Modul : Introduction to AI

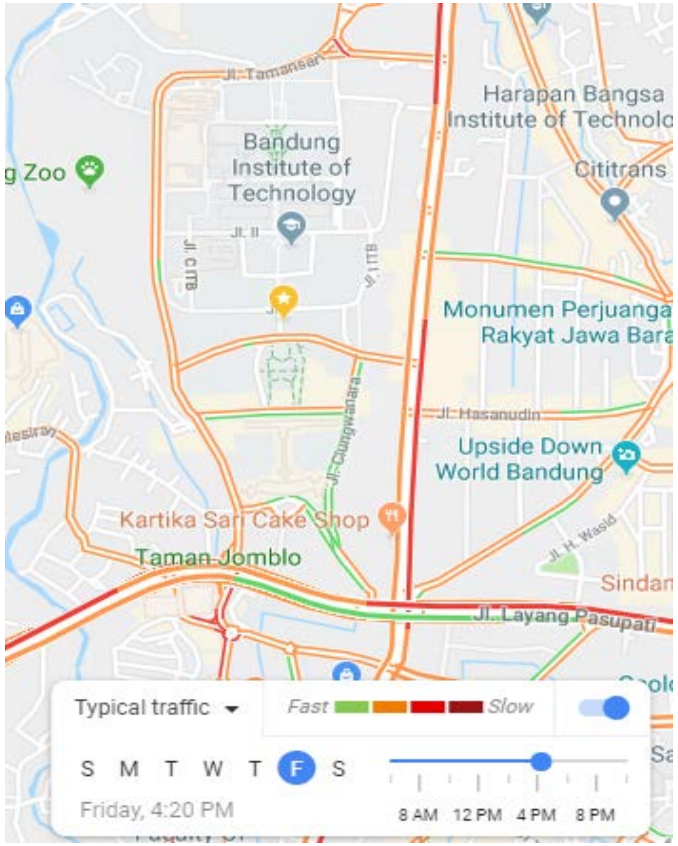
AI Application

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

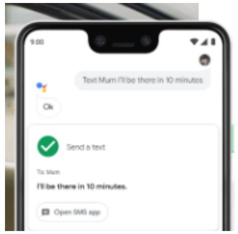
AI in Life: Path Finding

Path Finding / Direction

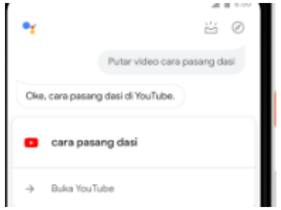


Traffic pattern

AI in Life: AI Assistant



“Hey Google, text Mum I'll be there in 10 minutes”

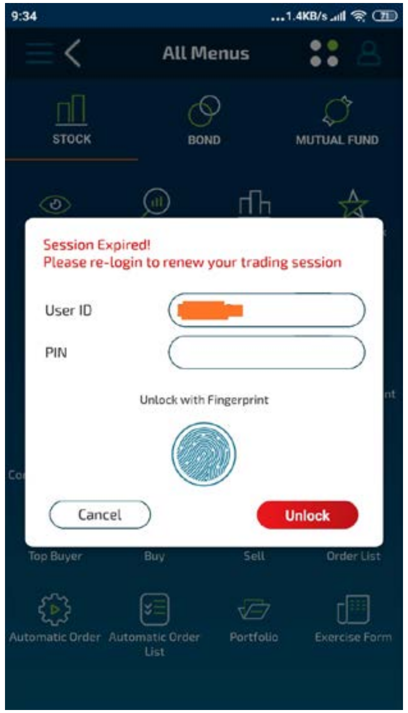


“Hey Google, play my morning playlist”

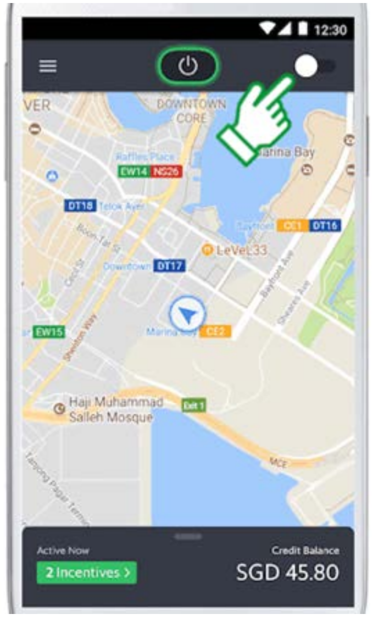
“Hey Google, dim the bedroom lights”

“Hey Google, set the temperature to 20 degrees”

AI in Life: Biometric Application



Fingerprint verification



Face verification

AI in Life: Transform Your Face

Pumarola, A., Agudo, A., Martinez, A.M., Sanfeliu, A., & Moreno-Noguer, F. (2019). *GANimation: One-Shot Anatomically Consistent Facial Animation. In International Journal of Computer Vision*

AI in Life: OpenAI Five for Dota 2

APRIL 15, 2019 • 7 MINUTE READ

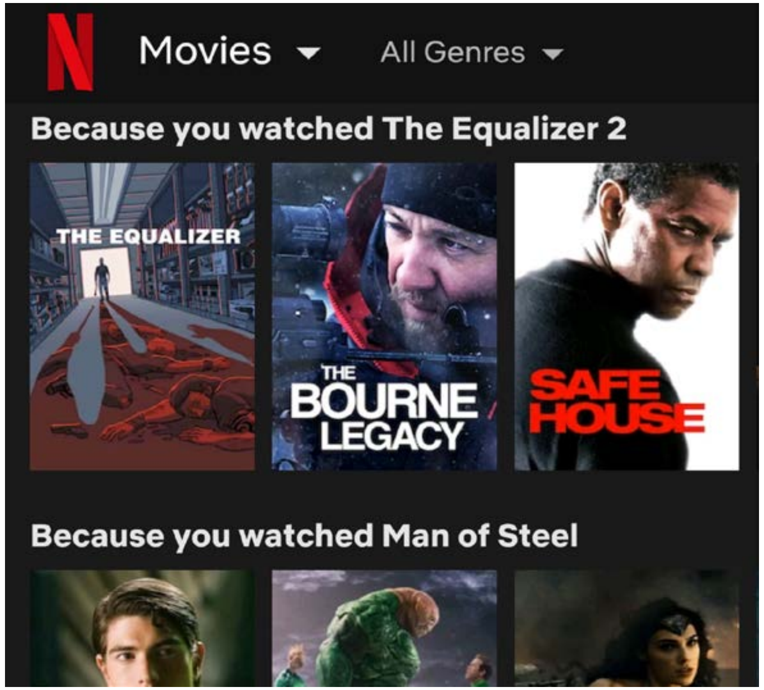
OpenAI Five Defeats Dota 2 World Champions

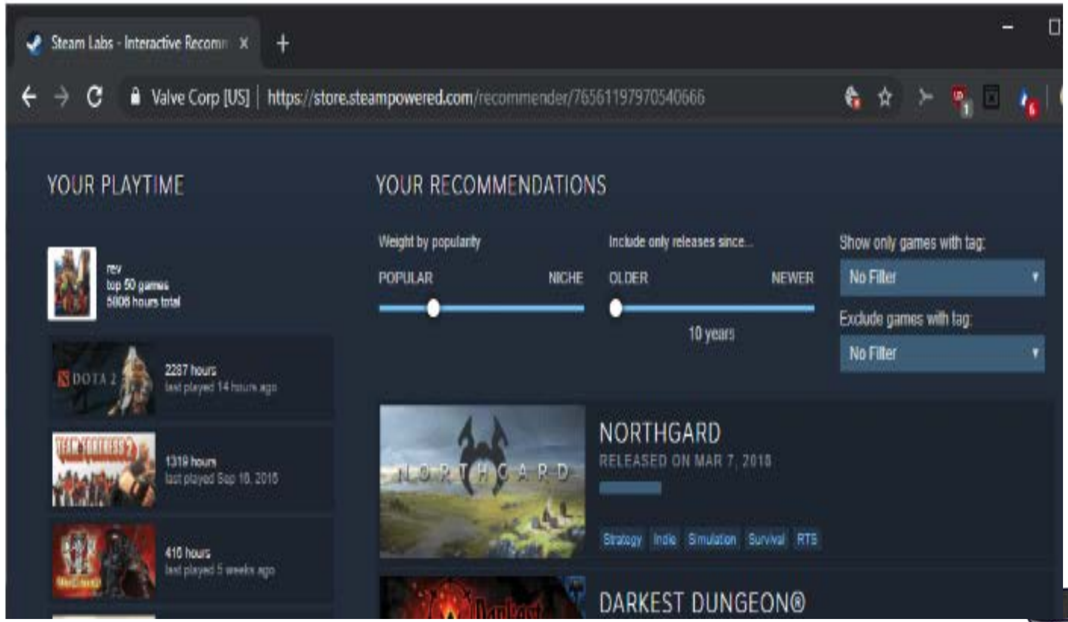
OpenAI Five is the first AI to beat the world champions in an esports game, having won two back-to-back games versus the world champion Dota 2 team, OG, at Finals this weekend. Both OpenAI Five and DeepMind's AlphaStar had previously beaten good pros privately but lost their live pro matches, making this also the first time an AI has beaten esports pros on livestream.

https://www.youtube.com/watch?v=UZHTNBMAfAA https://openai.com/blog/openai-five-defeats-dota-2-world-champions/

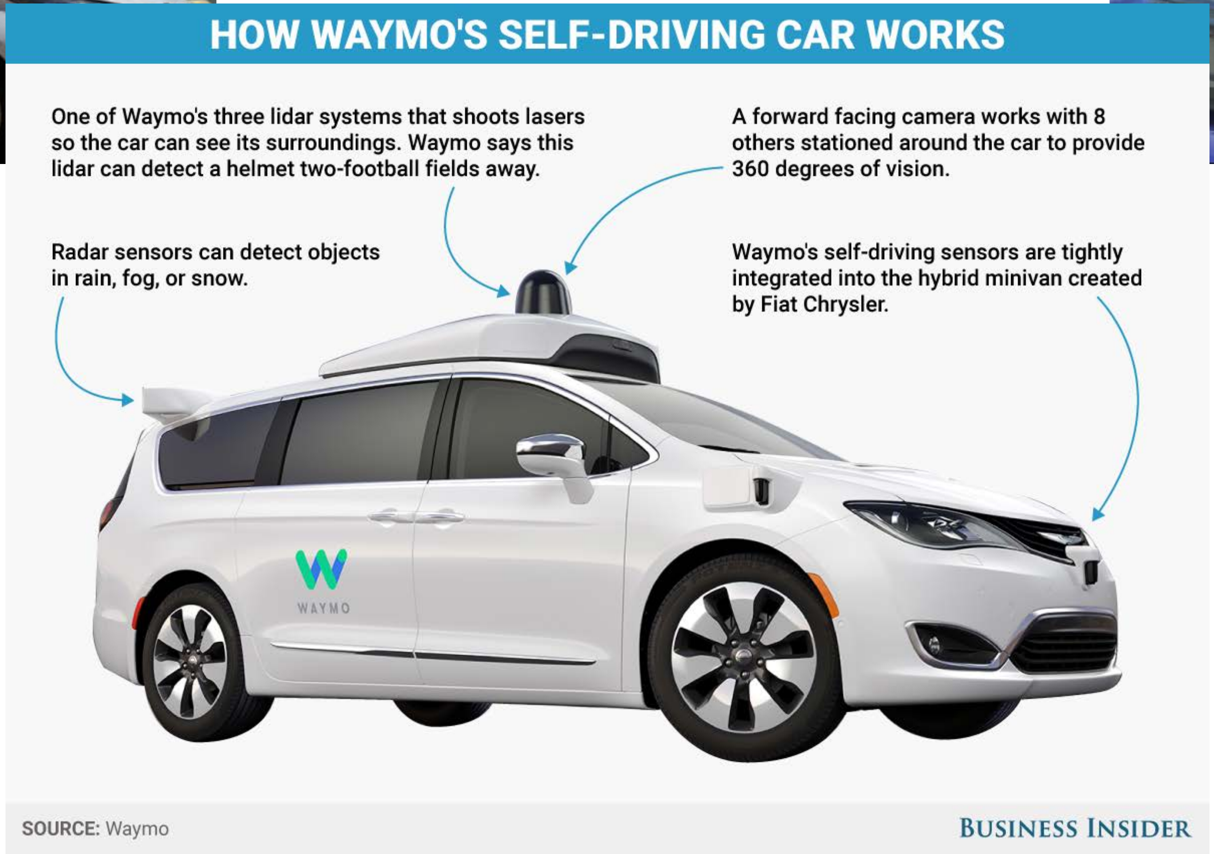
AI in Life: Recommender System

Spotify





AI in Life: Self Driving Car



http://www.businessinsider.sg/how-does-googles-waymo-self-driving-car-work-graphic-2017-1/?r=US&IR=T

VARIOUS AI APPLICATION

- AI for HEALTHCARE
- AI for EDUCATION
- AI for ENERGY INDUSTRY
- AI for MANUFACTURING

Modul: Introduction to AI

What is AI (1): Acting Humanly

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

4 Approaches in AI Definition

Humanly or Rationally

Thinking

or

Acting

Thinking Humanly

Thinking Rationally

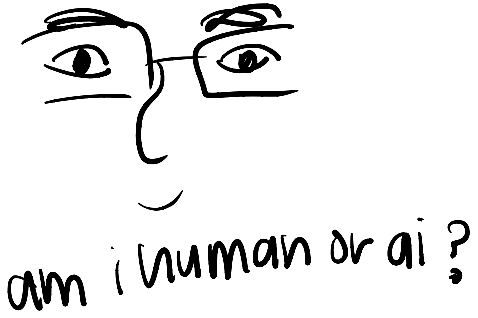
Acting Humanly

Acting Rationally

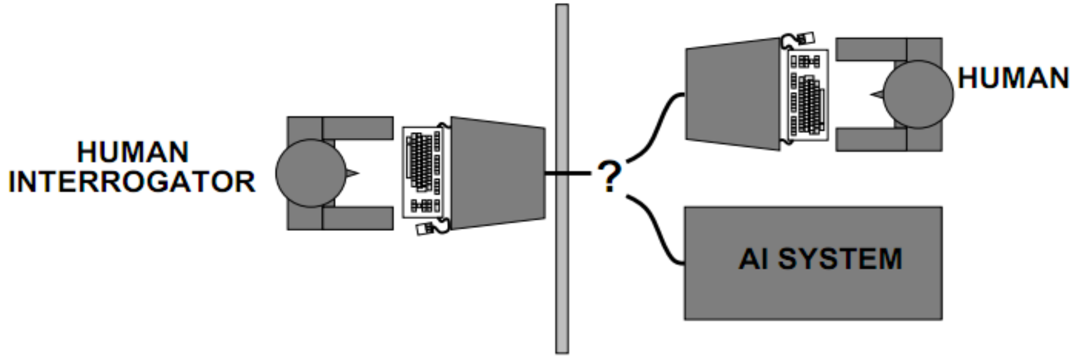
1st Approach of What is AI: ACTING HUMANLY

The art of creating machines that perform functions that require intelligence when performed by people (Kurzweil, 1990)

The study of how to make computers do things at which, at the moment, people are better (Rich and Knight, 1991)



1st Approach of What is AI: Acting Humanly (2)



TURING TEST

LOEBNER PRIZE

Annual Turing Test Competition

https://www.pandorabots.com/mitsuku/

Acting Humanly Approach: Applications Examples

PXDES, pneumoconiosis diagnosis from X-ray app, imitating a pneumoconiosis expert

AGREX, Agricultural Doctor (Agricultural Expert System)

Modul: Introduction to AI

What is AI (2): Thinking Humanly

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

4 Approaches in AI Definition

	Humanly or Rationally	
Thinking or Acting	Thinking Humanly	Thinking Rationally
	Acting Humanly	Acting Rationally

2nd Approach of What is AI: THINKING HUMANLY

[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ...
(Bellman, 1978)

The exciting new effort to make computers think ... machines with minds, in the full and literal sense
(Haugeland, 1985)

Thinking Humanly Approach: Applications Examples

NuPIC

Numenta Platform for Intelligent Computing

https://numenta.org/

Sighthound VIDEO

intelligent security camera software

Our patented deep learning computer vision technology teaches your cameras to see video just like the human brain.

https://www.sighthound.com/products/sighthound-video

Modul: Introduction to AI

What is AI (3): Thinking Rationally

KK IF – Teknik Informatika – STEI ITB

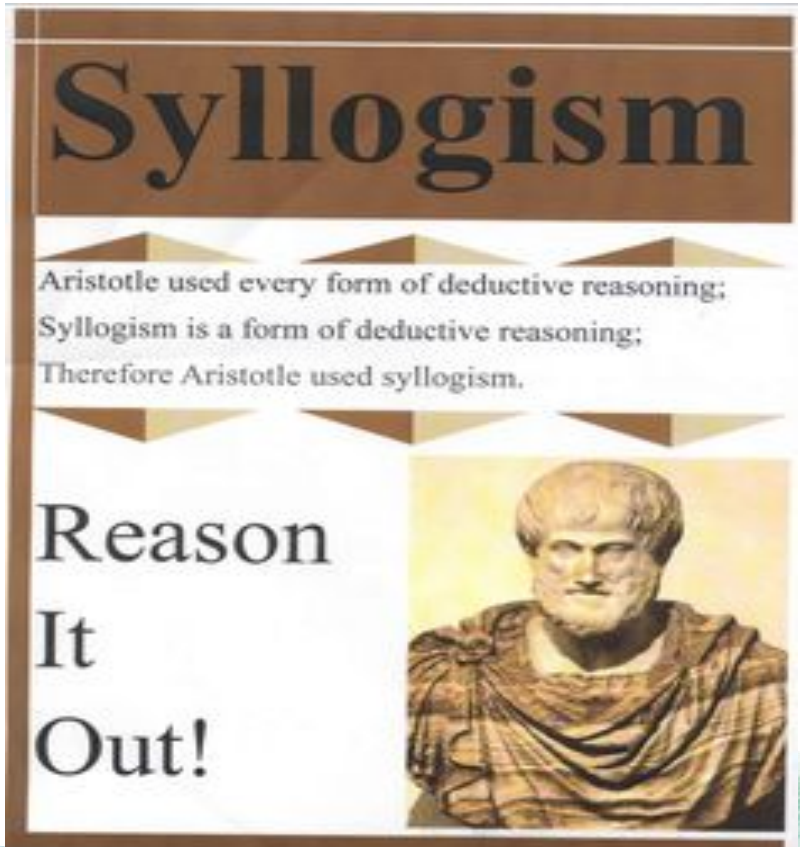
Inteligensi Buatan
(Artificial Intelligence)

4 Approaches in AI Definition

	Humanly	or	Rationally
Thinking or Acting	Thinking Humanly		Thinking Rationally
	Acting Humanly		Acting Rationally

3rd Approach of What is AI: THINKING RATIONALLY

The study of mental faculties through the use of computational models (Charniak and McDermott, 1985)



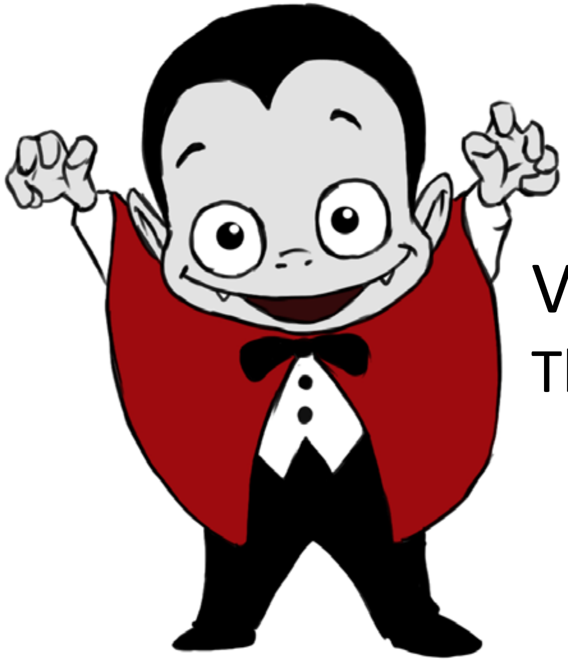
The study of the computations that make it possible to perceive, reason, and act (Winston, 1992)

Thinking Rationally Approach: Applications Examples

LEAN THEOREM PROVER

Microsoft Research

http://leanprover.github.io/



VAMPIRE
Theorem Prover

https://vprover.github.io/index.html

Modul: Introduction to AI

What is AI (4): Acting Rationally

KK IF – Teknik Informatika – STEI ITB

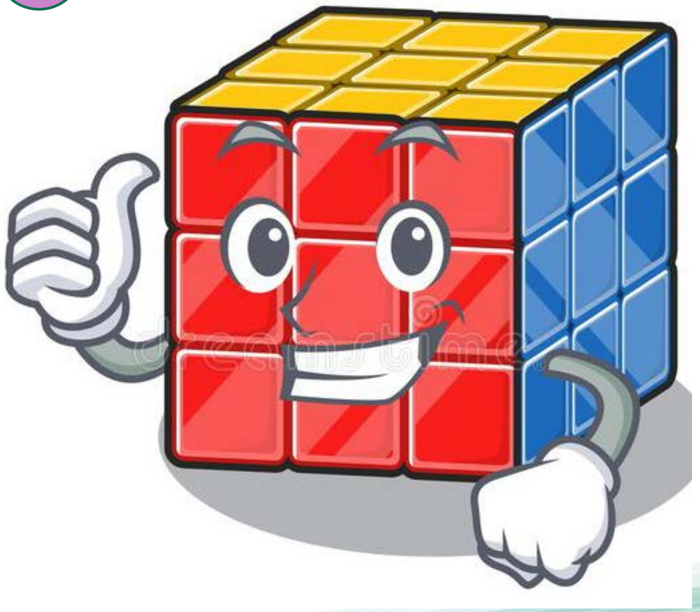
Inteligensi Buatan
(Artificial Intelligence)

4 Approaches in AI Definition

	Humanly or Rationally	
Thinking or Acting	Thinking Humanly	Thinking Rationally
	Acting Humanly	Acting Rationally

4th Approach of What is AI: ACTING RATIONALLY

Computational intelligence is the study of the design of intelligent agents(Poole et al., 1998)



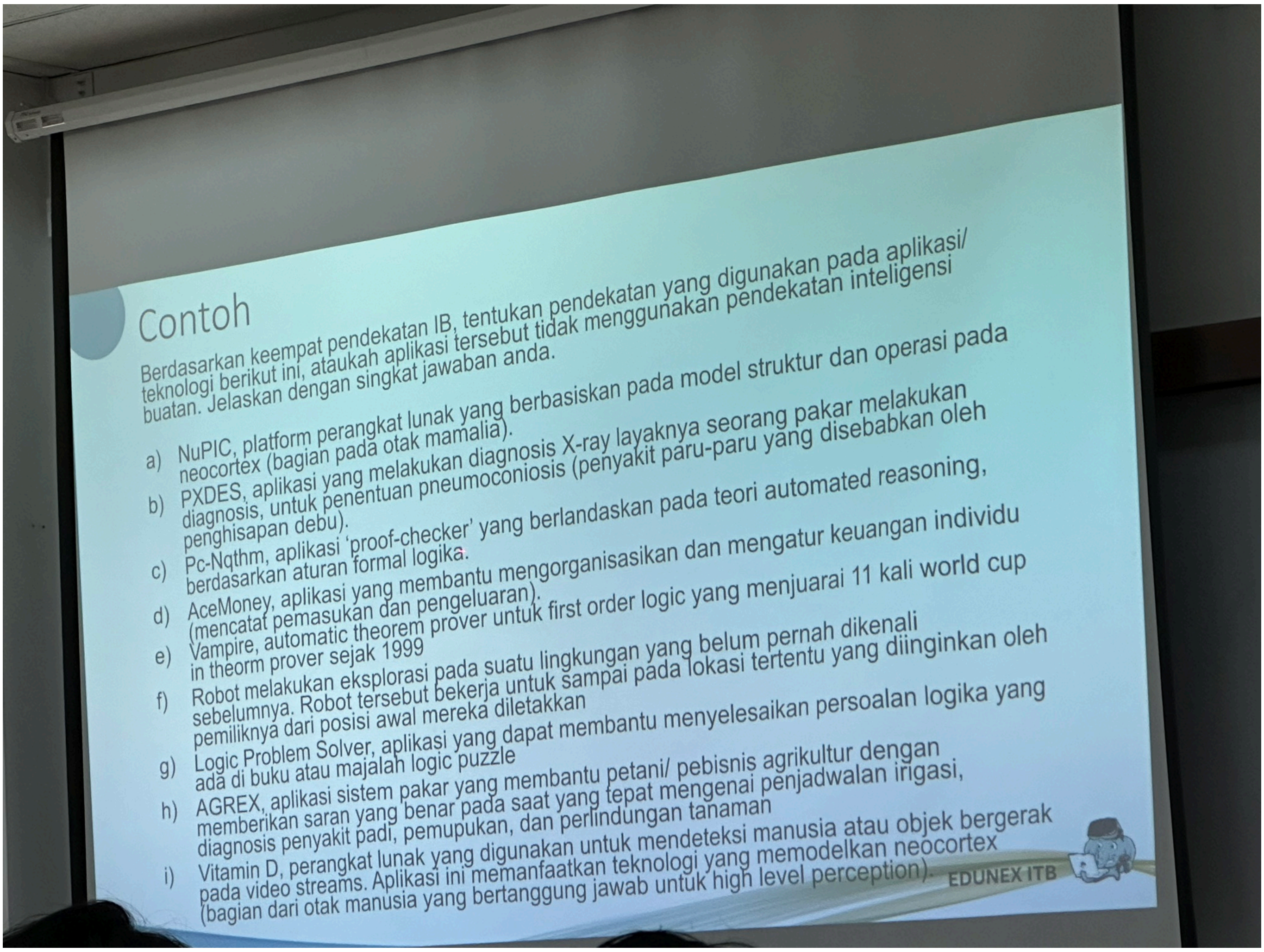
AI...is concerned with intelligent behavior in artifacts (Nilsson, 1998)

Acting Rationally Approach: Applications Examples

http://www.jobshop.72.sk/?m=0HU

https://blog.dota2.com/?l=english

THANK YOU



a) thinking humanly → meniru otak

b) acting humanly → meniru perilaku pakar

c) thinking rationally

d) bukan AI

e) thinking rationally

f) acting rationally

g) gak dinyatakan menyelesaikan persoalan pake pendekatan apa, bisa aja searching biasa

h) acting humanly

i) thinking humanly

→ pakai hukum² logika, bgmn seharusnya manusia berpikir

i love densu <3

thx

acting rationally
→ dfs, bfs, a*
→ model matematis yg telah ditemukan sebelumnya.

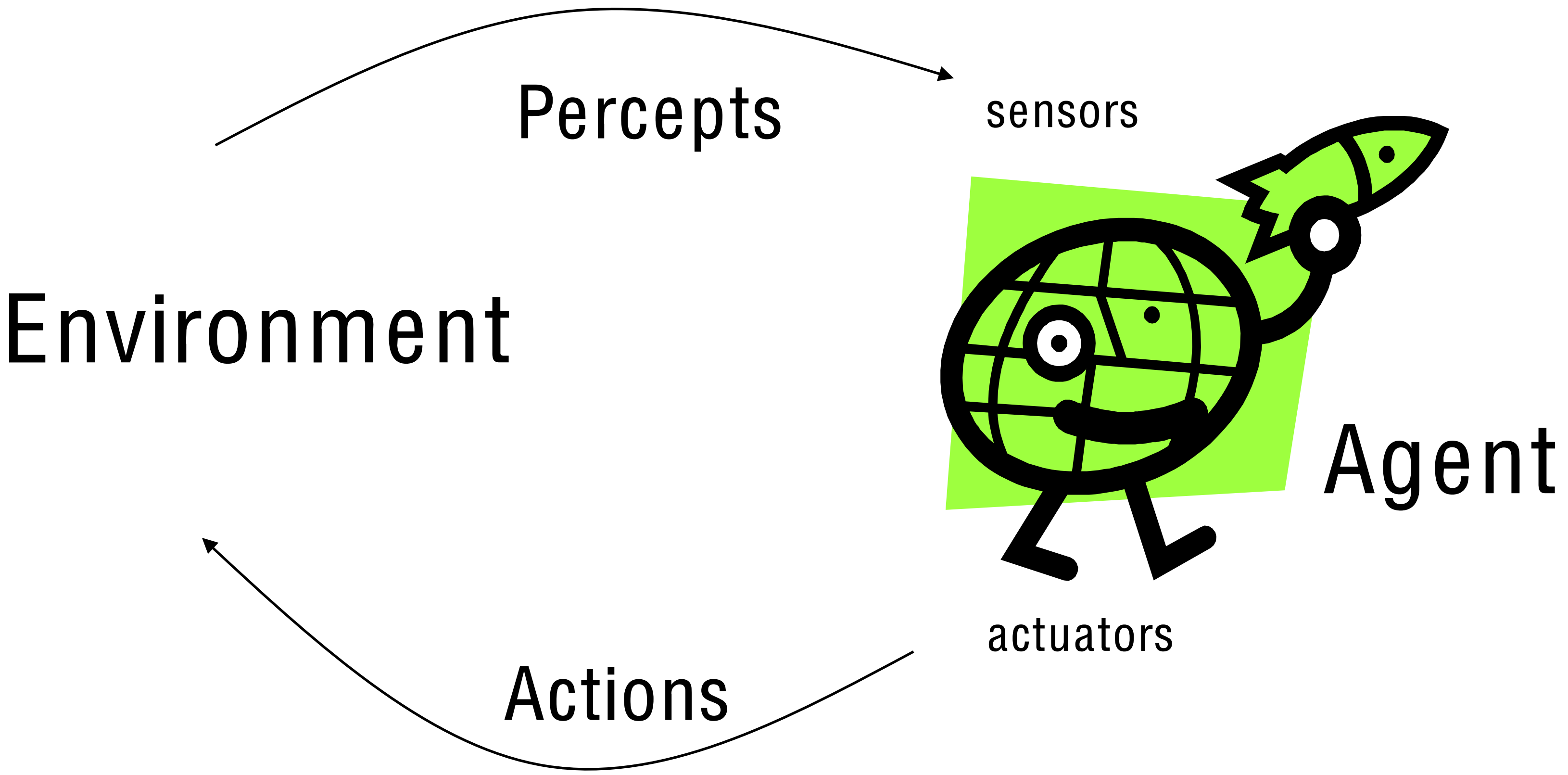
Modul : Intelligent Agent

Agent & Environment

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Agent & Environment



So long London

What is Agent?

Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

A robot

A factory

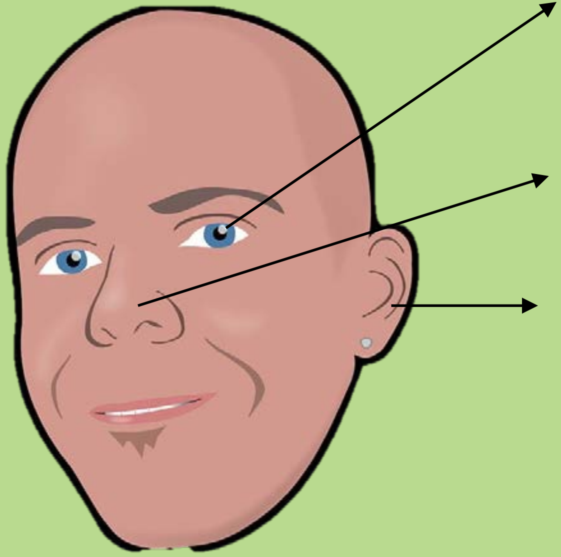
A web shopping program

…

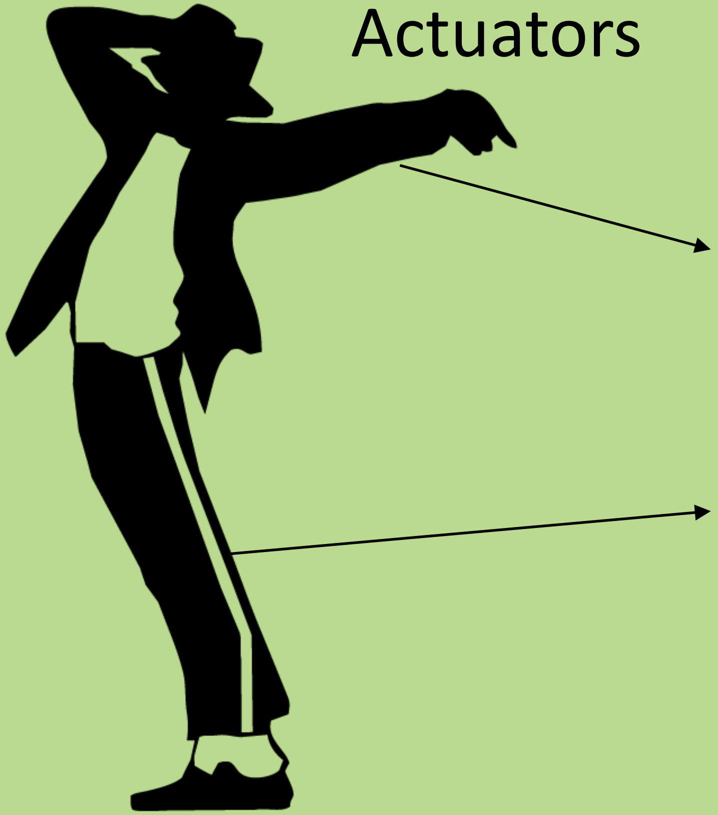
Computational agents that behave autonomously

Example: Human Agent

Sensors



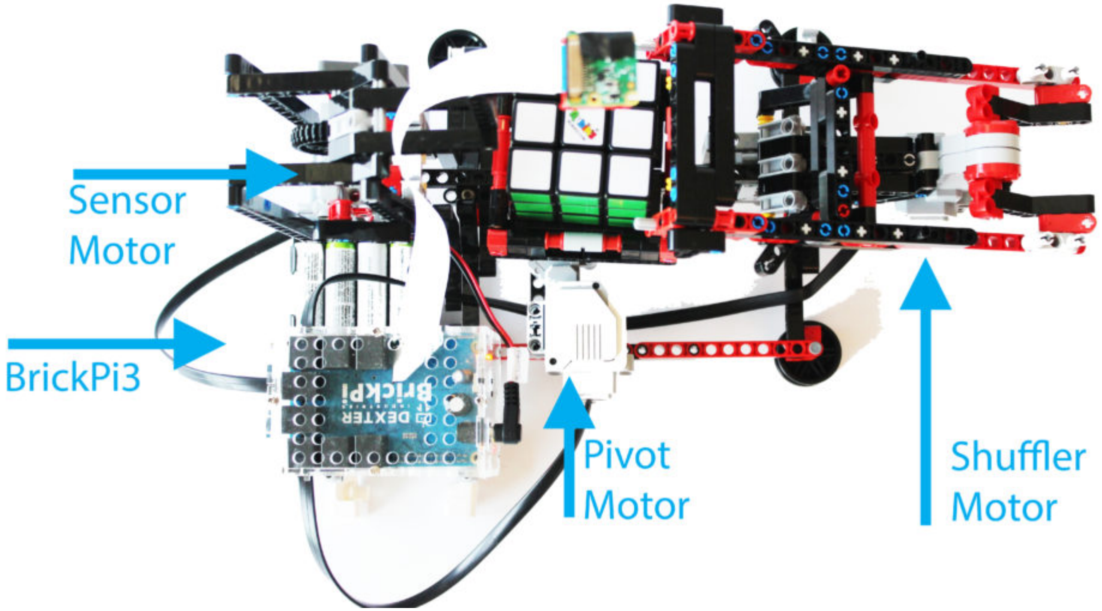
Actuators



Other Example: Rubic Solver Robot Agent

Sensors

Actuators



https://www.dexterindustries.com/projects/brickuber-project-raspberry-pi-rubiks-cube-solving-robot-project/

01

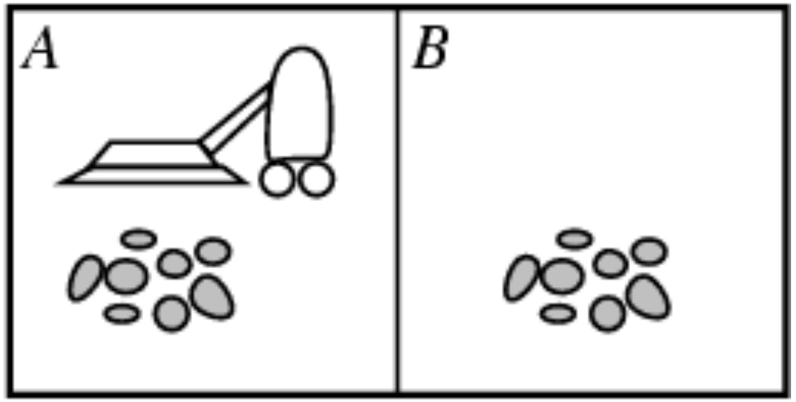
Modul : Intelligent Agent

Agent Model

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Vacuum-cleaner World



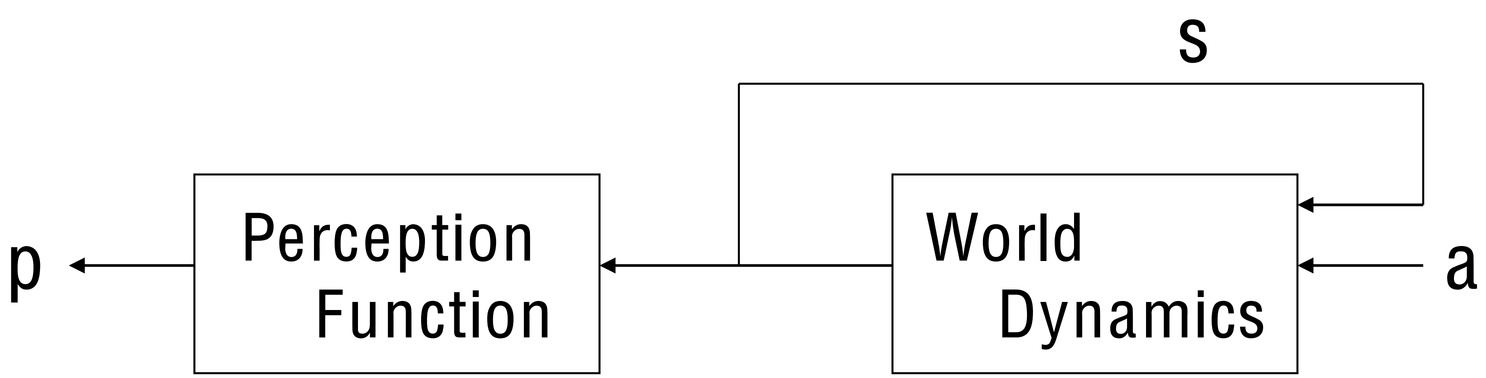
intelligent agent = cuman lakuin yg udah di program aja, kalau dia bikin yg lain = rusak

Percepts: something that is perceived by the agent **sensors**
→ location and contents: [A, Dirty]

Action: something that is carried out by the agent **actuators**
→ *Left*, *Right*, *Suck*, *NoOp*

How to Model the Agent World ??

- Percept Space
- Action Space
- Internal State
- Perception Function: S → P
- World Dynamics: S x A → S



Utility Function (U):
S → real
S* → real

Agent Design

Problem:
Find P* → A

Mapping sequence of percept (P*) to action (A)

Maximize U
(sequences of states S)

Remember:
S x A → S

Modul : Intelligent Agent

Rational Agent

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Rational Agent

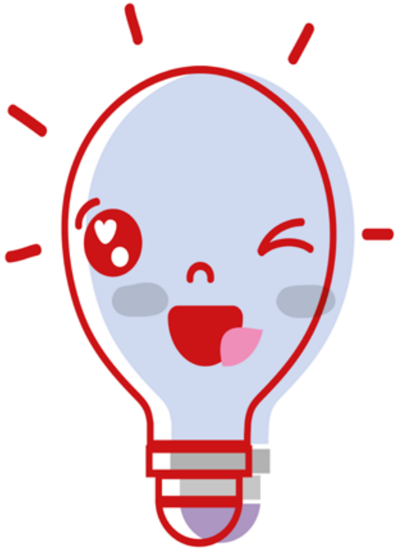
Strive to **DO THE RIGHT THING**

- Based on what it can perceived
- Based on what it can perform

<u>Performance Measure:</u>
Objective Criterion for Success of an Agent's behaviour

Rational Agent (2)

Sequences of Percepts



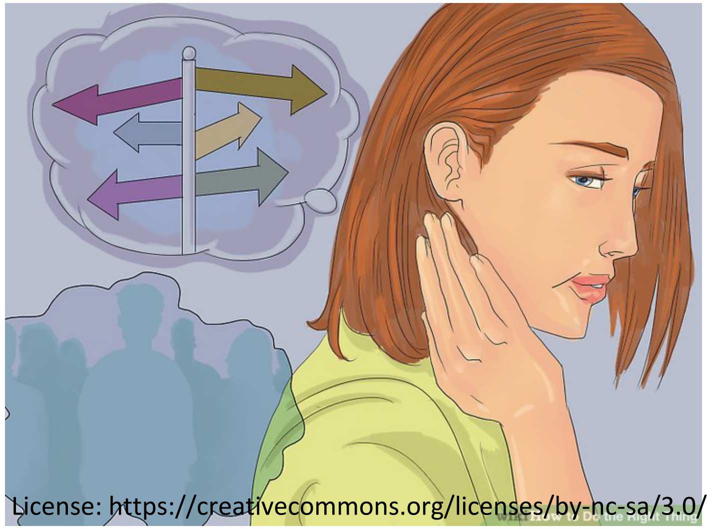
https://www.vecteezy.com/free-vector/kawaii

Actions

Performance Measurement

“Knowledge”

Rationality



blm tentu
tau semua,
bergantung sama
info yg kasih

Omniscience

Success

Limited Rationality

Rationality limitation:
Computational Constraint

Agent Design

Problem:
Find P* → A

Mapping sequence of percept (P*) to action (A)

Maximize U
(sequences of states S)
<u>Subject to Computational Constraints</u>

P E A S

THANK YOU

Modul : Intelligent Agent

P E A S

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

P E A S

Performance Measure

Environment

Actuators

Sensors

By: Strader

Example: Designing an Automated Taxi Driver

http://www.gettyimages.com/detail/83988175/Stone

- **P** Safe, fast, legal, comfortable trip, maximize profits
- **E** Roads, other traffic, pedestrians, customers
- **A** Steering wheel, accelerator, brake, signal, horn
- **S** Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

Example: Medical Diagnosis System Agent

P: Healthy patient, minimize costs, lawsuits

E: Patient, hospital, staff

S: Keyboard (entry of symptoms, findings, patient's answers)

A: Screen display (questions, tests, diagnoses, treatments, referrals)

01

Modul : Intelligent Agent

Task Environments

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Task Environment

Fully vs Partially Observable

fully : semua kondisi environment diketahui

partially : ada kondisi env yg tdk diketahui

Deterministic vs Stochastic

udh pasti partially

d : aksi yg menghasilkan hal pasti

s : hasil aksi berupa probabilitas

ex : main kartu, kita gtw next nya kartu orang bakal apa

Known vs Unknown

pengetahuan agent dlm melakukan aksi

agent tdk punya pengetahuan dlm melakukan aksi selanjutnya meski lingkungannya observable

Episodic vs Sequential

episodic : aksi sekarang gk mempengaruhi aksi selanjutnya

sequential : aksi skrg mempengaruhi yg selanjutnya

ex : main catur

Static vs Dynamic

static : kondisi environmentnya ga berubah

dynamic : kondisi environment berubah selama agentnya mikir

ex : autonomous driving, eh ada yg nyebrang tiba² red light

Semidynamic

→ environmentnya sbnrnya ga berubah tp ada pertimbangan luar yg bikin agent jd bikin hal yg lain

Discrete vs Continuous

discrete : lingkungannya bisa dipisah (statenya pasti)

continuous : tdk bisa dipisahkan secara diskrit lingkungannya

cth : automated driver (lingkungannya beda² terus)

Single vs Multi Agent

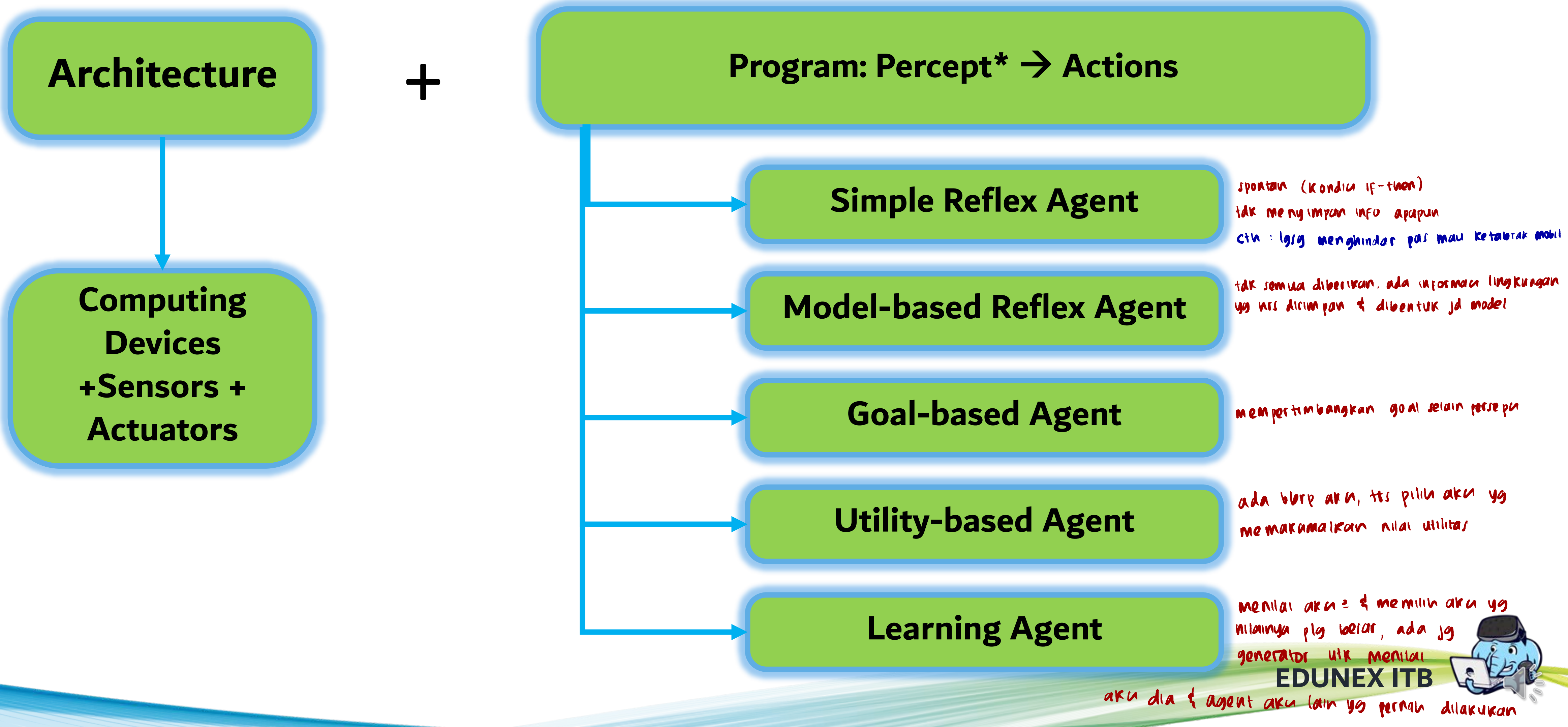
single : lingkungan berubah hanya berdasarkan aksi

multi agent : lingkungan berubah karena ga cuman aksi tapi ada faktor luar

Examples

	Chess with a clock	Chess without a clock	Taxi driving
Fully vs Partially Observable	Fully	Fully	Partially
Deterministic vs Stochastic	Deterministic	Deterministic	Stochastic
Episodic vs Sequential	Sequential	Sequential	Sequential
Static vs Dynamic	Semidynamic	Static	Dynamic
Discrete vs Continuous	Discrete	Discrete	Continuous
Single vs Multi Agent	Multi Agent	Multi Agent	Multi Agent

Agent Structure



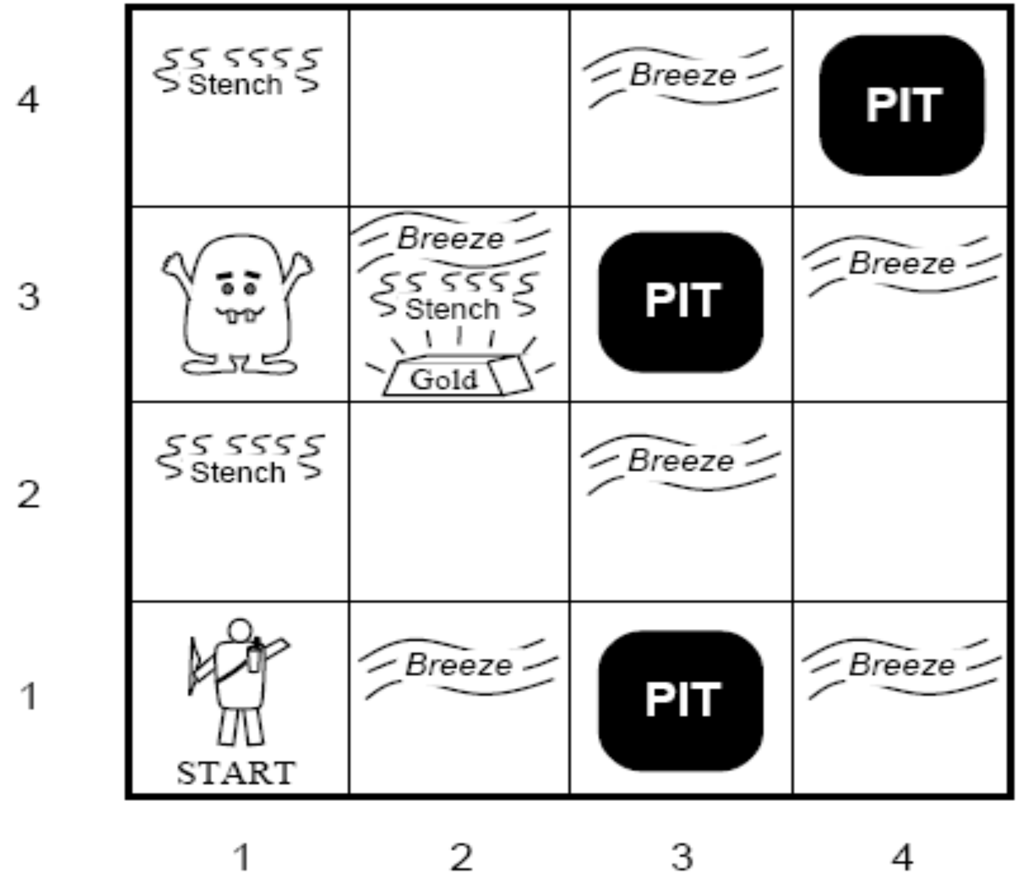
Modul : Intelligent Agent

Agent Level

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Wumpus World



Performance Measure: gold +1000, death -1000, -1 per step, -10 for using the arrow

Environment: cave, rooms, Wumpus, gold

Actuators: motor to move Left, Right, Forward, hands to Grab, Release, and Shoot arrow

Sensors: sensor to capture [Stench, Breeze, Glitter, Bump, Scream]

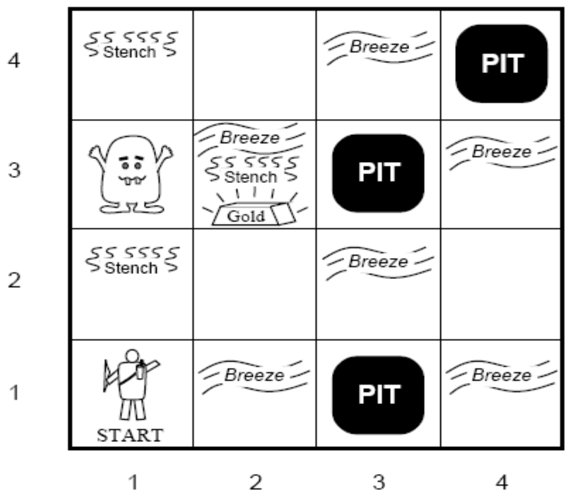
Level 1: Problem Solving Agent

agent punya informasi state scr lengkap

Agent has information about all of the states in Wumpus World

Agent has to 'search', the path that can lead agent to the goal, as fast as possible

There are many searching algorithms, and each algorithm is suitable for certain problem



DFS, BFS, IDS, UCS
A*, Greedy Best First,
Minmax search,
Genetic Algorithm, Hill Climbing,
Simulated Annealing,
Etc...

Level 2: Knowledge Based Agent

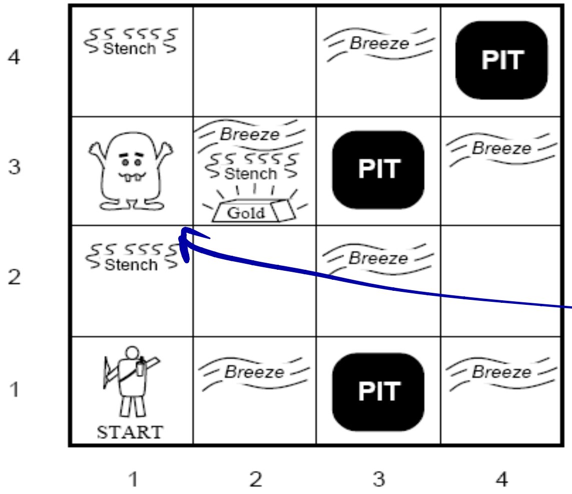
yg diberikan ke agent hanya knowledge saja

agent gapunya info state lengkap, jadi tiap ada state baru dia hrs mikir based on hal² yg udh diketahui

Agent doesn't have information about all of the states in Wumpus World. It only has 'basic knowledge/ premises'

When agent percept a state in a room, it will try to reason new facts/ states, this is how agent will step by step collecting all of the states of wumpus world in order to achieve its goal

Reasoning has to be done by Agent→ by deducting the premises with percepted fact.



contoh : if ada bau, maka ada wumpus

ini ya wumpus

Level 3: Learning Agent

bodoh di awal, pintar di akhir

agent gapunya state lengkap & gapunya knowledge

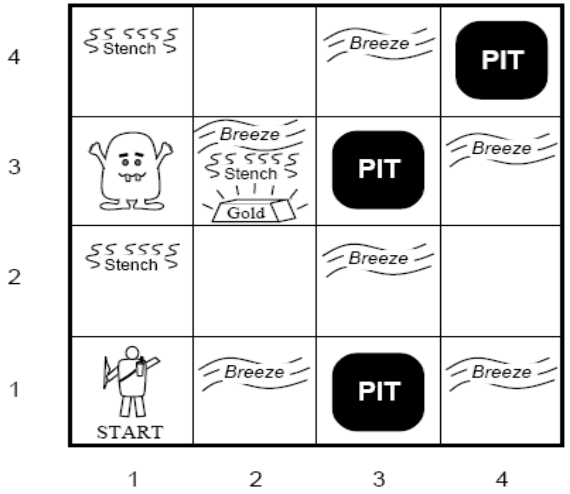
jd agent belajar bnr² dr tiap aksi yg dilakukan

di awal, agent bnr² gatau apa²

Agent doesn't have the information of all of the states and doesn't even have the basic knowldge of the wumpus world

Agent plays several times (perhaps dies several times) → The **observation data** from playing several times is the 'input' for learning process

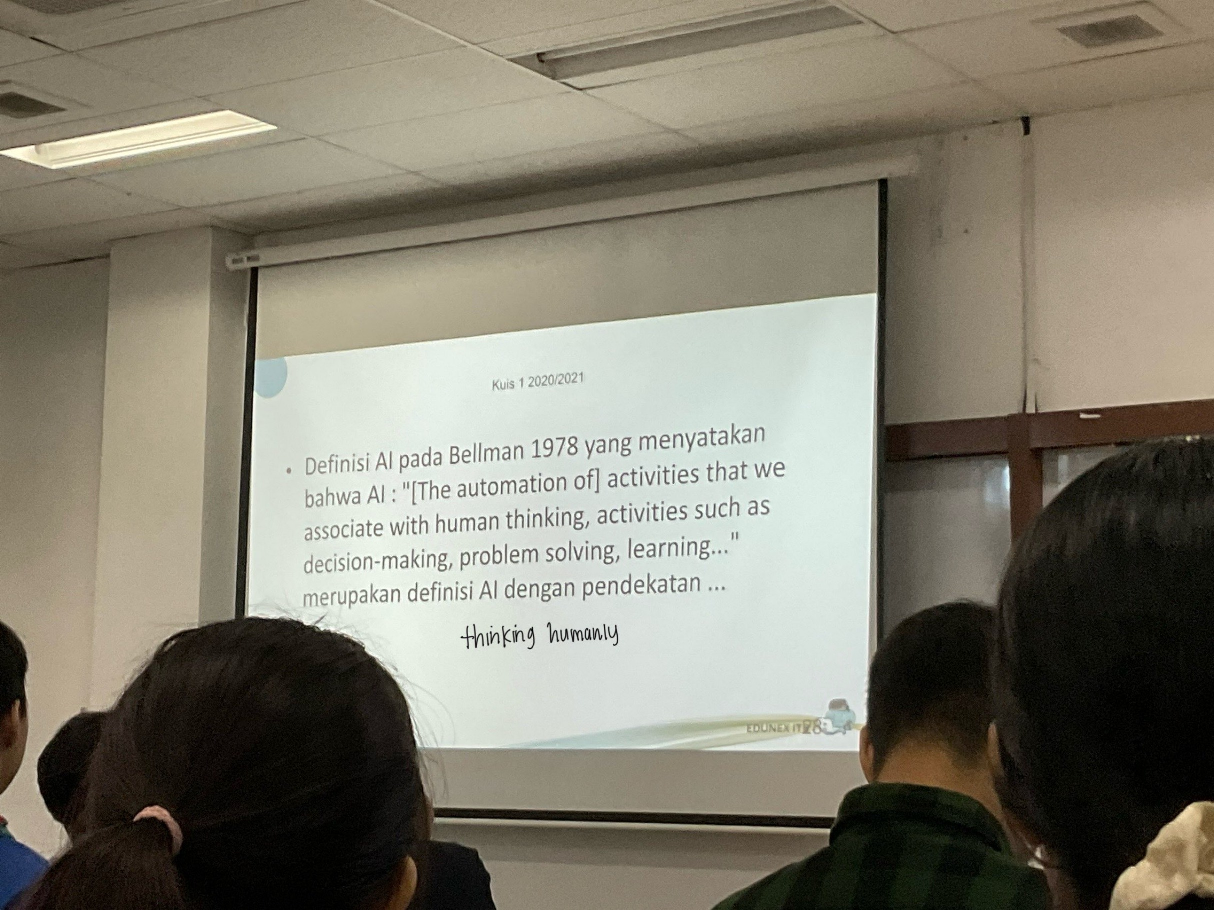
The result of the learning process, agent will have basic knowledge, e.g. Squares adjacent to pit are breezy



There are many learning algorithms, that suitable for certain purposes, and the 'availability' of the data/ feedback

- Supervised learning
- Unsupervised learning
- Reinforcement learning

THANK YOU


Kuis 1 2020/2021
Definisi AI pada Bellman 1978 yang menyatakan bahwa AI : "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning..." merupakan definisi AI dengan pendekatan ...
thinking humanly
28

Kuis 1 2020/2021

- Definisi AI pada Rich & Knight 1991 yang menyatakan bahwa AI : "The study of how to make computers do things at which, at the moment, people are better." merupakan definisi AI dengan pendekatan ...

acting humanly

No	Kalimat	Benar/ Salah (Nilai 1)	Alasan (Nilai 2)
a	Sebuah aplikasi chatbot harus dapat berpikir seperti manusia agar lolos Turing Test. → acting humanly	: salah	
b	Persoalan teka-teki logika yang diselesaikan dengan memanfaatkan algoritma Backtracking Search, merupakan contoh aplikasi berbasis AI dengan pendekatan thinking reationally. → Logika (law of thoughts)	: salah,	dia acting rationally
c	Bahasan Intelligent Agent dalam kuliah ini adalah agen yang bisa bersifat rational, artinya agen yang bekerja pada lingkungan tugas dengan properti 'partially observable' tidak akan bisa bersifat rational. ↳ tetap bisa bersifat rational	: salah	

2. Terdapat sebuah *intelligent agent* yang dibangun sebagai aplikasi web untuk membantu pembuktian suatu teorema matematika, dengan langkah sesedikit mungkin dan waktu secepat mungkin. Pembuktian ini memanfaatkan kaidah-kaidah inferensi yang sudah terdefinisi dalam domain matematika. Tentukan lingkungan tugas (*task environment*) PEAS dan 6 properti lingkungan tugas dari agen tersebut, dengan mengisi tabel berikut ini. Jawaban disertai alasan dengan singkat.

P : ketepatan, cepat, langkah
performance measure

E : orang (dia yg masukin teorema), web, kaidah² inferensinya
environment

A : layar, modul yg menerapkan kaidah inferensinya untuk membuktikan teorema math
actuator

S : keyboard
sensor

fully (di soal dh keliatan semua kaidah² jg dh tau)
deterministic (hasilnya udh pasti)
sequential (bisa pakai kaidah² sblmnya)
~~static (pas agent mikir, lingkungannya ga berubah)~~ ← semidynamic karena harus secepat mungkin
diskrit (lingkungan bs dipisah, statenya jelas. pas di apply kaidah, udh pasti hasilnya)
single

Lingkungan Tugas	Jawaban dengan alasan singkat
P	Pemilihan dan penerapan kaidah tepat, waktunya secepat mungkin dengan langkah sesedikit mungkin
E	Web, orang yang akan membuktikan teorema, kaidah
A	Modul inferensi yang dilengkapi dengan kaidah-kaidah, display untuk state antara dan hasil
S	Keyboard atau touch screen, bergantung asumsi device yang digunakan.

Problem Solving & Search

Informatics Engineering Study Program
School of Electrical Engineering and Informatics

Institute of Technology Bandung

Contents

- Review
- Problem Solving
- Example of Problem
- Formal Definition
- Search

Review

- What is AI → 4 approaches
  - For now we use 4th approach (acting rationally)
  - Rationality ≠ omniscience ≠ success
  - Limited rationality
- Intelligent Agent
  - PEAS
  - Task Environment :
    - Accessible (vs. Inaccessible) / Fully (vs partially) observable
    - Deterministic (vs. Non-Deterministic/ Stochastic)
    - Static (vs. Dynamic)
    - Discrete (vs. Continuous)
    - Episodic vs Sequential (non-episodic),
    - Single vs Multi agent
  - Agent Level

Problem Solving Agent

- Agent design:
  - formulate problem → search solution → execute
  - Task Environment: Remember PEAS
- Problem: satisfy goal (goal state)
  - Agent task: find out which sequence of actions will get it to a goal state
  - 5 components of **a problem formulation**:
    - initial states, intermediate states (state spaces),
    - goal state,
    - actions,
    - transition model (new_state = Result(ols_state,action)),
    - Action cost function
- Searching: process of looking for sequence of action
- Solution: sequence of action to goal state

Problem Solving

- Agent knows world dynamics
  - World states, actions
  - [when agent doesn't know → learning]
- World state is finite, small enough to enumerate
  - [when state is infinite →logic]
- World is deterministic
  - [when non-deterministic →uncertainty]
- Agent knows current state
  - [when agent doesn't know → logic, uncertainty]
- Utility for a sequence of states is a sum over path

Few real problems are like this, but this may be a useful abstraction of a real problem → solving problems by searching

Problem: Formal Definition

Problem components:

1. Initial State, State spaces: kota
   - State space forms graph (node: state, arc: action)
2. Goal State
3. Actions
4. Transition Model → S’ = Result(S,A) [deterministic]
5. Action Cost Function: (S,A)* ⟶ real
   - Sum of costs: Σ c(S,A)

- Solution: graph path
- Criteria for algorithms:
  - Computation time/space
  - Solution quality

Route Planning

Example: Route Planning in a Map

A map is a graph where nodes are cities and links are roads. This is an abstraction of the real world.

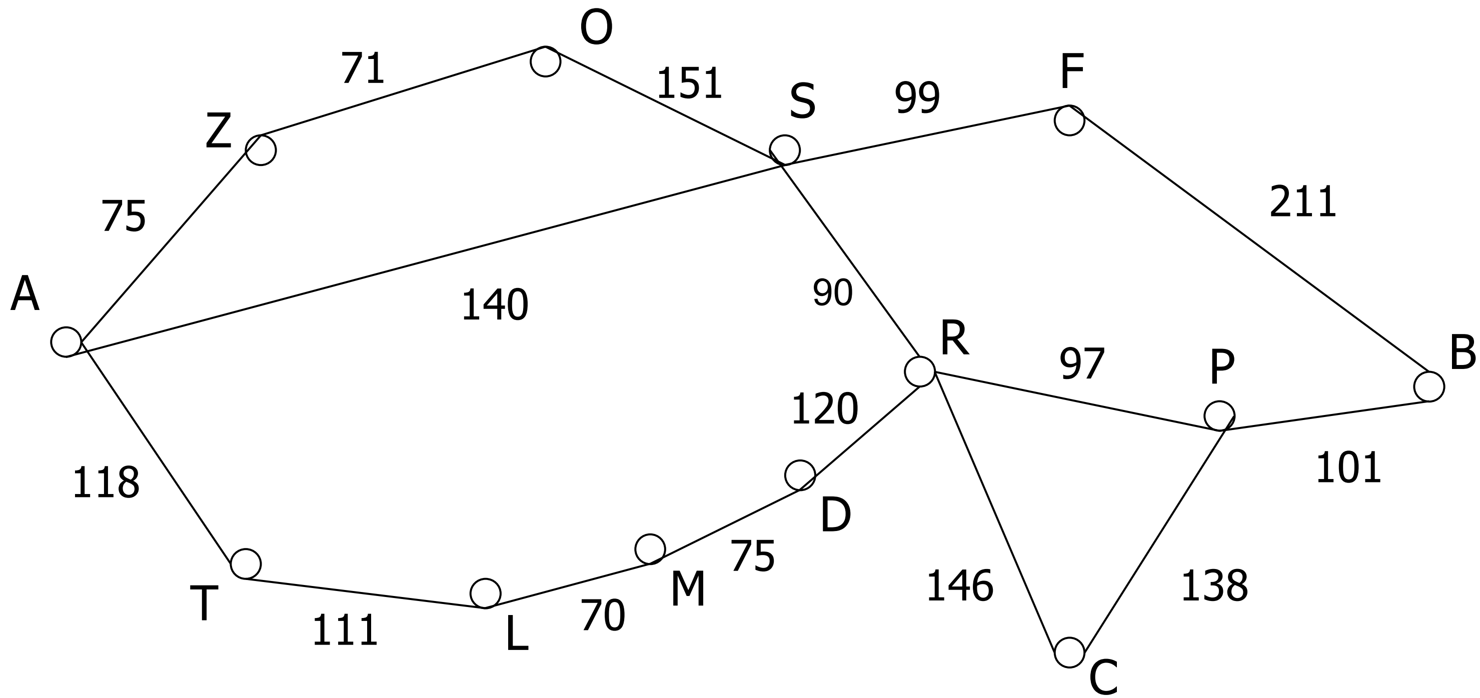
- Map gives world dynamics: starting at city X on the map and taking some road gets you to city Y.

