in [1,3] and [2,2] → pit in [1,3] and [2,2] OK
Set action: go to [2,2]

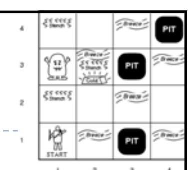




Exploring a wumpus world (4)

Percept [2,2] : [None, None, None, None, None]
No stench in [2,2] : No wumpus in [2,3] and [3,2]
No breeze in [2,2]: No pit in [2,3] and [3,2]
Set action: go to [2,3]

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 $\boxed{A}$ S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1



Exploring a wumpus world (5)

Percept [2,3] : [Stench, Breeze, Glitter, None, None]
Action: Grab

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

The Wumpus World: Summary

- Fundamental properties of logical reasoning
  - In each step, the agent draws a conclusion from available information
  - Conclusion is guaranteed to be correct if the available information is correct
- Knowledge-based agent

Simple Knowledge Based Agent

- Agent design: declarative approach
- TELL KB what it needs to know
- ASK itself what to do -- answers should follow from the KB

```
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
```

Intro to Knowledge Representation

- Instead of thinking about all the ways a world could be, we're going to work in a language of expressions that describe those sets
- It's one way of representing knowledge

Intro to Knowledge Representation (2)

- A language (to represent knowledge/ information)
  a set of syntactic and semantic conventions that makes it possible to describe things, and a way of manipulating expression in language
- Syntax: a description of what you're allowed to write down, what the expressions are, that are legal in a language.
- Semantic: which is some story about what those expressions mean.
- In short: Syntax is form and semantics is content.

Intro to Knowledge Representation (3)

- Examples:
  - Map → symbols, interpretation of symbols to represent real geographic condition
  - Natural Languages → collection of symbols to explain things
- Objectives of selection:
  - Processing → as simple as possible
  - Represent real-world problems into more comprehensible problems

Intro to Knowledge Representation (4)

- The representation should be:
  - Suitable for problem domain
    - Decision tree for classification
    - Skeletal construction for construction
    - Rule for all problem domain
  - Suitable for the tasks (inference)
    - Decision tree including interview process
    - Probability model for decision with uncertainty
  - Suitable for users (man or machine)
    - Semantic network for user, rule for machine

Intro to Knowledge Representation (5)

- Requirements of knowledge representation:
  - No contradiction
  - Each symbol must be unique
  - Explain certain objects, relations and attributes
  - Efficient manipulation in computer system
- Several examples → application oriented
  - *Logic: robotics*
  - *Production rules*: expert systems
  - *Semantic network, frame*: structured object representation → story understanding system
    - Information extraction

Deduction, Induction, Abduction

All men are mortal

Socrates is a man

∴ Socrates is mortal

This swan is white; That swan is white; Every swan that I've ever seen is white;

∴ All swans are white

Deduction, Induction, Abduction

If it is raining then the streets are wet
It is raining
∴ The streets are wet

If it is raining then the streets are wet
The streets are wet
∴ It is raining

The sun has risen every day so far
∴ The sun will rise tomorrow

Knowledge Representation Manipulation

- Deduction → logic principles

All men are mortal (*premise*);
Socrates is a man; (*premise*)
∴ Socrates is mortal (*deductive conclusion*)
2. If it is raining then the streets are wet (*premise*);
It is raining (*premise*);
Therefore the streets are wet (*deductive conclusion*)
- *truth preserving*

Knowledge Representation Manipulation (2)

- Induction → generalization of current observation

  1.This swan is white;That swan is white; Every swan that I've ever seen is white;

  Therefore all swans are white

  2.The sun has risen every day so far;

  Therefore, the sun will rise tomorrow

  - not always *truth preserving* →

  - for hypothesis → machine learning

Knowledge Representation Manipulation (3)

- Abduction

  1. If a person has a cold, then he has a runny nose;

  Jack has a runny nose;

  Therefore Jack has a cold

  - possibility of wrong conclusion

  - practical reasoning → diagnosis

INTRODUCTION TO LOGIC

What is a logic

- A formal language
  - Syntax: what expression are legal
  - Semantics: what legal expressions mean
  - Proof system: a way of manipulating syntactic expressions to get other syntactic expressions (which will tell us something new)
- Why proofs? Two kind of inferences an agent might want to make:
  - Multiple percepts → conclusion about the world
  - Current state & operator → properties of next state

