

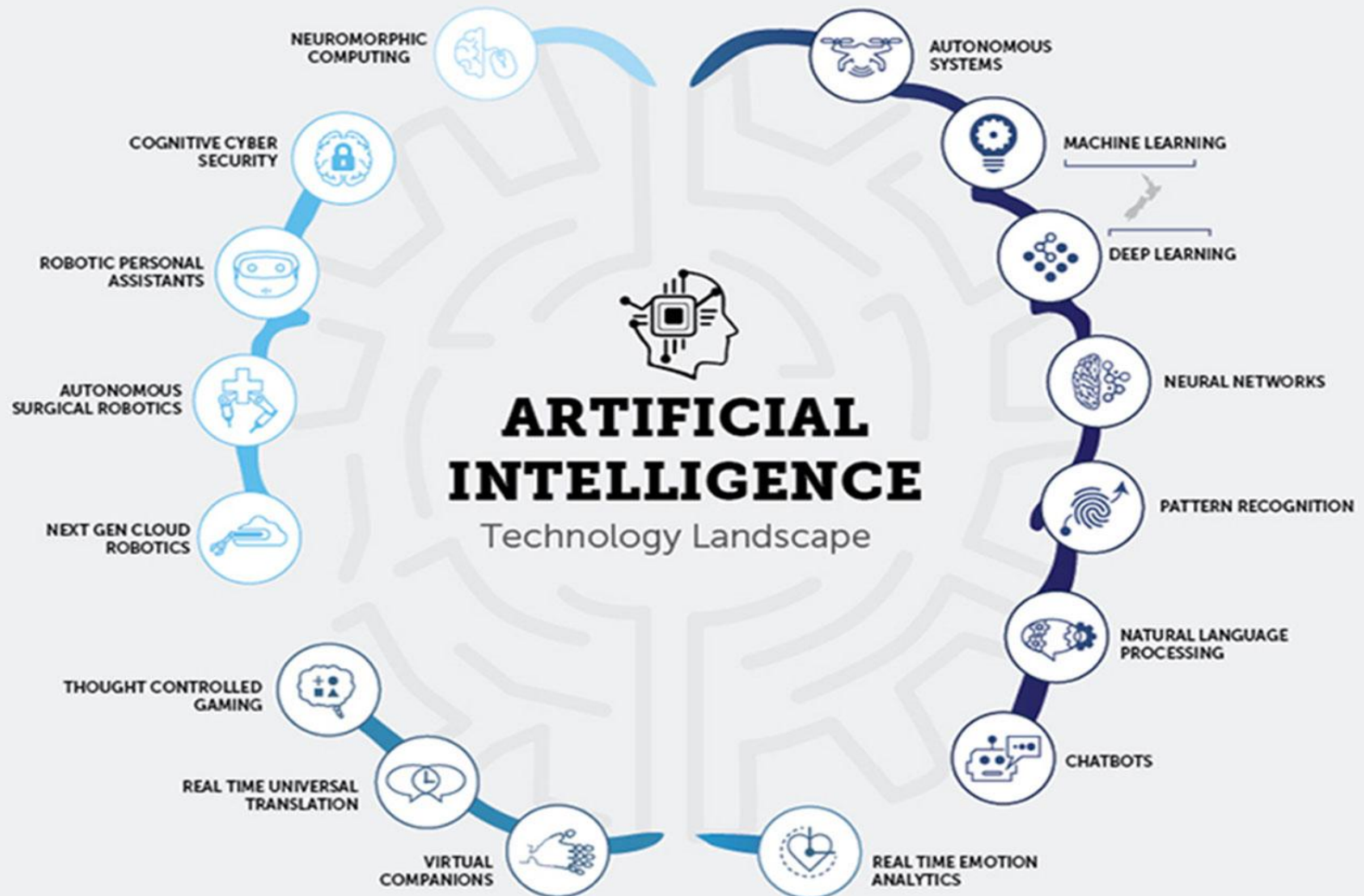
Introduction to Artificial Intelligence

Dr. Indrajit Pan

Associate Professor, Dept. of IT

RCC Institute of Information Technology, Kolkata

Application Domain of AI



Take a look on the Time Scale

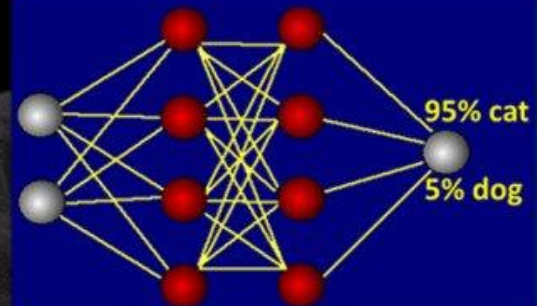
ARTIFICIAL INTELLIGENCE



MACHINE LEARNING



DEEP LEARNING



1950's

1960's

1970's

1980's

1990's

2000's

2010's

Scientists' view on AI

Rationally acting systems

- Field of studies aimed at clarifying and emulating intelligent behaviour in terms of computational processes.

[Schalkoff 1990]

- A branch of computer science concerning automation of intelligent behaviour.

[Luger, Stubblefield 1993]

Systems thinking like men

- Making computers thinking machines with a brain and in a every sense of the word.

[Haugeland 1985]

- Automation of tasks that are considered human: thinking, decision making, problem solving, learning, etc.

[Bellman 1978]

AI definition categories

Rationally thinking systems

- Testing the abilities and thought processes through the use of computational models.

[Charniak, McDermott 1985]

- Research on calculations that make them capable of perception, reasoning and action

[Winston 1992]

Systems acting like people

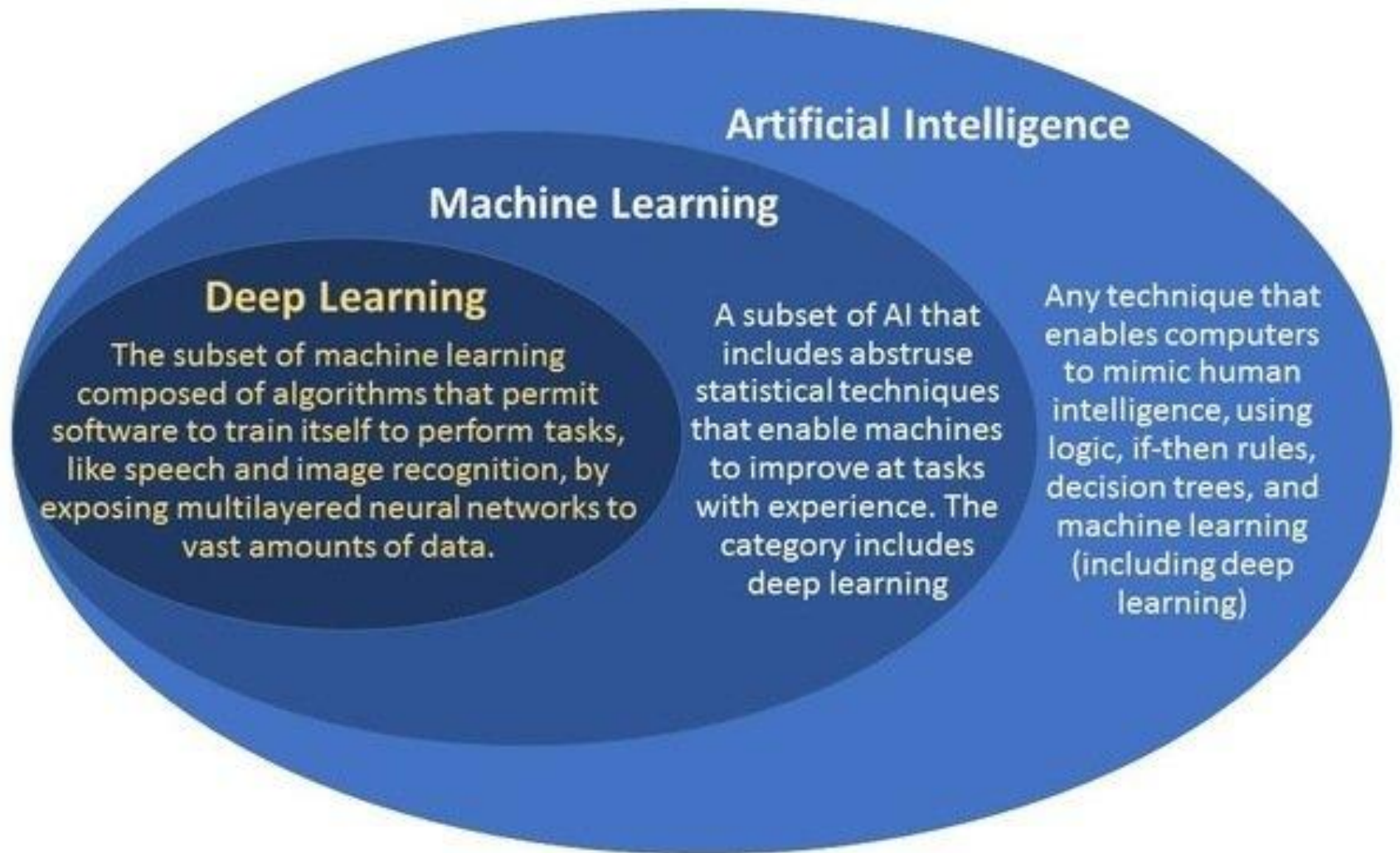
- The art of creating machines performing functions that require intelligence if done by humans.

[Kurzweil 1990]

- Research on how to make computers able to perform the things which at the moment are better performed by humans

[Rich, Knight 1991]

AI, ML and DL



Precise view on AI, ML and DL

Artificial Intelligence

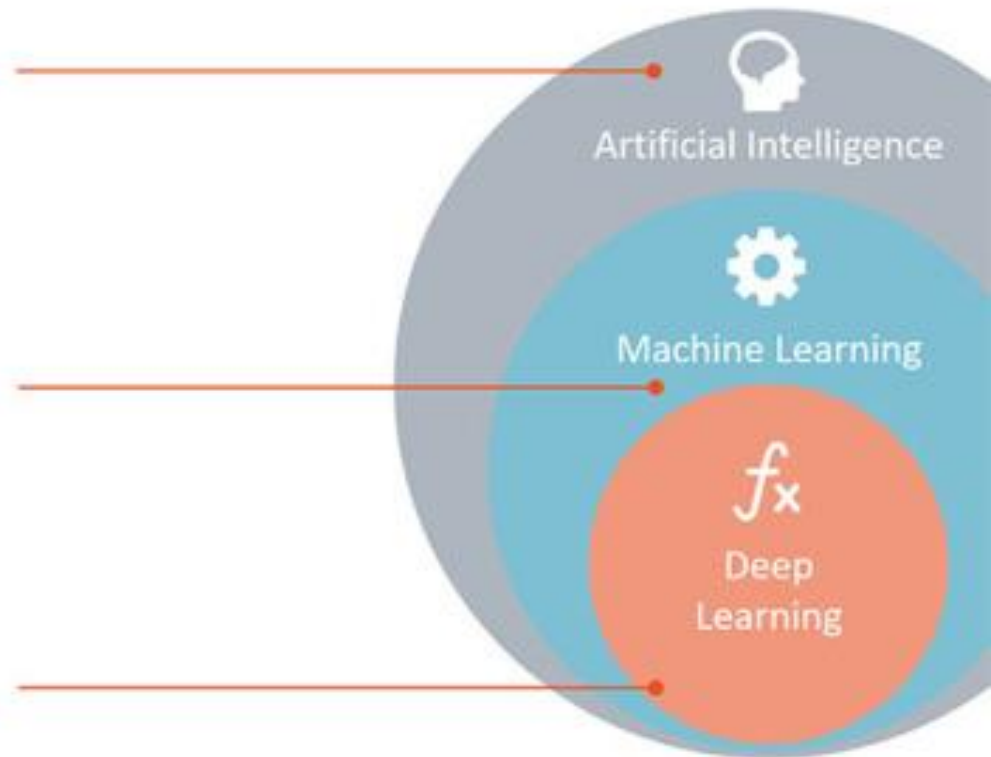
Any technique which enables computers to mimic human behavior.

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning

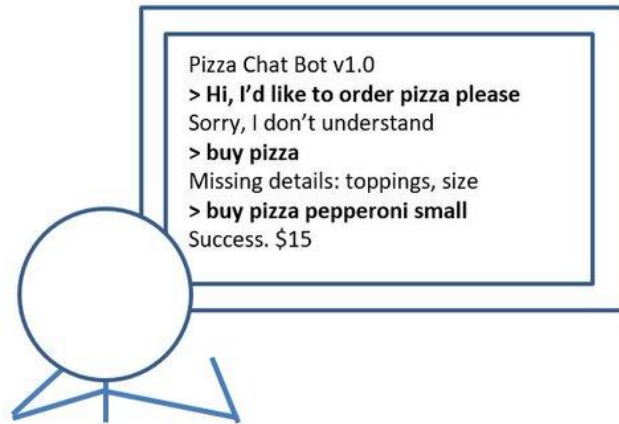
Subset of ML which make the computation of multi-layer neural networks feasible.



AI vs Human Interaction

Chat bot vs human conversation

In the past...



Hi, I'd like to order pizza please.

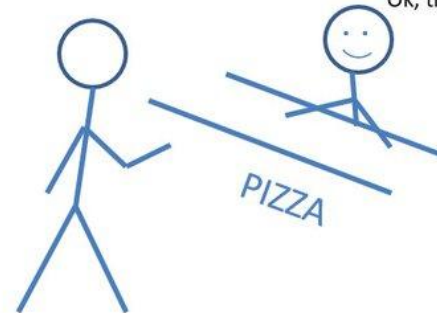
Sure, what do you want for the toppings?

I think I will take the pepperoni.

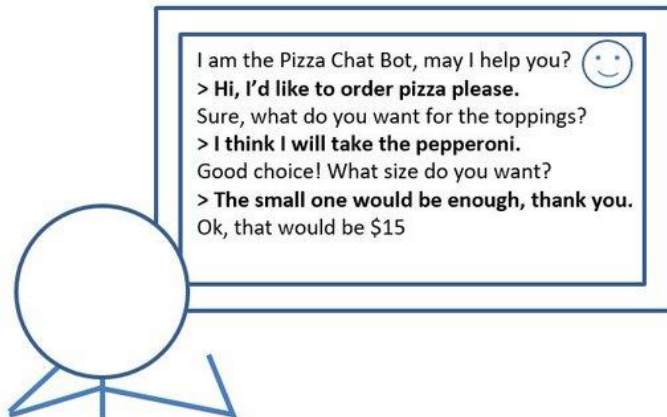
Good choice! What size do you want?

The small one would be enough, thank you.