Environment assumptions:

- Static: no change when solving problem
- Discrete: World (set of cities) is finite and enumerable.
- Deterministic: taking a given road from a given city leads to only one possible destination.
- Observable: information is complete
  - We assume current state is known.
- Utility for a sequence of states is usually either total distance traveled on the path or total time for the path.

Source: Russell's book

Search



S: set of cities
i.s: A (Arad)
g.s: B (Bucharest)
Goal test: s = B ?
Path cost: time ~ distance

Search

- UnInformed/Blind Search
  - Look around, don't know where to find the right answer
  - No additional information beyond that provided in problem definitional
  - Example: DFS, BFS, IDS, UCS , DLS
- Informed Search
  - Heuristic Search
    - Know some information that sometimes helpful
    - Know whether one non-goal state is "more promising" than another
    - Example: Best FS, A*,
- Local Search (for Optimization Problem) → Beyond Classical Search
    - Path to goal is irrelevant
    - Use very little memory
    - Can find reasonable solutions in large or infinite state spaces for which systematic algorithms are suitable
    - Example: Hill-climbing search, simulated annealing search, GA

Search

- It's time to do searching: covering the basic methods really fast.
  - Agenda: a list of states that are waiting to be expand

```
{Put start state (initial state) in the agenda}
AddState(Agenda, initial-state)
iterate
  GetState(Agenda, current-state)
stop: isGoal(current-state)
  if not isExpanded(current-state) then
     {put children in agenda}
     ExpandState(current-state, Agenda)
```

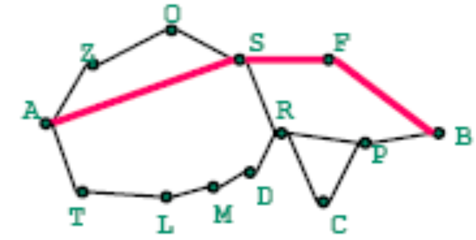
Search

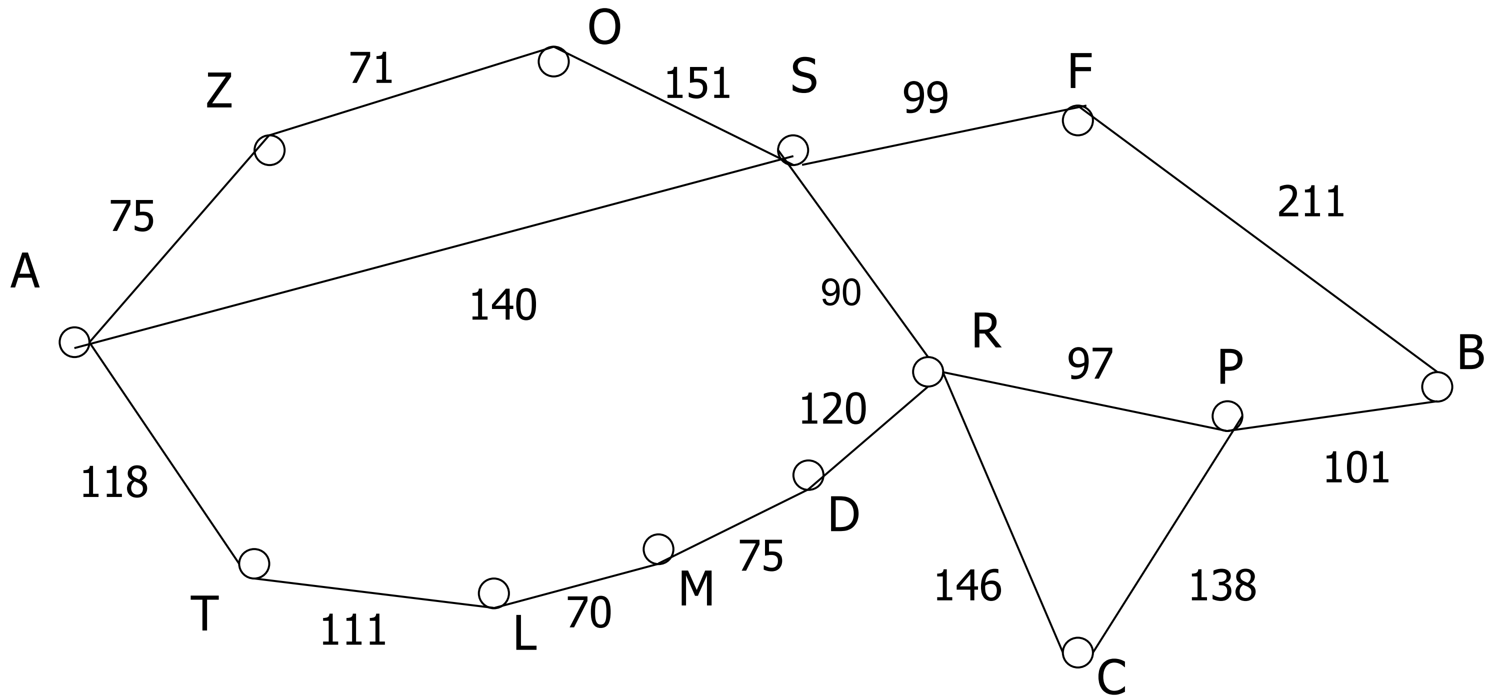
- **It's time to do searching: covering the basic methods really fast.**
  - Graph search
  - Agenda: a list of states that are waiting to be expand
  - Which state is chosen from the agenda defines the type of search & may have huge impact on effectiveness.

Uninformed Search

Breadth-First Search (BFS)

Treat agenda as a queue (FIFO)





$A \rightarrow Z_{A}, S_{A}, T_{A} \rightarrow S_{A}, T_{A}, O_{AZ} \rightarrow T_{A}, O_{AZ}, O_{AS}, F_{AS}, R_{AS} \rightarrow$
$O_{AZ}, O_{AS}, F_{AS}, R_{AS}, L_{AT} \rightarrow O_{AS}, F_{AS}, R_{AS}, L_{AT} \rightarrow F_{AS}, R_{AS}, L_{AT} \rightarrow$
$R_{AS}, L_{AT}, B_{ASF} \rightarrow L_{AT}, B_{ASF}, D_{ASR}, C_{ASR}, P_{ASR} \rightarrow B_{ASF}, D_{ASR}, C_{ASR}, P_{ASR}, M_{ATL}$
$\rightarrow$

Stop: B=goal, path: A → S → F → B, path-cost = 450

Breadth-First Search (BFS)

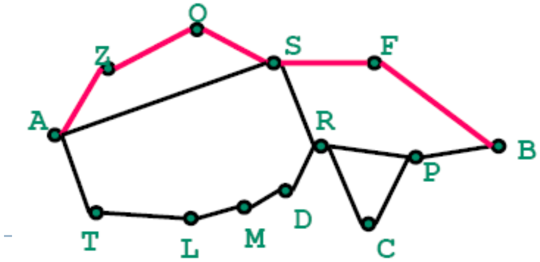
- Treat agenda as a queue (FIFO)
- Let's see what would happen if we did BFS on the graph G_1:
  - Start with initial state: A
  - Get A, expand it, add Z, S, T $\Rightarrow Z_A, S_A, T_A$
  - Get Z, expand it, add O $\Rightarrow S_A, T_A, O_{AZ}$
  - Get S, expand it, add O, F, R $\Rightarrow T_A, O_{AZ}, O_{AS}, F_{AS}, R_{AS}$
  - Get T, expand it, add L $\Rightarrow O_{AZ}, O_{AS}, F_{AS}, R_{AS}, L_{AT}$
  - Get O, expand it, nothing to add (already expanded) done twice!
  - Get F, expand it, add B $\Rightarrow R_{AS}, L_{AT}, B_{ASF}$
  - Get R, expand it, add D, C, P $\Rightarrow L_{AT}, B_{ASF}, D_{ASR}, C_{ASR}, P_{ASR}$
  - Get L, expand it, add M $\Rightarrow B_{ASF}, D_{ASR}, C_{ASR}, P_{ASR}, M_{ATL}$
  - Pop B, it is the goal state, and terminate.

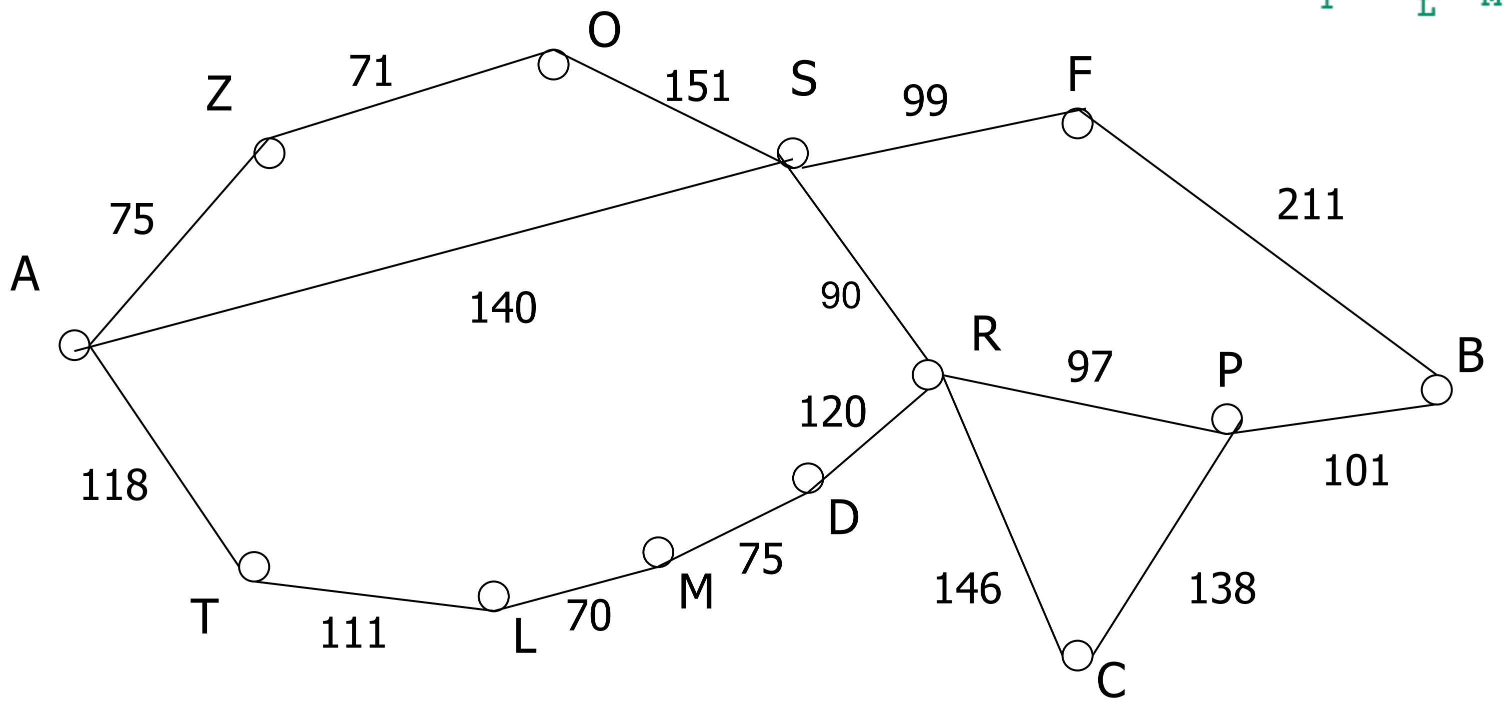
$\Rightarrow$ The RESULT is B_{ASF} with path: A, S, F, B

$\Rightarrow$ Path cost: 450

Depth-First Search (DFS)

Treat agenda as a stack (LIFO)





A → Z_A,S_A,T_A → O_{AZ}, S_A,T_A → S_{AZO},S_A,T_A → F_{AZOS}, R_{AZOS},S_A,T_A → B_{AZOSF}, R_{AZOS},S_A,T_A →

Stop: B=goal, path: A → Z → O → S → F → B, path-cost = 607

Depth-Limited Search (DLS)

- BFS finds min-step path but requires exponential space
- DFS is efficient in space, but has no path-length guarantee
  - DFS: can make a wrong choice and get stuck going down a very long (or even infinite) path when a different choice would lead to a solution near root of the search tree
- Solution: DFS-limited search
  - DFS with a predetermined depth limit l
  - Nodes at depth l are treated as if they have no successors.
  - Problem: the shallowest goal is beyond the depth limit
  - Depth limit can be based on knowledge of the problem

DLS Algorithm

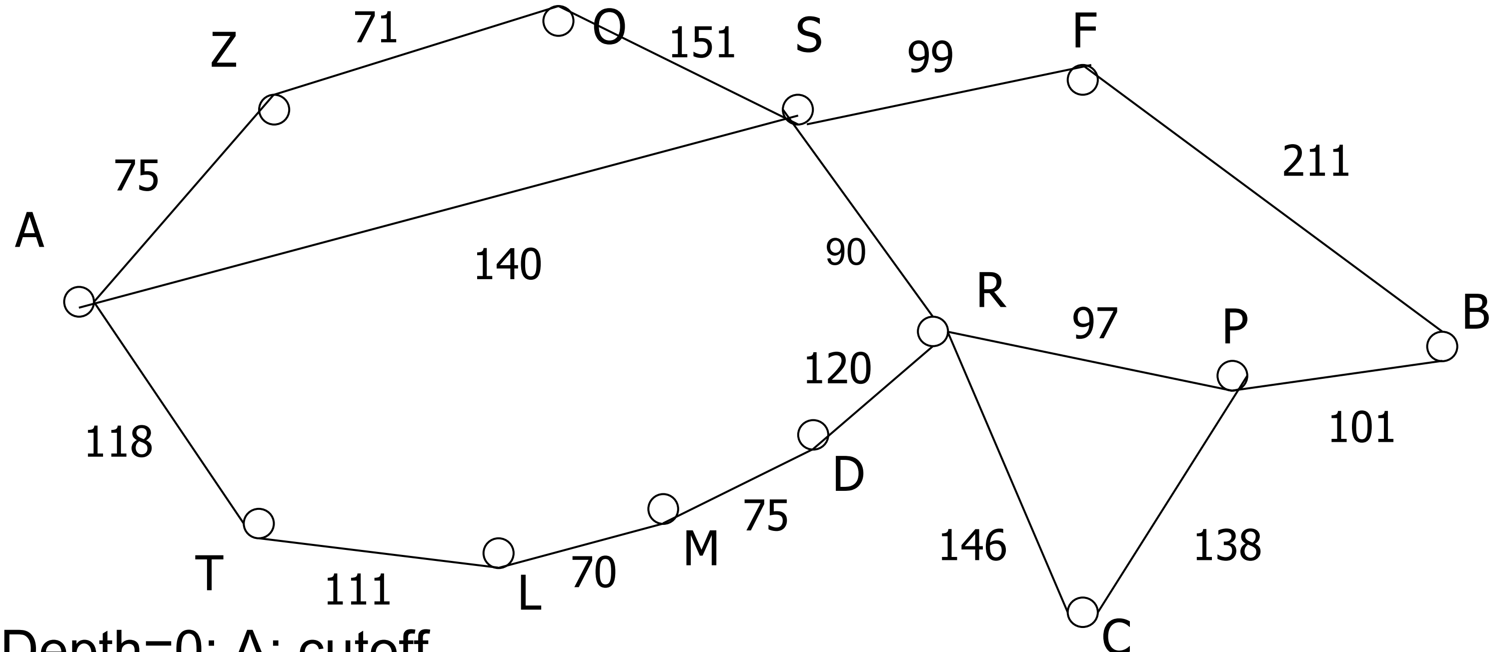
```
Function DLS (problem, limit) returns solution/ cutoff/
  failure
→ rec_DLS(make_node(init_state),problem,limit)

Function Rec_DLS (node,problem, limit) returns solution/
  cutoff/ failure
  if isGoal(node) then → solution(node)
  else if limit=0 then → cutoff
  else
      cutoff_occured ← false
      for each action in problem.Actions(node.State) do
         child ← CHILD-Node(problem, node, action)
         result ← rec_DLS(child,problem,limit-1)
         if result=cutoff then cutoff_occured ← true
         else if result≠failure then → result
      if cutoff_occured then → cutoff
      else → failure
```

Iterative Deepening Search (IDS)

- IDS: perform a sequence of DFS searches with increasing depth-cutoff until goal is found
- Assumption: most of the nodes are in the bottom level so it does not matter much that upper levels are generated multiple times.

```
Function Iterative-Deepening_Search(problem) returns
  solution/ failure
  for depth = 0 to ∞ do
       result ← DLS(problem, depth)
       if result ≠ cutoff then → result
```

IDS



Depth=0: A: cutoff

Depth=1: A → Z_A, S_A, T_A → Z_A: cutoff, S_A: cutoff, T_A: cutoff

Depth=2: A → Z_A, S_A, T_A → O_{AZ}, S_A, T_A → O_{AZ} : cutoff → F_{AS}, R_{AS}, T_A → F_{AS} : cutoff → R_{AS} : cutoff → L_{AT} → L_{AT} : cutoff

Depth=3: A → Z_A, S_A, T_A → O_{AZ}, S_A, T_A → S_{AZO}, S_A, T_A → S_{AZO}: cutoff → F_{AS}, R_{AS}, T_A → B_{ASF}, R_{AS}, T_A → B_{ASF}

Stop: B=goal, path: A → S → F → B, path-cost = 450

Uniform Cost Search (UCS)

- BFS & IDS find path with fewest steps
- If steps ≠ cost, this is not relevant (to optimal solution)
- How can we find the shortest path (measured by sum of distances along path)?
- UCS:
  - Nodes in agenda keep track of total path length from start to that node
  - Agenda kept in priority queue ordered by path length
  - Get shortest path in queue
- Explores paths in contours of total path length; finds optimal path

Uniform Cost Search (UCS)

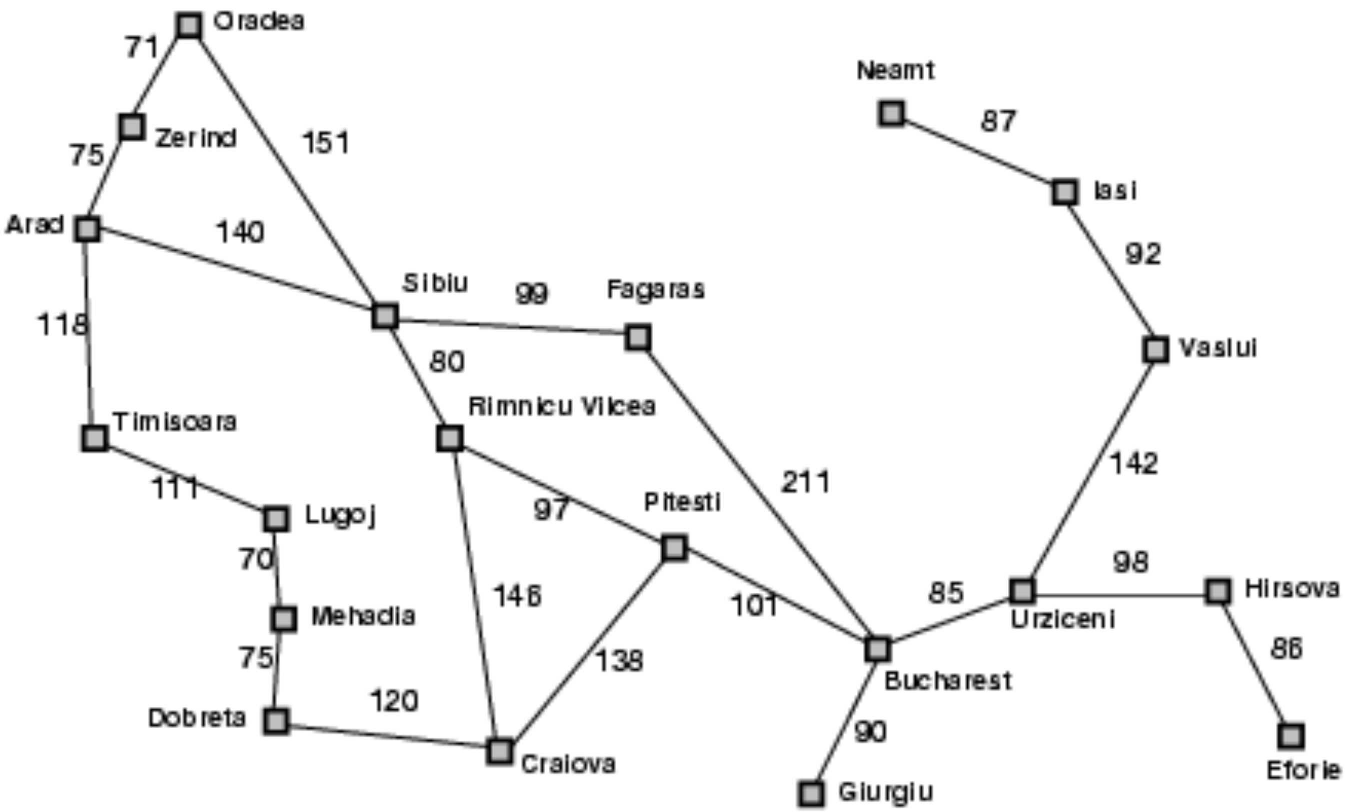
- Let's see what would happen if we did UCS on the graph G_1:
  - Start with start state: A
  - Remove A, add Z with cost 75, add T with cost 118, add S with cost 140 $\Rightarrow Z_{A-75}, T_{A-118}, S_{A-140}$
  - Remove Z (the shortest path), add its children: $O_{146} \Rightarrow T_{A-118}, S_{A-140}, O_{AZ-146}$
  - Remove T, add $L_{229} \Rightarrow S_{A-140}, O_{AZ-146}, L_{AT-229}$
  - Remove S, add O_{291}, F_{239}, $R_{230} \Rightarrow O_{AZ-146}, L_{AT-229}, R_{AS-230}, F_{AS-239}, O_{AS-291}$
  - Remove O, add nothing (already expanded)
  - Remove L, add $M_{299} \Rightarrow R_{AS-230}, F_{AS-239}, O_{AS-291}, M_{ATL-299}$
  - etc ...
- It seems clear that in the process of removing nodes from the agenda, we're enumerating all the paths in the graph in order of their length from the start state.

Informed Search

Best-first search

- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability“
  - Expand most desirable unexpanded node
- Implementation:
- Order the nodes in fringe in decreasing order of desirability
- Special cases:
  - greedy best-first search
  - A^* search

Romania with step costs in km



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

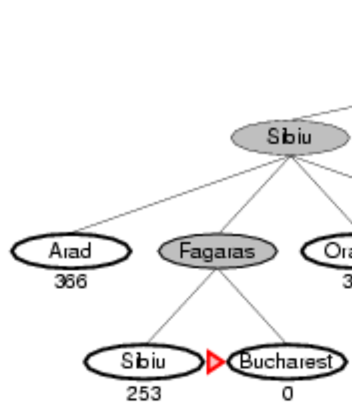
Greedy best-first search

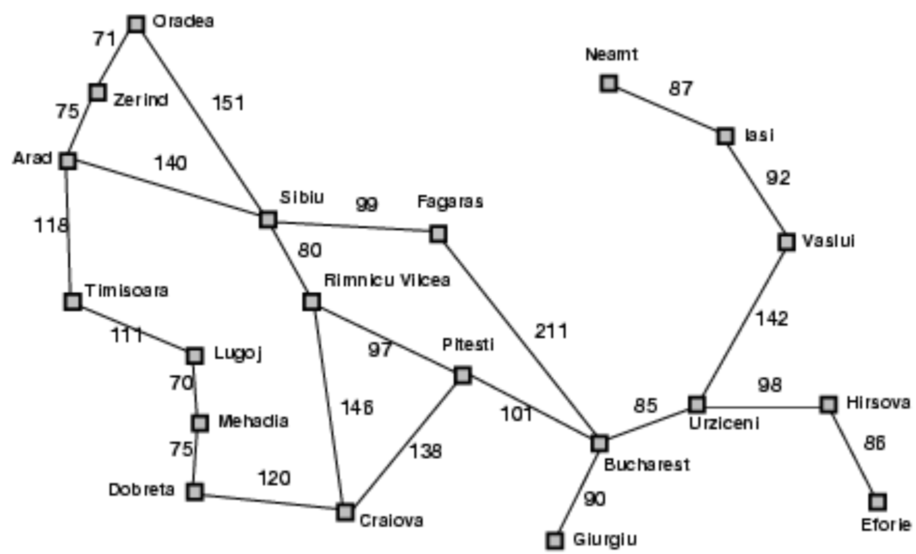
- Evaluation function $f(n) = h(n)$ (heuristic) = estimate of cost from n to *goal*

- e.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest

- Greedy best-first search expands the node that appears to be closest to goal

Greedy best-first search example





Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

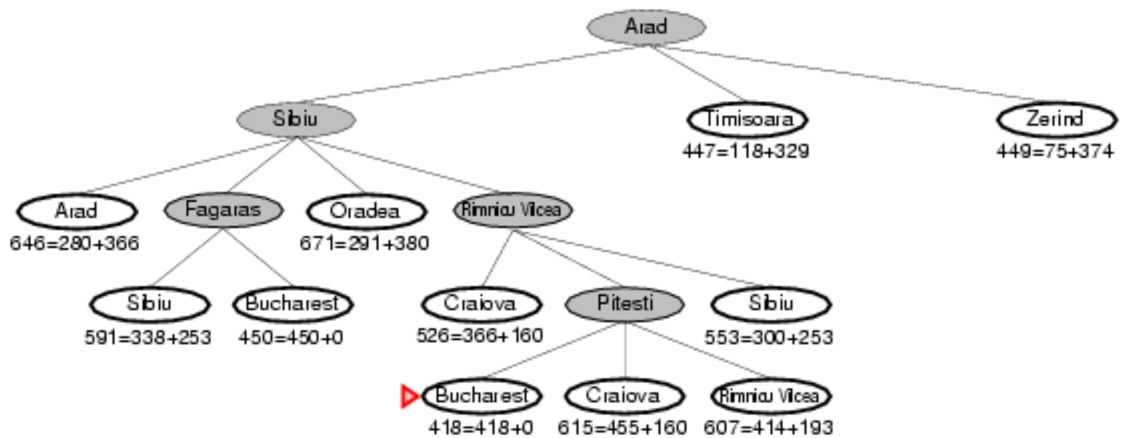
A^* search

- Idea: avoid expanding paths that are already expensive

- Evaluation function $f(n) = g(n) + h(n)$

  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost from n to goal
  - $f(n)$ = estimated total cost of path through n to goal

A* search example

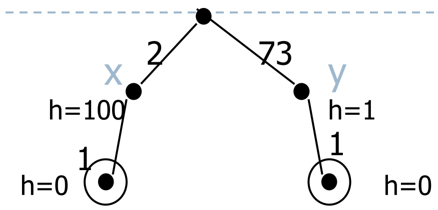


Admissible heuristics

- A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.
  An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)
- Theorem: If $h(n)$ is admissible, A* using `TREE-SEARCH` is optimal

Admissibility

- What must be true about h for A* to find optimal path?
- A* finds optimal path if h is admissable; h is admissible when it never overestimates.
- In this example, h is not admissible.
- In route finding problems, straight-line distance to goal is admissible heuristic.



g(X)+h(X)=2+100=102

G(Y)+h(Y)=73+1=74

Optimal path is not found!

Because we choose Y, rather than X which is in the optimal path.

THANK YOU

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

Modul 3: Beyond Classical Search

Classical vs Local Search

Inteligensi Buatan
(*Artificial Intelligence*)

Classical Search

transition state → kalau ada aksi, selama blm sampe goal, dia pindah state

Problem

semua state sudah diketahui

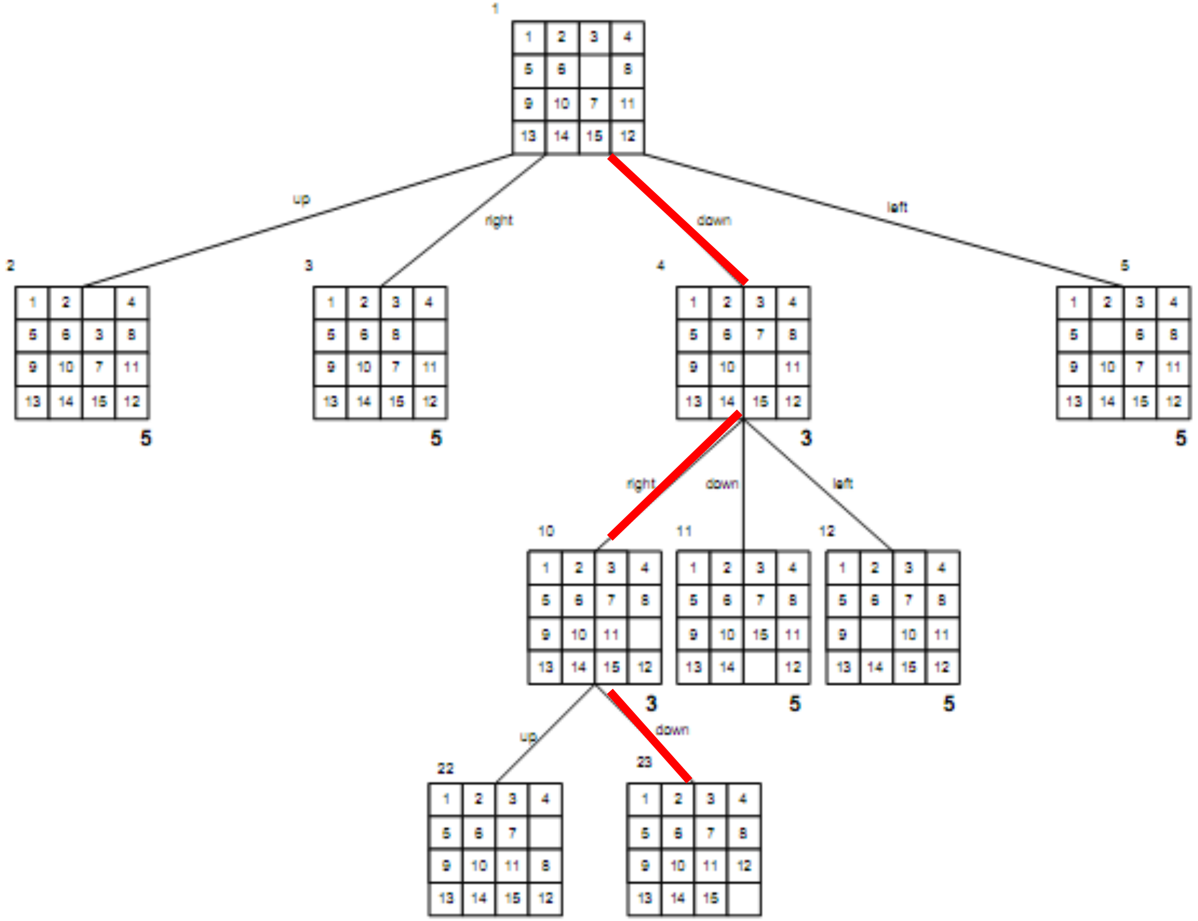
Known environment (observable, deterministic)
States (initial), operators, path cost, goal test

Explore search space systematically

Solution (path to goal)

Solution: sequence of actions

N-Puzzle Problem



Solution: down → right → down

Path to Goal as Solution

- In many problems, the path to goal is irrelevant.
- Solution: X=(8,6,4,2,7,5,3,1)
- Q1=8 → Q2=6 → … → Q8=1

path relevant : path jadi bagian dari solusi
contoh : rubiks, water jug.

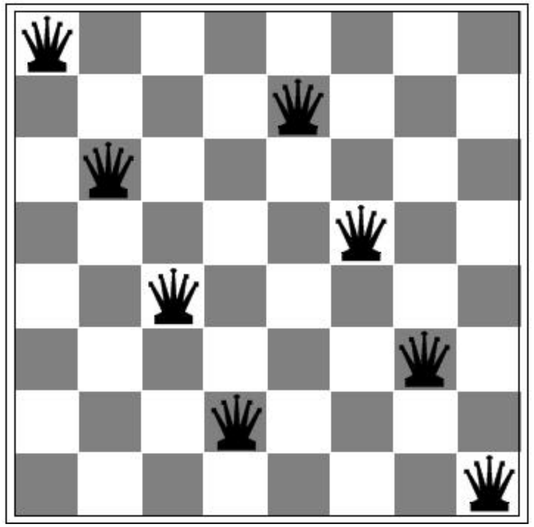
path irrelevant : path dari initial state ke goal state tidak menjadi bagian dari solusi. contoh : N-queens, graph-coloring, knapsack, cryptarithmetic
(ganti huruf mana aja dulu doesn't matter)

variable : setiap karakter
↳ nilai yg diassign itu angka.

N-Queen Problem

objective : menempatkan queen ke bidak agar tidak bisa saling makan.

variabel : queen1, queen2, ... queen-N.
↳ nilai yg di assign ke variabel itu posisinya.