Propositional logic

Syntax: what you're allowed to write

- Sentences → wffs
- true and false are sentences (base cases)
- Propositional variables are sentences: p,q,r,z
- If Φ and Ψ (metavariables) are sentences, then so are:
  (Φ), ¬ Φ, Φ ∨Ψ, Φ ∧Ψ, Φ→ Ψ, Φiff Ψ
- Nothing else is a sentence

Precedence

¬	highest
∧	
∨	
→	
↔	lowest

A ∨ B ∧ C	A ∨ (B ∧ C)
A ∧ B → C ∨ D	(A ∧ B) → (C ∨ D)
A → B ∨ C ↔ D	(A → (B ∨ C)) ↔ D

- Precedence rules enables 'shorthand' form of sentences, but formally only the fully parenthesized form is legal
- Syntactically ambiguous forms allowed in shorthand only when semantically equivalent: A ∧ B ∧ C is equivalent to (A ∧B) ∧C and A ∧(B ∧C)

Propositional logic

Semantics: meaning of sentences

- Truth value {**t**,**f**}
- Interpretation is an assignment of truth values to the propositional variables

  holds(Φ,i) [sentence Φ is **t** in interpretation i]

  fails(Φ,i) [sentence Φ is **f** in interpretation i]

Terminology

- A sentence is valid iff its truth value is **t** in all interpretation
  - Examples: true, ¬false, p ∨ ¬p
- A sentence is satisfiable iff its truth value is **t** in at least one interpretation
  - Examples: p, true, ¬p
- A sentence is unsatisfiable iff its truth value is **f** in all interpretation
  - Examples: p ∧ ¬p, false, ¬true

All are finitely decidable

Satisfiability

- Related to constraint satisfaction
- Given a sentence S, try to find an interpretation i such that *holds*(S,i)
- Analogous to finding an assignment of values to variables such that the constraints hold
- Brute force method: enumerate all interpretations and check
- Better methods: heuristic search, constraint propagation, local search

Checking Interpretation

Knowledge we have (knowledge base):

- If today is sunny, then Tomas will be happy
  ($s \rightarrow h$)
- If Tomas is happy, the lecture will be good
  ($h \rightarrow g$)
- Today is sunny (s)

Should we conclude that the lecture will be good?

Checking Interpretation

S	H	G	S →H	H→G	S	G
t	t	t	t	t	t	t
t	t	f	t	f	t	f
t	f	t	f	t	t	t
t	f	f	f	t	t	f
f	t	t	t	t	f	t
f	t	f	t	f	f	f
f	f	t	t	t	f	t
f	f	f	t	t	f	f

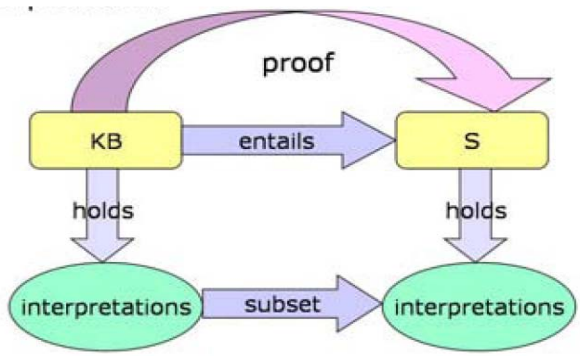
Entailment

A Knowledge base (KB) entails a sentence S iff every interpretation that makes KB true also makes S true

- Enumerate all interpretations
- Select those in which all element of KB are true
- Check to see if S true in all of those interpretations
- Problems → too many interpretations, in general!

Entailment and Proof

A proof is a way to test whether a KB entails a sentence, without enumerating all possible interpretations



Proof

- Proof is a sequence of sentences
- First one are premises
- Write down next line as result of applying inference rule to previous lines
- When S is on line, you have proved S from KB
- Inference rule: natural deduction
  - Modus ponens, Modus tolens, And-introduction, And-elimination
- Example KB: $P \wedge Q$, $P \rightarrow R$, $(Q \wedge R) \rightarrow S$

Proof Systems

- There are many natural deduction systems; typically proof checker → sound but not complete
- Natural deduction uses lots of inference rules which introduces a large branching factor in the search for a proof
- In general, you need to do 'proof by cases' which introduces even more branching

→ Resolution (inference rule that is sound and complete) : require conjunctive normal form

Conjunctive Normal Form (CNF)

- Satisfiability problems are written as CNF formulas; example:

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

  - $(A \lor B \lor \neg C)$ is clause, which is a disjunction of literals
  - A, B and ¬C are literals, each of which is a variable or the negation of a variable
  - Each clause is a requirement which must be satisfied and it has different ways of being satisfied
  - Every sentence in propositional logic can be written in CNF