Ok, that would be \$15

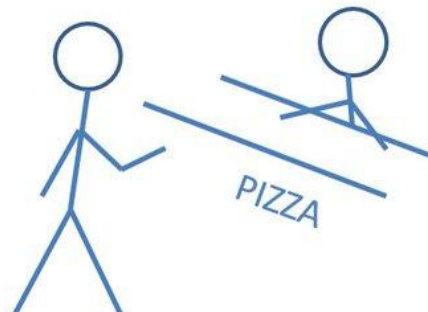


In the near future... (after the advancement in AI, ML and NLP)

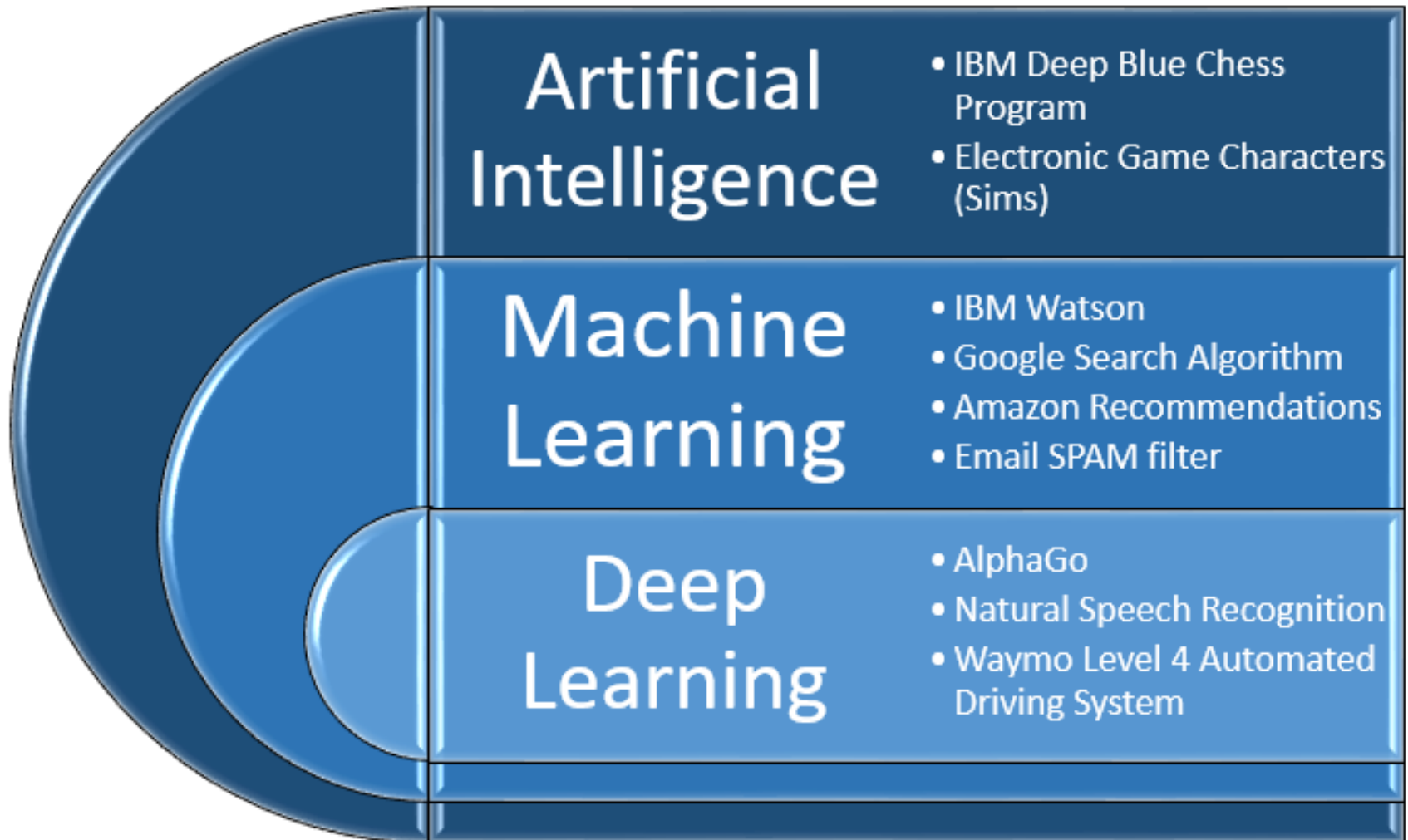


Pizza, pepperoni, small

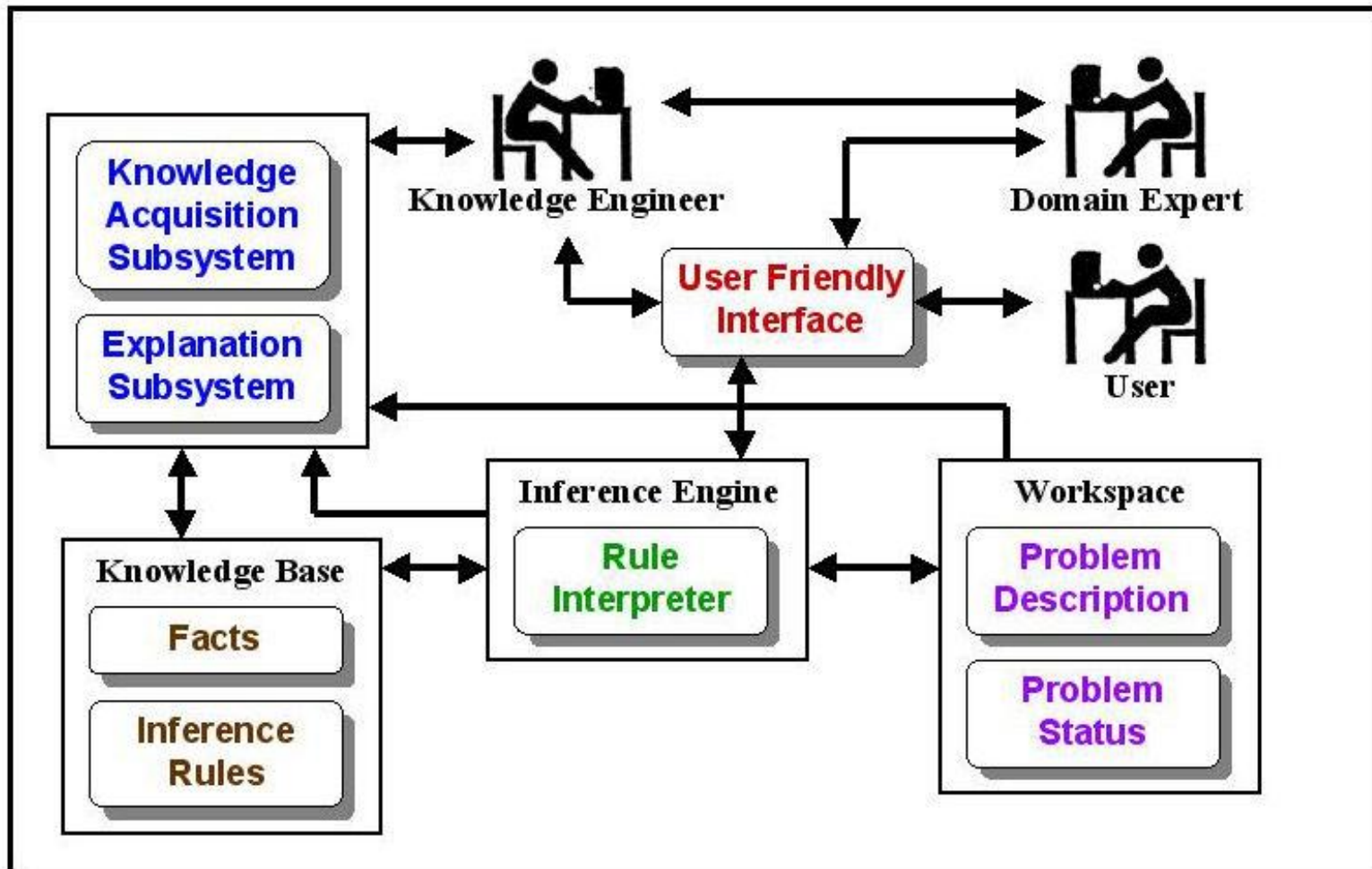
\$15



Real Applications



Components of AI



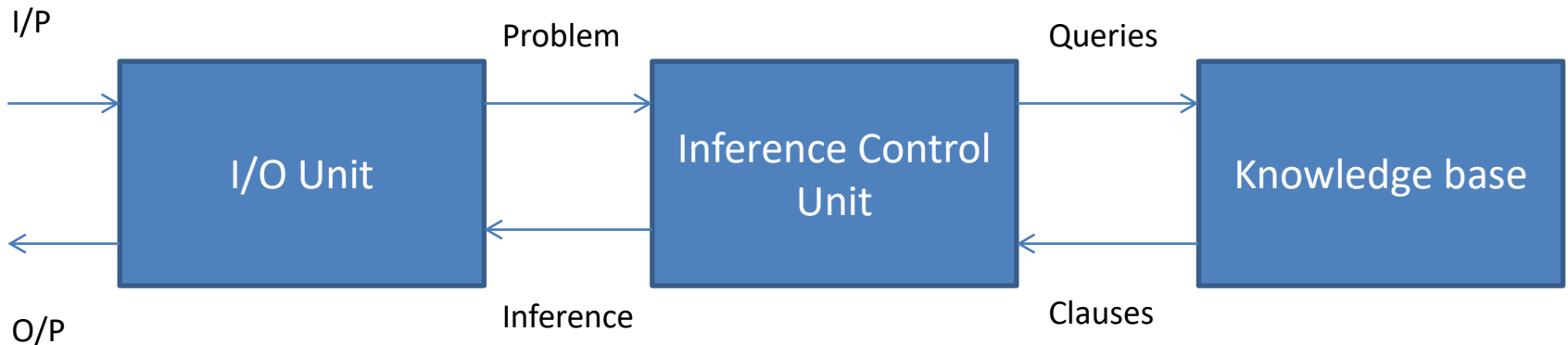
Focus on the Core Components



Knowledgebase

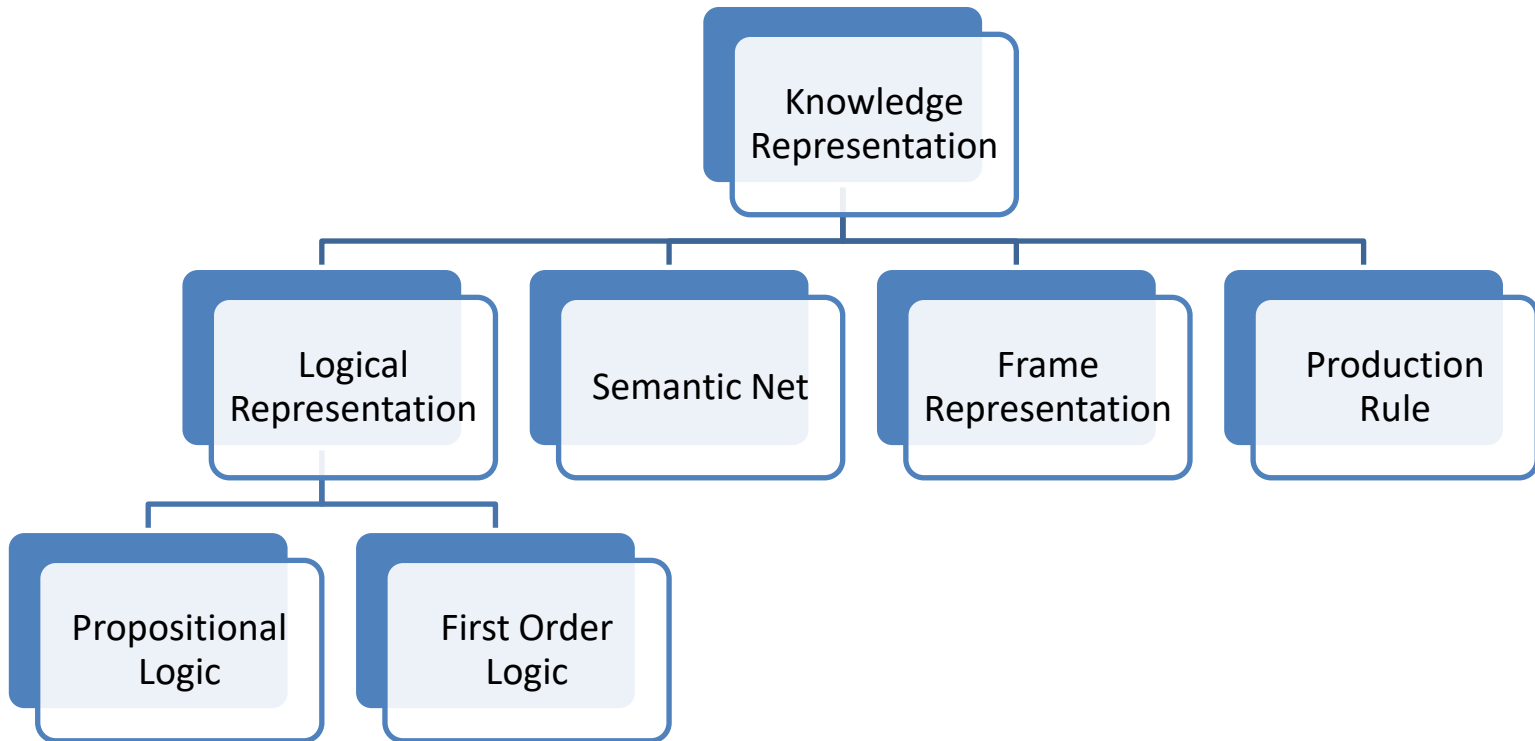
- It is used to store different facts associated with a specific domain
- It helps in building logic
- Contributes in decision making
- Knowledge is of two types
 - Declarative Knowledge
 - My height is 172cm and Vikash is 170 cm
(Statement)
 - Procedural Knowledge
 - I am 2cm taller than Vikash *(Derived and conclusive)*

Structure of Knowledgebase System



- A **knowledge base** (KB) is a technology used to store complex structured and unstructured information used by a computer system. The initial use of the term was in connection with **expert systems** which were the first **knowledge-based** systems.

Knowledge Representation



Propositional Logic (PL)

- Represent sentential form
- It tells either true or false
- Can't represent relation
- Use of PL is less
- PL can't be directly represented in predicated form
- Quantifiable facts (all, some, every) can't be written in PL
- Not suitable for representing single fact

Propositional Logic (PL)

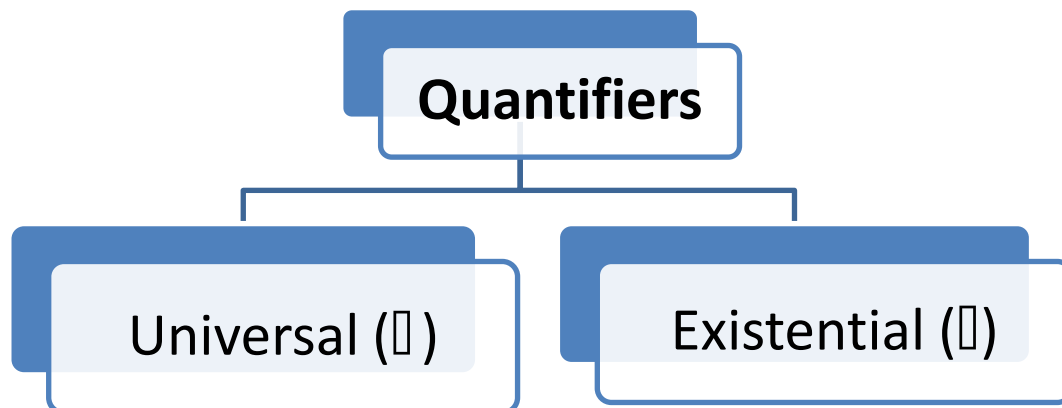
Word	Symbol	Example (Let A and B are two sentences)
not	\neg	$\neg A$ (not A)
and	\wedge	$A \wedge B$ (A and B)
or	\vee	$A \vee B$ (A or B)
implies	\rightarrow	$A \rightarrow B$ (if A then B)
equivalence	\leftrightarrow	$A \leftrightarrow B$ (A if and only if B)

PL - Example

Statement	Sentence
It is hot	A
It is humid	B
It is raining	C
Using the set of above three facts, following statements are represented in PL	
Statement	PL
If it is humid then it is hot	$B \rightarrow A$
If it is hot and humid then not raining	$(A \wedge B) \rightarrow \neg C$

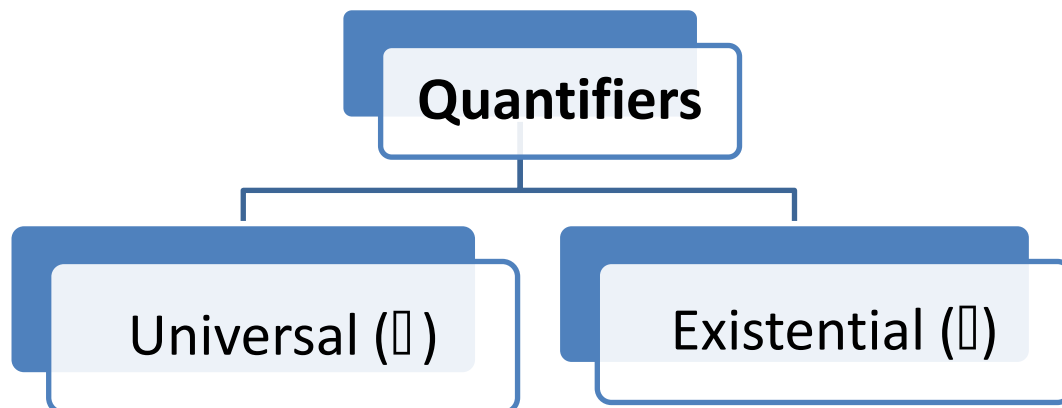
First Order Predicate Logic (FOL/ FOPL)

Gorilla is black	<ul style="list-style-type: none">➤ black(Gorilla)➤ $\text{Gorilla}(x) \rightarrow \text{black}(x)$
John is tall	tall (John)
Boys like cricket	like (Boys, Cricket)
What if Generalization/ Specialization/ Pattern representation	
All boys like cricket	?
Some boys like cricket	?