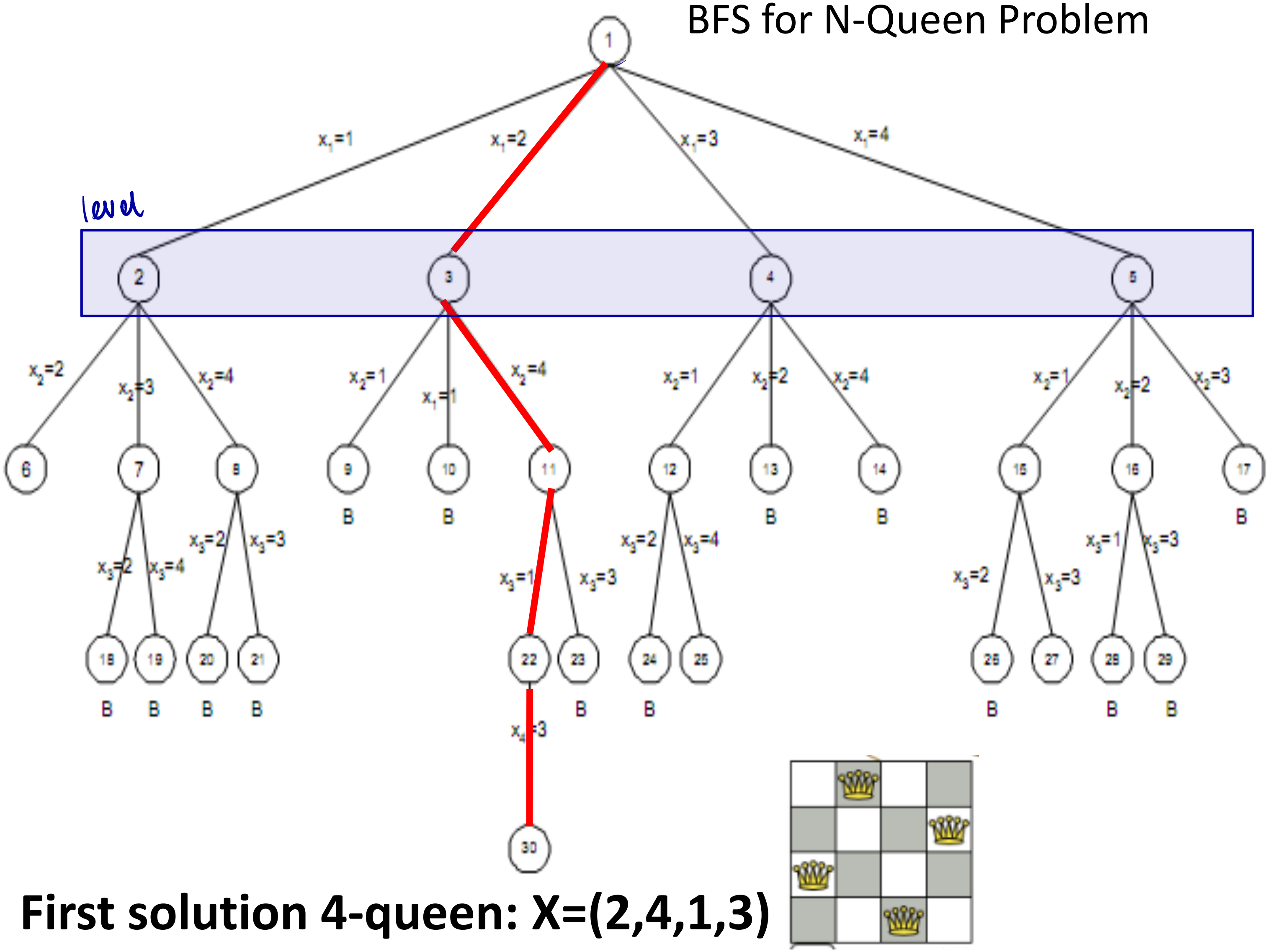
n-queens are on the board, none attacked. one queen per column

Path to Goal as Solution (2)

- For example, in n-queen
  - BFS solution: x1=2 → x2=4 → x3=1 → x4=3
- What matters is the final configuration of queens, not the order in which they are added.

kalau gagal dia backtrack.

BFS for N-Queen Problem



First solution 4-queen: X=(2,4,1,3)

Local Search

Jika path ke goal nya TIDAK PENTING (path irrelevant)

pada initial state, complete configuration
↳ tiap variable sudah diisi nilai secara random, tiap statenya jd perbaika nilai var nya smpe dh ga ada yg melanggar constraints.

pasti ada nilai dlm state.

- If path to goal does not matter, we might consider different class of algorithms.
- Local search: complete state formulation
  - State: "complete" configurations
  - Keep single current state, not paths.
  - Action: move only to neighbors of current state.
  - No path cost → state value: value according to objective function or heuristic cost function
  - No goal test → maximum state value

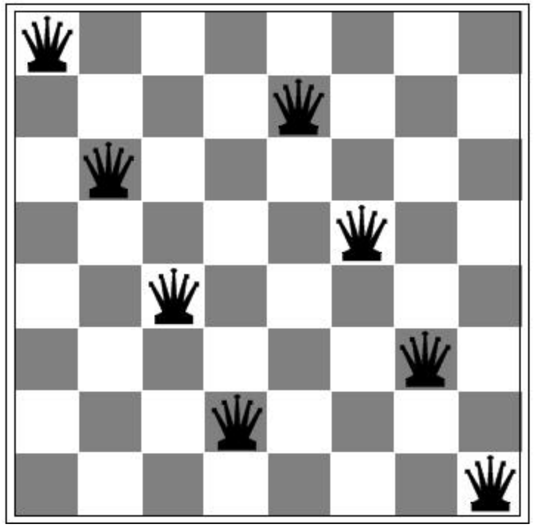
    berhenti pas nilai statenya sudah maksimum
  - Solution: final state.

Problem
States (initial), neighbors
Path to goal is irrelevant

↓ Find the best state by moving to neighbors of current state

Solution
Final state/configuration

N-Queen: Classical vs Local Search



Classical search	Local search
State: any arrangement of **0 to n** queens on the board (incremental)	State: any arrangement of **n** queens on the board (complete)
Initial state: no queens on the board. Goal test: n-queens are on the board, none attacked	Initial state: a [random] state. Goal test: state value = global maximum
Action: add a queen to any empty square	Action: move a single queen to another square (neighbor)
Solution: path to goal	Solution: final state

Init state: complete configuration, tp mungkin masih melanggar constraints.

nilai maks

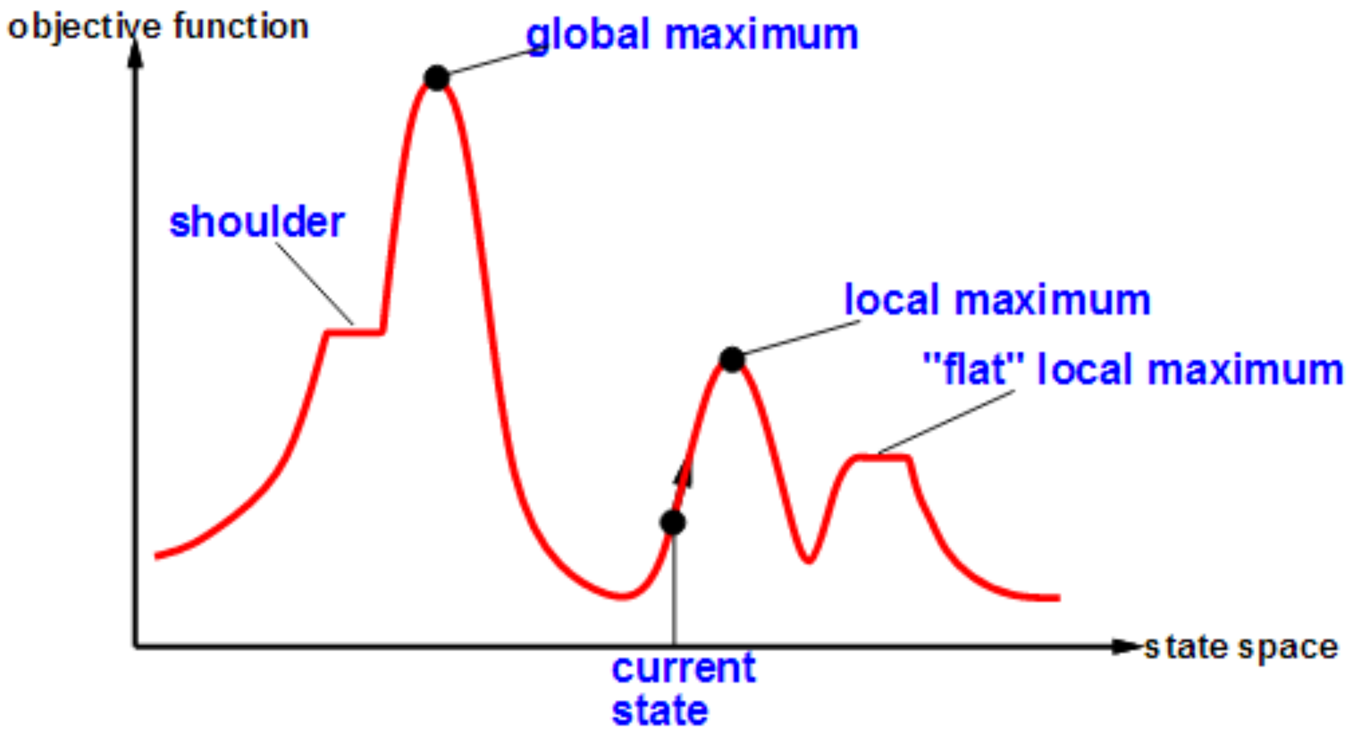
the worst objective function untuk kasus 8 queen : −28.

$\frac{-56}{2}$

$8 \times (-7)$ 1 ratu bisa serang 7 queen lain

$= -56$, dibagi 2 karena kasus 1 attack 2 = 2 attack 1

Local Search Explore State-space Landscape



A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. (Russel & Norvig, 2010)

- A landscape has "location" (defined by state) and "elevation" (value of objective or heuristic cost value).
- Local search aims to find global optimum.
- Problem: depending on initial state, can get stuck in local optimum.

Summary: Local Search

Keep single current state
semua var dh ada nilainya.
State: "complete" configurations
Complete formulation

Find the best state (global optimum)
Action: move to neighbors
untuk mendapat state dgn nilainya paling optimal

Path to goal is irrelevant
Solution: final state

Next:
- State
- Successor
- Neighbors

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

Modul 3: Beyond Classical Search

State: Value, Successor, and Neighbor

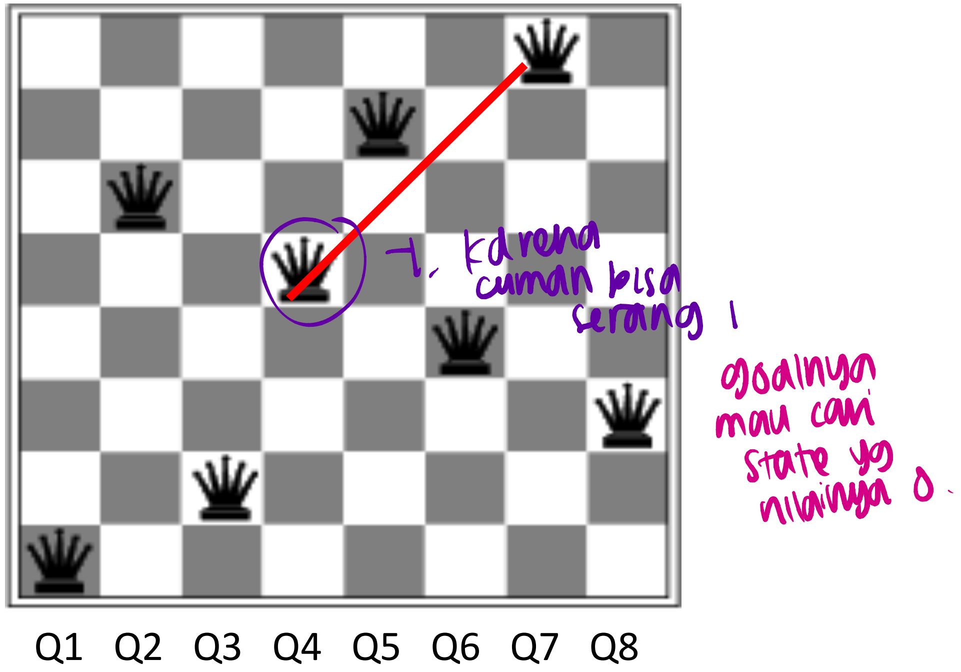
Inteligensi Buatan
(*Artificial Intelligence*)

Local Search: State

Keep single current state
State: "complete" configurations
(one queen per column)
Solution: final state

Find the best state (global optimum)
Action: move to neighbors

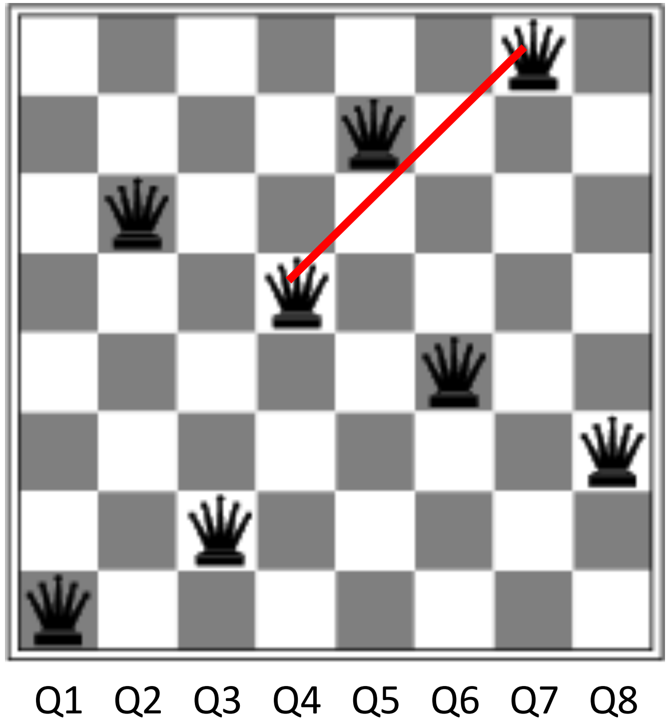
State for 8-queens problem



Each state has state value based on heuristic cost function.

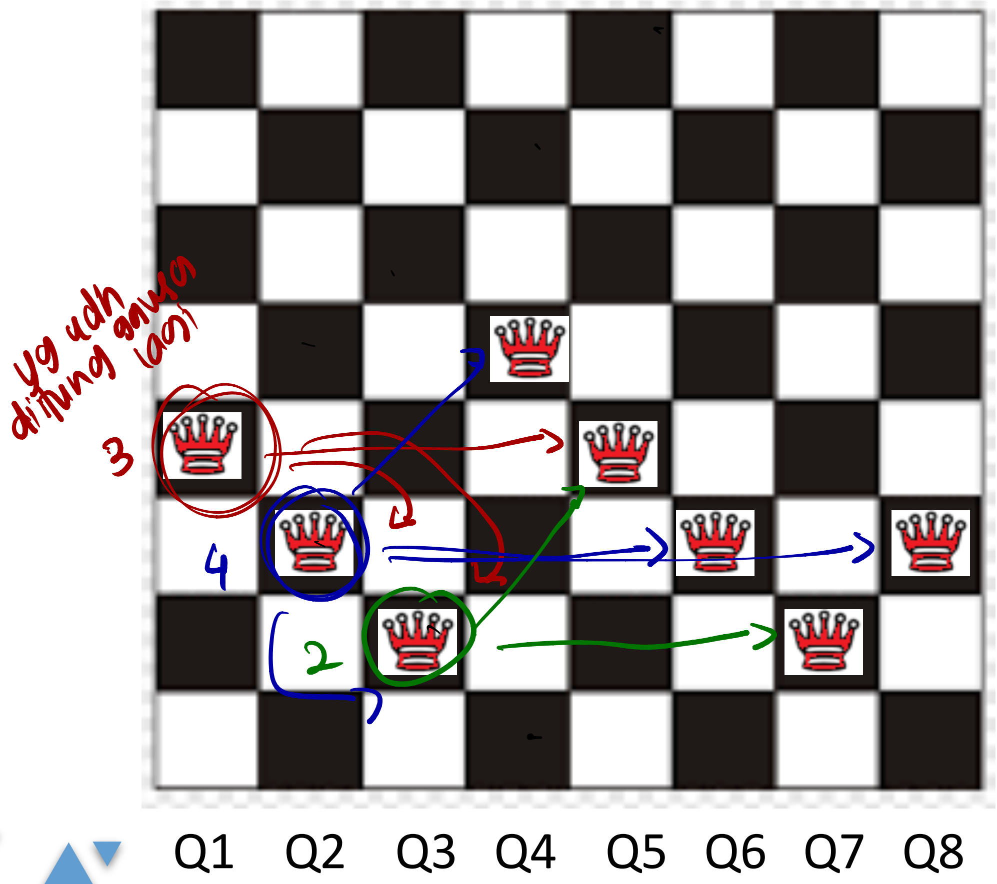
State Value (h) for 8-queens problem

- *h* = - number of pairs of queens that are attacking each other, either directly or indirectly.
- h=-1: only 1 pair of queens (Q4 attacks Q7)
- Optimum solution has global maximum, h=0.



h = *-1* for the above state

Exercise: Determine h



List pairs of queens that are attacking each other, either directly or indirectly.

h = - number of pairs of queens that are attacking each other, either directly or indirectly.

ada 56 tetangga yg mungkin
(1 ratu bs punya 7 kondisi lain)

Local Search: Action

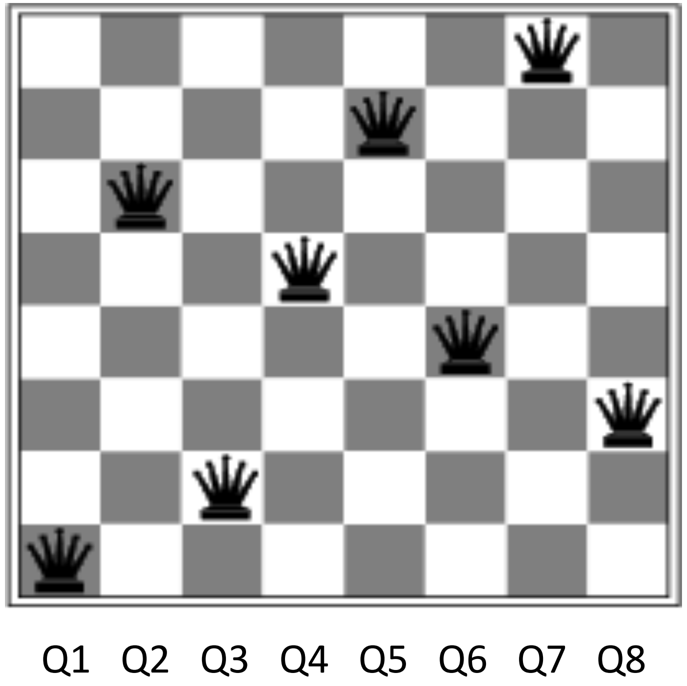
Keep single current state
State: "complete" configurations
(one queen per column)
Solution: final state

Find the best state (global optimum)
Action: move to neighbor

What is neighbor of a state ?

mengembalikan semua state yg mungkin dari pindah 1 ratu ke kolom yg sama, baris beda

Neighbor and Successors for 8-queens



Neighbor: highest-valued successor

- Successor function returns all possible states generated by moving a single queen to another square in the same column.

semua successor jd neighbor state

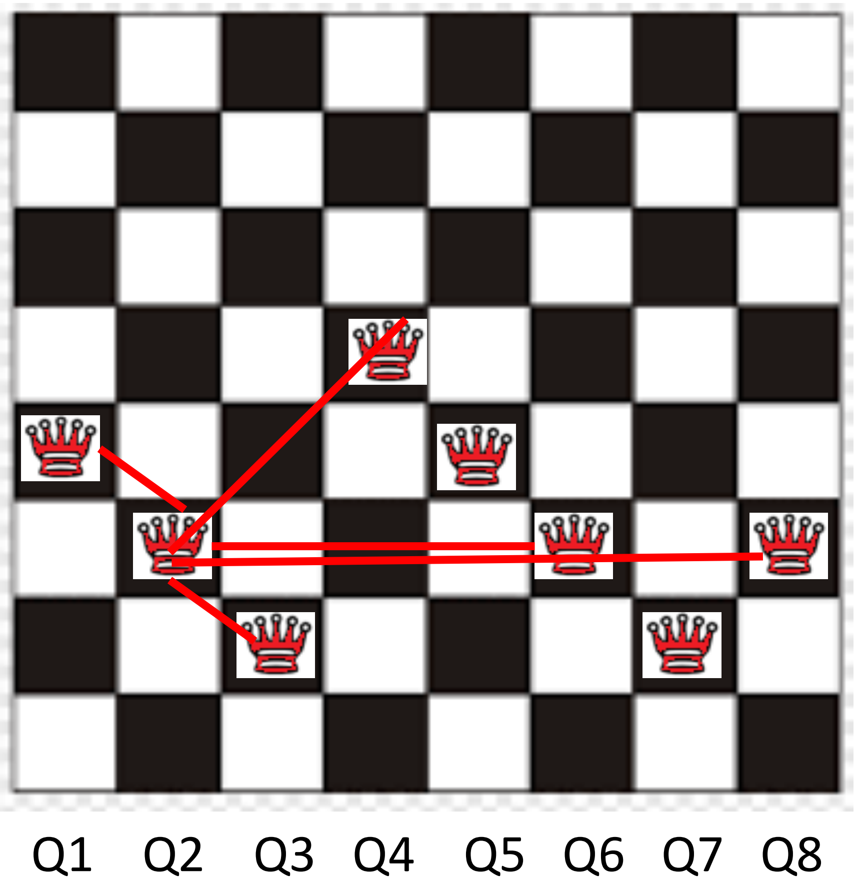
- Each state has 8*7 = 56 successors
- Choose randomly among the set of best successors if there is more than one.

Neighbor: random successor

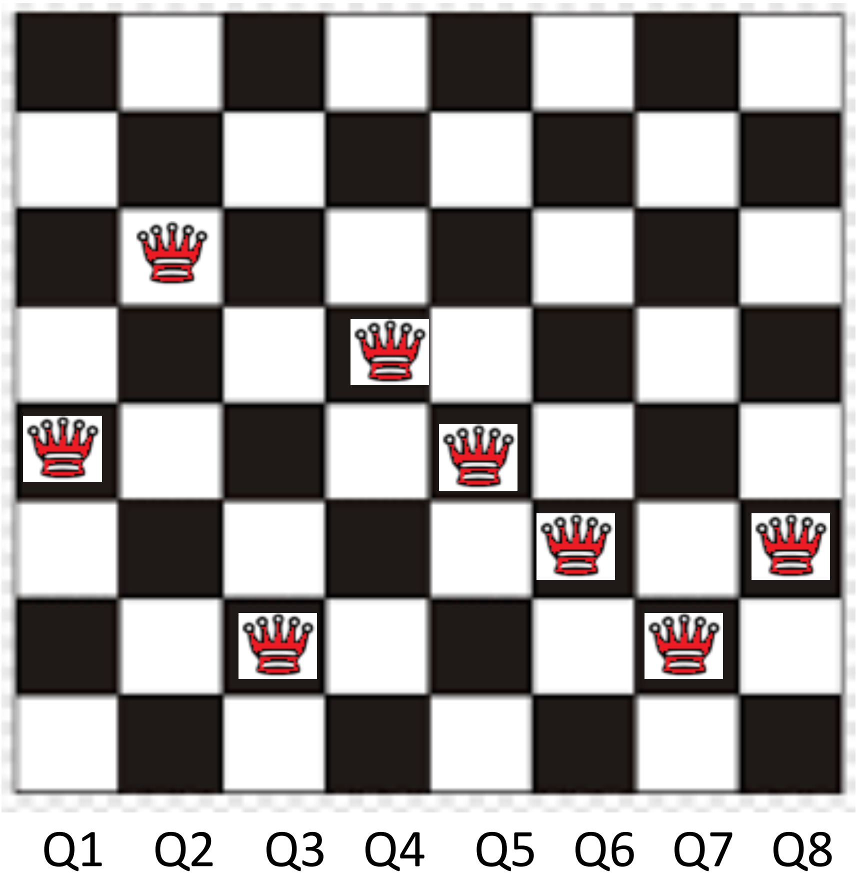
salah 1 dr successor aja yg jd neighbor state.

- Successor function returns a random state generated by moving a random single queen to another square in the same column.

Neighbor: Random Successor



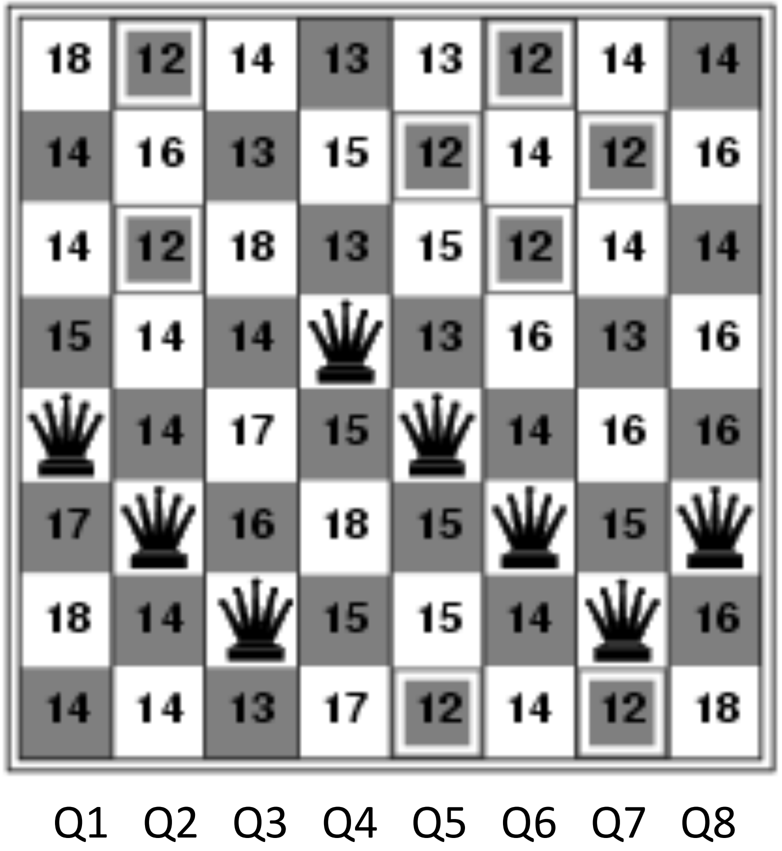
Column: Q2
Row: 3➔6



Neighbor: Highest-valued Successor

Each number indicates h if we move a queen in its corresponding column

State value dari setiap successor belum diberi tanda negative (cost)



Summary: State, Neighbor

Keep single current state
State: "complete" configurations
(one queen per column)
Solution: final state

Find the best state (global optimum)
Action: move to neighbors

Next:

- Hill climbing Search
- Simulated annealing
- Genetic algorithm

01

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

Modul 3: Beyond Classical Search

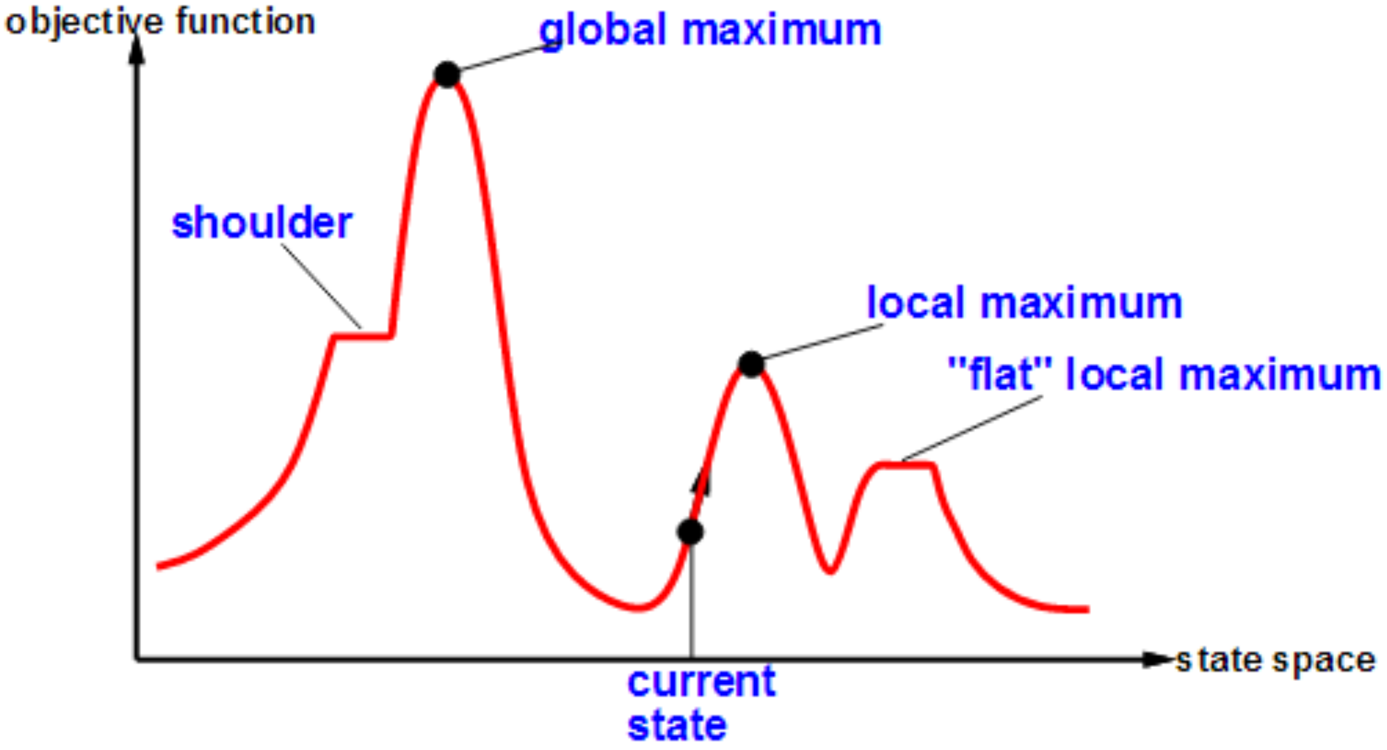
Hill-climbing Search

Inteligensi Buatan
(*Artificial Intelligence*)

→ tetangga yg lebih baik ni lainya

Hill-climbing Search

"Like climbing Everest in thick fog with amnesia"



Starting from a randomly generated initial state

Loop that continually moves in the direction of increasing value (objective) or decreasing value (cost)

Terminates when it reaches a "peak" where no neighbor has a higher value

Hill-climbing Search: Steepest Ascent

(Russel & Norvig, 2010)

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

current ← MAKE-NODE(*problem*.INITIAL-STATE)

loop do

neighbor ← a highest-valued successor of *current*

if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE

current ← *neighbor*

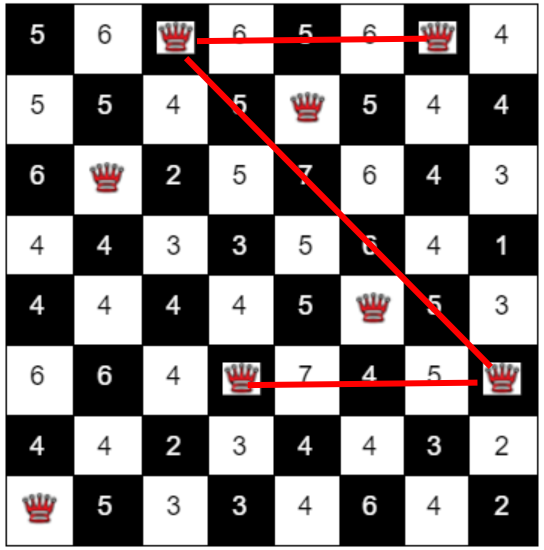
Starting from a randomly generated initial state

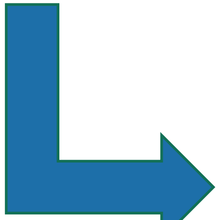
Loop that continually moves in the direction of increasing value (objective) or decreasing value (cost)

Terminates when it reaches a **"peak"** including **"flat"** where no neighbor has a higher value

Hill-climbing: Illustration

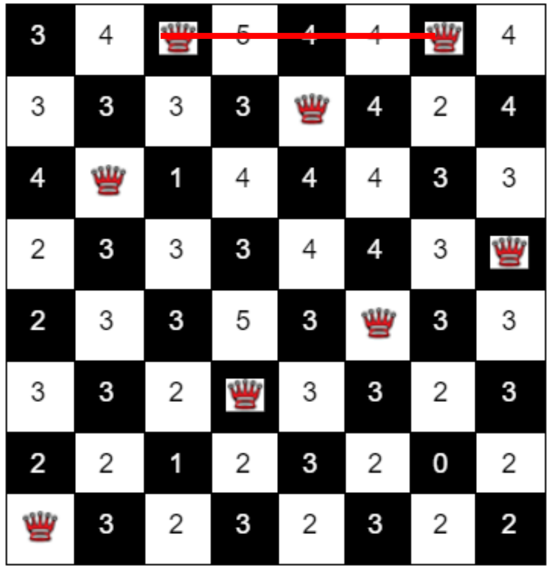
h=-3



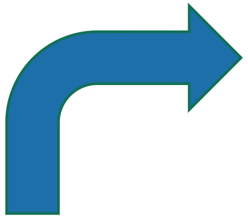


Q8: 3→5

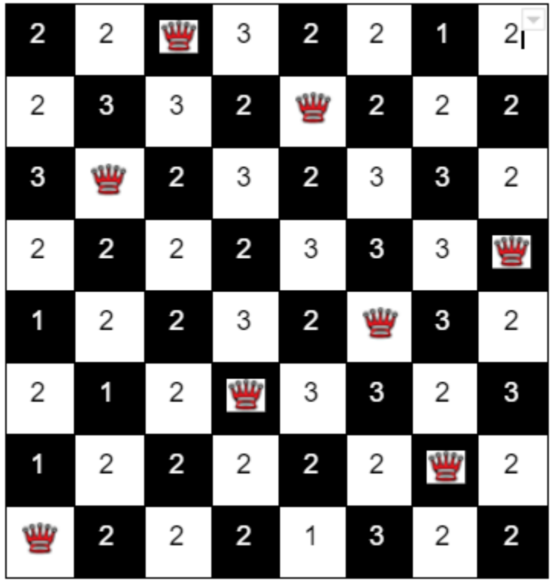
h=-1 (Q8: 3 →5)