Converting to CNF

1. Eliminate arrows using definitions
2. Drive in negations using De Morgan's Laws
   - $\neg(E \lor F) \equiv \neg E \land \neg F$
   - $\neg(E \land F) \equiv \neg E \lor \neg F$
3. Distribute or over and
   - $A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$
4. Every sentence can be converted to CNF, but it may grow exponentially in size

CNF Conversion Example

$(A \lor B) \Rightarrow (C \Rightarrow D)$

1. Eliminate arrows
   - $\neg(A \lor B) \lor (\neg C \lor D)$
2. Drive in negations
   - $(\neg A \land \neg B) \lor (\neg C \lor D)$
3. Distribute or over and
   - $(\neg A \lor \neg C \lor D) \land (\neg B \lor \neg C \lor D)$

Propositional Resolution

- Resolution rule:

$$\frac{\begin{array}{l}\alpha \lor \beta \\ \neg\beta \lor \gamma\end{array}}{\alpha \lor \gamma}$$

⇒ It turns out that one rule is all you need to prove things.

⇒ At least to prove that a set of sentences is not satisfiable (by contradiction)

Propositional Resolution

- Resolution refutation:
  1. Convert all sentences to CNF
  2. Negate the desired conclusion (converted to CNF)
  3. Apply resolution rule until either
     - Derive false (a contradiction)
     - Can't apply any more
- Resolution refutation is sound & complete
  - If we derive a contradiction, then the conclusion follows from the axioms
  - If we can't apply any more, then the conclusion cannot be proved from the axioms.

Propositional Resolution Example

Prove R

1	P V Q
2	P ⇨ R
3	Q ⇨ R

false V R

¬R V false

false V false

Step	Formula	Derivation
1	P V Q	Axiom
2	¬P V R	Axiom
3	¬Q V R	Axiom
4	¬R	Negated conclusion
5	Q V R	1,2
6	¬P	2,4 (not needed)
7	¬Q	3,4
8	R	5,7
9	● (empty clause)	4,8

First Order Logic (FOL)

- Statements that cannot be made in propositional logic but can be made in FOL
- In FOL variables refer to things in the world and can be quantified over them

FOL Syntax

- Term
  - Constant symbols: Fred, Japan, Bacterium39
  - Variables: x, y, a
  - Function symbols: applied to one or more terms →

  F(x), f(f(x)), mother-of(John)
- Sentences
  - A predicate symbol applied to zero or more terms:

  On(a,b), Sister(Jane, Joan), Sister(mother-of(John),Jane)
  - $t_1 = t_2$
  - If v is a variable and Φ is a sentence, then ∀v. Φ and ∃v. Φ are sentences
  - Closure under sentenial operators: ¬ ∧ ∨ ⇨ ⇔ ( )

FOL Semantic

- Interpretation I
  - U set of objects (called ‘domain of discourse’ or ‘universe’)
  - Maps constant symbols to element of U
  - Maps predicate symbols to relations on U (binary relation is a set of pairs)
  - Maps function symbols to function on U (function is a binary relation with a single pair for each elemen in U, whose first item is that element)

Semantics of Quantifiers

- Extend an interpretation I to bind variable x to element a ∈ U: $I_{x/a}$
  - holds(∀x. Φ,I) iff holds(Φ, $I_{x/a}$) for all a ∈ U
  - holds(∃x. Φ,I) iff holds(Φ ,$I_{x/a}$) for some a ∈ U
- Quantifier applies to formula to right until an enclosing right parenthesis:

  (∀x.P(x) ∨ Q(x)) ∧ (∃x.R(x) ⇒ Q(x))

Writing FOL

- Cats are mammals [cat, mammal]
  - ∀x.cat(x) ⇒ mammal(x)
- A nephew is a sibling's son [nephew, sibling, son]
  - ∀xy.[nephew(x,y) ⇔ ∃z.[sibling(y,z) ∧ son(x,z)]]
- Everybody loves somebody [$loves^2$]
  - ∀x.∃y.loves(x,y)
  - ∃y.∀x.loves(x,y)
- Nobody loves Jane
  - ∀x. ¬loves(x,Jane)
  - ¬∃x.loves(x,Jane)
- Whoever has a father, has a mother
  - ∀x.[[∃y.father(y,x)] ⇒ [∃z.mother(z,x)]]