First Order Predicate Logic (FOL/ FOPL)

Gorilla is black	$\triangleright \text{black}(\text{Gorilla})$ $\triangleright \text{Gorilla}(x) \rightarrow \text{black}(x)$
John is tall	$\text{tall}(\text{John})$
Boys like cricket	$\text{like}(\text{Boys}, \text{Cricket})$
What if Generalization/ Specialization/ Pattern representation	
All boys like cricket	$\forall x: \text{boys}(x) \rightarrow \text{like}(x, \text{Cricket})$
Some boys like cricket	$\exists x: \text{boys}(x) \wedge \text{like}(x, \text{Cricket})$



Difference between PL & FOL

Propositional Logic	First Order Logic
It uses propositions. A complete sentence is denoted by symbol.	FOL uses predicated form. This involves constant, variables, functions, relations.
PL can't represent individual sentence.	FOL can represent individual properties.
<i>e.g. John is tall</i> – Can't be expressed by PL	<i>e.g. John is tall</i> => tall (John)
It can't express generalization, specialization, patterns (e.g. Quantifiers)	It can express generalization, specialization pattern
<i>e.g. Triangles have three sides</i>	<i>e.g. Triangles have three sides</i> no_of_sides(triangle, 3)

Exercise

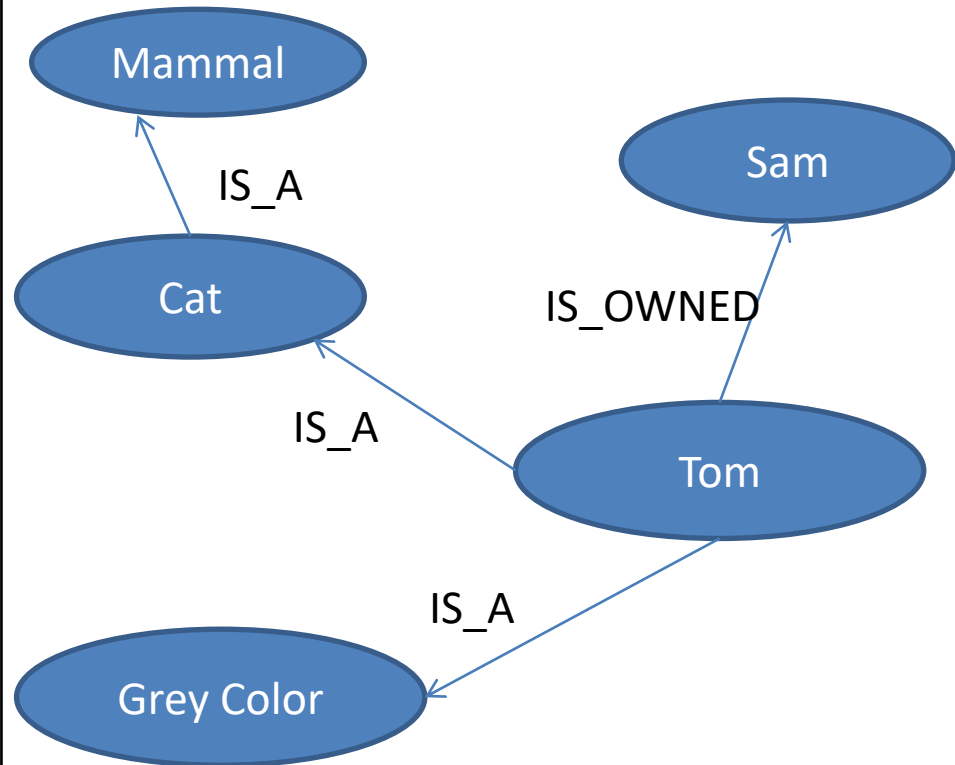
Convert following sentences in FOL

- Ravi likes all kinds of foods
- Apple and chicken are food
- Anything anyone eats and not killed is food
- Ajay eats peanuts and still alive
- Rita eats that Ajay eats

Semantic Net

- Representation of knowledge using graphical diagram/ graphical network

- Tom is cat
- Tom is grey color
- Tom is mammal
- Tom is owned by Sam
- Cat is a mammal



Frame Representation

- Frames are report like structures
- It consists of a collection of slots as attributes and the corresponding slot values
- Slots have **names** and values called **facts**

e.g. [Profession (**value:** *Engineer*)]

 [Age (**value:** 25)]

 [City (**value:** *Kolkata*)]

 [State (**value:** *WB*)]

Production Rule

if *proposition1* **and** *proposition2* **are true then**
proposition3

- It provides a set of rules and action specifiers
- Production rules are mainly used within a **production system**
- Production system also consists a **state space** description
- A State space consist of a starting configuration (Initial state) and ending/ final configuration (goal state) along with all intermediate states

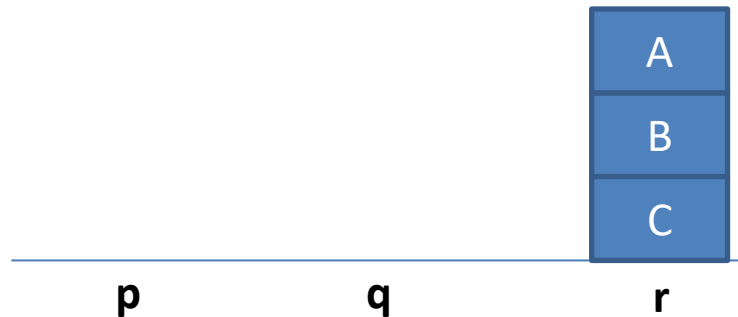
Production System: Example 1

Problem Name: *Shifting of blocks*

Initial State:



Goal State:



Moves: (i) Shift and place a block on another block
(ii) Shift and place a block on empty stack

A		
B		C
P	Q	R

Shift(A, Q)

Shift(A, C)

Shift(C, A)

Shift(C, Q)

B	A	C
P	Q	R

		A
B		C
P	Q	R

C		
A		
B		
P	Q	R

A		
B	C	
P	Q	R

..... Shift(B, C)

		B
	A	C
P	Q	R

Shift(A, B)





		A
		B
		C
P	Q	R

.....

Production System: Example 2

Problem Name: *Wolf, Goat and Cabbage Problem*





Initial State:

	R I V E R	
		
		
		

wgc(Boat Pos, Occupancy left bank, Occupancy right bank)

wgc(left, [w,g,c], [])

Goal State:

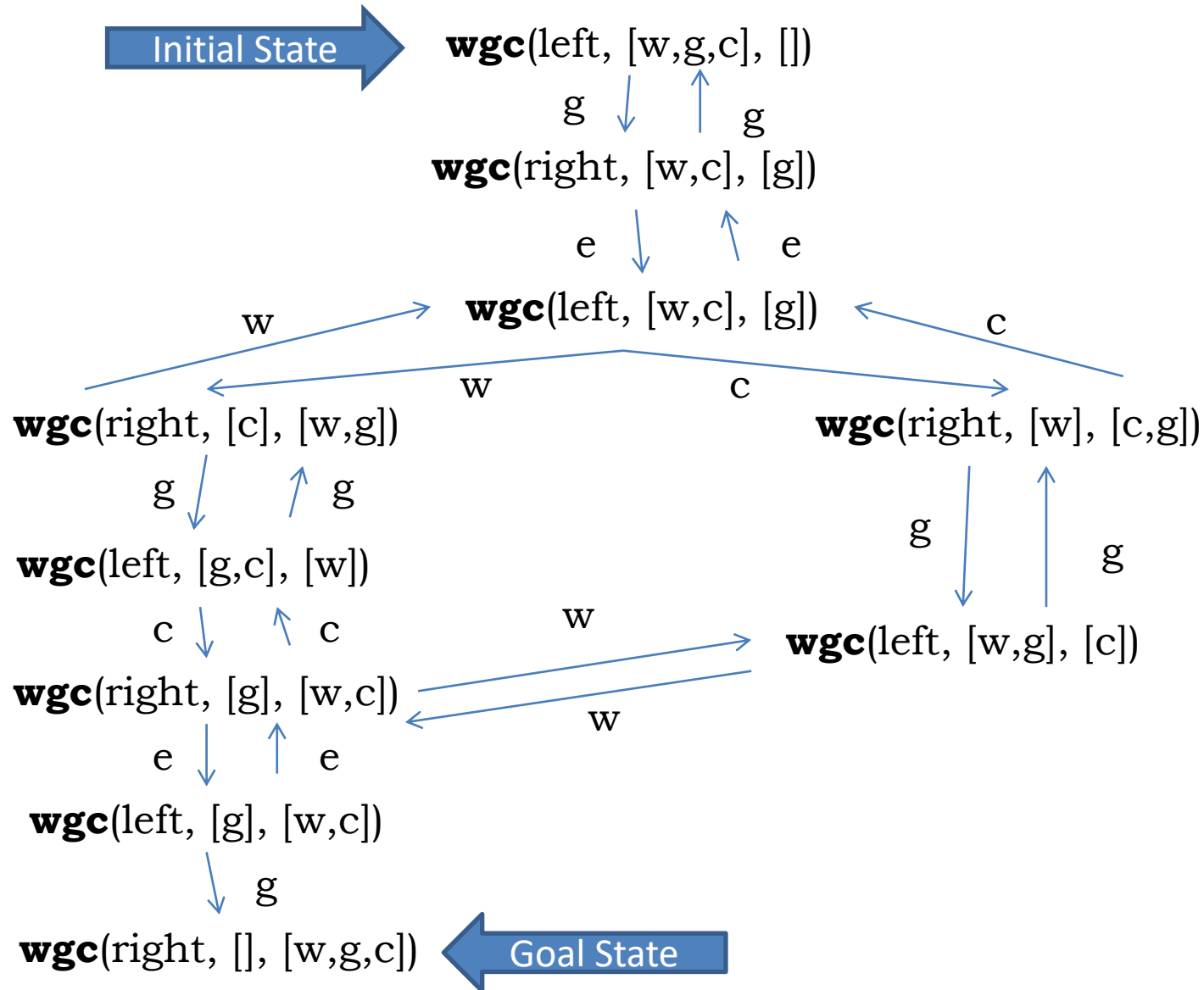
	R I V E R	
		
		
		

wgc(right, [], [w,g,c])

Constraints: (i) Only anyone can be carried by boat at a time
(ii) Can't left **wolf with goat** or **goat with cabbage** alone

Move: Boat can move either empty (e) or with wolf (w), or goat (g) or cabbage (g)

State Space: Example 2

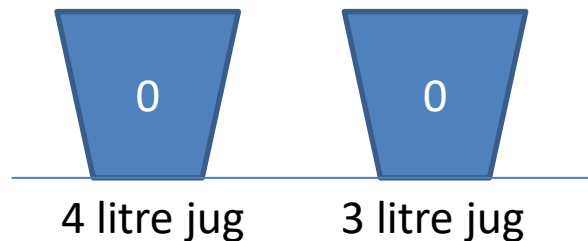


Production System: Assignment

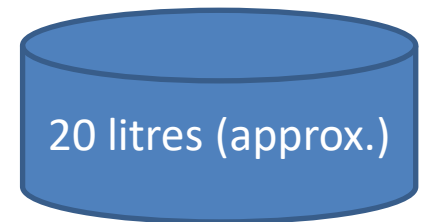
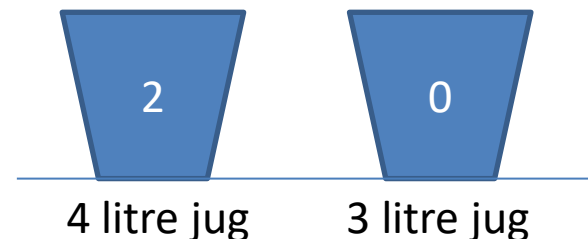
Problem Name: *Water Jug Problem*

Statement: You have one 4 litre jug and one 3 litre jug both with no measuring marks. Also there is a Vat containing approximate 20 litres of water. Fill out 4 litre jug with exactly 2 litres

Initial State:



Goal State:



Production System: Assignment

Possible operations:

1. $\text{fill}(n) \rightarrow$ fill n^{th} jug to its capacity
2. $\text{empty}(n) \rightarrow$ n^{th} jug to vat
3. $\text{transfer}(m, n) \rightarrow$ transferring the content of m jug to n jug until n fills up or m empties out

Production System: Assignment

Possible moves:

- a) Fill 4 litre jug
- b) Fill 3 litre jug
- c) Empty 4 litre jug on vat
- d) Empty 3 litre jug on vat
- e) Transfer 3 litre jug to 4 litre jug up to capacity
- f) Transfer 4 litre jug to 3 litre jug up to capacity

Features of Production System

1. Each rule should be triggered (applied) on proper state so tht resultant state is unambiguous
2. There will be finite number of intermediate states
3. There will be at least one path starting from initial state to goal state
4. Each of the path should consist of finite number of steps which means the path length should be finite

Inference Engine

Inference engine is involved in following tasks

- Query generation in clausal form
- Deduction of new facts from the existing knowledge
- Resolution
- Decision making

Forward Chaining

- ✓ It is process of deriving new facts from the known set of facts
- ✓ If the goal is discussed and deduced from a given set of facts then forward chaining is suitable

Forward Chaining: Example 1

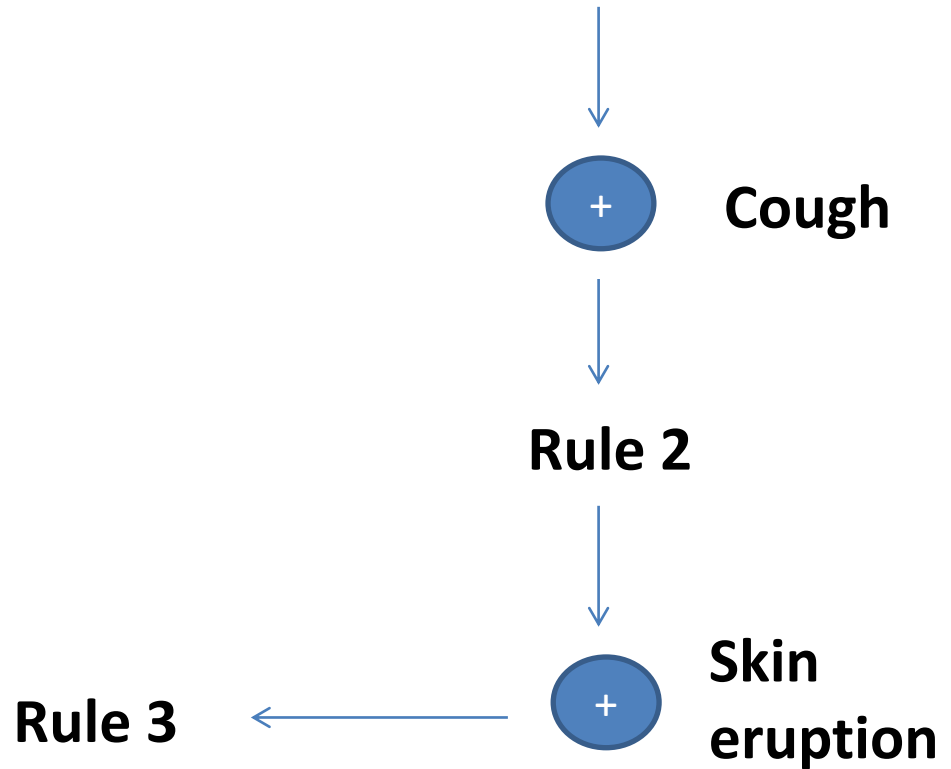
Rule	If	Then
1	Person has headache and temperature	Fever
2	Fever with cough	Viral fever
3	Viral fever with skin eruption	Missiles
4	Fever and suffers more than a week	Typhoid

A person comes with following symptoms:

- Headache and temperature
- Cough
- Skin eruption
- What is the final diagnosis?

Example 1: Continued

Headache and Temperature → Rule 1



Decision: *The person has Missiles*

Forward Chaining: Example 2

Rule	If	Then
1	Parent of father	Grand parent
2	Parent of mother	Grand parent
3	Parent and female	Mother
4	Parent and male	Father
5	Grand parent and female	Grand mother
6	Grand parent and male	Grand father
7	Ofspring of parent	Child
8	Child and male	Son
9	Child and female	Daughter

Forward Chaining: Assignment

- Sashi is male
- Jyoti is male
- Jyoti is parent of Sashi
- Sashi is parent of Varun
 - What is the relation between Jyoti and Varun

Backward Chaining

- ✓ In backward chaining a conclusion is hypothesized and antecedent consequence rules are used to resolve backward.
- ✓ If the purpose is to deny or verify one particular hypothesis then backward chaining is done.

Backward Chaining: Example 1

Rule	If	Then
1	Person has headache and temperature	Fever
2	Fever with cough	Viral fever
3	Viral fever with skin eruption	Missiles
4	Fever and suffers more than a week	Typhoid

A person comes with following symptoms:

- Headache and temperature
- Suffering more than a week
- What is the final diagnosis?