Q7: 8→2



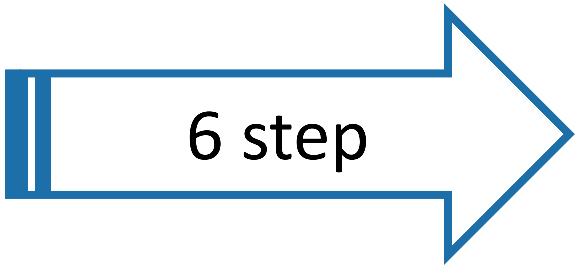
h=0 (Q7: 8 →2)



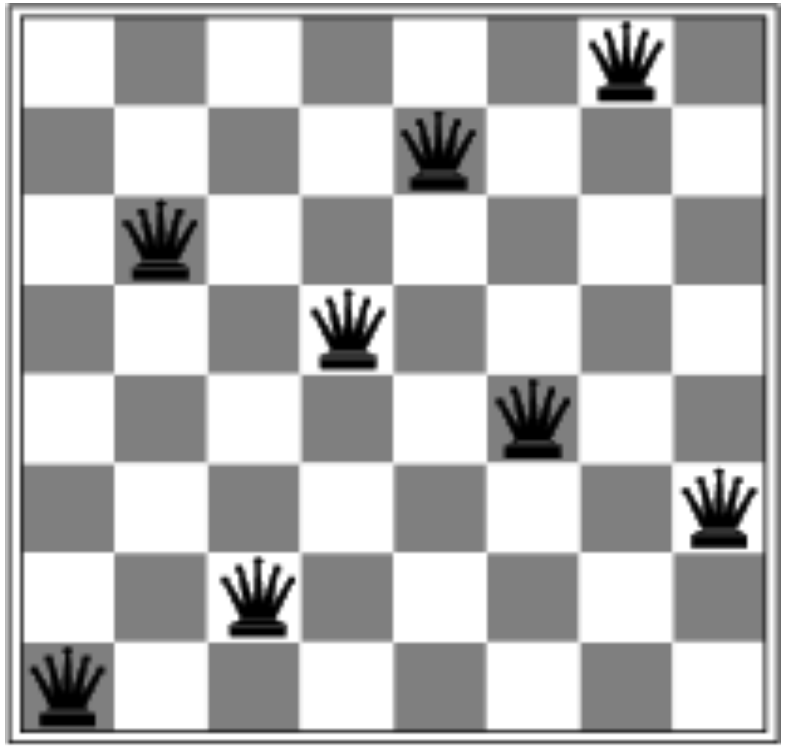
stop in global optimum (solution)

Hill-climbing: Stuck In Local Optimum

Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8



Step 1: h=-17 → h=-12
Step 2: h=-12 → h=-7
Step 3: h=-7 → h=-4
Step 4: h=-4 → h=-3
Step 5: h=-3 → h=-1
Step 6: h=-1 stop



Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8

Hill-climbing for 8-Queen Problem

```
function HILL-CLIMBING(problem) returns a state that is a local maximum

  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
     neighbor ← a highest-valued successor of current
     if neighbor.VALUE ≤ current.VALUE then return current.STATE
     current ← neighbor
```

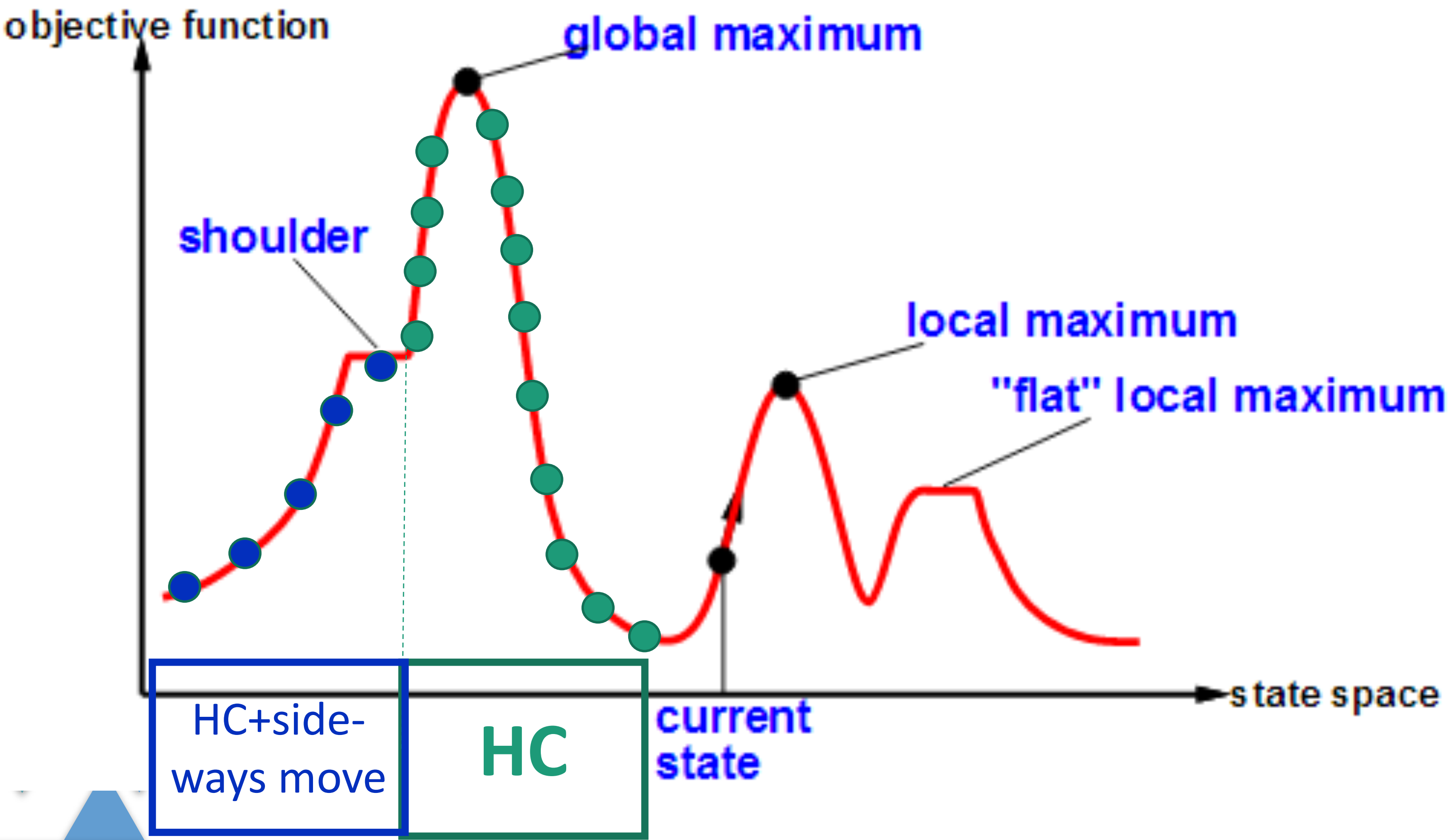
State space:
$8^8 \approx 16.8$ million states

Average case: works quickly
when **success**: avg 4 steps
when **stuck**: avg 3 steps

Best case:
initial state = goal state
Prob: 92 / 8^8 = 0.00054%

Get **stuck** 86%,
solving only 14% of problem instances

Hill-climbing Search: Final State



Success: global maximum

Stuck: 1) local maximum, 2) flat local maximum, 3) shoulder

Shoulder: still possible to global max.
Variant HC: + **sideways move** with limit on number of consecutive ways

bolehin tetangga nilainya sama, jd gn langsung berhenti

Variant 1: Hill-climbing with Sideways Move

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
  neighbor ← a highest-valued successor of *current*
  if neighbor.VALUE $\boxed{<}$ current.VALUE **then return** *current*.STATE
  current ← *neighbor*

Terminates when it reaches a "peak" ~~including "flat"~~

Increase success for 8-queens problem.
Limit=100:
14% ➔ 94% success

Works **slower**:
when **success**: avg 4 ➔ 21 steps
when **stuck**: avg 3 ➔ 64 steps

cara kerjanya sama persis dgn steepest tp diulang2.

Variant 2: Random Restart Hill-climbing

QOTD

If at first you don't succeed, try, try again. It conducts **a series of hill climbing** searches (random initial states, until a goal is found)

Expected nb of restarts = $1/p$ → p=0.14: 7 restart

Expected nb of steps = $s+f(1-p)/p$ → p=0.14, s=4, f=3: 22 steps.

Very effective for n-queens problem: solving 3 million queens in under a minute (Luby et al., 1993) hanya membangkitkan 1 successor secara random.

Variant 3: Stochastic Hill-climbing

cek nilai tetangga lebih bagus.
kalau worse (≤), statenya tetap selama iterasinya belum sampai nmax.

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

current ← MAKE-NODE(*problem*.INITIAL-STATE)
repeat nmax times
neighbor ← a **random** successor of *current*
if neighbor.VALUE **>** current.VALUE **then** *current* ← *neighbor*

Terminates when it reaches **nmax iteration**

Generating a successor randomly (not all successor)

Move to neighbor if it is better than current state.

Works **slower** (more steps)

Summary: Hill-climbing

continually moves in the direction of increasing value (objective) or decreasing value (cost)

Depending on initial state, can get stuck in local maxima.

Variant: steepest ascent HC, HC with sideways move, stochastic HC, random restart HC

Next:

- Simulated annealing

THANK YOU

Modul 3: Beyond Classical Search

Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

Simulated Annealing

Inteligensi Buatan
(*Artificial Intelligence*)

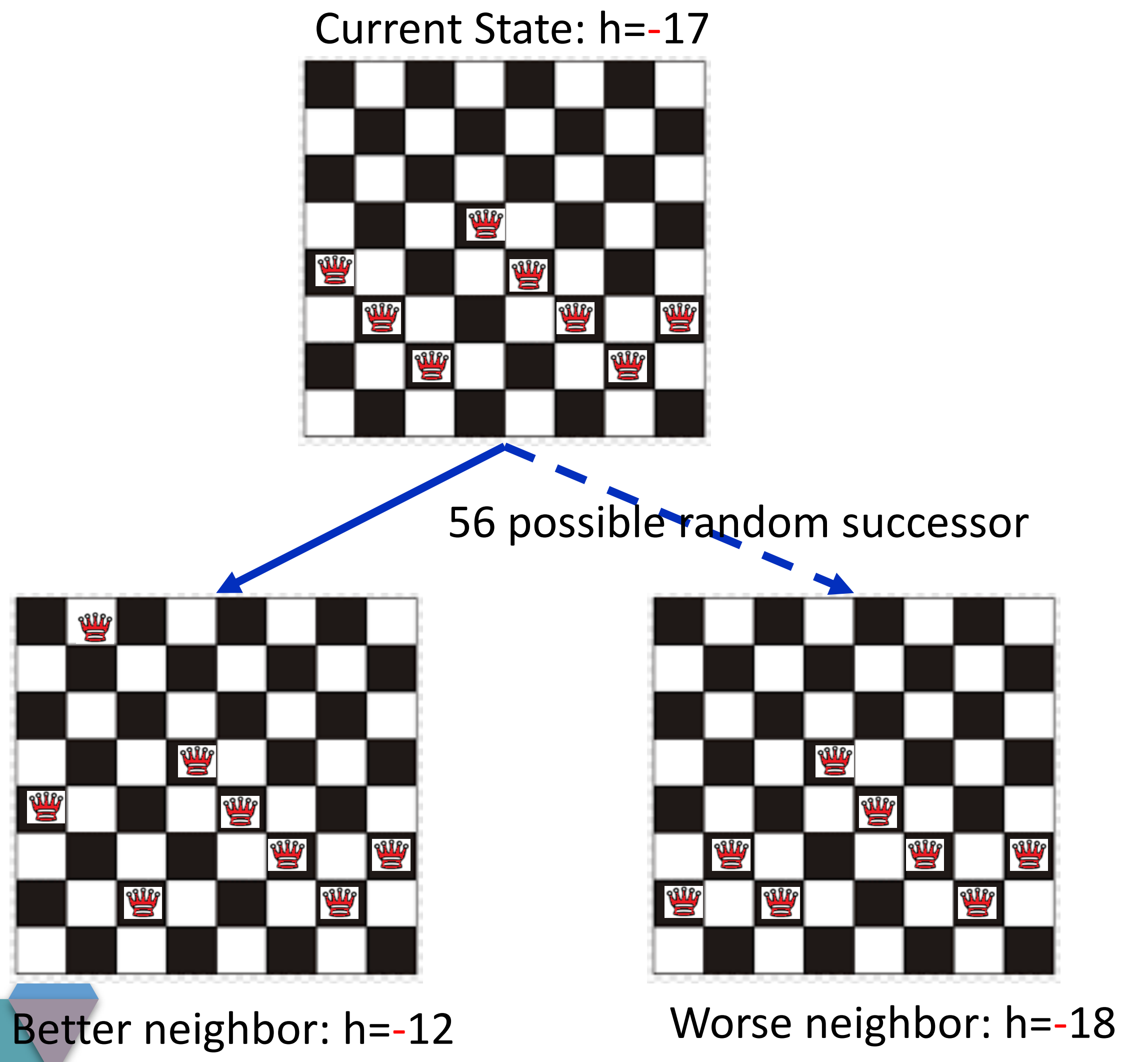
bolehin pindah ke state tetangga yg lebih buruk dgn probabilitas tertentu yg semakin lama semakin menurun.

SA: Combining Completeness and Efficiency

tetangga cuman 1

- Simulated annealing combines hill climbing (efficient) with a random walk (complete)
- Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency
- Simulated annealing is a version of stochastic hill climbing where some downhill moves are allowed.

Neighbor: One Random Successor of Current



- Stochastic hill-climbing:
  - Only move to better neighbor, and skip worse neighbor.
- Simulated annealing:
  - move to better neighbor, and allow move to worse neighbor that has probability

$$e^{\Delta E / T}$$

ΔE → nilai / utilitas state tetangga terpilih – nilai / utilitas state sekarang.

T → temperatur

$T \uparrow$, nilai e (prob.) jd makin besar. karena ΔE negatif

hanya digunakan ketika state tetangga yg dibangkitkan lebih buruk.

Simulated Annealing

ΔE selalu negatif karena dia lebih buruk.

https://id.m.wikipedia.org/wiki/Berkas:Annealing_a_silver_strip.JPG

Annealing: heat (metal or glass) and allow it to cool slowly, in order to remove internal stresses and toughen it.

- T is the "temperature" of annealing that gradually decreases.
- ΔE=neighbor.value-current.value
- If better neighbor ($\Delta E>0$): move to neighbor (stochastic HC)

  kalo lebih > 0 pasti pindah
- If worse neighbor ($\Delta E<0$): probability move $e^{\Delta E/T}$
  - When T is high (e.g. $\Delta E=-5$, $T=100$, prob=0.95), there is a lot of random motion ➔ random walk
  - When T approaches 0 (e.g. $\Delta E=-5$, $T=1$, prob=0.007), randomness is decreased ➔ stochastic hill climbing.

Simulated Annealing (Russel & Norvig, 2010)

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

beres artinya (selesai).

Terminates T=0

T: temperature as a function of time t

Move to better neighbor (stochastic HC)

allowing some "bad" moves, depends on probability

pindah/ enggak tergantung sama probability

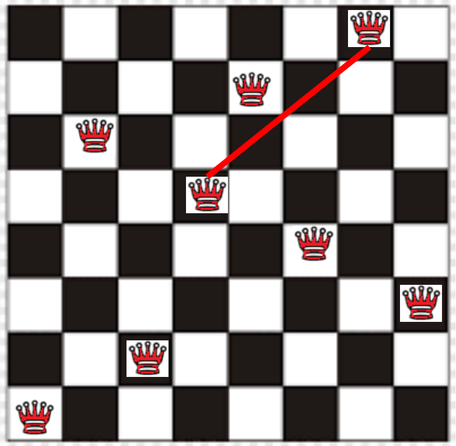
pendekatan.

set threshold probability.

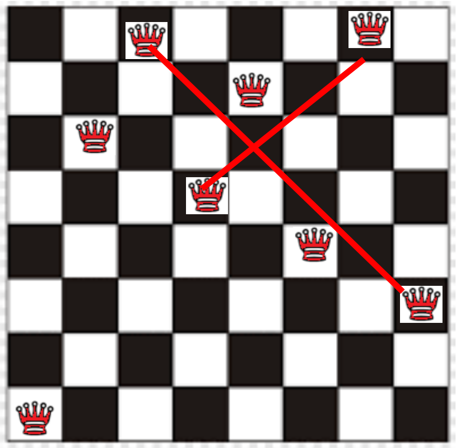
bangkitkan nilai random dari 0-1.

Move to Worse State

Current State: h=-1



Column: C3
Row: 2 → 8



$current \leftarrow next$ only with probability $e^{\Delta E/T}$

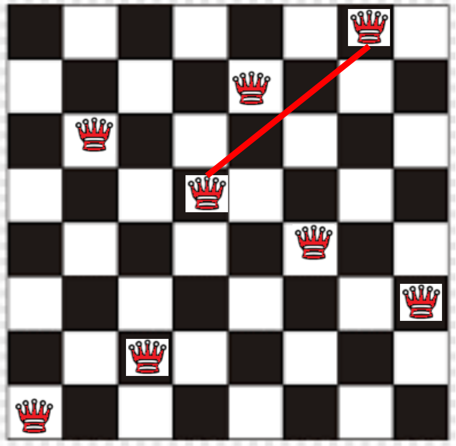
<u>Compare to static value</u>:
move probability > 0.5

<u>Compare to random value</u>:
move probability > random(0,1)

Next: h=-2, T=10
$\Delta E<0$: prob=$e^{(-2-(-1))/10}=e^{-0.1}$ =0.9

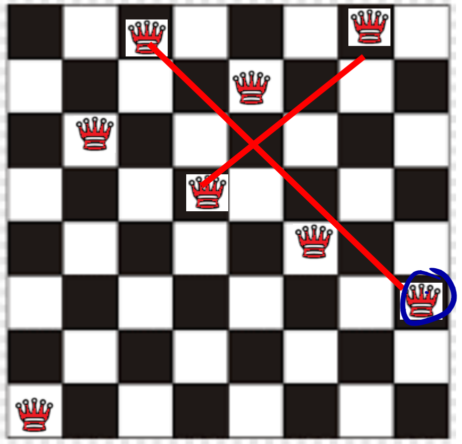
Simulated Annealing: Illustration

Current State: h=-1



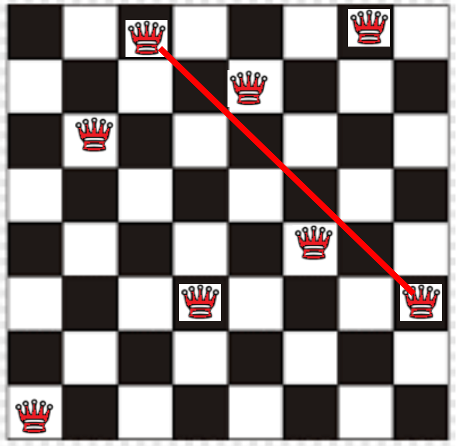
Column: C3
Row: 2➔8

Next: h=-2, T=10
Prob=$e^{(-2-(-1))/10}=e^{-0.1}=0.9$
Current ← next



Column: C4
Row: 5➔3

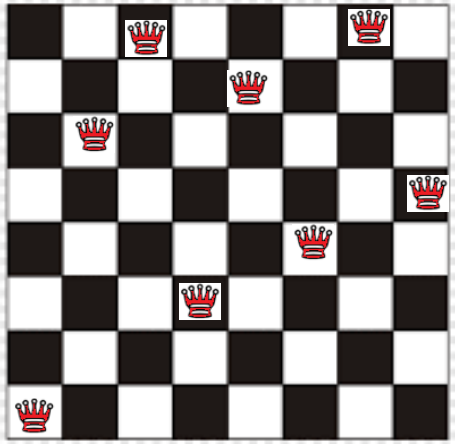
Next: h=-1
Current ← next



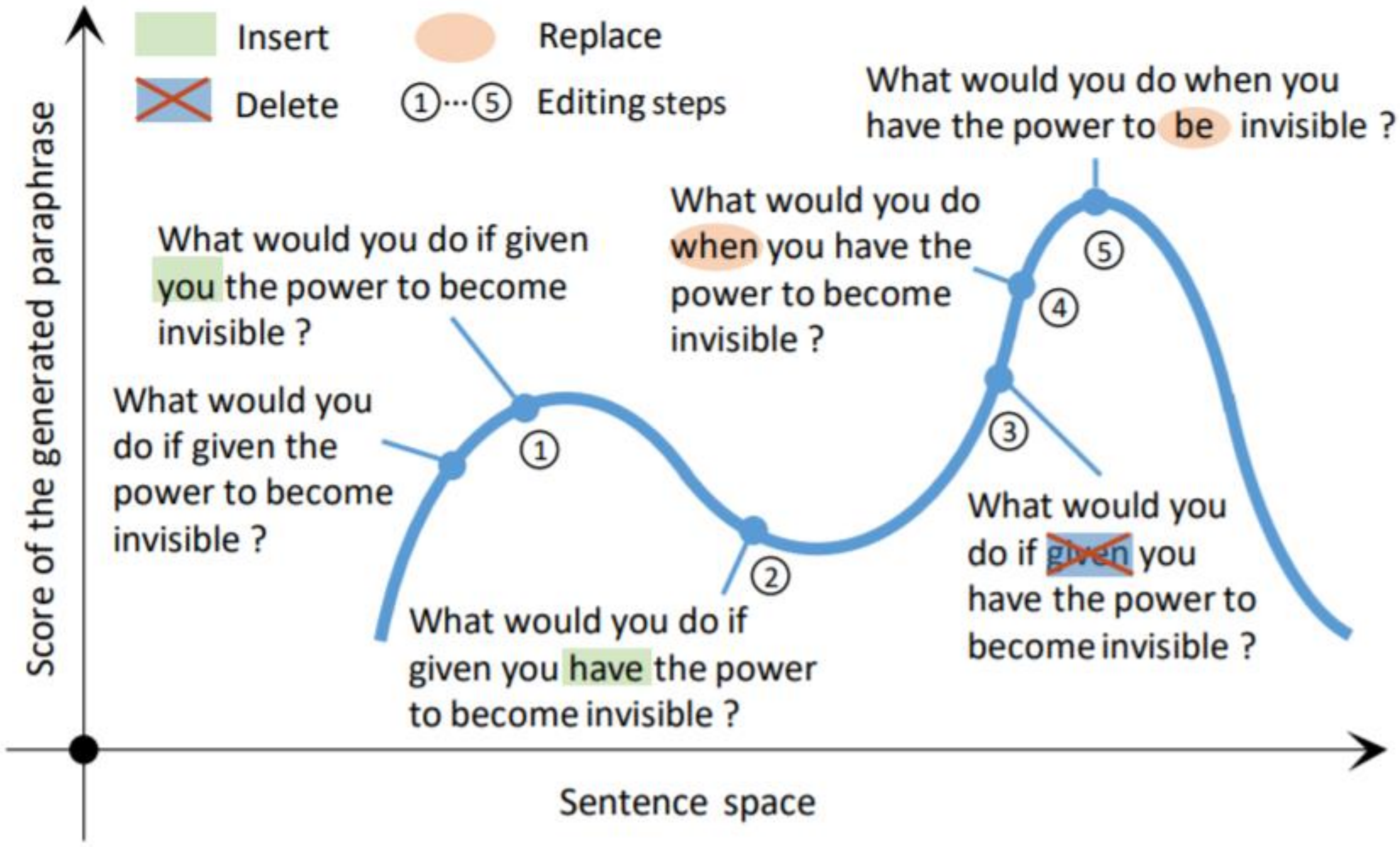
Column: C8
Row: 3➔5

→ stop pas 0 .

Next: h=0
Current ← next



Application: Simulated Annealing for Paraphrase



Liu, X., Mou, L., Meng, F., Zhou, H., Zhou, J., & Song, S. (2020). Unsupervised paraphrasing by simulated annealing. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 302–312 July 5 - 10, 2020. c 2020 Association for Computational Linguistics. *arXiv preprint arXiv:1909.03588*.

Properties of simulated annealing search

One can prove: If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1

Widely used in VLSI layout, airline scheduling, etc

Summary: Simulated Annealing

Simulated annealing is a version of stochastic hill climbing where some downhill moves are allowed.

If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1

Next:

- Genetic Algorithm

THANK YOU

INSTITUT TEKNOLOGI BANDUNG

1920

Modul 3: Beyond Classical Search

Genetic Algorithm

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Genetic Algorithm

Random Restart Hill-climbing

- parallel instead of in sequence
- not independent: passing *useful information*

informasi successor dari setiap statenya masih di pertimbangkan.

Local Beam Search

chooses k successors at random

mulai dengan sejumlah k state.
jd seolah² random restart tapi berjalannya paralel.

misal ada 4 state. casus 8 ratu, successornya ada 56+56+56+56, diambil 4 terbaik

Stochastic Beam Search

successor states: combining *two* parent states

↳ tidak harus lebih baik tetangganya, boleh lebih buruk dgn probabilitas tertentu.
(tidak selalu memilih k best successor)

Genetic Algorithm

Local Beam Search → masih ada kemungkinan stuck di local optima

```
function BEAM-SEARCH(problem, k) returns a solution state
    start with k randomly generated states
    loop
        generate all successors of all k states
        if any of them is a solution then return it
        else select the k best successors
```

begins with *k* randomly generated states.

all the successors of all *k* states are generated.

If any one is a goal, the algorithm halts. Otherwise, it selects the *k* best successors from the complete list and repeats.

In a local beam search, useful information is passed among the parallel search k threads

Genetic Algorithm: Illustration (Russel & Norvig, 2010)

State / Individual

A state is represented as a string over a finite alphabet (often a string of 0s and 1s)

One character ~ one variable

contoh representasi individu / state

32752411

State Value: Fitness Function

evaluasi semua individu.

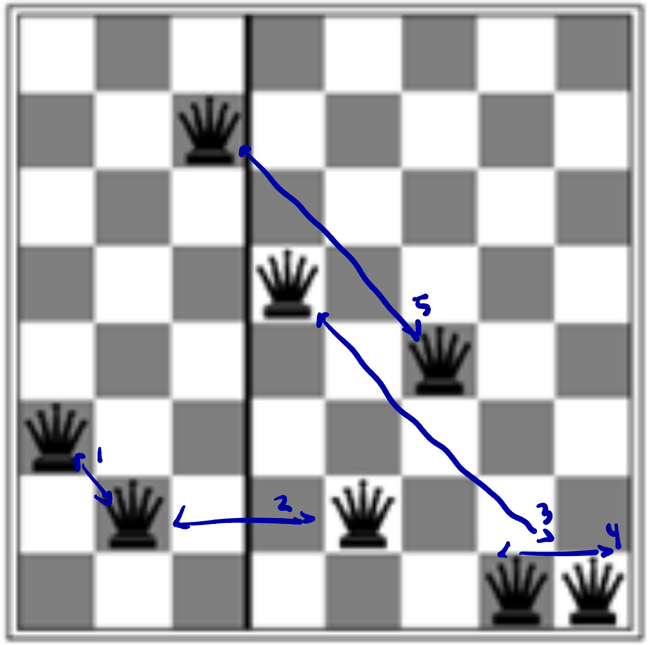
- Evaluation function (fitness function).
- Higher values for better states.
- Fitness function for n-queen: number of non-attacking pairs of queens
  - min = 0
  - max = (8 × 7)/2 = 28 (global maximum)

8 ratu

bisa tidak serang 7.

1 serang 2 = 2 serang 1.

32752411



F(32752411)=28-5=23

Initial Population & Successor Function

- GA starts with *k* randomly generated states.
- A successor state is generated by combining two parent states
- Produce the next generation of states by selection, crossover, and mutation

24748552	• F=24
32752411	• F=23
24415124	• F=20
32543213	• F=11

pendekatan threshold / roulette wheel.

Random Selection of Parent States

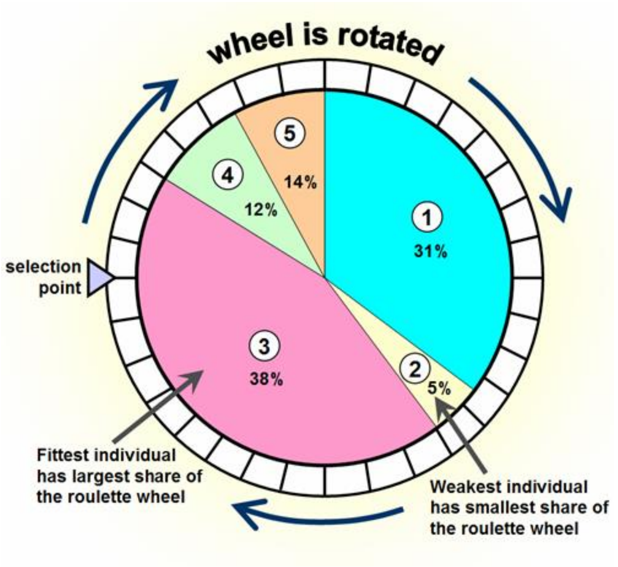
Probability of random selection

24748552
- 24/(24+23+20+11)
- = 31%

32752411
- 23/(24+23+20+11)
- =29%

24415124
- 20/(24+23+20+11)
- =26%

32543213
- 11/(24+23+20+11)
- =14%



0-31	32-60	61-86	87-100

random(0,100)

32752411

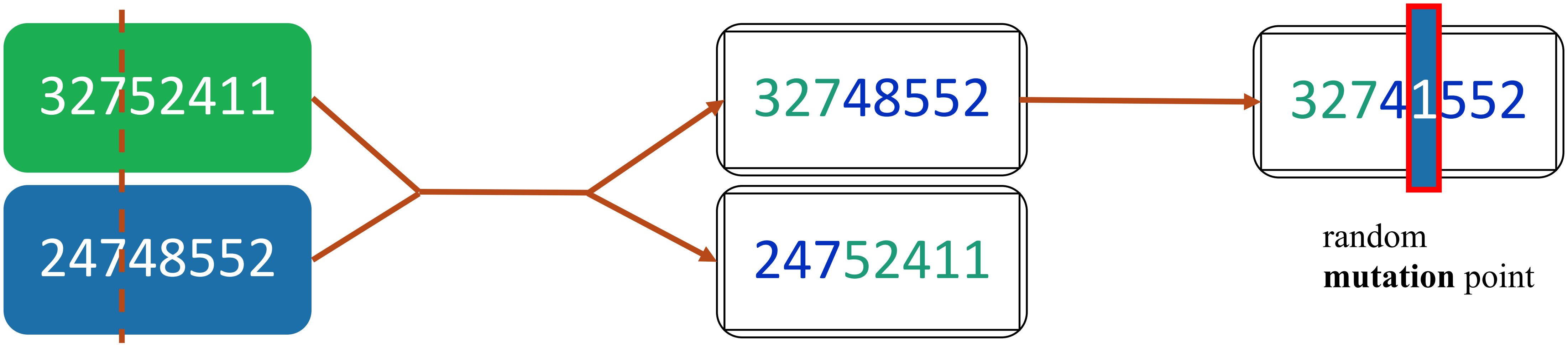
24748552

32752411

24415124

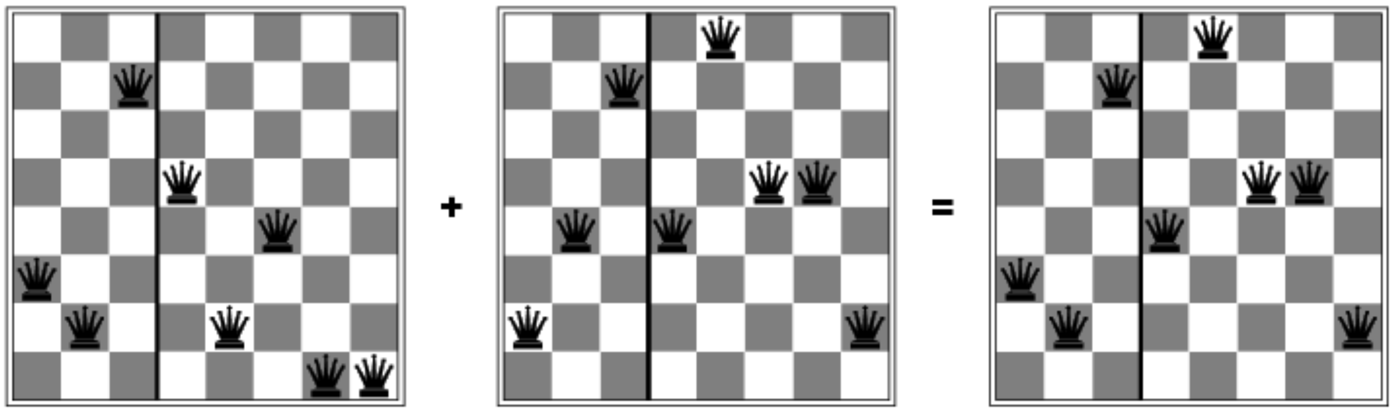
Cross Over / Reproduce & Mutation

bergantung ke probability juga



random **crossover** point

random **mutation** point



32752411 24748552 32748552

In more popular version, each mating of two parents produces only one offspring, not two (Russel & Norvig, 2010)

Genetic Algorithm: Illustration (Russel & Norvig, 2010)

Genetic Algorithm

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
     new_population ← empty set
     for i = 1 to SIZE(population) do
        x ← RANDOM-SELECTION(population, FITNESS-FN)
        y ← RANDOM-SELECTION(population, FITNESS-FN)
        child ← REPRODUCE(x, y)
        if (small random probability) then child ← MUTATE(child)
        add child to new_population
     population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```

Summary

- *k* randomly generated states (population: k individual)
- Fitness function (state value)
- successor function: combining two parent states (selection, cross-over, mutation)

pilih teknik tertentu

Terima Kasih

INSTITUT TEKNOLOGI BANDUNG

1920

1. Teknik local search yang mengenumerasi semua state tetangga (successor) dan current state pindah ke state tetangga terbaik jika nilai evaluasi state tetangga terbaik lebih baik dari nilai evaluasi current state adalah:

generate semua, tp ambil yg paling baik

→ dia hanya mengenumerasi 1 state tetangga, ga semua.

a. steepest-ascend hill-climbing; b. hill-climbing with sideways move; c. stochastic hill-climbing

2. Simulated annealing (SA) merupakan kombinasi dari hill-climbing dan random walk. Manakah karakteristik dari SA ?

a. Jika penurunan temperatur T cukup lambat, SA menjamin akan menemukan optimum global dengan peluang mendekati 1.

b. SA memiliki peluang yang membesar (mengecil) secara eksponensial untuk memilih state tetangga yang lebih buruk nilai evaluasinya.

c. SA membangkitkan secara random satu state tetangga, bukan semua state tetangga untuk setiap iterasinya.