Entailment in FOL

- KB entails S: for every interpretation I, if KB holds in I, then S holds in I
- Computing entailment is impossible in general, because there are infinitely many possible interpretations
- Even computing holds is impossible for interpretations with infinite universes
- So, we'll do proofs
- In FOL, if S is entailed by KB, then there is a finite proof of S from KB → satisfiability problems
- We need Clausal Form (as in Propositional Logic)

First-Order Resolution

Two new things:

- Converting FOL to clausal form
- Resolution with variable substitution

Converting to Clausal Form

1. Eliminate arrows using definitions
2. Drive in negations
   $\neg\forall x.\ \alpha$ becomes $\exists x.\ \neg\ \alpha$
3. Rename variables apart
   $\forall x.\exists y.(\neg P(x) \vee \exists x.Q(x,y))$ becomes
   $\forall x_1.\exists y_2.(\neg P(x_1) \vee \exists x_3.Q(x_3,y_2))$
4. Skolemize
   - Substitute brand new name for each existentially quantified variable
     - $\exists x.\ P(x) \Rightarrow P(Fred)$
     - $\exists x.\ P(x,y) \Rightarrow P(x_{11},y_{12})$
     - $\exists x.\ P(x) \wedge Q(x) \Rightarrow P(Blue) \wedge Q(Blue)$
     - $\exists y.\ \forall x.\ Loves(x,y) \Rightarrow \forall x.\ Loves(x,Alice)$
     - $\forall x.\ \exists y.\ Loves(x,y) \Rightarrow \forall x.\ Loves(x,Beloved(x))$

Converting to Clausal Form (2)

5. Drop universal quantifiers
6. Convert to CNF
7. Rename the variables in each clause
   $\forall x.\ P(x) \wedge Q(x) \Rightarrow \forall y.P(y) \wedge \forall z.Q(z)$

Example: Converting to Clausal Form

1. John owns a dog
   a. $\exists x.\ D(x) \land O(J,x)$
   b. $D(Fido) \land O(J,Fido)$
2. Anyone who owns a dog is a lover-of-animals
   a. $\forall x.\ (\exists y.\ D(y) \land O(x,y)) \Rightarrow L(x)$
   b. $\forall x.\ \neg(\exists y.\ D(y) \land O(x,y)) \lor L(x)$
   c. $\forall x.\ (\forall y.\ \neg D(y) \lor \neg O(x,y)) \lor L(x)$
   d. $\neg D(y) \lor \neg O(x,y) \lor L(x)$

Proving Validity

- How do we use resolution refutation to prove something is valid?
  - Normally, we prove a sentence is entailed by the set of axioms
  - Valid sentences are entailed by the empty set of sentences
    - E is valid
    - $\{\} \models E$ [empty set of sentences entails E]
    - $\{\} \vdash E$ [empty set of sentences proves E]
  - To prove validity by refutation, negate the sentence and try to derive contradiction

Proving Validity: Example

Proving validity of:

∃x. (P(x) ⇨ P(A)) ∧ (P(x) ⇨ P(B))

¬ (∃x. (P(x) ⇨ P(A)) ∧ (P(x) ⇨ P(B)))
¬ (∃x. (¬P(x) ∨ P(A)) ∧ (¬P(x) ∨ P(B)))
∀x. ¬ ((¬P(x) ∨ P(A)) ∧ (¬P(x) ∨ P(B)))
∀x. ¬(¬P(x) ∨ P(A)) ∨ ¬(¬P(x) ∨ P(B))
∀x. (P(x) ∧ ¬P(A)) ∨ (P(x) ∧ ¬P(B))
(P(x) ∧ ¬P(A)) ∨ (P(x) ∧ ¬P(B))
(P(x) ∨ P(x)) ∧ (P(x) ∨ ¬P(B)) ∧ (¬P(A) ∨ P(x)) ∧ (¬P(A) ∨ ¬P(B))

1	P(x)	
2	P(x) ∨ ¬P(B)	
3	¬P(A) ∨ P(x)	
4	¬P(A) ∨ ¬P(B)	
5	¬P(B)	4,1 {x/A}
6	•	5,1 {x/B}

References

1. George F. Luger, William A. Stubblefield, *Artificial Intelligence Structures and Strategies for Complex Problem Solving 2nd Edition*, The Benjamin/ Cummings Publishing Company Inc., California, 1993

2. Efraim Turban, *Decision Support and Expert Systems*, Macmillian Publishing Company, New York, 1988

3. Russel S., Norvig P., Artificial Intelligence A Modern Approach 1st edition, Prentice Hall, New Jersey, 1995

THANK YOU