Example 1: Continued

- ✓ **Hypothesis: Typhoid**
- **Typhoid** must trigger **Rule 4**
- **Rule 4** requires (a) Must have fever and (b) Suffers more than a week
- **(a)** requires to trigger **Rule 1**
- **Rule 1** requires (a) Headache and (b) Temperature

Example 1: Continued

- ✓ **Now**, to verify hypothesis ask questions to patient
 - **Q1:** Do you have Headache? [**Ans. Yes**]
 - **Q2:** Do you have temperature? [**Ans. Yes**]
 - **Q3:** Are you suffering more than a week? [**Ans. Yes**]
- ✓ Since all the questions are ratified positively hence the Hypothesis stands correct

Resolution

Rules for resolution:

1. Convert the given facts in FOL
2. Convert FOL in CNF (Conjunctive Normal Form)
3. Negate the statement to be Proved
4. Draw the resolution graph
5. Prove by contradiction

Conversion of FOL to CNF

Rule 1:

If FOL has ' \rightarrow ' or ' \leftrightarrow ' then eliminate by

- i. $[a \rightarrow b] \Rightarrow [\neg a \vee b]$
- ii. $[a \leftrightarrow b] \Rightarrow [a \rightarrow b \wedge b \rightarrow a]$

Rule 2:

- i. $\neg (\forall x P) \Rightarrow \exists x \neg P$
- ii. $\neg (\exists x P) \Rightarrow \forall x \neg P$
- iii. $\neg (a \vee b) \Rightarrow \neg a \wedge \neg b$
- iv. $\neg (a \wedge b) \Rightarrow \neg a \vee \neg b$
- v. $\neg (\neg a) \Rightarrow a$

Conversion of FOL to CNF

Rule 3:

Rename variables, i.e. Different variables for next sentence

Rule 4:

Replace existential quantifiers by Skolem constant

$\exists x \text{ Rich}(x) \Rightarrow \text{Rich}(G1)$

Rule 5:

Drop universal quantifiers

Resolution: Example 1

Sentence:

All lecturers are determined. Anyone who is determined and intelligent will give good service. Mary is an intelligent lecturer. ***Show that*** Mary will give good service

Example 1: Continued

Rules	Resolution
(1) All lecturers are determined	Mary will give good service
(2) Anyone who is determined and intelligent will give good service	
(3) Mary is an intelligent lecturer	

Now convert all these sentences in FOL

Example 1: Continued

(Step 1: Convert all to FOL)

Rules	Resolution
(1) All lecturers are determined	Mary will give good service
(2) Anyone who is determined and intelligent will give good service	
(3) Mary is an intelligent lecturer	

Rules in FOL	Resolution in FOL
[R1] $\forall x: \text{lecturer}(x) \rightarrow \text{determined}(x)$	good_service (Mary)
[R2] $\forall x: \text{determined}(x) \wedge \text{intelligent}(x) \rightarrow \text{good_service}(x)$	
[R3] $\text{intelligent}(\text{Mary}) \wedge \text{lecturer}(\text{Mary})$	

Now convert all these FOL to CNF

Example 1: Continued

(Step 2: FOL to CNF)

Rules in FOL

[R1] $\forall x: \text{lecturer}(x) \rightarrow \text{determined}(x)$

[R2] $\forall x: \text{determined}(x) \wedge \text{intelligent}(x) \rightarrow \text{good_service}(x)$

[R3] $\text{intelligent}(\text{Mary}) \wedge \text{lecturer}(\text{Mary})$

Rules in CNF

[R1] $\neg \text{lecturer}(x) \vee \text{determined}(x)$ {by Conversion Rule 5 and 1. (i)}

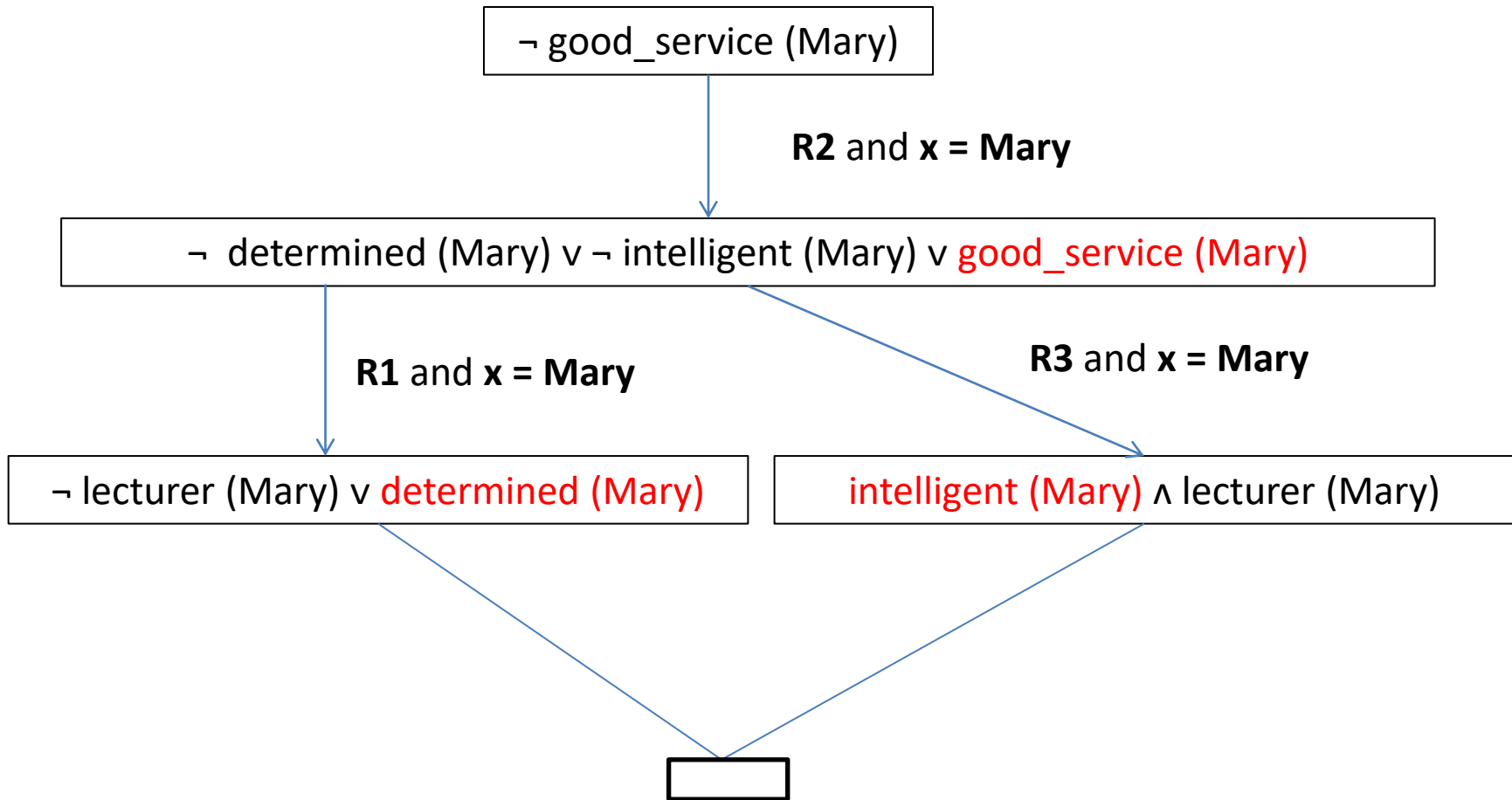
[R2] $\neg [\text{determined}(x) \wedge \text{intelligent}(x)] \vee \text{good_service}(x)$ {by Conversion Rule 5 and 1. (i)}

Then, $\neg \text{determined}(x) \vee \neg \text{intelligent}(x) \vee \text{good_service}(x)$ {by Conversion Rule 2. (iv)}

[R3] $\text{intelligent}(\text{Mary}) \wedge \text{lecturer}(\text{Mary})$ {Converted form is same}

Example 1: Continued

(Step 3, 4 & 5: Resolution Graph)



Proved by Contradiction as two left over clauses are contradicting to each other

Resolution: Assignment

Sentence:

Ravi likes all kinds of food. Apple and chicken are food. Anything anyone eats and not killed is food. Ajay eats peanut and still alive. ***Show that*** Ravi likes peanut.

Answer of Assignment: Page 1

(Step 1: Convert all to FOL)

Rules	Resolution
(1) Ravi likes all kinds of food	Ravi likes peanut
(2) Apple and chicken are food	
(3) Anything anyone eats and not killed are food	
(4) Ajay eats peanut and still alive	

Rules in FOL	Resolution in FOL
[R1] $\forall x: \text{food}(x) \rightarrow \text{likes}(\text{Ravi}, x)$	likes (Ravi, peanut)
[R2] $\text{food}(\text{Apple}) \wedge \text{food}(\text{chicken})$	
[R3] $\forall x, \forall y: \text{eats}(y, x) \wedge \neg \text{killed}(y) \rightarrow \text{food}(x)$	
[R4] $\text{eats}(\text{Ajay}, \text{peanut}) \wedge \text{alive}(\text{Ajay})$	
[R5] $\forall x: \text{alive}(y) \rightarrow \neg \text{killed}(y)$ [derived rule]	

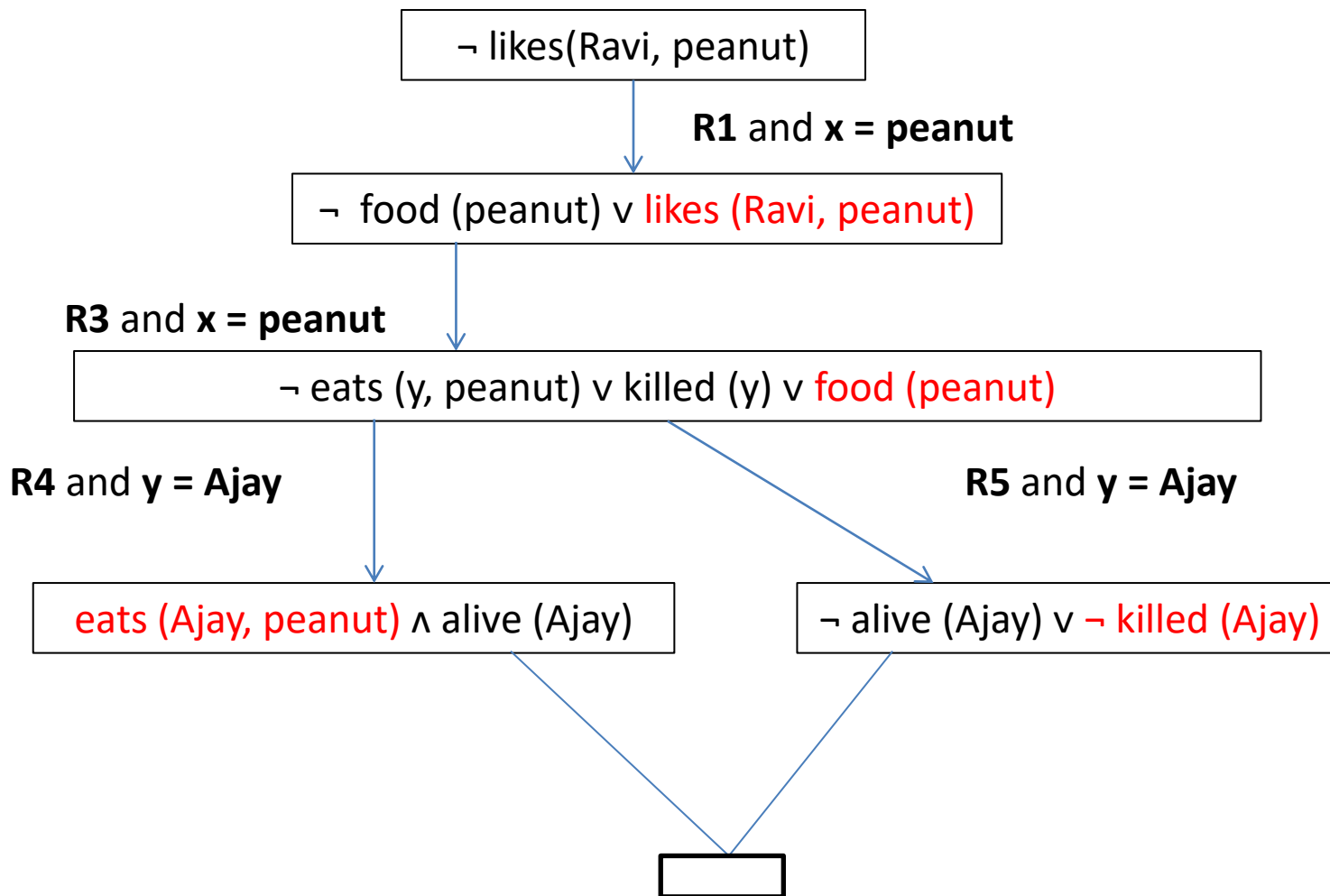
Answer of Assignment: Page 2

(Step 2: Convert FOL to CNF)

Rules in FOL	Resolution in FOL
[R1] $\exists x: \text{food}(x) \rightarrow \text{likes}(\text{Ravi}, x)$	likes (Ravi, peanut)
[R2] $\text{food}(\text{Apple}) \wedge \text{food}(\text{chicken})$	
[R3] $\exists x, \exists y: \text{eats}(y, x) \wedge \neg \text{killed}(y) \rightarrow \text{food}(x)$	
[R4] $\text{eats}(\text{Ajay}, \text{peanut}) \wedge \text{alive}(\text{Ajay})$	
[R5] $\exists y: \text{alive}(y) \rightarrow \neg \text{killed}(y)$ [derived rule]	
Rules in CNF	Resolution in CNF
[R1] $\neg \text{food}(x) \vee \text{likes}(\text{Ravi}, x)$	likes (Ravi, peanut)
[R2] $\text{food}(\text{Apple}) \wedge \text{food}(\text{chicken})$	
[R3] $\neg \text{eats}(y, x) \vee \text{killed}(y) \vee \text{food}(x)$	
[R4] $\text{eats}(\text{Ajay}, \text{peanut}) \wedge \text{alive}(\text{Ajay})$	
[R5] $\neg \text{alive}(y) \vee \neg \text{killed}(y)$	

Answer to Assignment: Page 3

(Step 3, 4 & 5: Resolution Graph)



Proved by Contradiction as two left over clauses are contradicting to each other

Searching Techniques

Problem as a State Space Search

- To build a system to solve a particular problem, we need:
 - **Define the problem**: must include precise specifications ~ initial solution & final solution.
 - **Analyze the problem**: select the most important features that can have an immense impact.
 - **Isolate and represent** : convert these important features into knowledge representation.
 - **Problem solving technique(s)**: choose the best technique and apply it to particular problem.

Impact Factors

- Typical questions that need to be answered:
 - Is the problem solver **guaranteed** to find a solution?
 - Will the system always terminate or caught in a infinite loop?
 - If the solution is found, it is **optimal**?
 - What is the **complexity** of searching process?
 - How the system be able to reduce searching complexity?
 - How it can effectively utilize the representation paradigm?

Important Terms

- **Search space** → possible conditions and solutions.
- **Initial state** → state where the searching process started.
- **Goal state** → the ultimate aim of searching process.
- **Problem space** → “what to solve”
- **Searching strategy** → strategy for controlling the search.
- **Search tree** → tree representation of search space, showing possible solutions from initial state.

Searching Strategies

- **Blind search** → traversing the search space until the goal node is found (might be doing exhaustive search).

- *Techniques* : **Breadth First Uniform Cost , Depth first, Interactive Deepening search.**

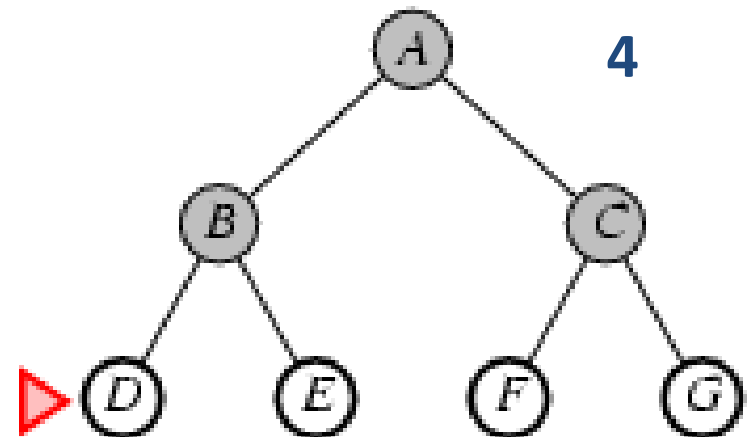
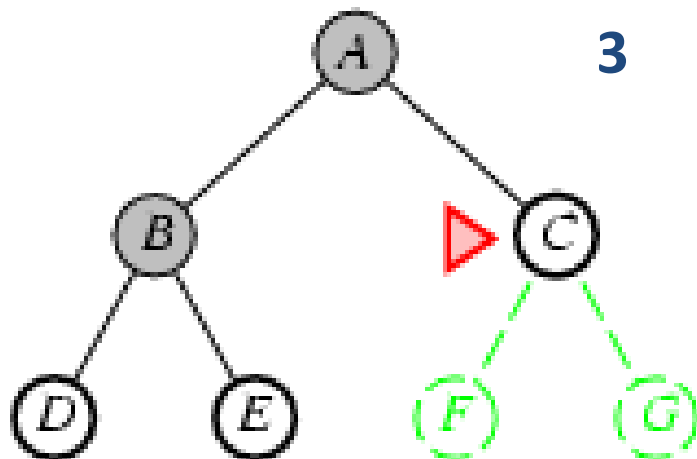
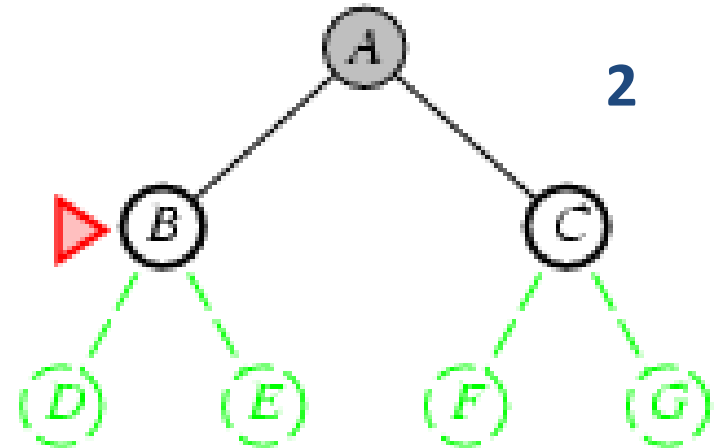
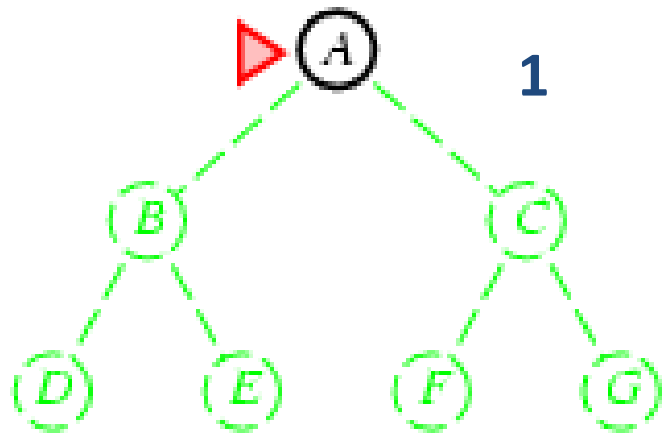
- Guarantees solution.