3. Misalkan current state memiliki nilai evaluasi h, dan objektifnya mencari nilai h maksimum. Manakah perpindahan current state ke tetangga yang benar.

a. Pada steepest-ascend hill climbing, current state akan selalu pindah ke tetangga terbaik dari semua tetangga yang ada, dengan nilai evaluasi h_next jika h_next >h. Jika h_next <=h, masih ada peluang untuk berpindah state. → ga ada peluang pindah tetangga.

b. Pada simulated annealing, current state akan selalu pindah ke tetangga yang diambil secara acak dengan nilai evaluasi h_next jika h_next>h. Jika h_next <=h, masih ada peluang untuk berpindah state.

c. Pada algoritma genetika, current state akan selalu pindah ke tetangga dengan nilai evaluasi h_next yang dihasilkan dari proses seleksi, cross-over, dan mutasi tanpa mempedulikan apakah h_next >h.

Pada simulated annealing, diberikan sejumlah kondisi nilai temperatur (T), current.value, dan neighbor.value.

Diketahui batas move probability adalah 0.6. Pada ketiga kasus berikut, tentukanlah peluang neighbor state akan

dipilih menggantikan current state, serta current state diset dengan apa. (Nilai 10)

→ pindah ke neighbor

a. T = 10, current.value = 5, neighbor.value = 3 → tetangga is worse, $e^{\frac{\Delta E}{T}} = e^{\frac{3-5}{10}} = e^{\frac{-2}{10}} = $ 0,8 an.

→ pindah ke neighbor

b. T = 3, current.value = 3, neighbor.value = 5 → tetangga is better. pasti pindah.

c. T = 0, current.value = 5, neighbor.value = 3 → ga ngapa²in

soalnya T nya = 0 .. pasti return parent.

- Pada algoritma genetika dengan populasi 4 individu (A,B,C,D), misalkan nilai fitness setiap individu secara berturut-turut adalah 10,8,7,5. Jika menggunakan roulette wheel dengan urutan individu tetap, lakukanlah proses seleksi pembentukan populasi untuk cross-over dengan urutan bilangan acak 0.55; 0.25; 0.85; 0.35

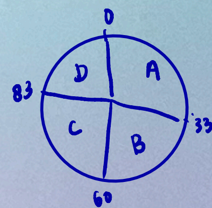
cari probability.
proporsi fitness function

$A : \frac{10}{30} = 0{,}33$

$B : \frac{8}{30} : 0{,}27$

$C : \frac{7}{30} : 0{,}23$

$D : \frac{5}{30} : 0{,}17$.



A	B	C	D
0-33	34-60	61-83	84-100

$0{,}55 \rightarrow B$

$0.25 \rightarrow A$

$0{,}85 \rightarrow D$

$0{,}35 \rightarrow B$

INSTITUT TEKNOLOGI BANDUNG

1920

Modul 4: Adversarial Search

Adversarial Search

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Adversarial Search

Non-Adversarial Search

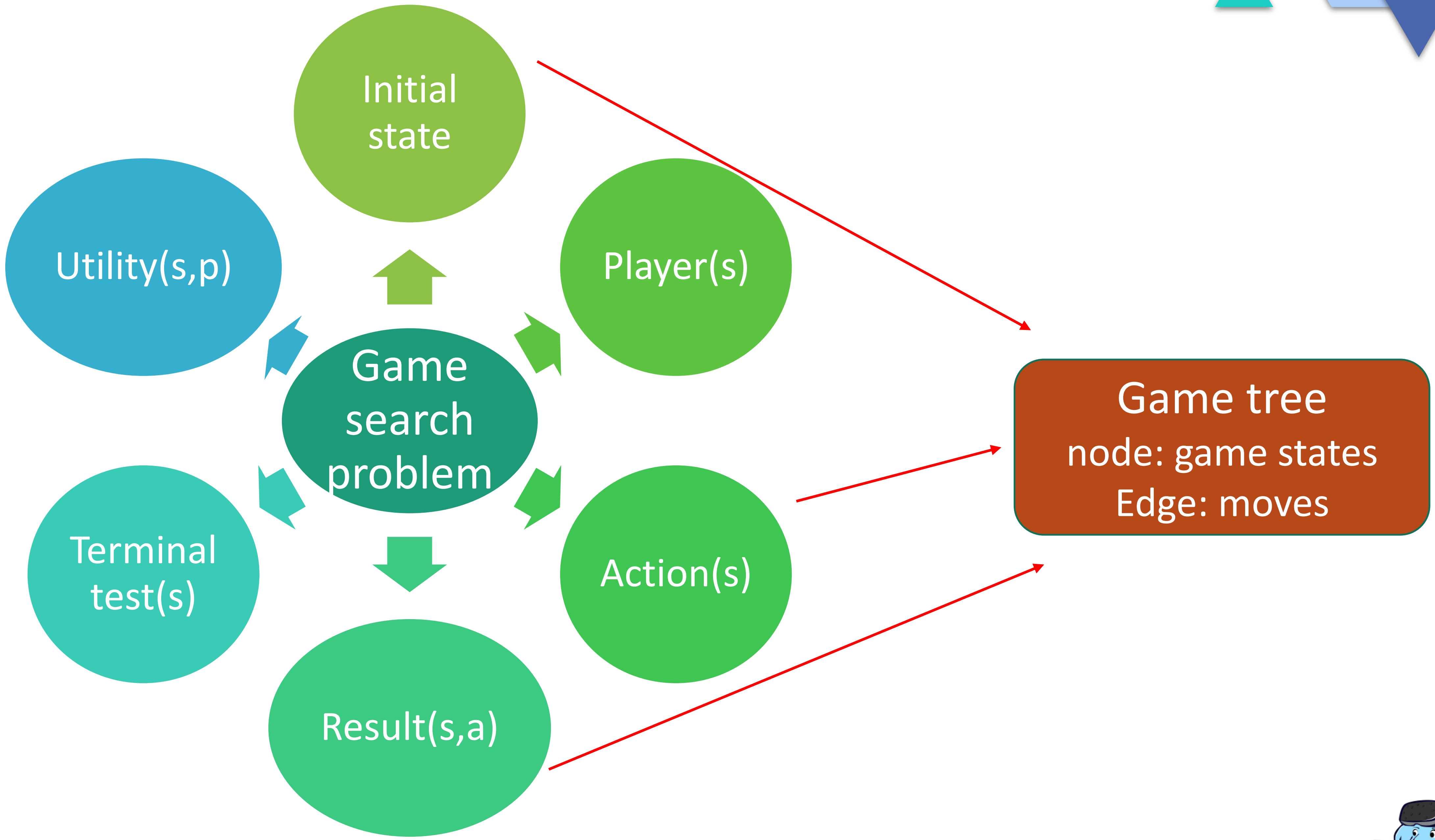
Deterministic and fully observable

Turn-taking

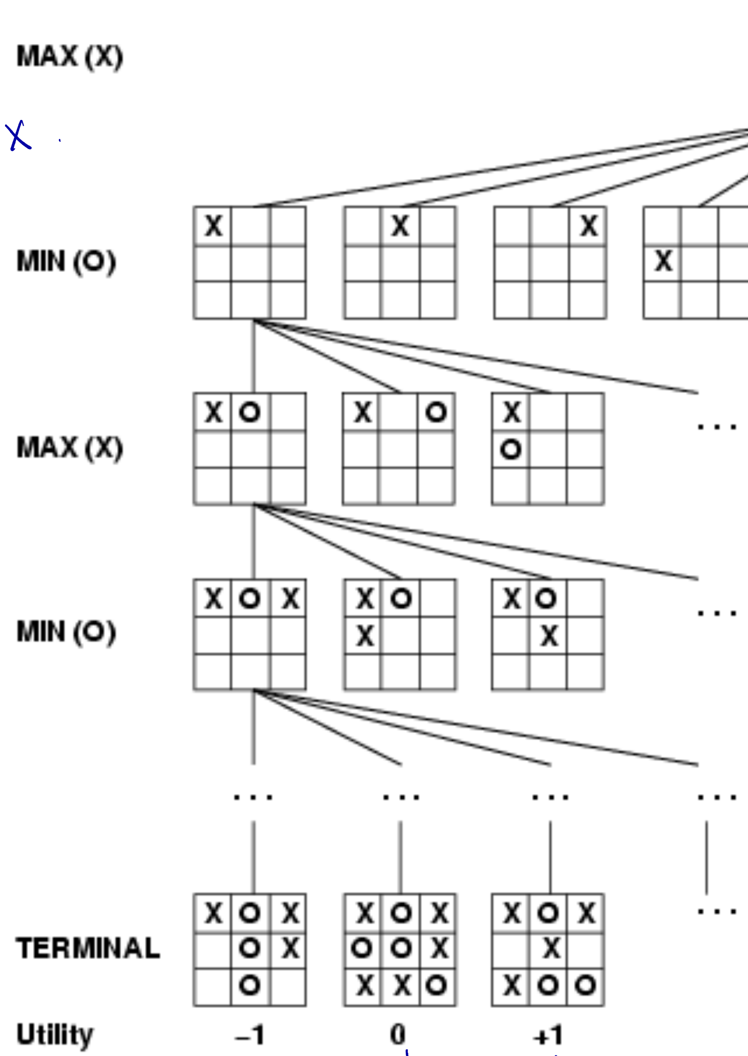
Approximate solution: strategy

Multiagent: competitive environment (games)

Optimize solution: path to goal or solution state


Initial state
Utility(s,p)
Player(s)
Game search problem
Terminal test(s)
Action(s)
Result(s,a)
Game tree
node: game states
Edge: moves

Game tree (2-players, tic-tac-toe)



Initial state

Kita X

we lose

draw

we win

node: game states
Edge: moves

+1: win
-1: loss
0: draw

Optimal Decision with Minimax Algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT(*state*, *a*))

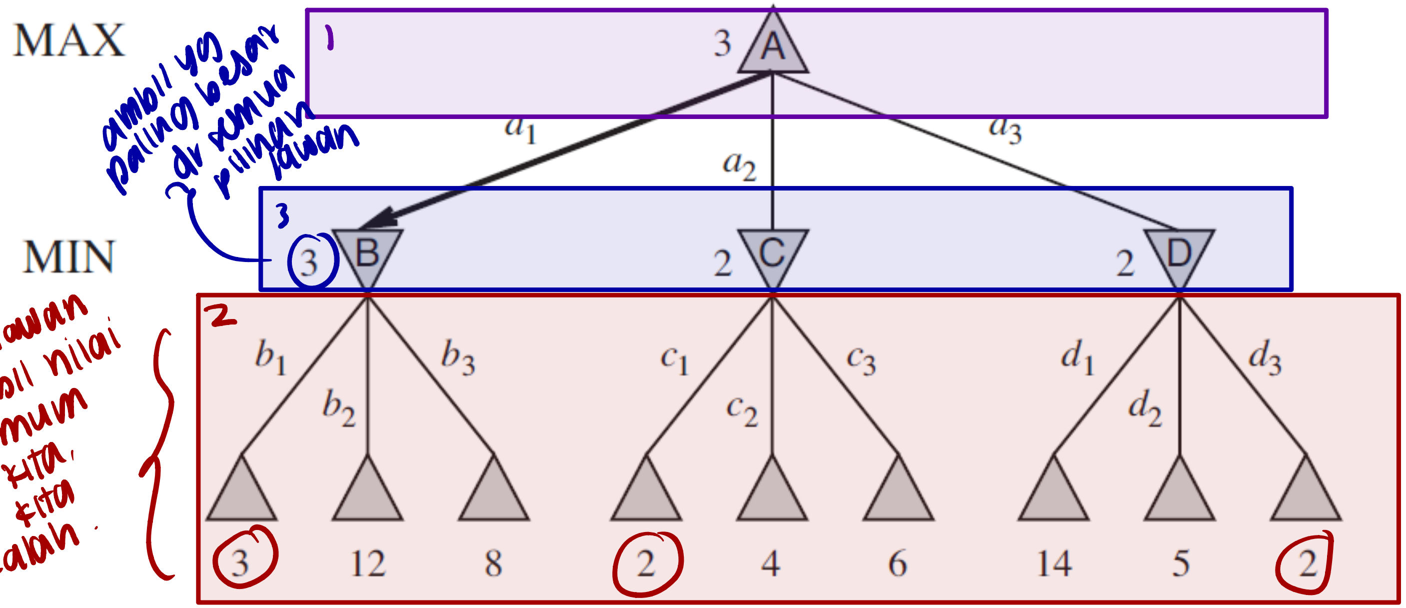
function MIN-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$ → nilai maks.
  for each *a* **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return v

function MAX-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for each *a* **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

Minimax: 2-ply Game Tree

ambil yg paling besar dr semua pilihan lawan

POV lawan, lawan pasti ambil nilai yg minimum buat kita, biar kita kalah

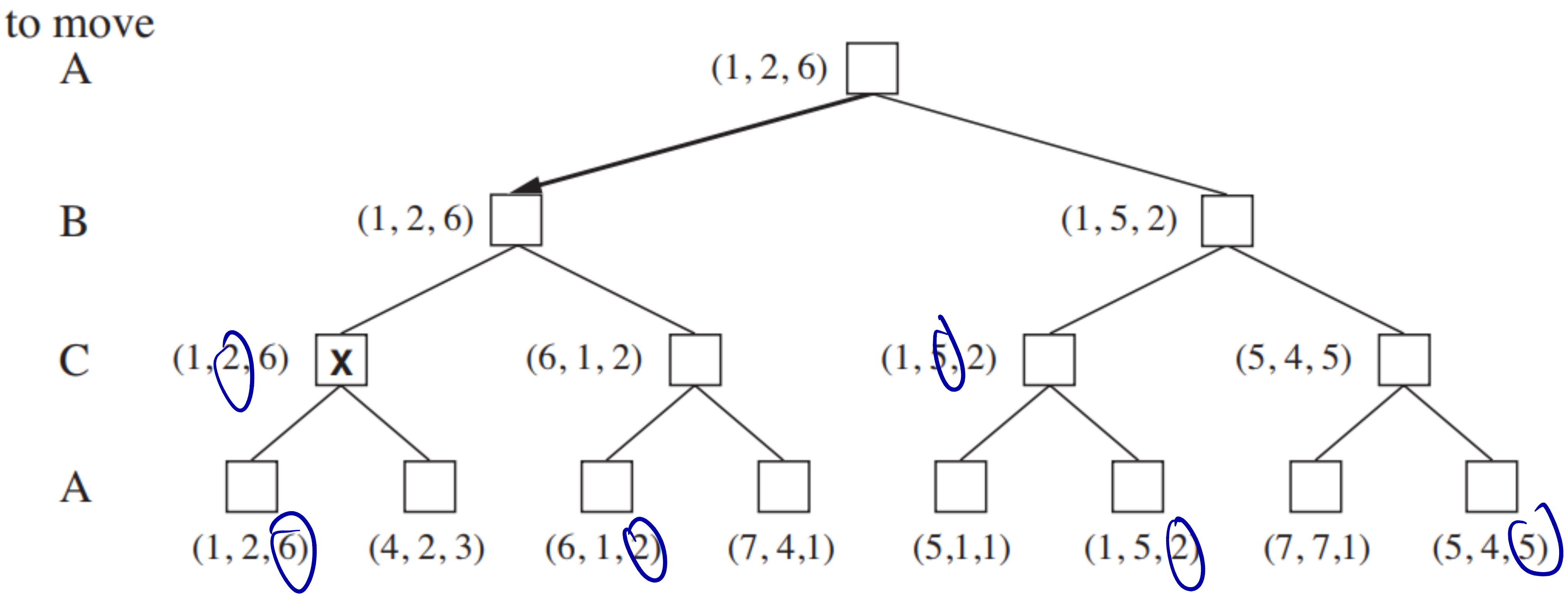


Minimax(B)=min(
Minimax(result(B,b1)),
Minimax(result(B,b2)),
Minimax(result(B,b3)))
=min(3,12,8)=3

Minimax(A)=max(Minimax(B),
Minimax(C),Minimax(D))
=max(3,2,2)=3

$$
\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}
$$

Minimax 3-ply in Multiplayer Games



Minimax Properties

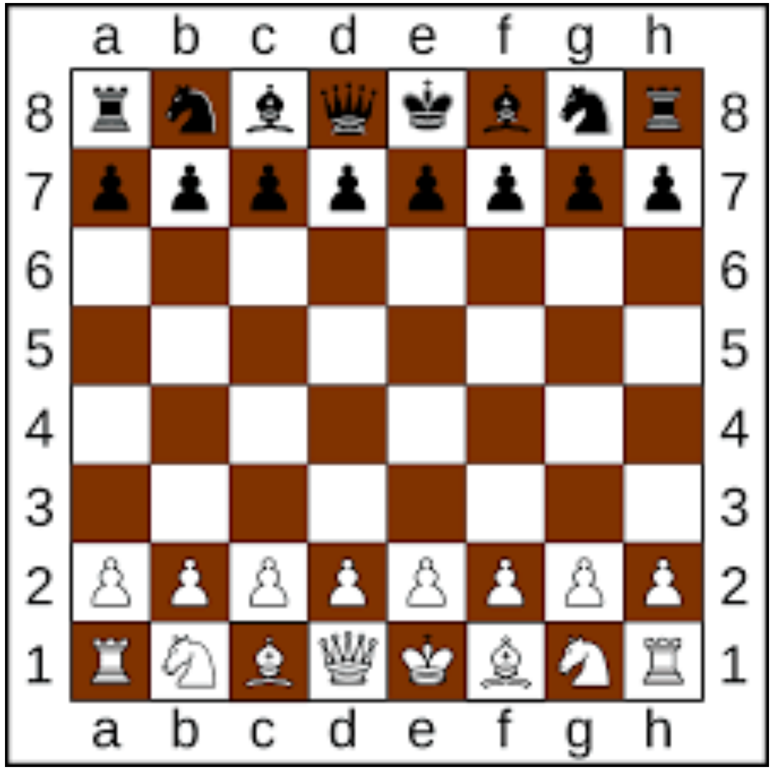
Complete?
Yes (if tree is finite)

Optimal?
Yes (against an optimal opponent)

Time complexity?
$O(b^m)$

Space complexity?
O(bm) (depth-first exploration)

Minimax for Chess

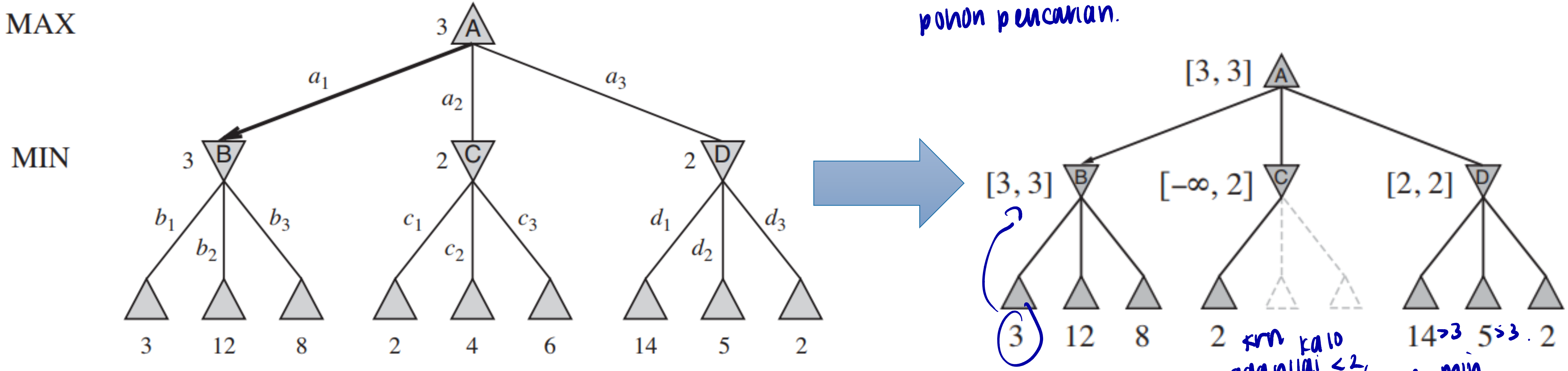


https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg

- Branching factor: 35 (avg)
- Games often 50 moves for each player → m=100
- Game states is exponential in the depth of game tree.
- Exact solution is completely infeasible

αβSearch: Minimax with Pruning

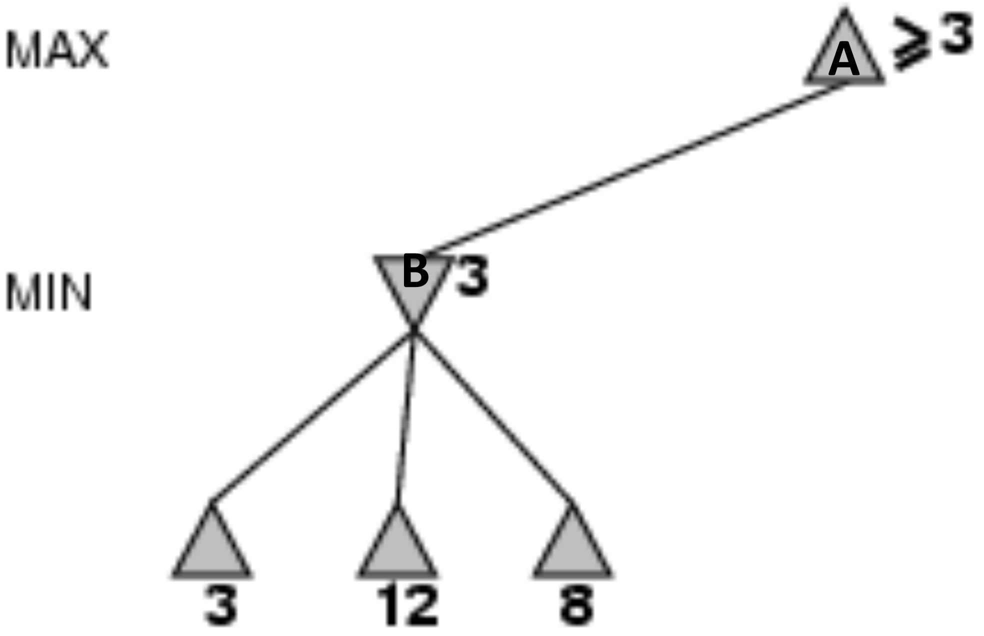
tidak mengubah hasil akhir,
hanya memangkas
pohon pencarian.



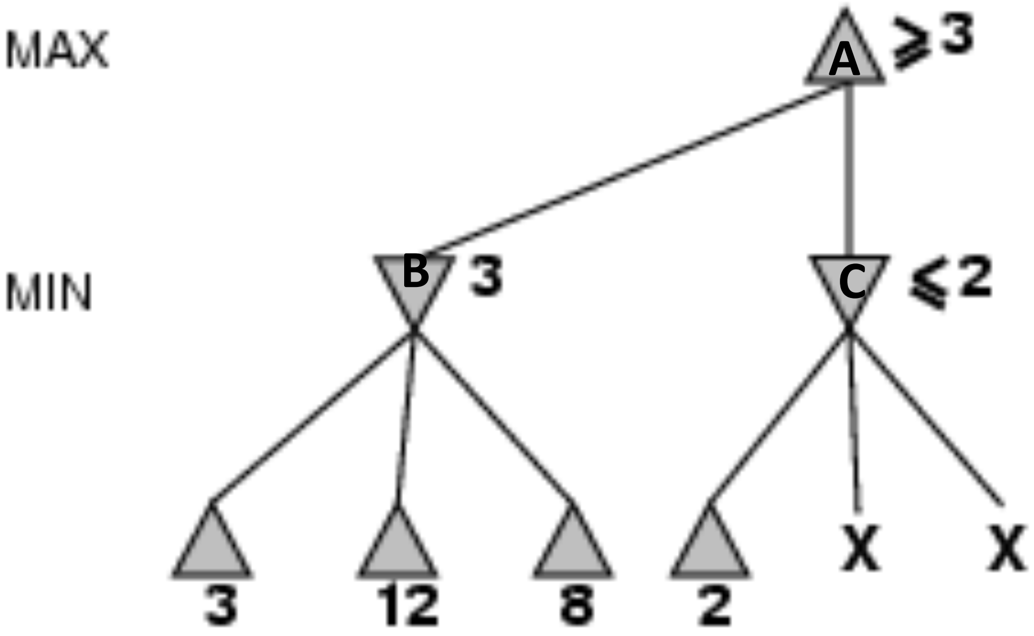
krn kalo
ada nilai < 2,
akan dipilih sama min,
tapi maks ga akan milih itu.
karena skrg nilai kita
dh ada 3.

Pruning does not affect final result. It returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.