- **Heuristic search** → search process takes place by traversing search space with applied rules (information).

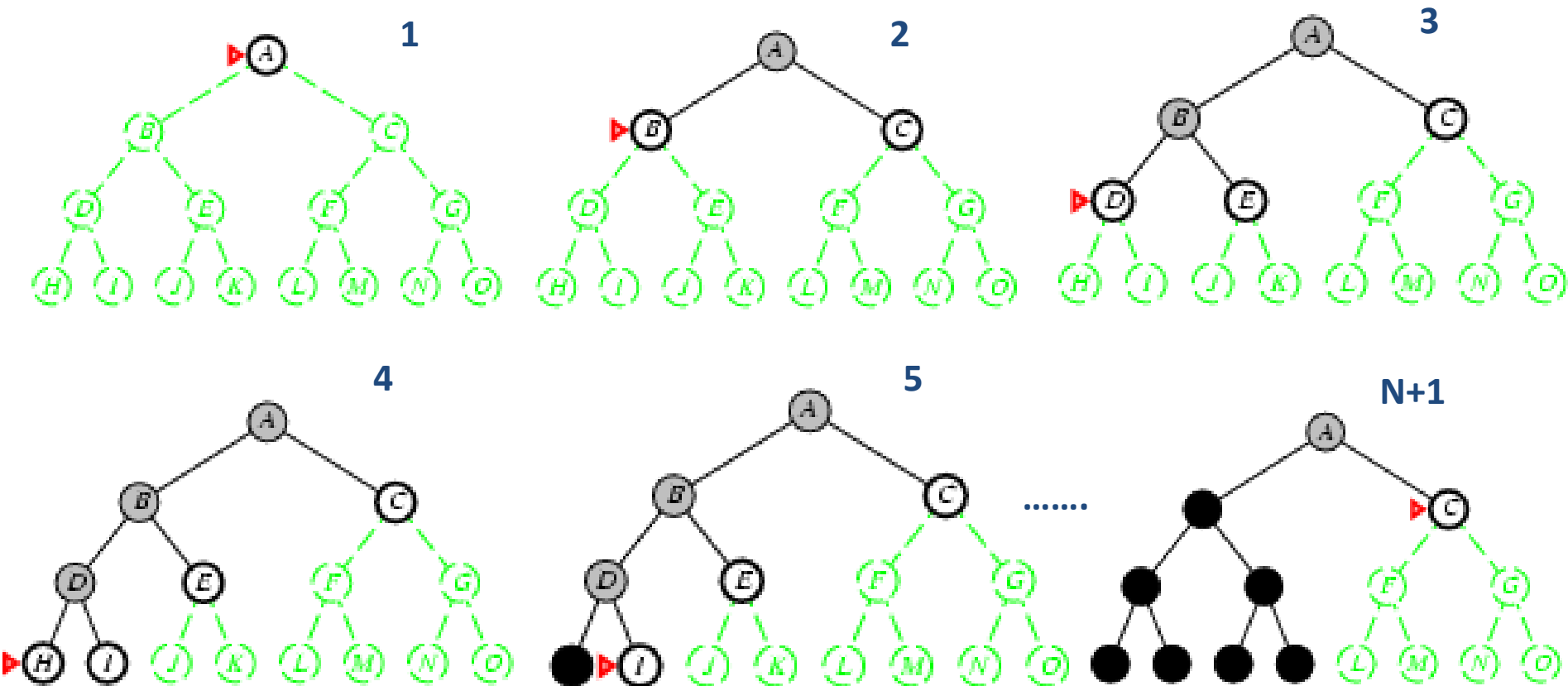
- *Techniques*: **Greedy Best First Search, A* Algorithm**

- There is no guarantee that solution is found.

Blind Search : Breadth First Search



Blind Search : Depth First Search (DFS)

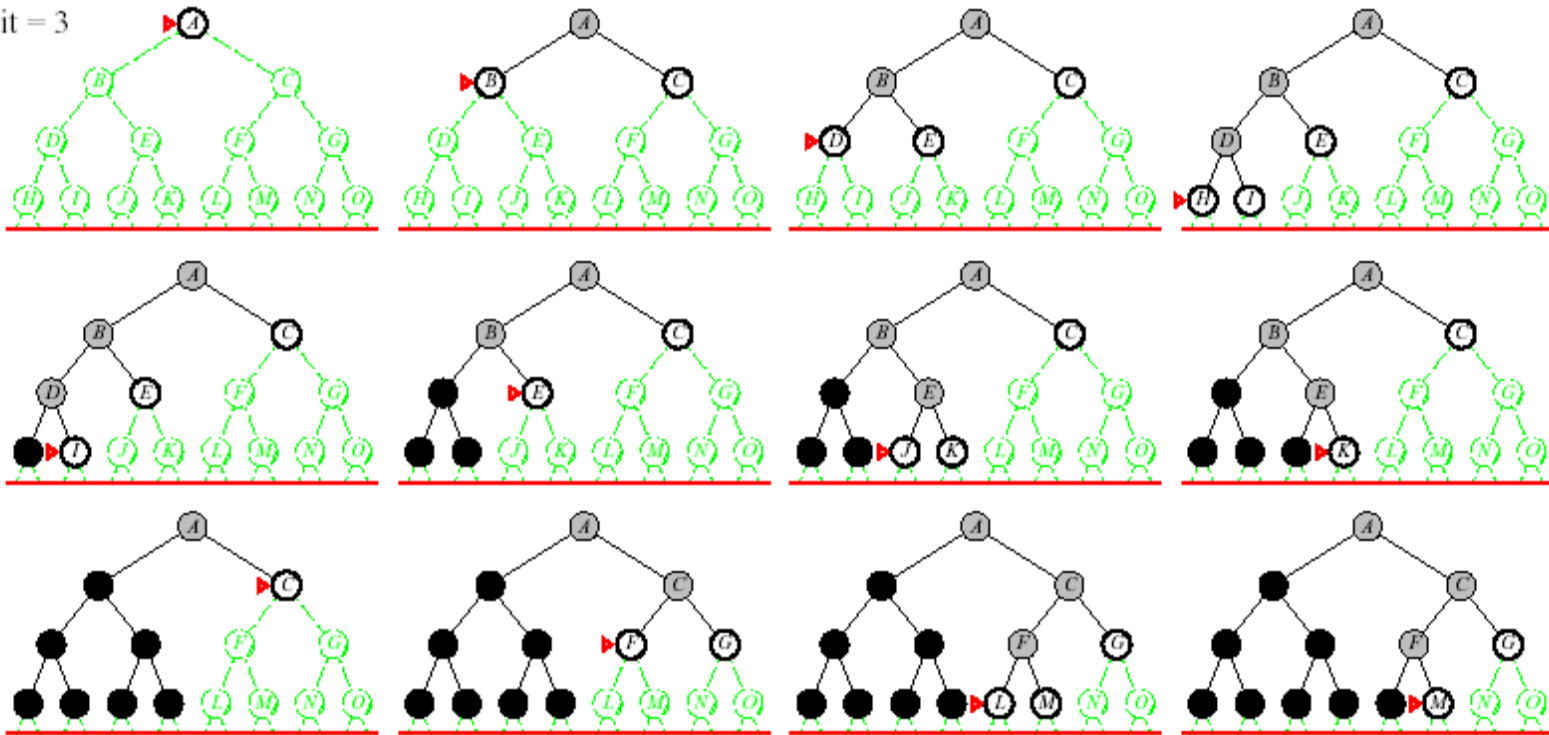


Blind Search : Iterative Deepening DFS (ID-DFS)

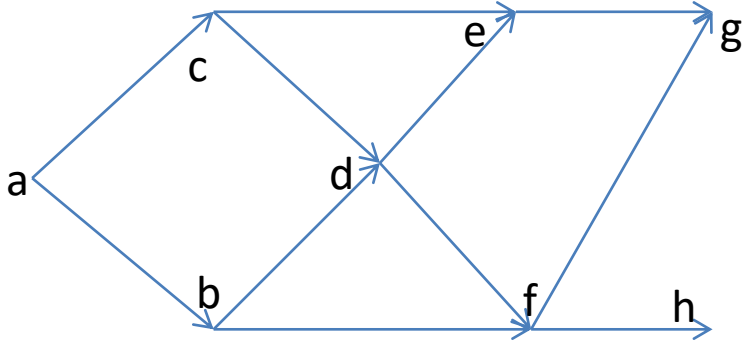
Limit = 1



Limit = 3



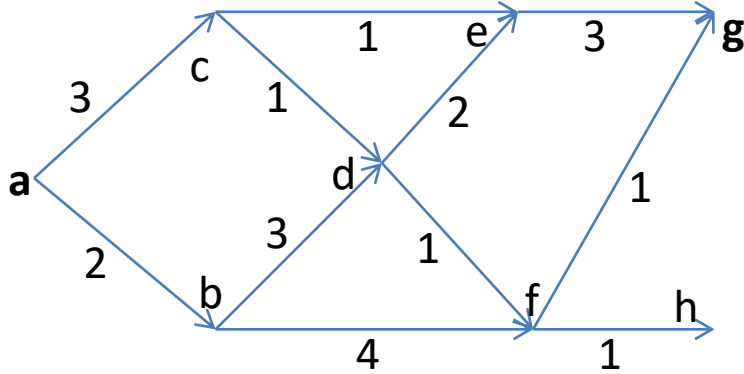
BFS and DFS - Example



BFS	DFS
Queue	Stack
Start State: <i>a</i>	
Goal State: <i>g</i>	

BFS		DFS	
OPEN	CLOSED	OPEN	CLOSED
a		a	
b, c	a	b, c	a
c, d, f	a, b	d, f, c	a, b
d, f, e	a, b, c	e, f, c	a, b, d
f, e	a, b, c, d	g, f, c	a, b, d, e
e, g, h	a, b, c, d, f		a, b, d, e, g
g, h	a, b, c, d, f, e		
	a, b, c, d, f, e, g		
Trace: <i>a, b, c, d, f, e, g</i>		Trace: <i>a, b, d, e, g</i>	

Uniform Cost Search- Example



- $g(n)$ is currently known best path from a to n
- $g(n') = g(n) + c(n, n')$, where n' is predecessor of n
- **e.g.** $g(b) = g(a) + c(a, b) = 0 + 2 = 2$

Instant	a	b	c	d	e	f	g	h
0	0							
1	0	2	3					
2		2	3	5		6		
3			3	4	4	6		
4				4	4	6	7	
5				4		5	7	
6						5	6	6
7							6	

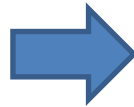
Path: $a \rightarrow c \rightarrow d \rightarrow f \rightarrow g$

Heuristic Search

➤ Evaluation function or Heuristic function

2	8	3
6		4
1	7	5

Present state



1	2	3
8		4
7	6	5

Goal state

Total shifts required = $2 + 1 + 2 + 1 + 2 = 8$
~ *hint of a heuristic estimate*

Heuristic Search

➤ Some more examples

1	2	3
4	5	6
7	8	

Goal state

1	2	3
4	5	6
7		8

$h(\text{state}) = 1$

3	2	8
4	5	6
7	1	

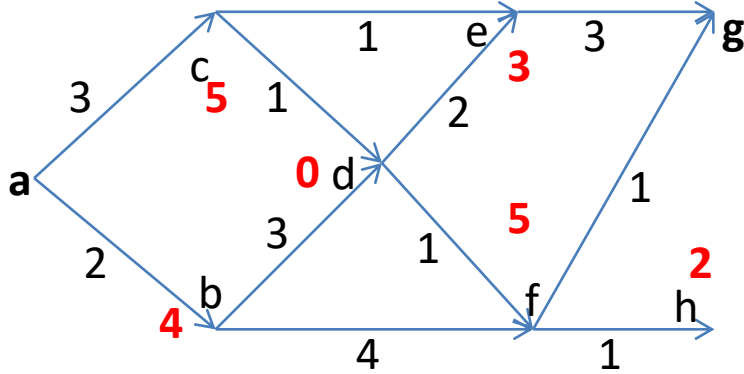
$h(\text{state}) = 8$

Measured on the basis of *Manhattan distance*

Heuristic Search

- Tries to expand the node that is closest to the goal.
- Evaluates using heuristic function
- Possibly lead to the solution very fast.
- Problem ? ~ can end up in sub-optimal solutions (doesn't take notice of the distance it travels).
- Global Maxima and Local Maxima
- Hill climbing
- Local Maxima Challenges: Plateau, Foothill and Ridge Problem

A* Search- Example



• $g(n)$ is currently known best path from a to n

• $h(n)$ = Heuristic value associated with each node

$$f(n) = g(n) + h(n)$$

Instant	a	b	c	d	e	f	g	h
0	0 + 0							
1	0 + 0	2 + 4	3 + 5					
2		2 + 4	3 + 5	5 + 0		6 + 5		
3			3 + 5	5 + 0	7 + 3	6 + 5		
4			3 + 5	4 + 0	4 + 3	6 + 5		
5				4 + 0	4 + 3	5 + 5		
6					4 + 3	5 + 5	7 + 0	
7							7 + 0	

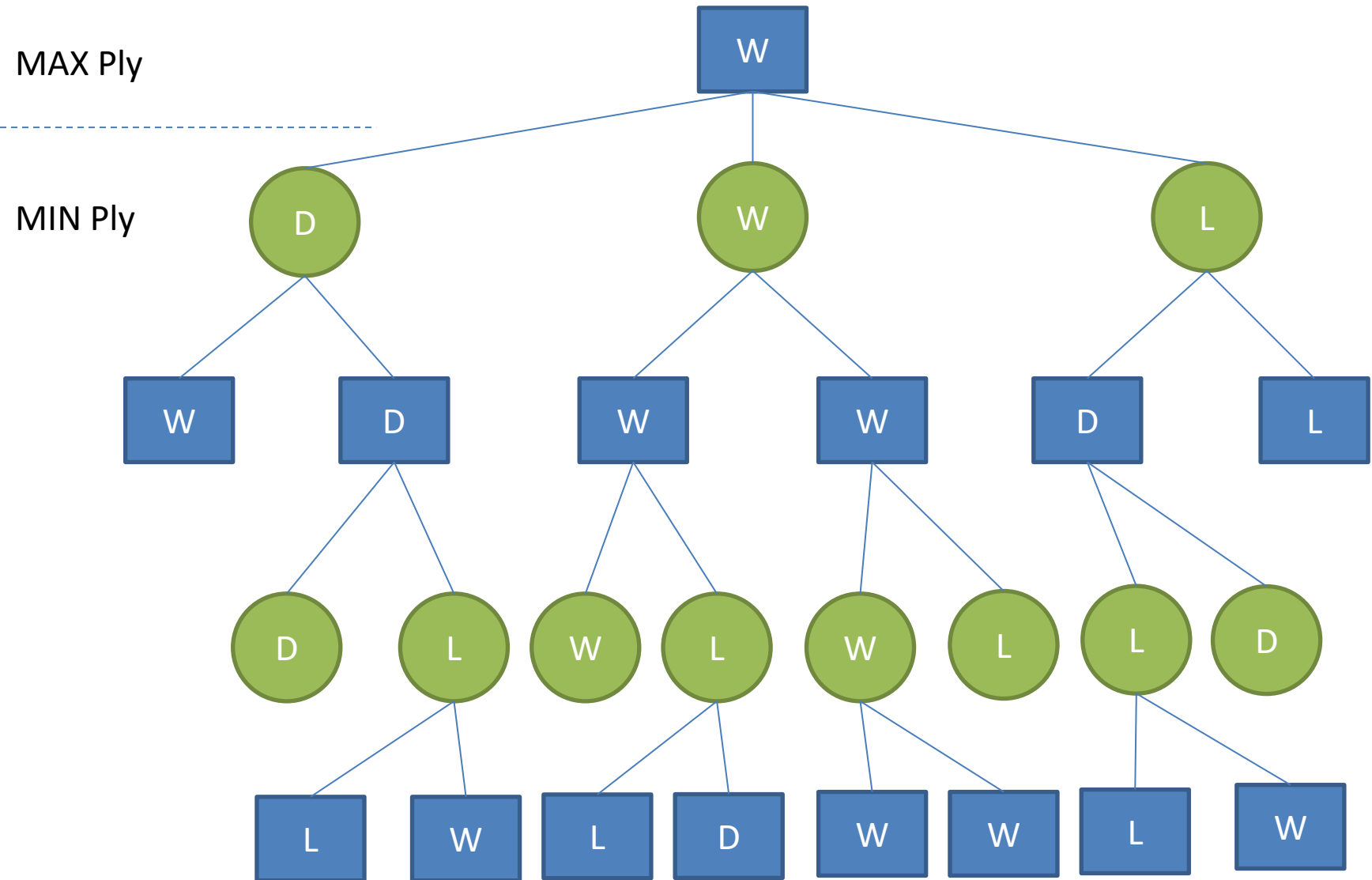
Path: **a → c → e → g**

Game Playing

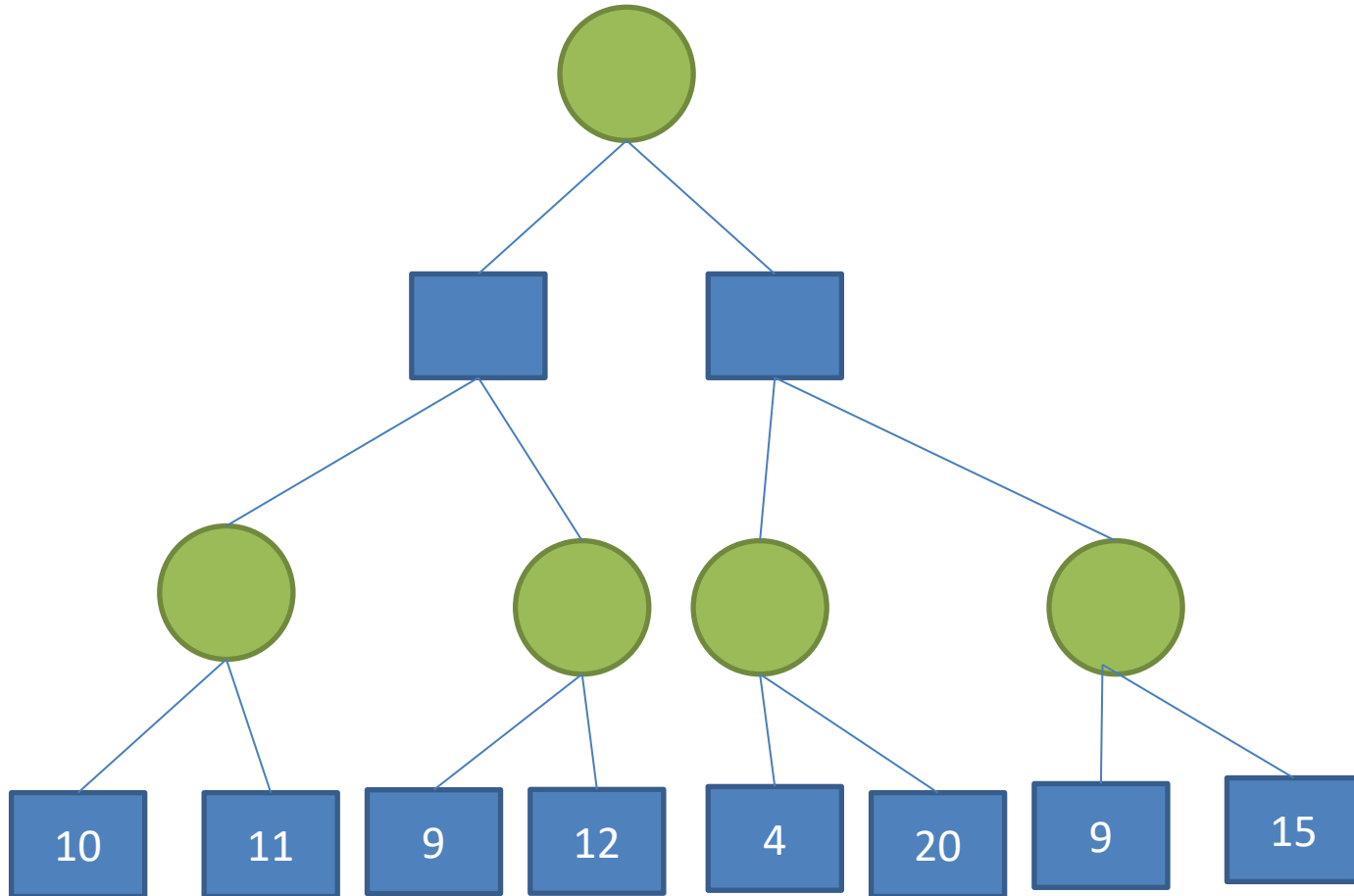
Strategic Game

- A game is expanded through a game tree
- A game tree is AND-OR tree, where OR nodes are represented by Square and AND nodes by Circle
- AND node is considered as minimizing node
- OR node is considered as maximizing node
- Every node is looking for a winning strategy.
- AND and OR nodes appear at alternative levels

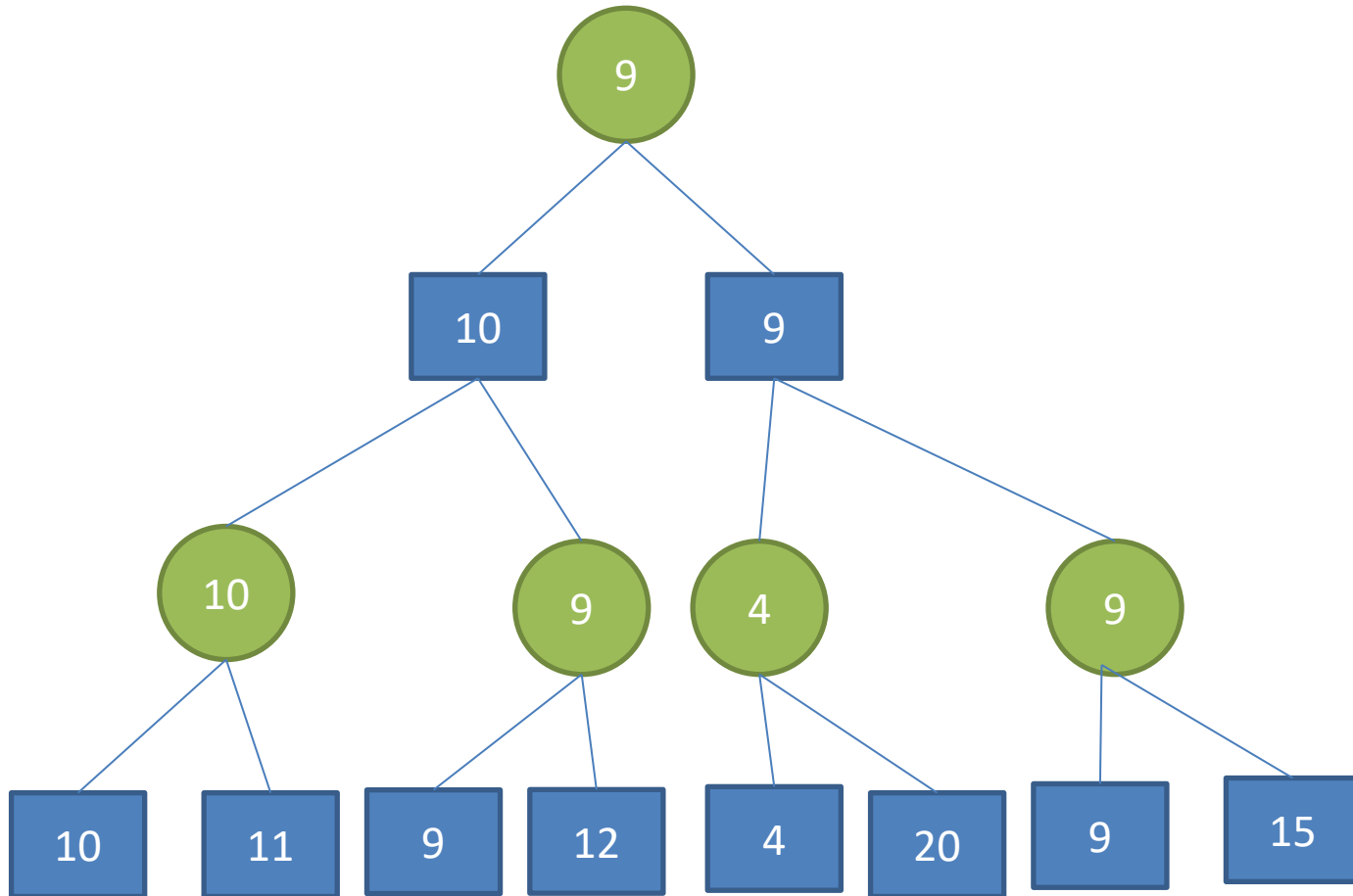
Strategic Game



MINIMAX Algorithm

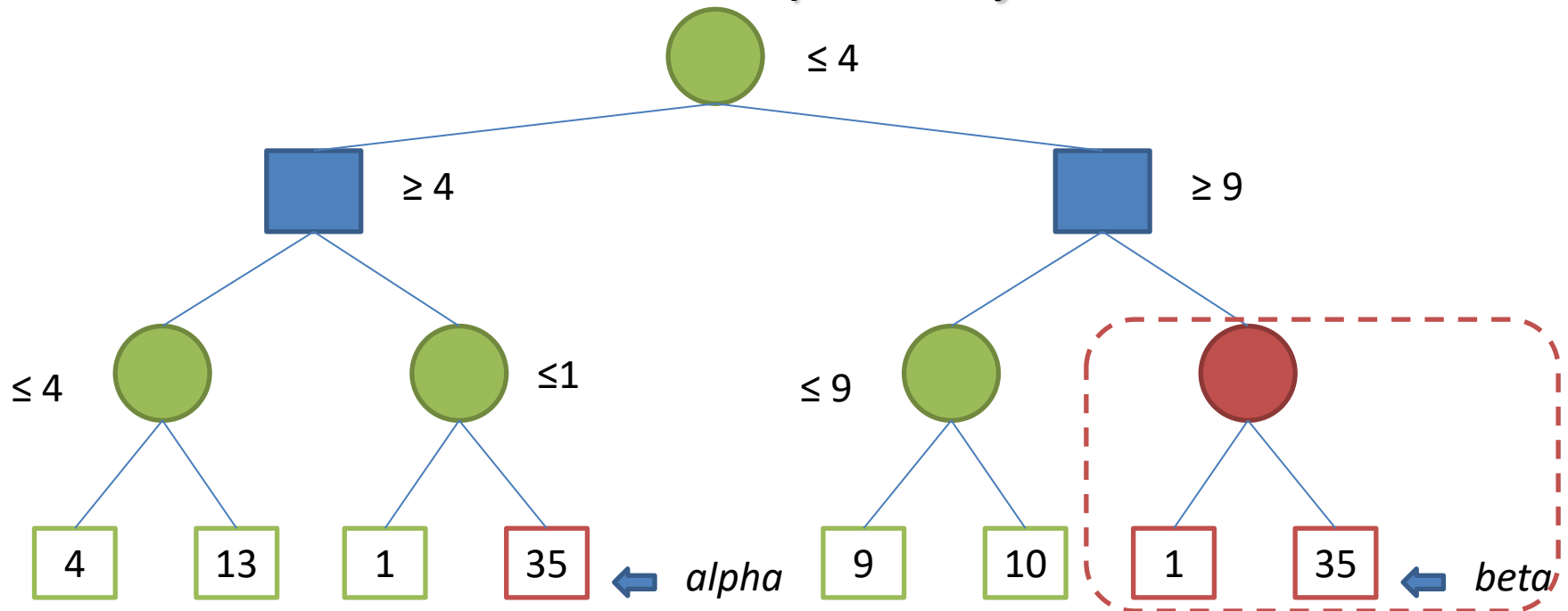


MINIMAX Algorithm

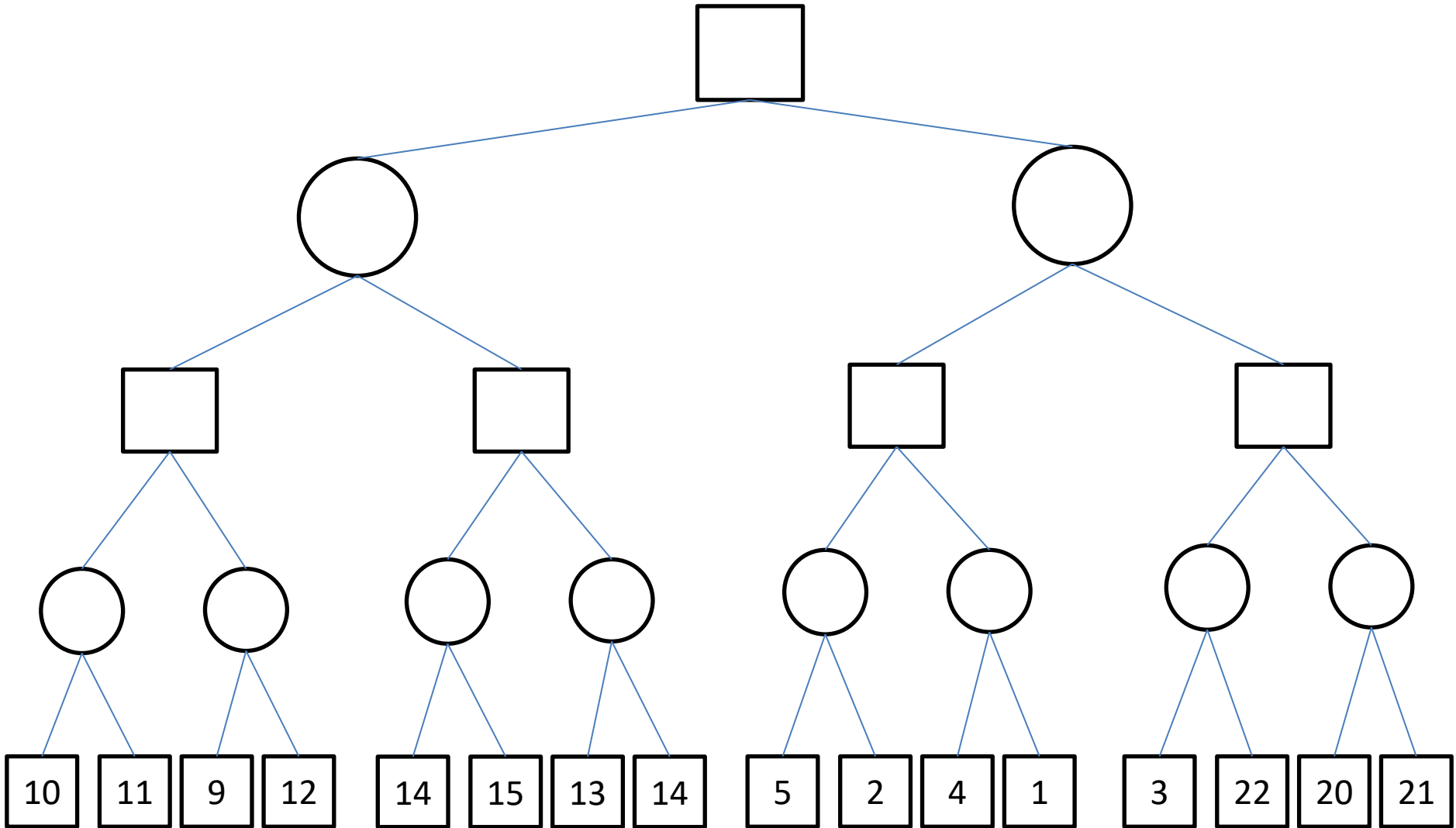


Alpha – Beta Cutoff

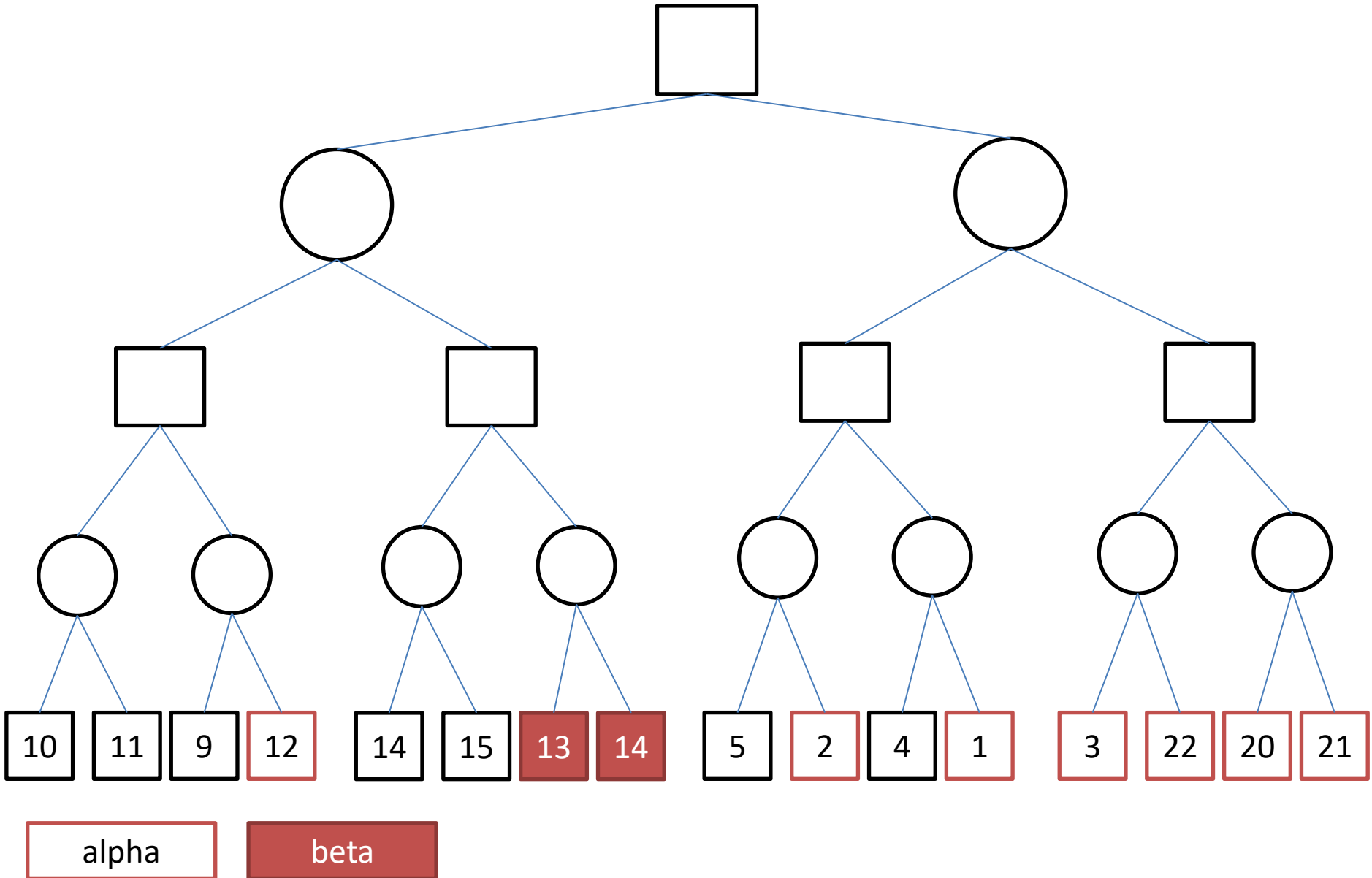
- Minimizing node puts upper threshold
- Maximizing node puts lower threshold
- Max node threshold basis comparison yields alpha cut off
- Min node threshold basis comparison yields beta cut off