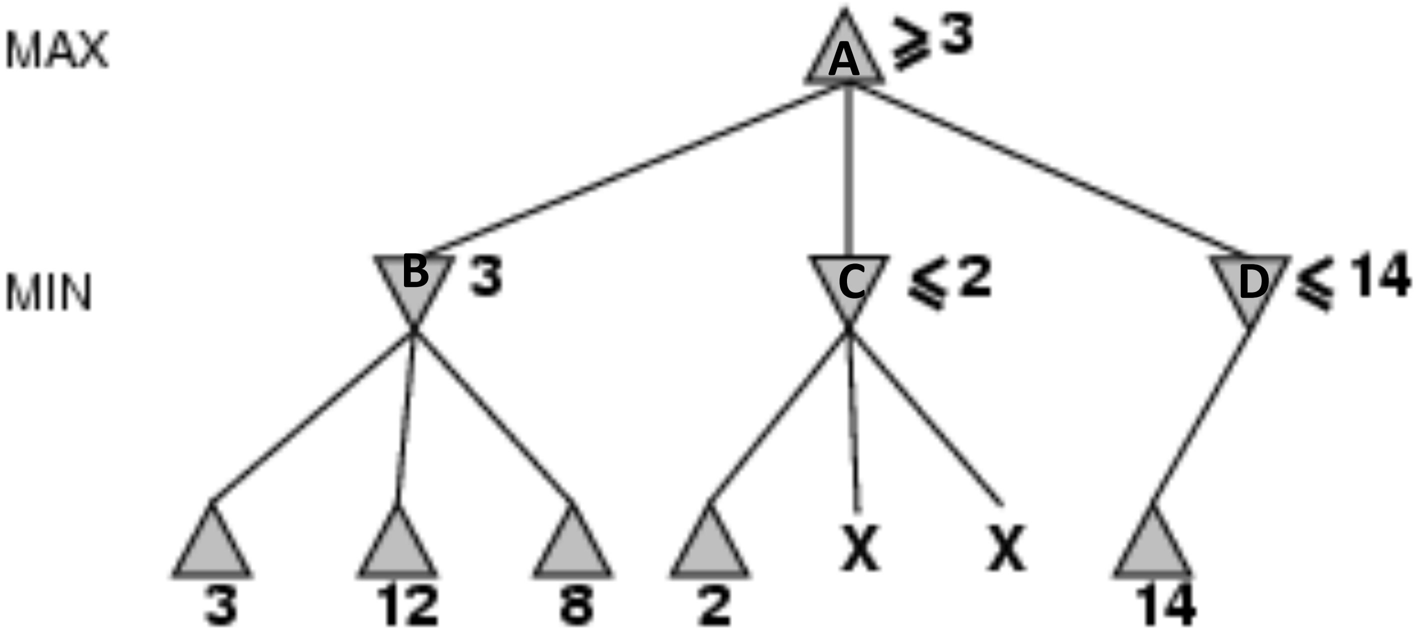
α-β pruning example



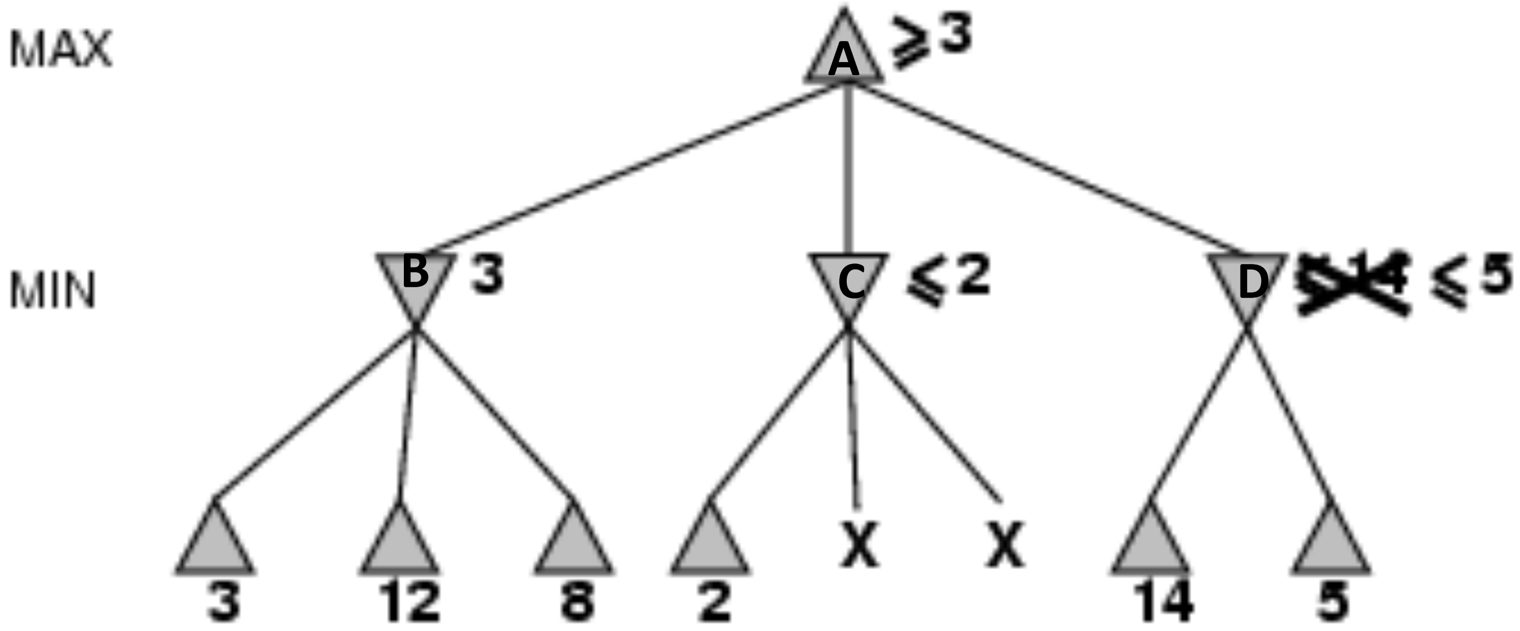
α-β pruning example



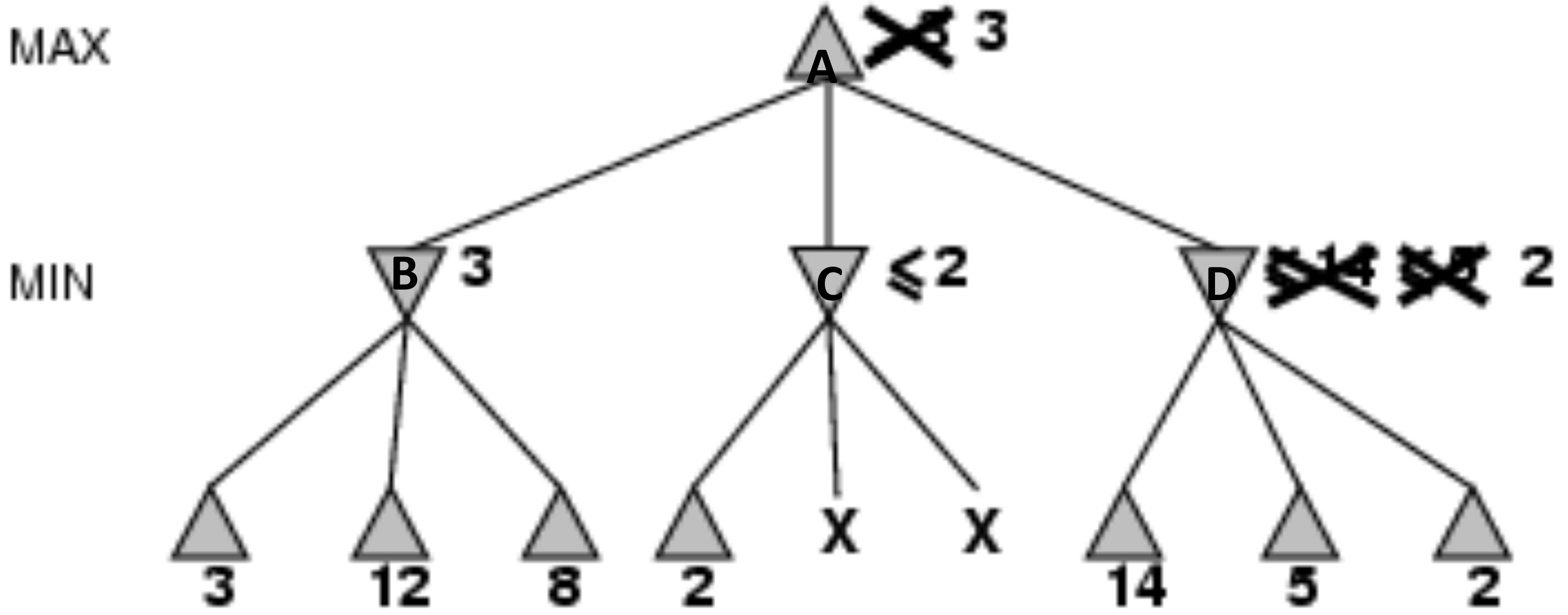
α-β pruning example



α-β pruning example

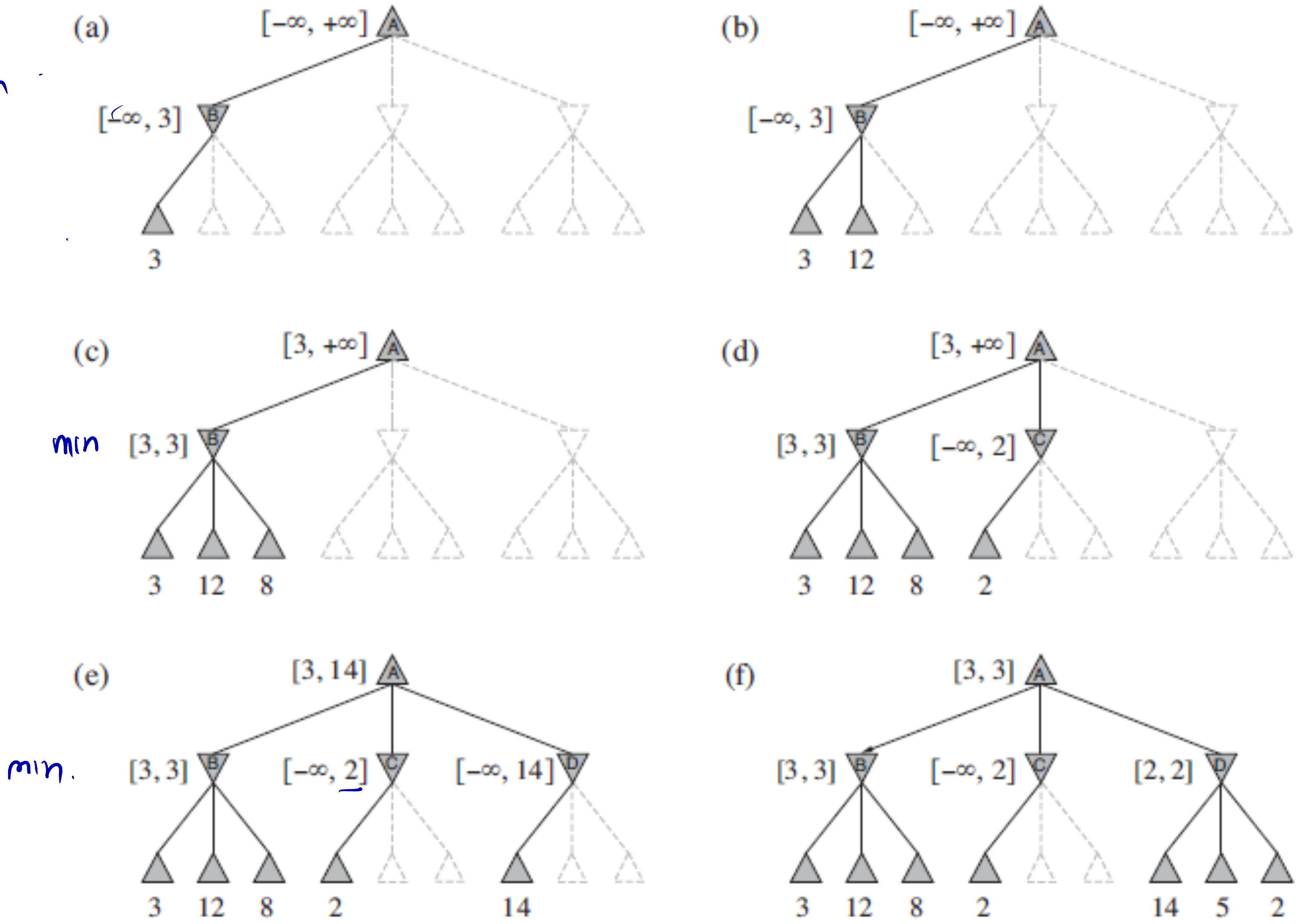


α-β pruning example



Algorithm illustration with α-β value

α : max
β : min



Summary

Adversarial search

Minimax search

αβSearch

INSTITUT TEKNOLOGI BANDUNG

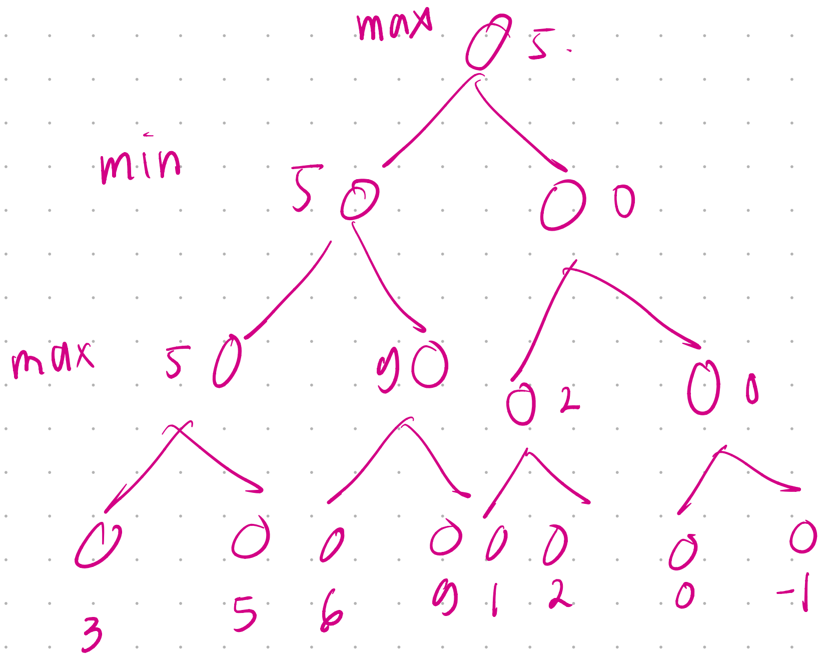
1920

Soal Adversarial Search

Terdapat suatu permainan berupa *3-ply game* sebagai berikut. Langkah pertama MAX yang bergerak dan memiliki dua kemungkinan langkah yaitu L atau R. Langkah berikutnya dilakukan MIN dengan dua kemungkinan langkah juga yaitu L atau R. Langkah terakhir dilakukan oleh MAX dengan dua kemungkinan langkah yaitu L atau R. Banyaknya kemungkinan urutan langkah adalah 8. Nilai pay-offs untuk MAX pada setiap kemungkinan urutan langkah adalah sebagai berikut: LLL = 3; LLR = 5; LRL = 6; LRR = 9; RLL = 1; RLR = 2; RRL = 0; dan RRR = -1. MAX berusaha memaksimalkan nilai payoffs nya, sedangkan MIN berusaha meminimalkan nilai payoffs MAX.

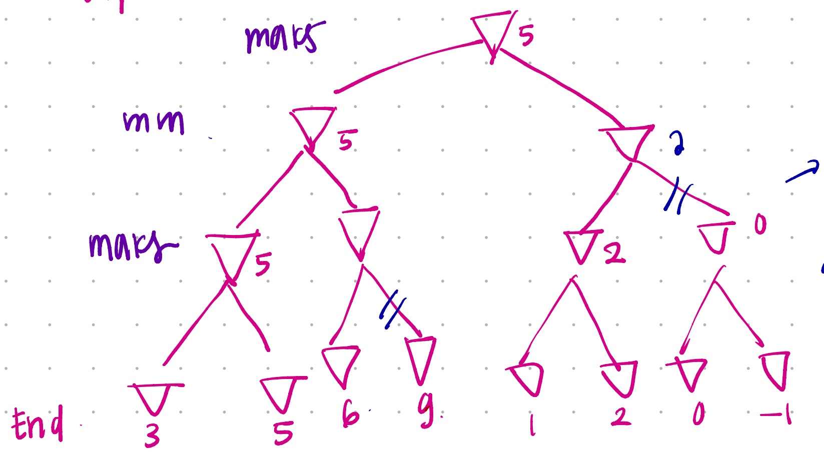
a) Gambarkan pohon *3-ply game* tersebut lengkap dengan nilai payoffs dari MAX pada setiap simpul pohon, dengan asumsi semua pemain memilih aksi secara rasional.

b) Terapkan *alpha-beta pruning* saat melakukan pencarian, dan ilustrasikan dengan gambar pohon untuk **setiap *pruning*** yang mungkin dilakukan pada cabang pohon dan jelaskan alasannya. Aksi L dieksplorasi terlebih dahulu sebelum aksi R dalam tahapan pencarian.



max
5
min
5
0
max
5
9
2
0
3
5
6
9
1
2
0
-1

alpha beta-pruning



0 gaperlu lg dicek karena kalau di > 2, ga bakal di pilih si min tapi kalo < 2, ga bakal dipilih si maks.

kalo misal dia > 6 ga bakal dipilih min.

< 6, ga bakal dipilih maks.

α mengubah nilai maks.

Modul : Constraint Satisfaction Problem (CSP)

Terminologi Dalam CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Constraint Satisfaction Problem (CSP)

- Termasuk dalam Problem Solving
- Formal Representation Language
- Allow General-Purpose Algorithm with more power

State

Variabel (X) ← Value from Domain (D)

Goal Test

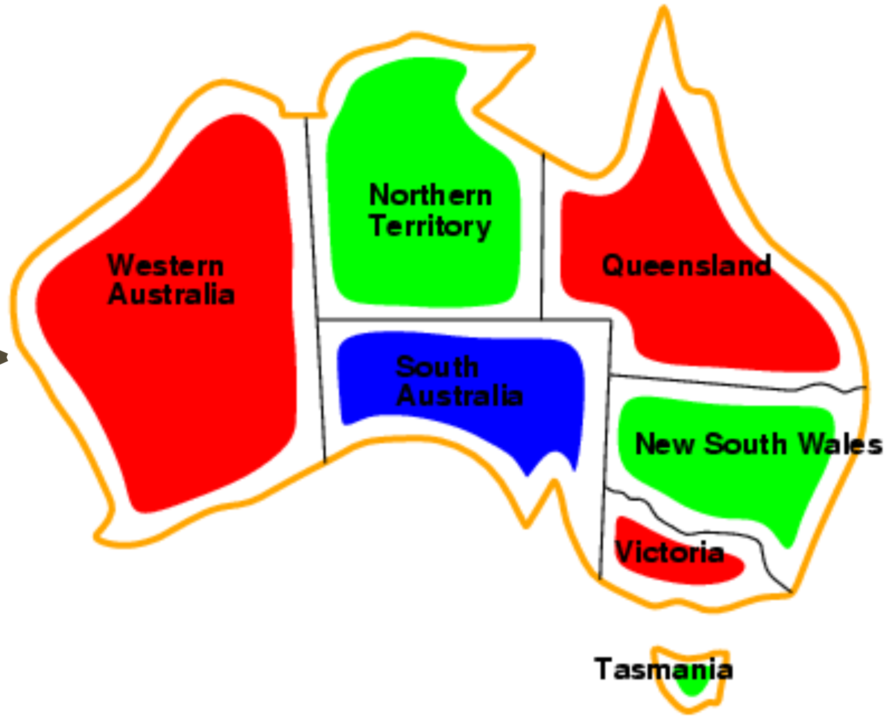
Allowable Combinations of values

Example: Map-Coloring Problem

Variables:
WA, NT, Q, NSW, V, SA, T

Domain:
D_i = {red,green,blue}

Constraints:
adjacent regions must have different colors



Solution: complete and consistent assignments

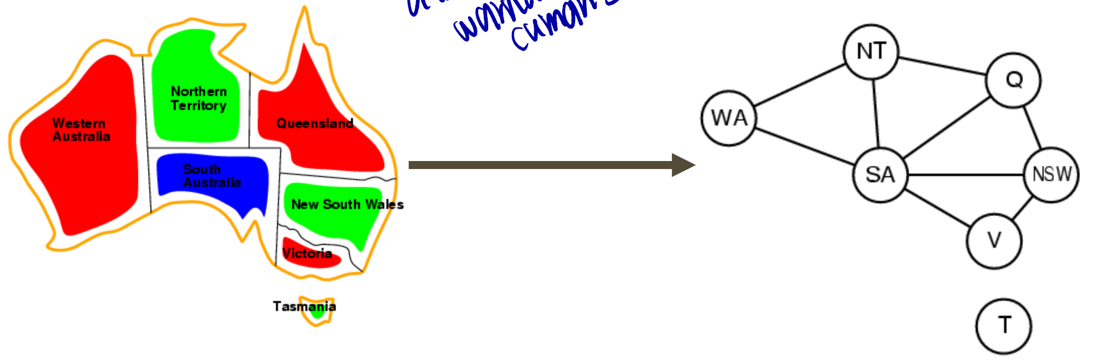
CSP Visualization

Variables → Nodes

Constraints → Link/ Arc

Constraints → Box

domain : warnanya cuman 3



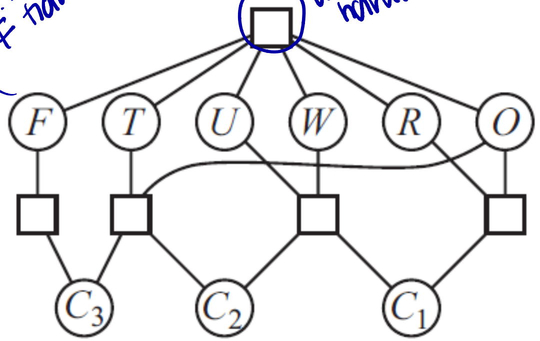
Constraint Graph

semua huruf domain sama yaitu angka (0..9).

↳ tapi bisa bikin 2 alternatif
① bikin constraint F ≠ 0 dan T ≠ 0
② bikin domain F dan T dari awal {1..9} F tidak boleh 0.

$$
\begin{array}{r}
T\ W\ O \\
+\ T\ W\ O \\
\hline
F\ O\ U\ R
\end{array}
$$

constraint semua angkanya harus beda



Constraint Hypergraph

Variations of CSP Formalism

- Variables
  - Discrete
    - Finite Domain
    - Infinite Domain
      - Constraint Language: $StartJob_1 + 5 \leq StartJob_3$
  - Continuous
    - Linear Programming, Quadratic Programming, etc
- Constraints
  - Unary (SA ≠ green)
  - Binary (SA ≠ WA)
  - Higher-Order (cryptarithmetic)
  - Preference
    - COP

ITB

Example: Cryptarithmetic Puzzle

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

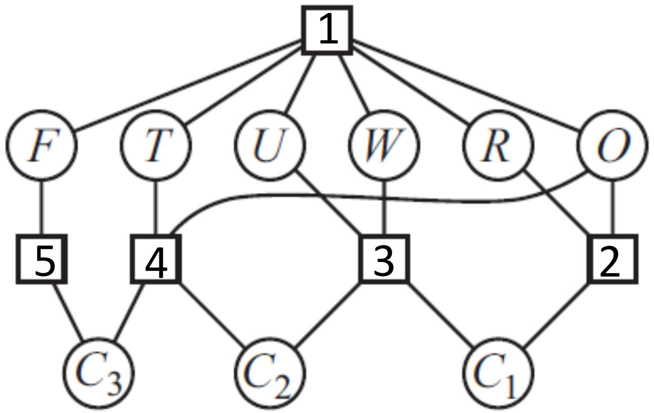
Variables

F,T,U,W,R,O

C_1, C_2, C_3 : auxiliary variables

Domain

F,T,U,W,R,O ={0,1,2,3,4,5,6,7,8,9}



Constraints

1. Alldiff *(F,T,U,W,R,O)*
2. $O + O = R + 10 \cdot C_1$
3. $W + W + C_1 = U + 10 \cdot C_2$
4. $T + T + C_2 = O + 10 \cdot C_3$
5. F = C_3 , $T \neq 0$, $F \neq 0$

Modul : Constraint Satisfaction Problem (CSP)

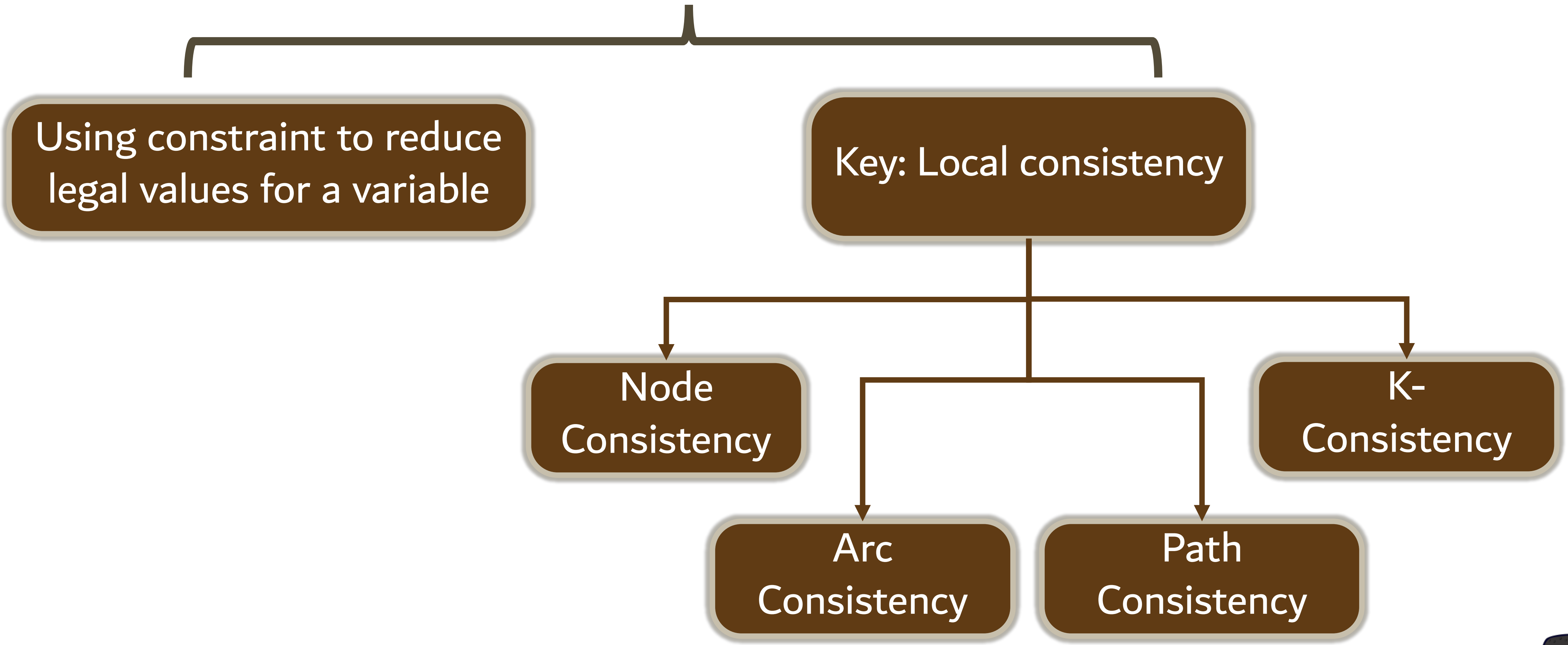
Inference in CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Constraint Propagation

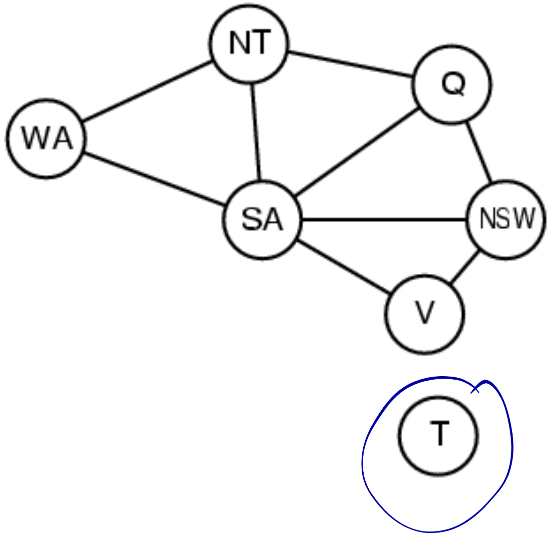


Node Consistency

All the values in the variable's domain satisfy the variable's **unary constraints**

Example: SA ≠ green

SA = {red, blue}

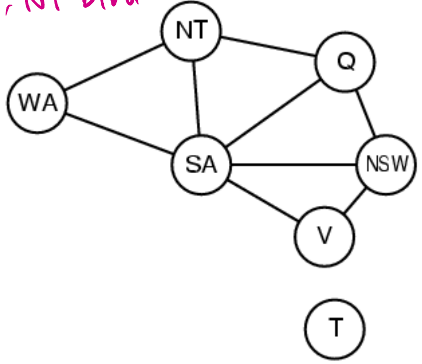


↓ T mau warna apapun bebas (selama ada dlm domain) bcs dia ga terikat ke manapun

A network is node-consistent if every variable in the network is node-consistent

Arc Consistency (AC)

SA merah, WA biru OK
SA merah, NT biru OK



A variable in a CSP is **arc-consistent** if every value in its domain satisfies the variable's binary constraints

Example: SA ≠ WA

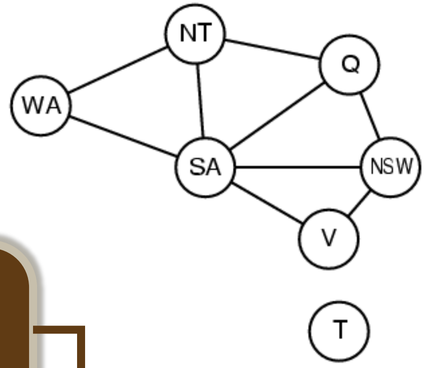
(SA,WA) = {(red , green), (red , blue), (green, red), (green, blue), (blue, red), (blue, green)}

Has no effect in this example (no reduction in the domain)

A network is arc-consistent if every variable is arc consistent with every other variable

Path Consistency (PC)

Arc Consistency: solve the problem if each variable has only 1 value left after the process OR finds that CSP can not be solved



Does not work for map coloring with only 2 values in the domain → (Path Consistency)

{Xi,Xj} is path-consistent to Xm if:

- Assignment {Xi = a,Xj = b} consistent with constraints on {Xi,Xj}
- There is assignment to Xm that satisfies constraints on {Xi,Xm} and {Xm,Xj}.

Example: Coloring Map with 2 colors (red, blue) → PC: {WA,SA} wrt NT

- {WA = red ,SA = blue} or {WA = blue,SA = red}
- No valid choice for NT

→ Eliminate both assignment → No solution

K-Consistency

A CSP is k-consistent if: any set of k – 1 variables & any consistent assignment to those variables, there is a consistent value to be assigned to k^{th} variable

- 1-consistency: given empty set, can make any set of one variable consistent
- 2-consistency = Arc Consistency
- 3-consistency = Path Consistency

Modul : Constraint Satisfaction Problem (CSP)

Backtracking Search for CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Backtracking Search

Use Depth First Search → Solution for n variables at depth n

Path is irrelevant → variable assignment commutative

Only consider assignments to a single variable at each node

Basic uninformed algorithm for CSPs

classical dimulai dengan kosong (tdk ada variabel yg diassign)

local dimulai dengan random configuration

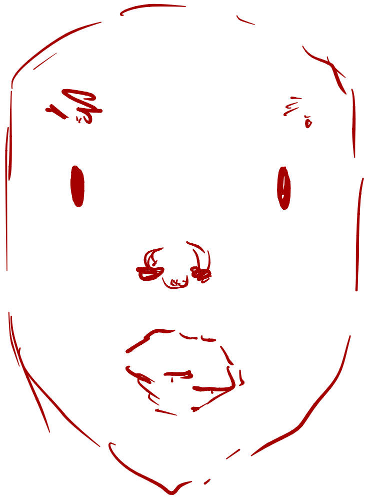
Algorithm

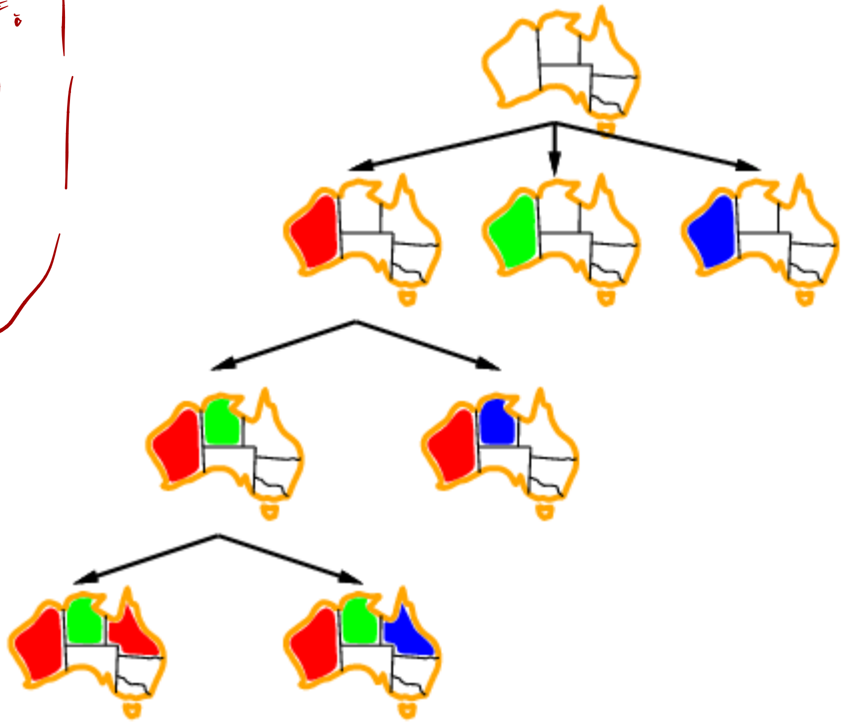
```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
   return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
   if assignment is complete then return assignment
   var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
   for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
      if value is consistent with assignment then
         add {var = value} to assignment
         inferences ← INFERENCE(csp, var, assignment)
         if inferences ≠ failure then
            add inferences to csp
            result ← BACKTRACK(csp, assignment)
            if result ≠ failure then return result
            remove inferences from csp
         remove {var = value} from assignment
   return failure
```

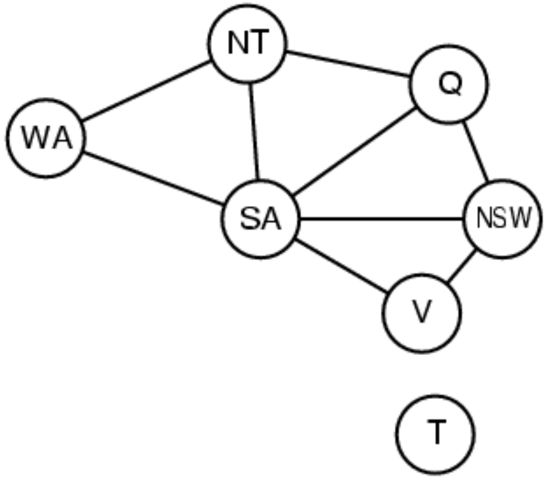
misal SA udah merah, WA dan NT kan gaboleh merah, jadi merah itu diapus dr domain

inference: menghapus nilai yg membuat inconsistent dr domain

Example: Map Coloring Problem







Improving Backtracking Efficiency

Which variable should be assigned next?

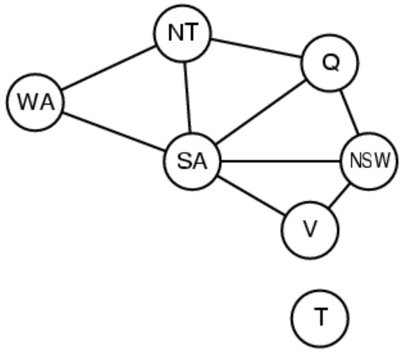
In what order should its values be tried?

Detect inevitable failure early?

Without Domain Specific Knowledge

Variable Ordering

$$var \leftarrow \text{SELECT-UNASSIGNED-VARIABLE}(csp)$$



1. Static Variable Ordering: {*WA, NT, SA, Q, NSW, V, T*}

2. Minimum Remaining Values Heuristic

sudah assign nilai. ada nilai² yg hilang dr domain variabel tertentu. ⇒ inference.

→ jd pilih variabel yg nilai dalam domainnya paling dikit.

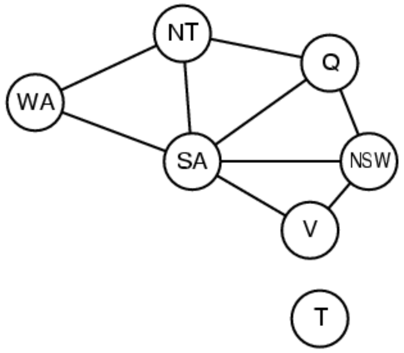
After WA=red and NT=green, SA or Q?

SA={blue}
Q={blue, red}

Choose SA (has minimum remaining value)

Most Constrained Variable

Variable Ordering - 2



$var \leftarrow \text{SELECT-UNASSIGNED-VARIABLE}(csp)$

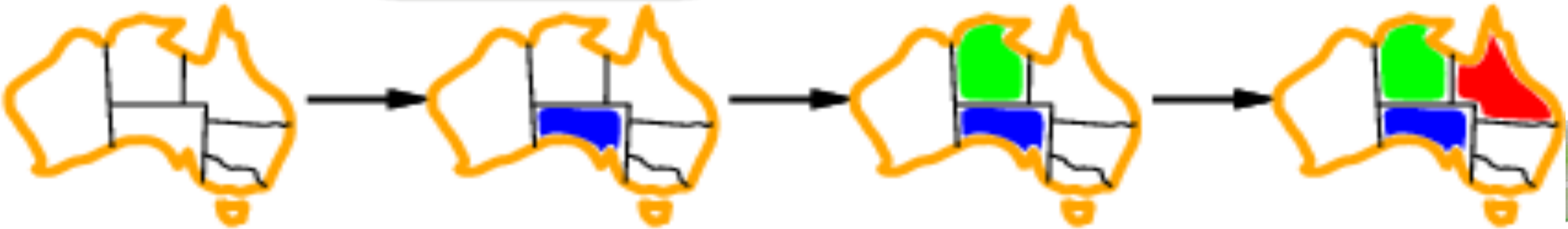
2. Minimum Remaining Values Heuristic : First Variable to Assign?

Use degree heuristic → Number of constraints involved in a variable on other unassigned variable

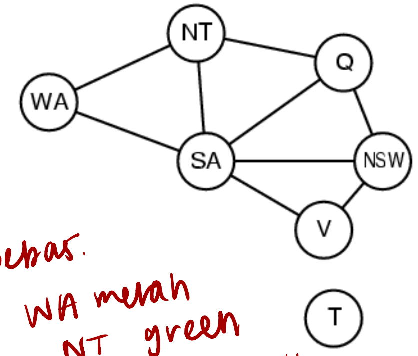
SA di assign duluan karena constraintnya paling besar (5). T belakangan aja bcs no constraint.

SA: 5
NT, Q, NSW: 3
WA, V: 2
T: 0

→ Choose SA First!!



Value Ordering



Least Constraining Value Heuristic

memberi kebebasan lebih buat variabel selanjutnya / yg belum di assign.

misal: urutan bebas.
isi WA merah
NT green
Q bisa merah / biru.
→ kalau Q biru, maka ketika inferensi SA habis.
→ kalau Q merah, SA masih ada.
jd pilih merah.

Prefer value that rules out fewest choice for neighboring variables

After WA=red and NT=green, What color for Q?