Alpha - Beta Cutoff



Alpha - Beta Cutoff



Strategic Game: NIM

➤ 5 sticks will be given in a two players game where any of the player at its turn can pickup 1, 2 or 3 sticks at a time. The player who will pickup the last stick will loose the game.

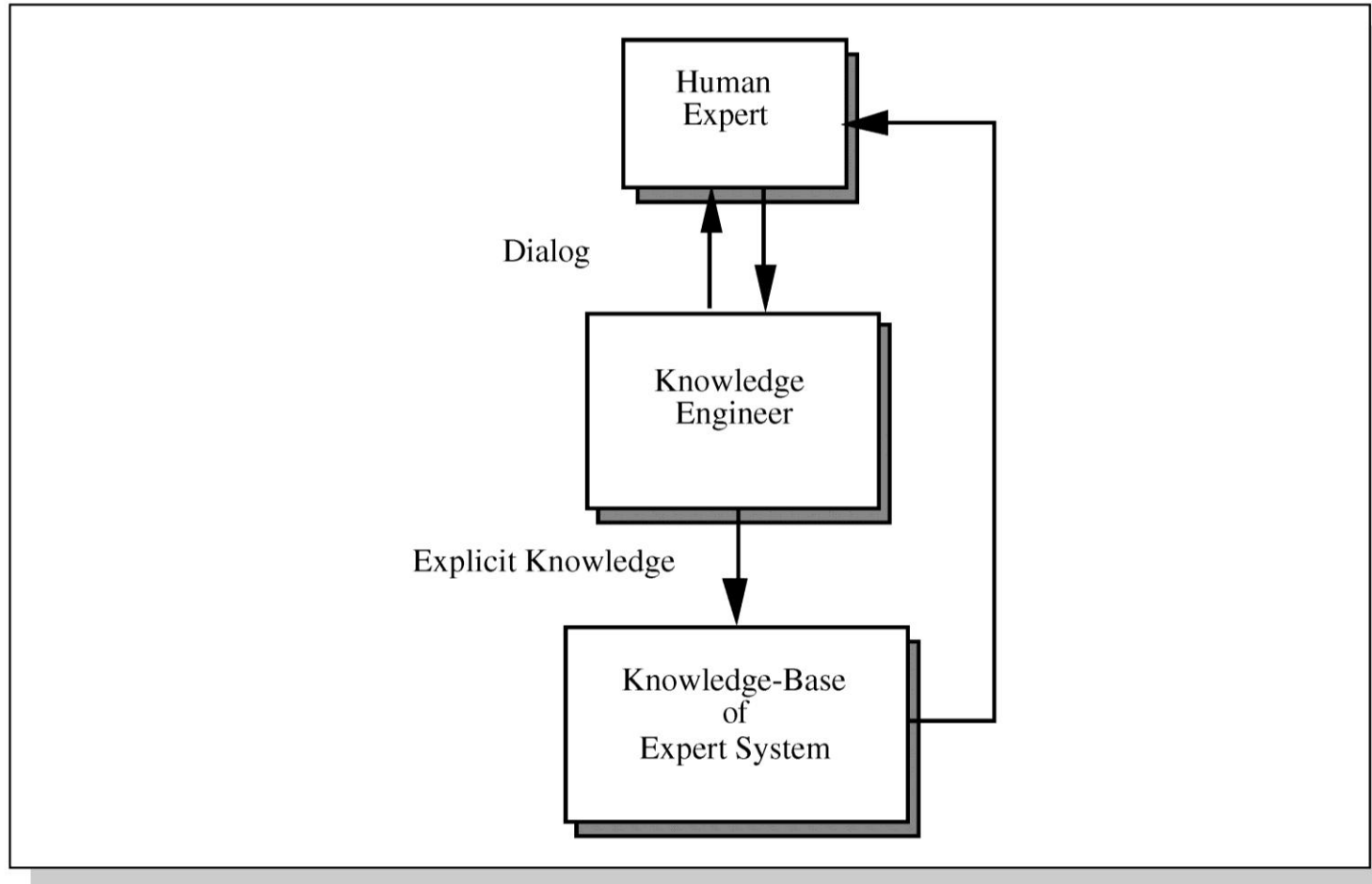
Expand the game tree

Expert System

What is an Expert System?

- A system that uses human expertise to make complicated decisions.
- Simulates reasoning by applying knowledge and interfaces.
- Uses expert's knowledge as rules and data within the system.
- Models the problem solving ability of a human expert.

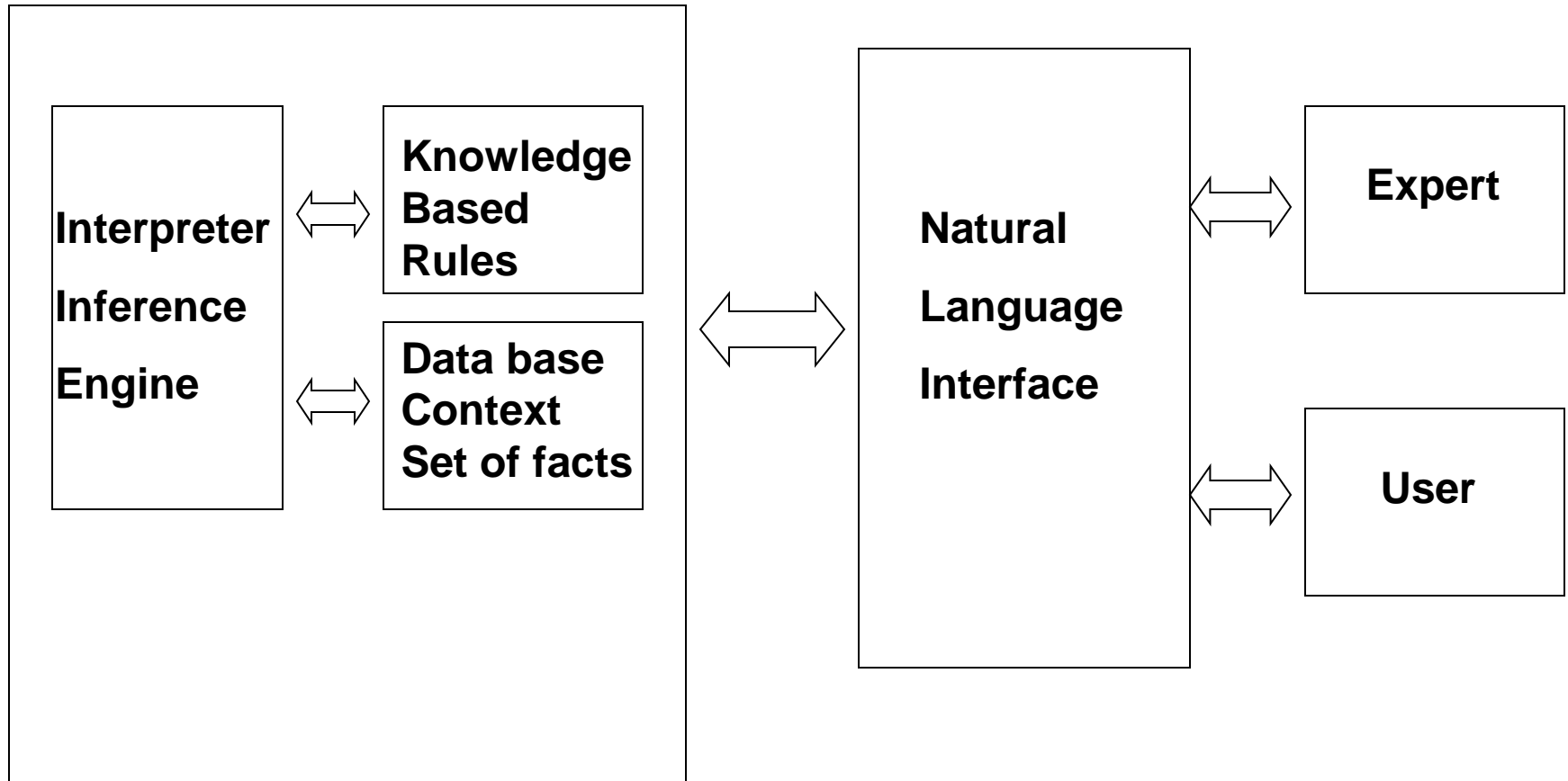
Development of an Expert System



Components of an ES

1. Knowledge Base
2. Reasoning or Inference Engine
3. User Interface
4. Explanation Facility

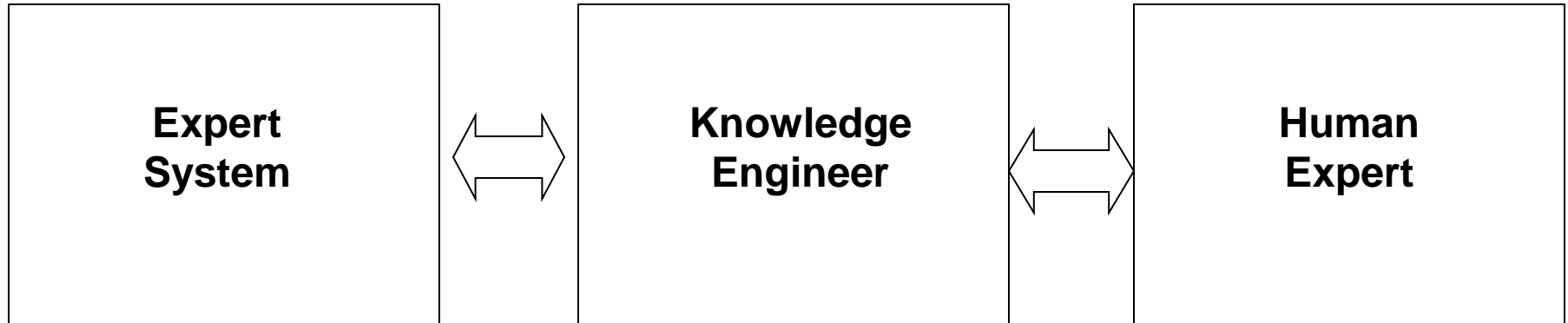
Expert System Structure



Knowledge Base

- Represents all the data and information imputed by experts in the field.
- Stores the data as a set of rules that the system must follow to make decisions.

Knowledge Acquisition



Inference Engine

- Asks questions to the user about what they are looking for.
- Applies the knowledge and the rules held in the knowledge base.
- Appropriately uses this information to arrive at a decision.

User Interface

- Allows the expert system and the user to communicate.
- Finds out what it is that the system needs to answer.
- Sends the user questions or answers and receives their response.

Explanation Facility

- Explains the systems reasoning and justifies its conclusions.

***Does anyone know any examples
of expert systems used in
everyday life?***

Medical Diagnosis & Expert Systems

- Pathology is the study of the origin, nature, and course of diseases.
- Pathology reports explain the outcomes of tests on the patient and various information on the diseases.
- Pathologists were in need of a comprehensive interpretative service to help them interpret these reports.

PEIRS

- PEIRS (Pathology Expert Interpretative Reporting System) was created as a user-maintained expert system for automating the interpretation of chemical pathology reports.
- The system uses over 2500 rules to generate interpretive comments on the reports.

Ethical Implications

- Do you want to put a decision about your life in the hands of a computer?
- When do you think you'd trust a computer's judgment over a doctors?

Turbo Tax

- Expert System designed to help users to complete tax returns

<http://youtube.com/watch?v=5b7KhvQSNcY>



Early Expert Systems

- DENDRAL – used in chemical mass spectroscopy to identify chemical constituents
- MYCIN – medical diagnosis of illness
- DIPMETER – geological data analysis for oil
- PROSPECTOR – geological data analysis for minerals
- XCON/R1 – configuring computer systems

Problems with Expert Systems

- There is no expert on the field
- The expert is unable to communicate his/her ideas
- Sometimes the expert is unwilling to communicate his/her ideas
- Must have all information on a subject
- Can all the testing be accomplished?
- User acceptance

What is a Decision Support System?

- An information system that utilizes analytical modeling and helps executives to make strategic decisions
- DSS is user-driven and rely on the knowledge possessed by its user

DSS vs. Expert Systems

- DSS uses analytical modeling for decision making
- Expert Systems are interactive and make decisions for you

Advantages of Expert Systems

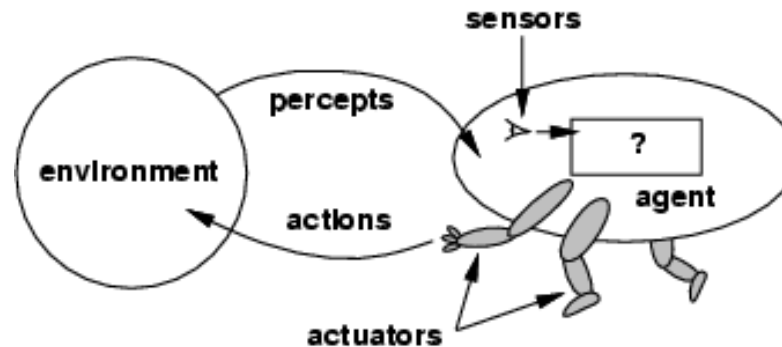
- Can be simple to use
- Efficient results
- Accurate results
- Adaptation and adjustments to changing conditions
- Cost effective

Intelligent Agent

Agents

- An **agent** is anything that can be viewed as **perceiving** its **environment** through **sensors** and **acting** upon that environment through **actuators**
- Human agent: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators
- Robotic agent: cameras and infrared range finders for sensors;
- Various motors for actuators

Agents and environments

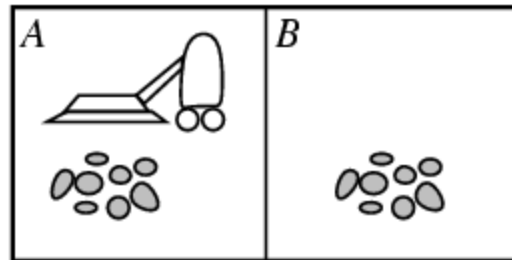


- The **agent function** maps from percept histories to actions:

$$[f: P^* \rightarrow \mathcal{A}]$$

- The **agent program** runs on the physical **architecture** to produce agent actions
 - agent = architecture + program

Vacuum-cleaner world



- Percepts: location and contents, e.g., [A,Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

Rational agents

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful
- Performance measure: An objective criterion for success of an agent's behavior
e.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

Rational agents

- **Rational Agent:** In each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rational agents

- Rationality is distinct from omniscience (all-knowing with infinite knowledge)
- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration)
- An agent is **autonomous** if its behavior is determined by its own experience (with ability to learn and adapt)

PEAS

- PEAS: Performance measure, Environment, Actuators, Sensors
- Must first specify the setting for intelligent agent design
- Consider, the task of designing an automated taxi driver:
 - Performance measure
 - Environment
 - Actuators
 - Sensors

PEAS

- Setting for intelligent agent design
 - ***Performance measure:*** Safe, fast, legal, comfortable trip, maximize profits
 - ***Environment:*** Roads, other traffic, pedestrians, customers
 - ***Actuators:*** Steering wheel, accelerator, brake, signal, horn
 - ***Sensors:*** Cameras, speedometer, GPS, odometer, engine sensors, keyboard

PEAS

- **Agent:** Medical diagnosis system
- **Performance measure:** Healthy patient, minimize costs, lawsuits
- **Environment:** Patient, hospital, staff
- **Actuators:** Screen display (questions, tests, diagnoses, treatments, referrals)
- **Sensors:** Keyboard (entry of symptoms, findings, patient's answers)

PEAS

- **Agent:** Part-picking robot
- **Performance measure:** Percentage of parts in correct bins
- **Environment:** Conveyor belt with parts, bins
- **Actuators:** Jointed arm and hand
- **Sensors:** Camera, joint angle sensors

PEAS

- **Agent:** Interactive English tutor
- **Performance measure:** Maximize student's score on test
- **Environment:** Set of students
- **Actuators:** Screen display (exercises, suggestions, corrections)
- **Sensors:** Keyboard

Environment types

- **Fully observable** (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- **Deterministic** (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is **strategic**)
- **Episodic** (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

Environment types

- **Static** (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is **semi-dynamic** if the environment itself does not change with the passage of time but the agent's performance score does)
- **Discrete** (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- **Single agent** (vs. multi-agent): An agent operating by itself in an environment.

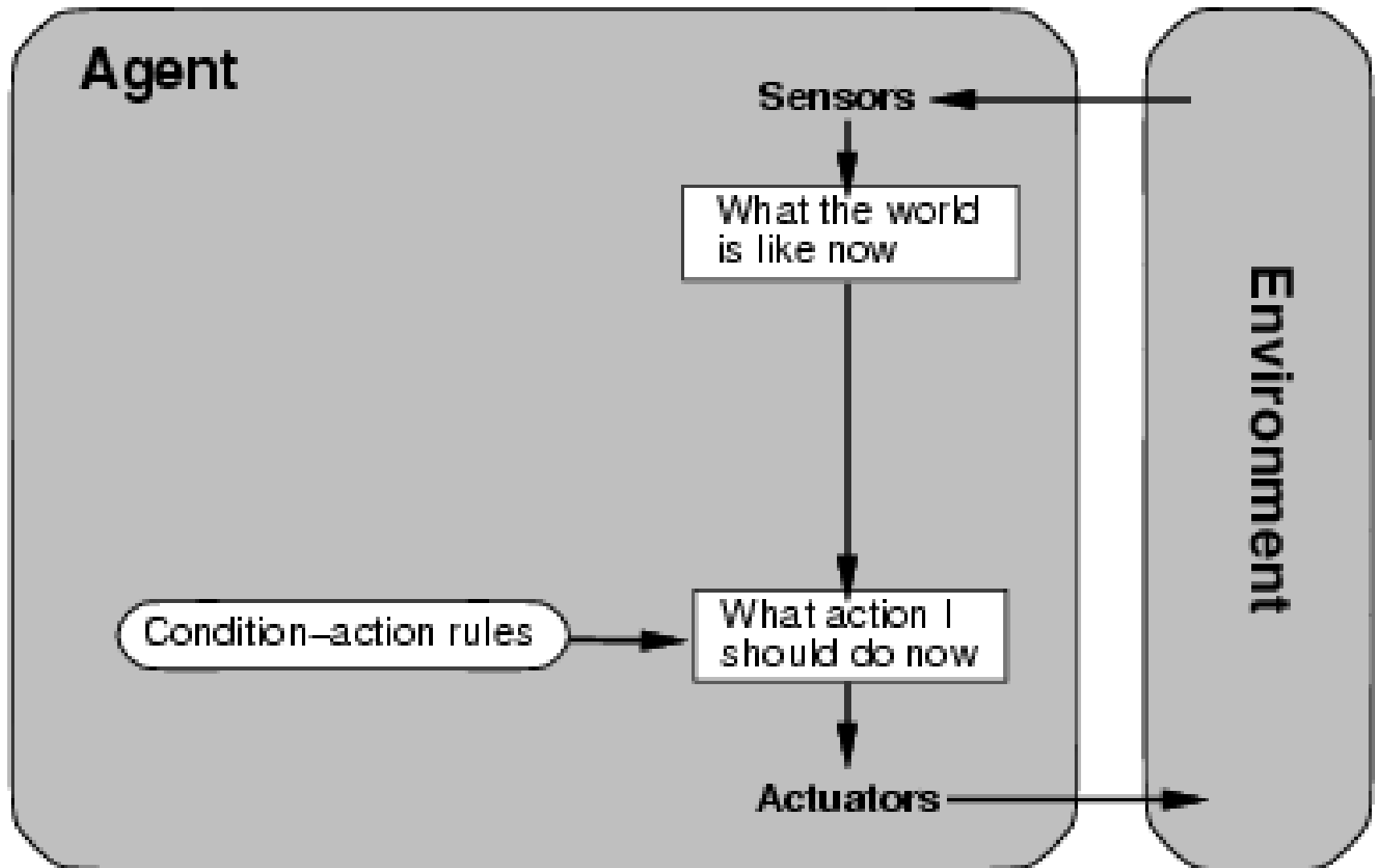
Agent functions and programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- One agent function (or a small equivalence class) is rational
- Aim: find a way to implement the rational agent function concisely

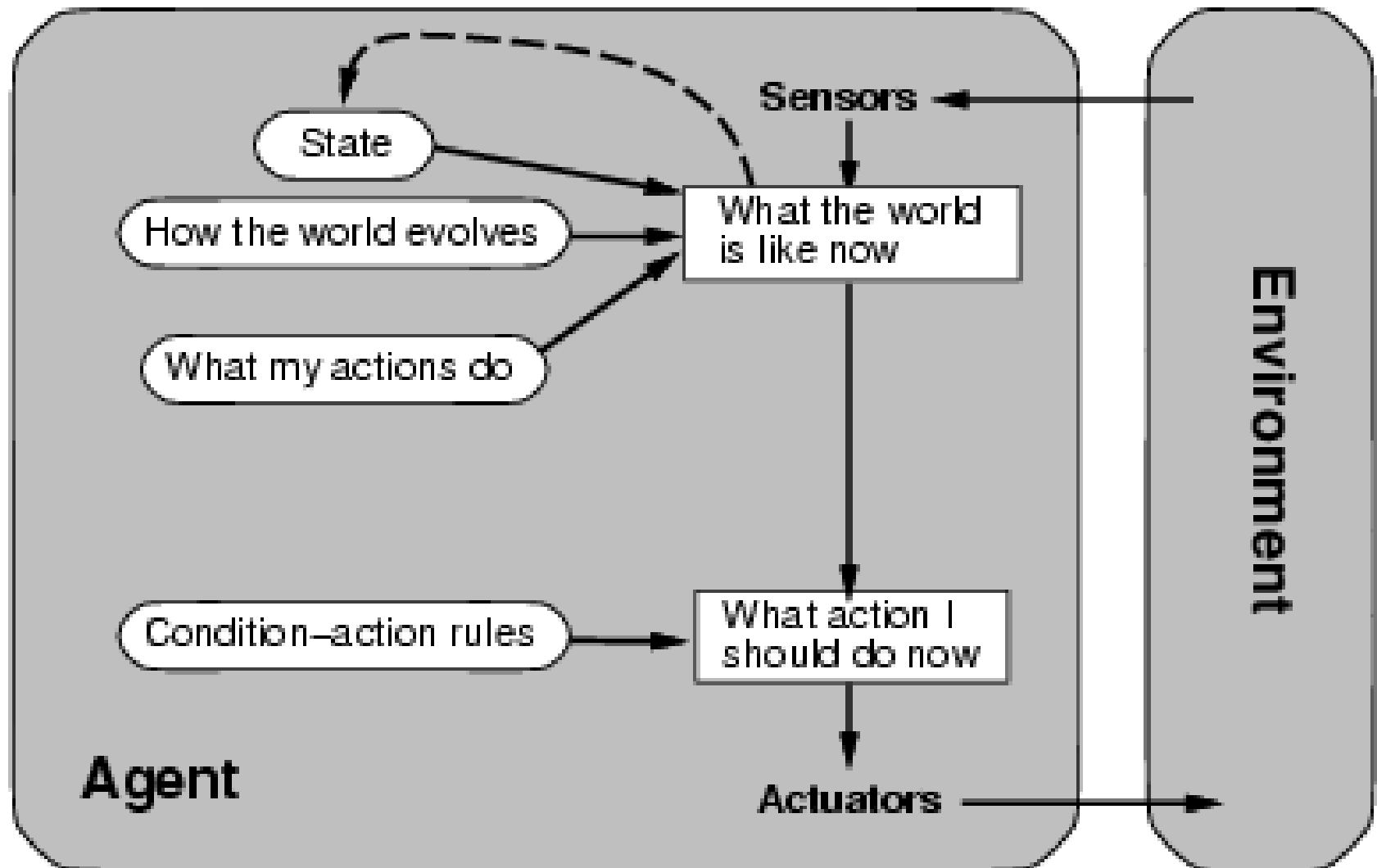
Agent types

- Four basic types in order of increasing generality:
 - Simple reflex agents
 - Model-based reflex agents
 - Goal-based agents
 - Utility-based agents

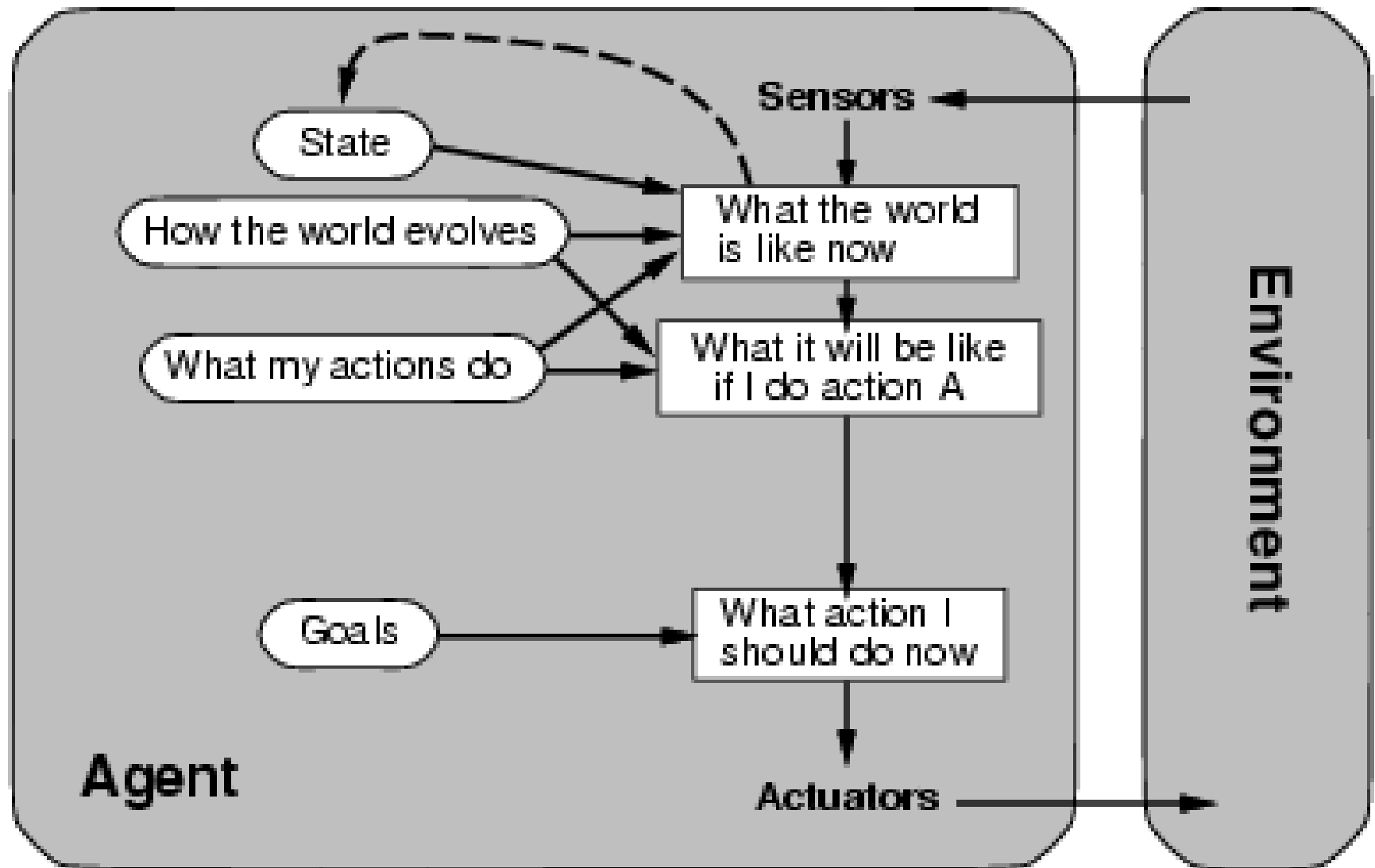
Simple reflex agents



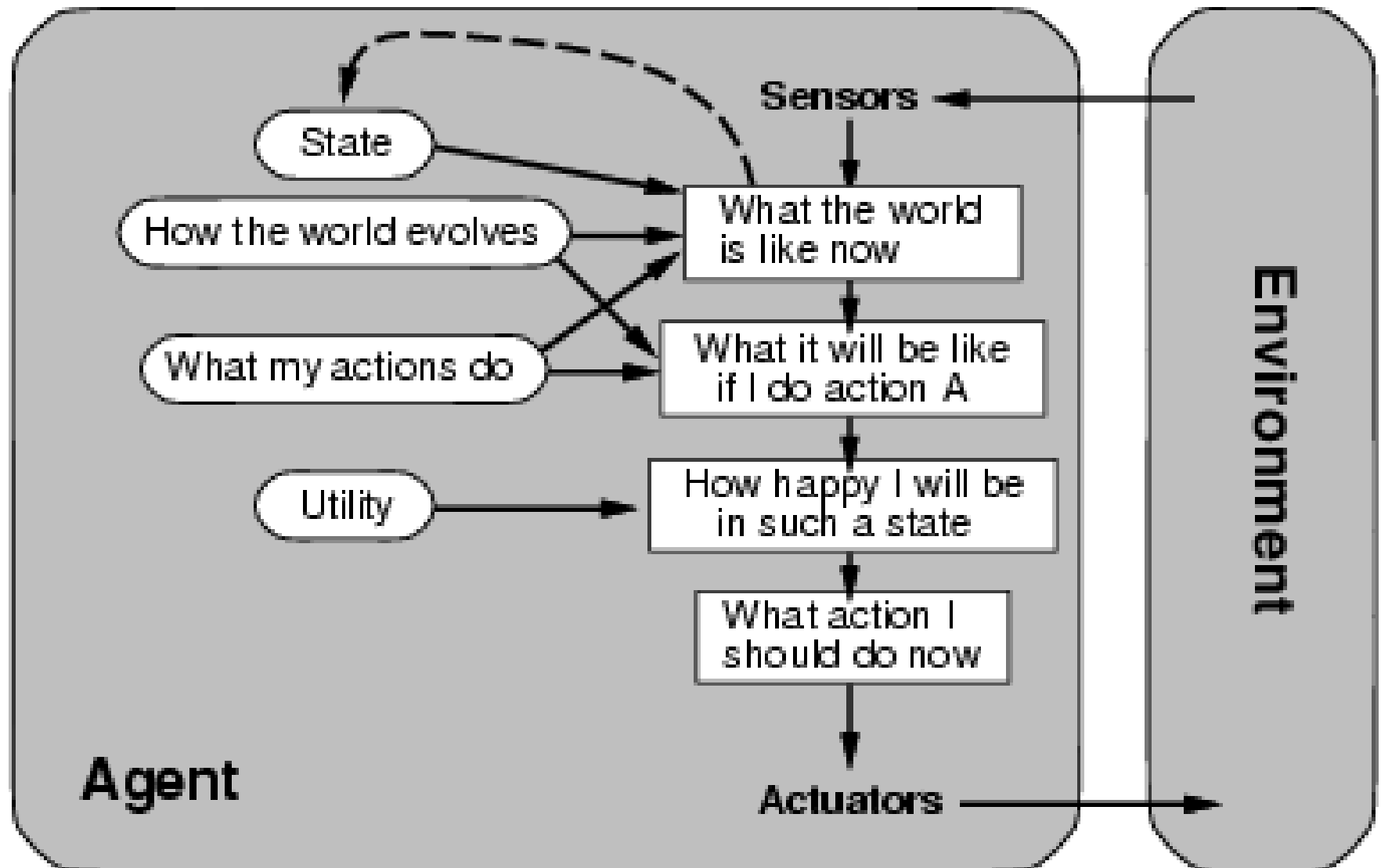
Model-based reflex agents



Goal-based agents



Utility-based agents



Learning agents