Q=blue → SA = { }
Q=red → SA = {blue}

Choose value red for Q

Value Ordering is irrelevant if we want to have all possible solutions

01

Modul : Constraint Satisfaction Problem (CSP)

Interleaving Search and Inference in CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Interleaving search and inference

bisa dilakukan sblm pencarian juga, lho!

Inference can be done before searching

Interleaving search and inference → detect failure early

- Forward Checking: establishes arc consistency for binary constraint
- Constraint Propagation → Maintaining Arc Consistency (MAC)

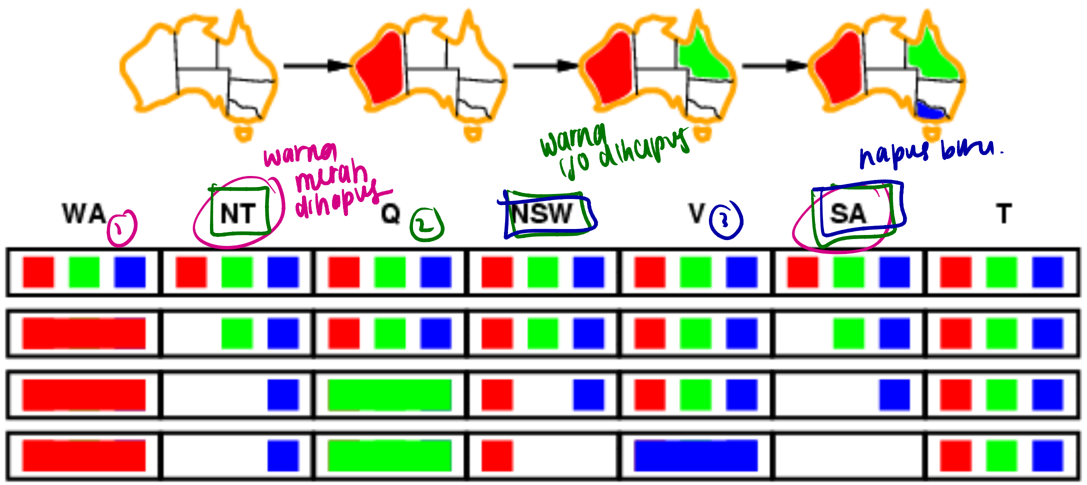
Let's see an example

Example: Interleave Search and Inference

Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values

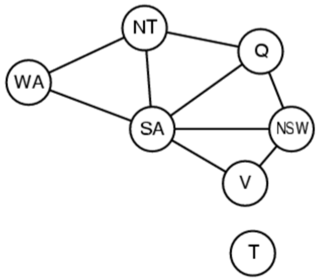
contoh ini ga pakai heuristic tadi!



gagal sblm ngisi SA.

→ no need to continue.

FAIL!



MAC: NT and SA cannot both be blue → MAC: repeatedly enforces constraints locally

04

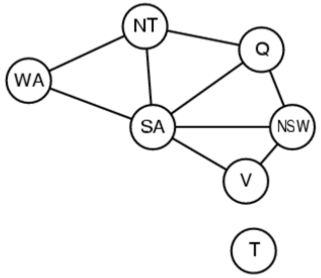
Example: Interleave Search and Inference

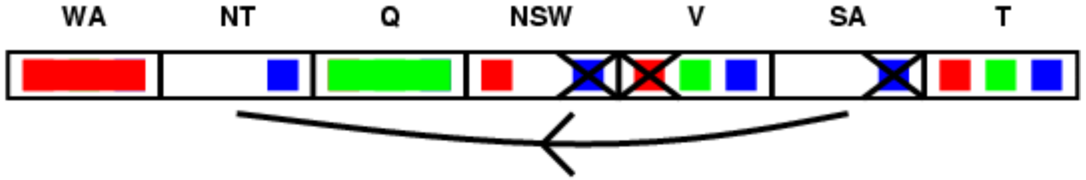
Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values

MAC: repeatedly enforces constraints locally

mencegah dari awal assignment yg gagal dari awal.





FAIL!

misal WA merah
Q ijo
NT sisa biru.
SA yg sisa biru
uda pasti gagal kan?
artinya Q jgn di assign ijo.

If variable *X* loses a value, neighbors of *X* need to be rechecked

Modul : Constraint Satisfaction Problem (CSP)

Local Search for CSP

Nur ULFA Maulidevi

KK IF - Teknik Informatika- STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Local Search

Complete-state formulation → initial state assigns a value to every variable

The search changes the value of one variable at a time

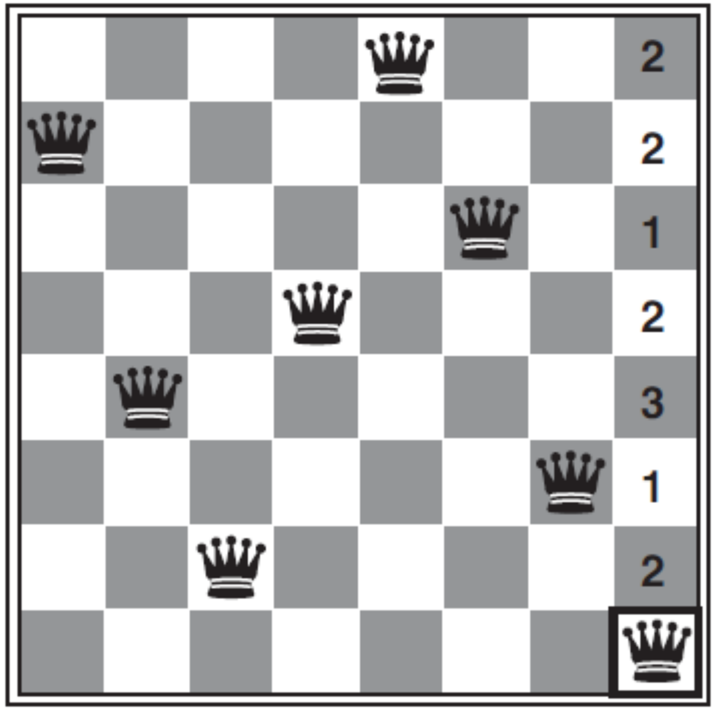
Variable selection: randomly select any conflicted variable

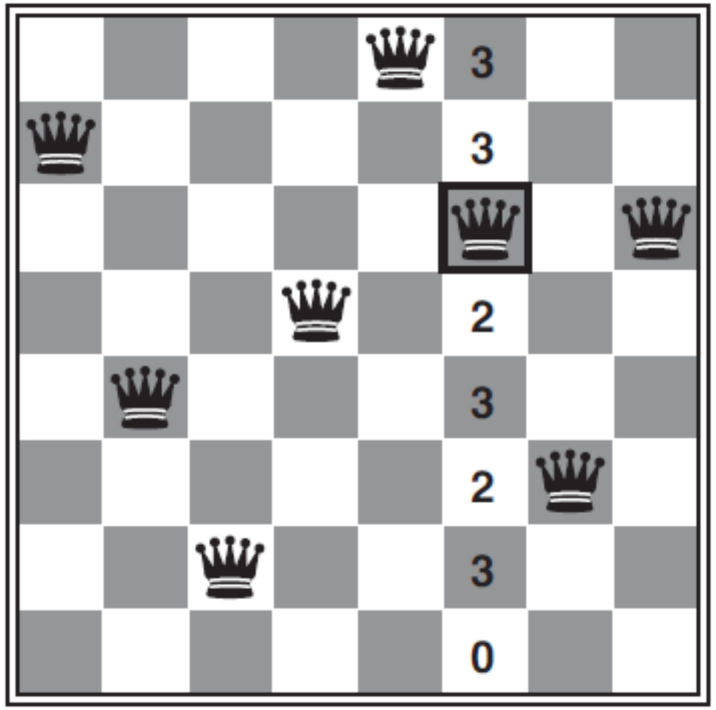
Value selection by min-conflicts heuristic

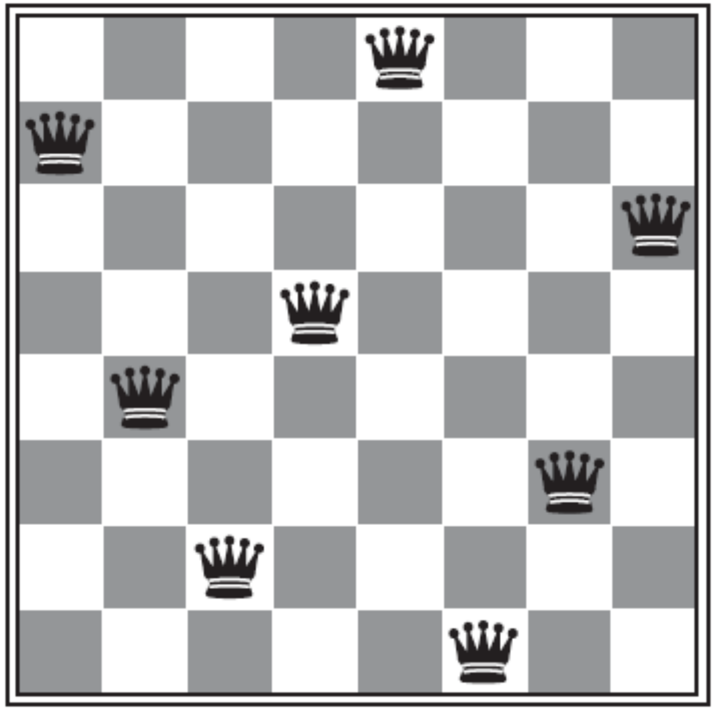
Min-Conflict Heuristic

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
   inputs: csp, a constraint satisfaction problem
           max_steps, the number of steps allowed before giving up

   current ← an initial complete assignment for csp
   for i = 1 to max_steps do
      if current is a solution for csp then return current
      var ← a randomly chosen conflicted variable from csp.VARIABLES
      value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
      set var = value in current
   return failure
```

Example: n-Queens Problem







Applications

Solve *n*-queens in almost constant time for arbitrary *n* with high probability

Online setting → scheduling

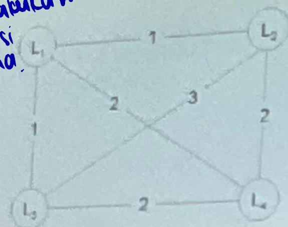
THANK YOU

Meeting Scheduling Problem:

tidak ada meeting dilakukan di lokasi yg sama.

Meeting	Location	Attending agents
m_1	L_1	A_1, A_3
m_2	L_2	A_2, A_3, A_4
m_3	L_3	A_1, A_4
m_4	L_4	A_1, A_2

Gambar 1.



Gambar 2.

- Durasi tiap pertemuan: 1 time slot
- Jarak tempuh antar lokasi: sesuai Gambar 2 (dalam time slot)
- Dicari alokasi waktu tiap pertemuan dalam 1 hari

Tentukan:

a. Variable
b. Domain
c. Constraints

$$|m_i - m_j| \geq 1 + jarak_{i-j}$$

yg mencari jarak terdekat

INSTITUT TEKNOLOGI BANDUNG

1920

Modul 5: Knowledge-based System

01 What & Why

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

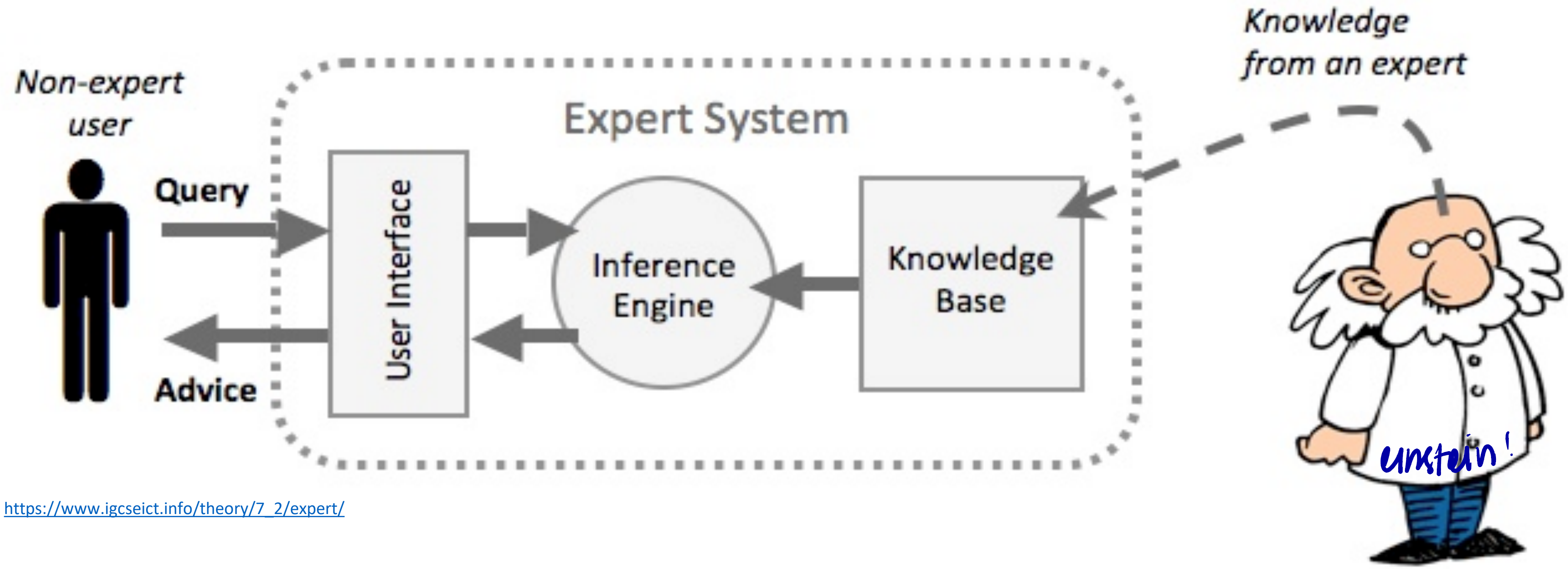
Knowledge-based System (KBS): What

Apply knowledge in solving problem

Reconstruct expertise and reasoning capabilities of qualified specialists within limited domains

Logical reasoning

Knowledge-based System ≠ Expert System



https://www.igcseict.info/theory/7_2/expert/

Knowledge-based System (KBS): Why

Approach in developing AI agent

Logical reasoning: thinking rationally

Template-based pattern recognition

Statistical-based pattern recognition

Structural/syntactic pattern recognition

Deep learning-based pattern recognition

pengenalan berbasis neural network

KBS	Conventional Program
Ill-structured problem	Well-structured problem
Expert determine actions, but execution order by interpreter	Programmer determines actions and execution order
Problem solving method + domain knowledge + data	Algorithm + data

persoalan susah buat distrukturkan, gbs eksplisit tau dia persoalan apa (cth: CSP / apala yg lain)

1. Instruction-based programming style:
   program = sequence of instructions and queries

instruction 1
↓
instruction 2
↓
query
↙ ↘
instruction 3   instruction 4

   The programmer determines what is done and in what order it is done.

2. Rule-based programming style:
   program = set of rules and rule interpreters

Rule 1: If situation X1, then action Y1.
Rule 2: If situation X2, then action Y2.
Rule 3: If situation X3, then action Y3.

   The expert determines what is done, and the rule interpreter determines the order.

Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Problem Characteristics

Well-formed problem	Ill-structured problem
Exact / certain solution	Uncertain solution
Explicit goal	Undefined goal
Explicit operator	Unknown operator

Summary

What is KBS	KBS ≠ ES	Why KBS	KBS vs conventional

Reasoning in Knowledge-based Agent

Modul 5: Knowledge-based System

02 Knowledge-based Agent

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

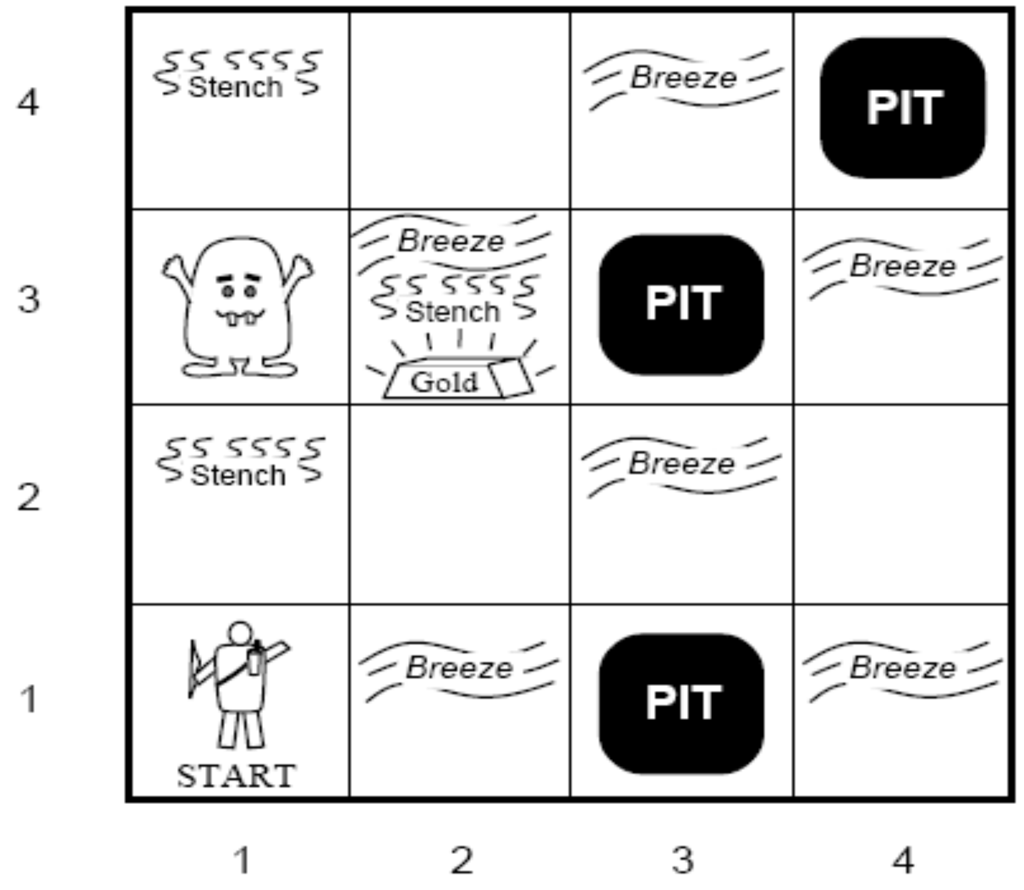
Knowledge-based Agent

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

Wumpus World



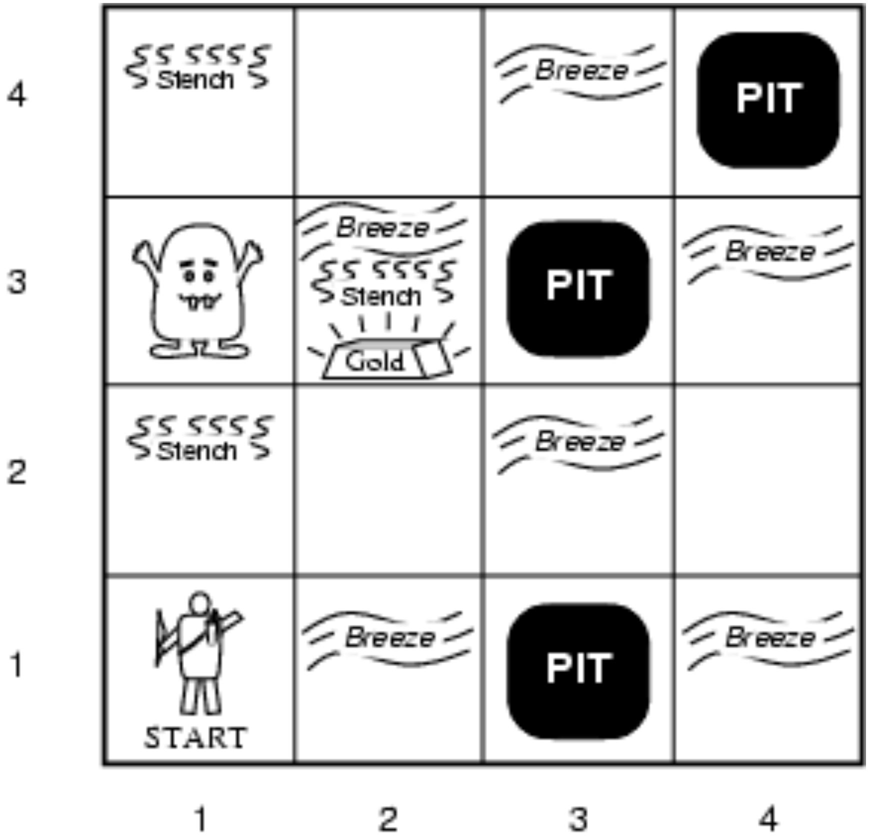
Performance Measure: gold +1000, death -1000, -1 per step, -10 for using the arrow

Environment: cave, rooms, Wumpus, gold

Actuators: motor to move Left, Right, Forward, hands to Grab, Release, and Shoot arrow

Sensors: sensor to capture [Stench, Breeze, Glitter, Bump, Scream]

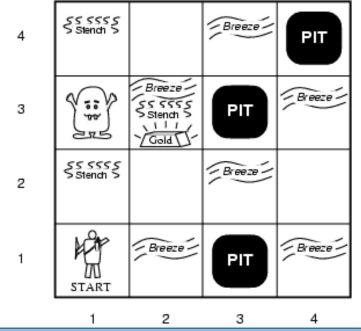
Exploring a wumpus world

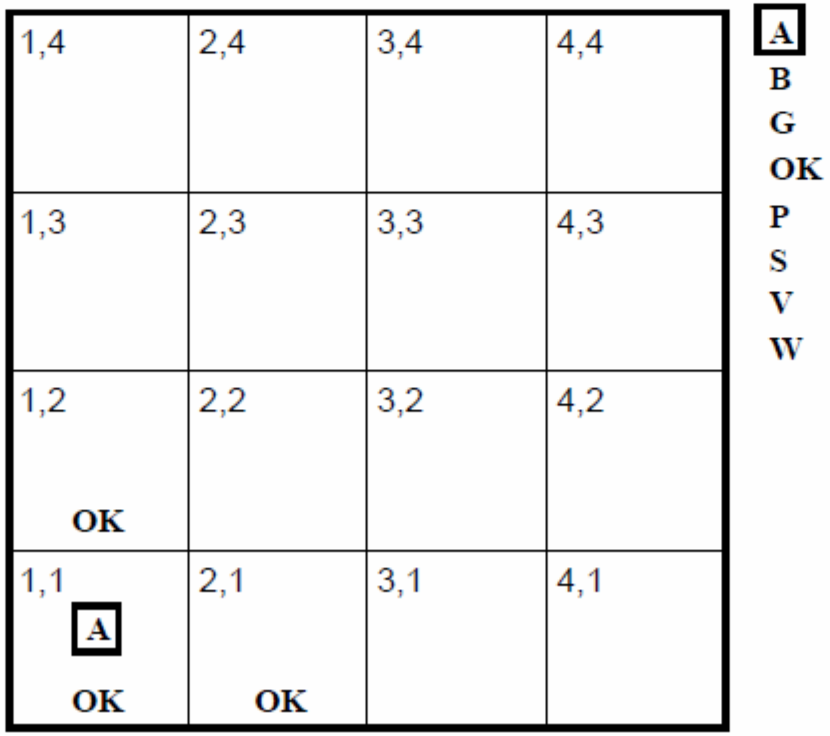


Percept [1,1] : [None, None, None, None, None]
No stench in [1,1] : No wumpus in [1,2] and [2,1]
No breeze in [1,1]: No pit in [1,2] and [2,1]
Action: forward to [2,1]

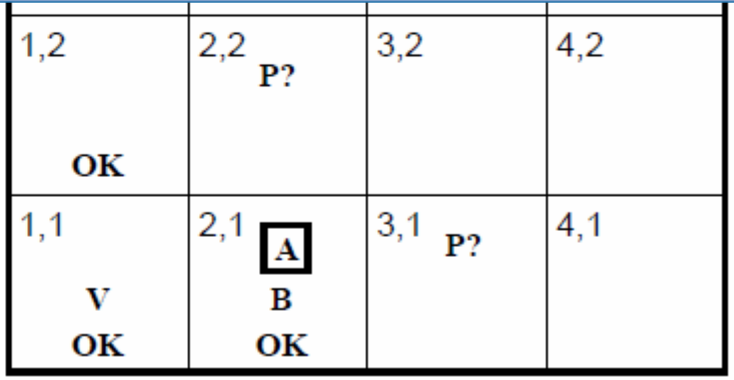
[1,1] : OK (safe)

Exploring a wumpus world (2)





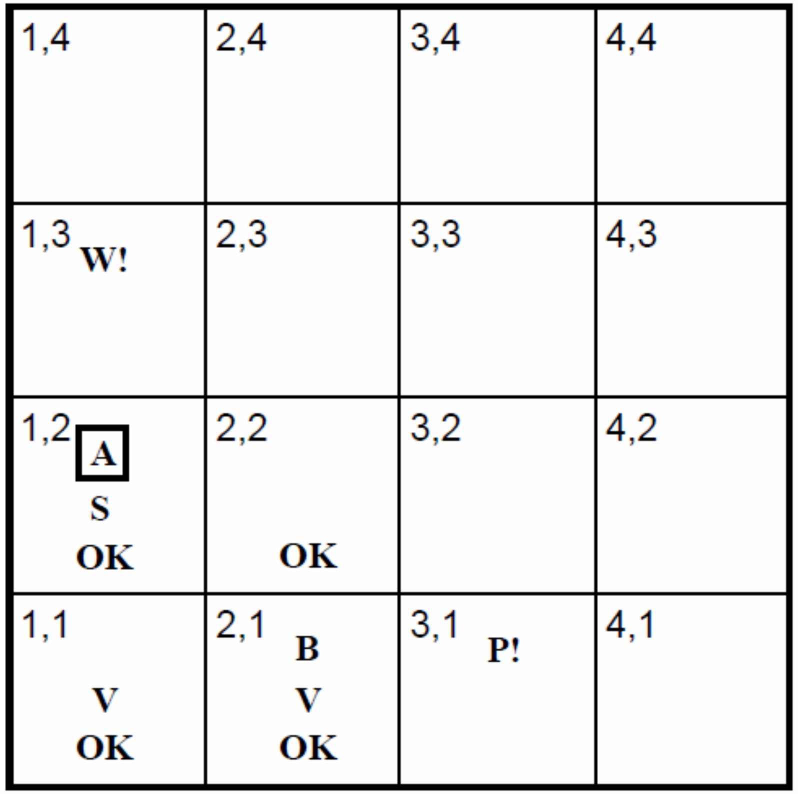
(a)

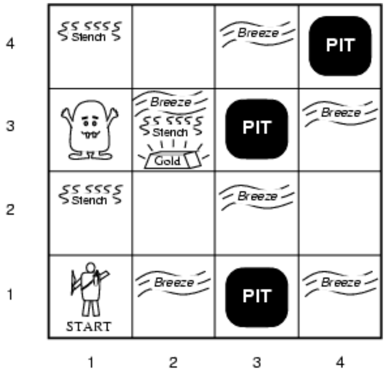


(b)

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]
<u>Set action</u>: go back to [1,1] and forward to [1,2]

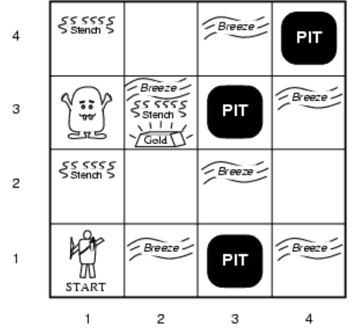
Exploring a wumpus world (3)





Percept [1,2] : [Stench, None, None, None, None]
Stench in [1,2] : 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 [3,1] and [2,2] OK
<u>Set action</u>: go to [2,2]

Exploring a wumpus world (4)



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

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]

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

Generic Knowledge-based Agent

Domain knowledge
→ static

working memory
→ dynamic

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*)) {assert percept}
    action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*)) {reasoning}
    TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*)) {assert action}
    $t \leftarrow t + 1$
    return *action*

Knowledge-based Agent Development

Starting with an empty knowledge-base

Agent designer can TELL sentences one by one

agent knows how to operate in its environment

The designers have no idea about the solution

The designers cannot anticipate all possible situations

The designers cannot anticipate all changes over time

notes :
proposisi : sesuatu yg bisa ditentukan kebenarannya (true/false)

Knowledge Representation

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.

Requirements of Knowledge Representation

- No contradiction
- Each symbol must be unique
- Explain certain objects, relations and attributes
- Efficient manipulation in computer system

Production rules

Semantic networks dan frames

Selecting Knowledge Representation

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

Summary

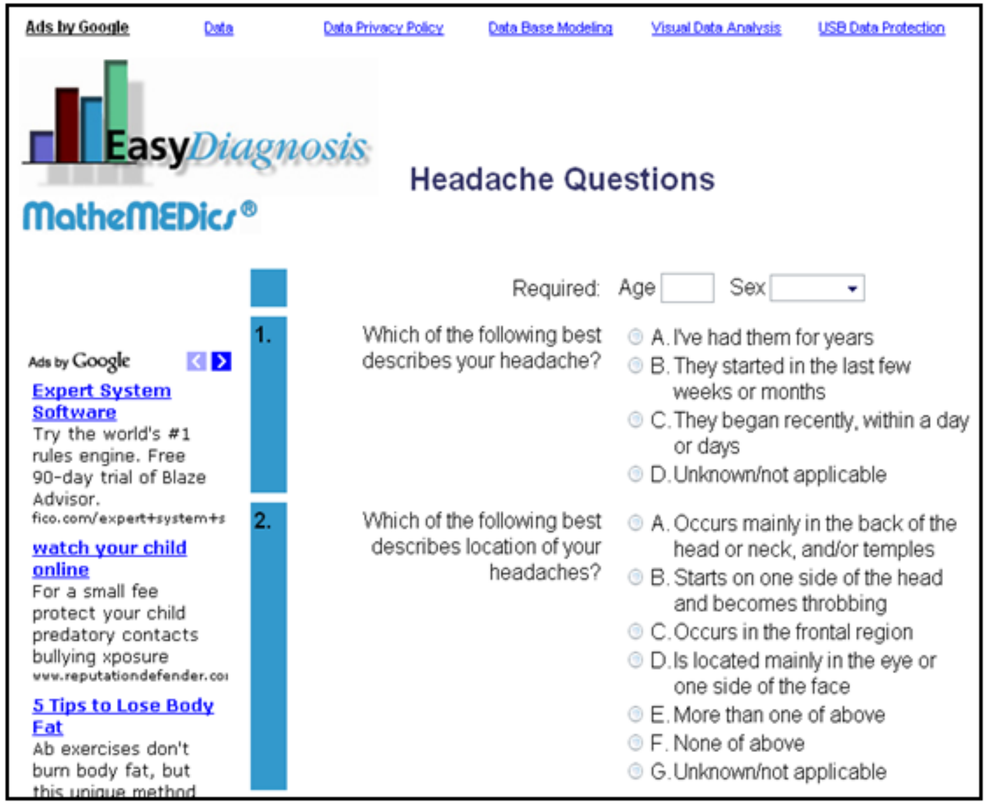
Logical reasoning
Reasoning in Wumpus world
Generic knowledge-based agent
KB agent development
Knowledge representation
KR: requirement & selection
KBS Architecture

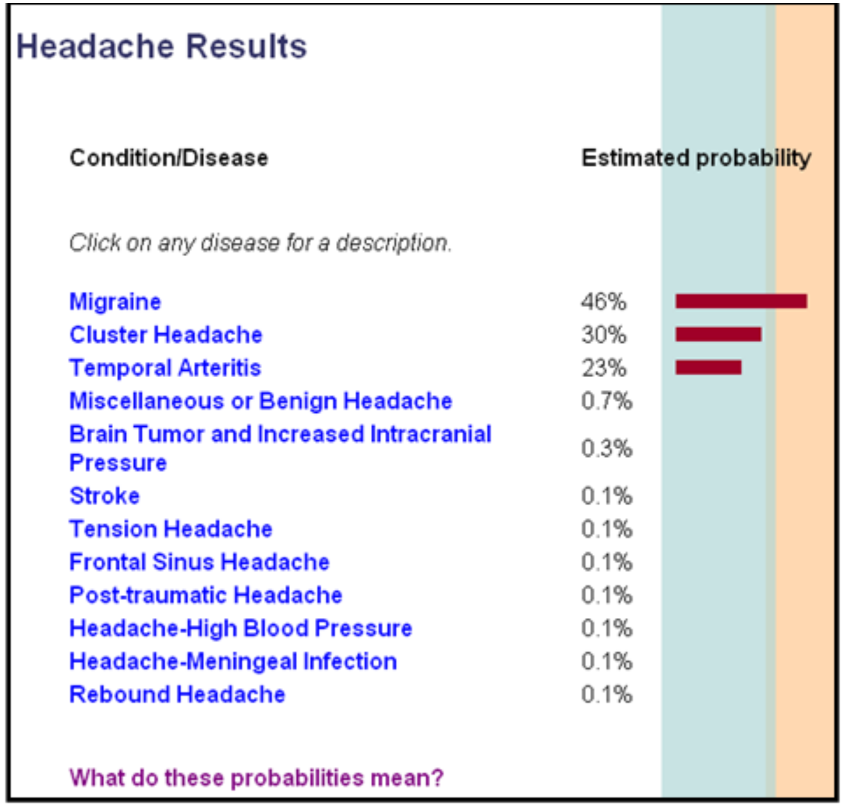
KBS Examples

Contoh Aplikasi

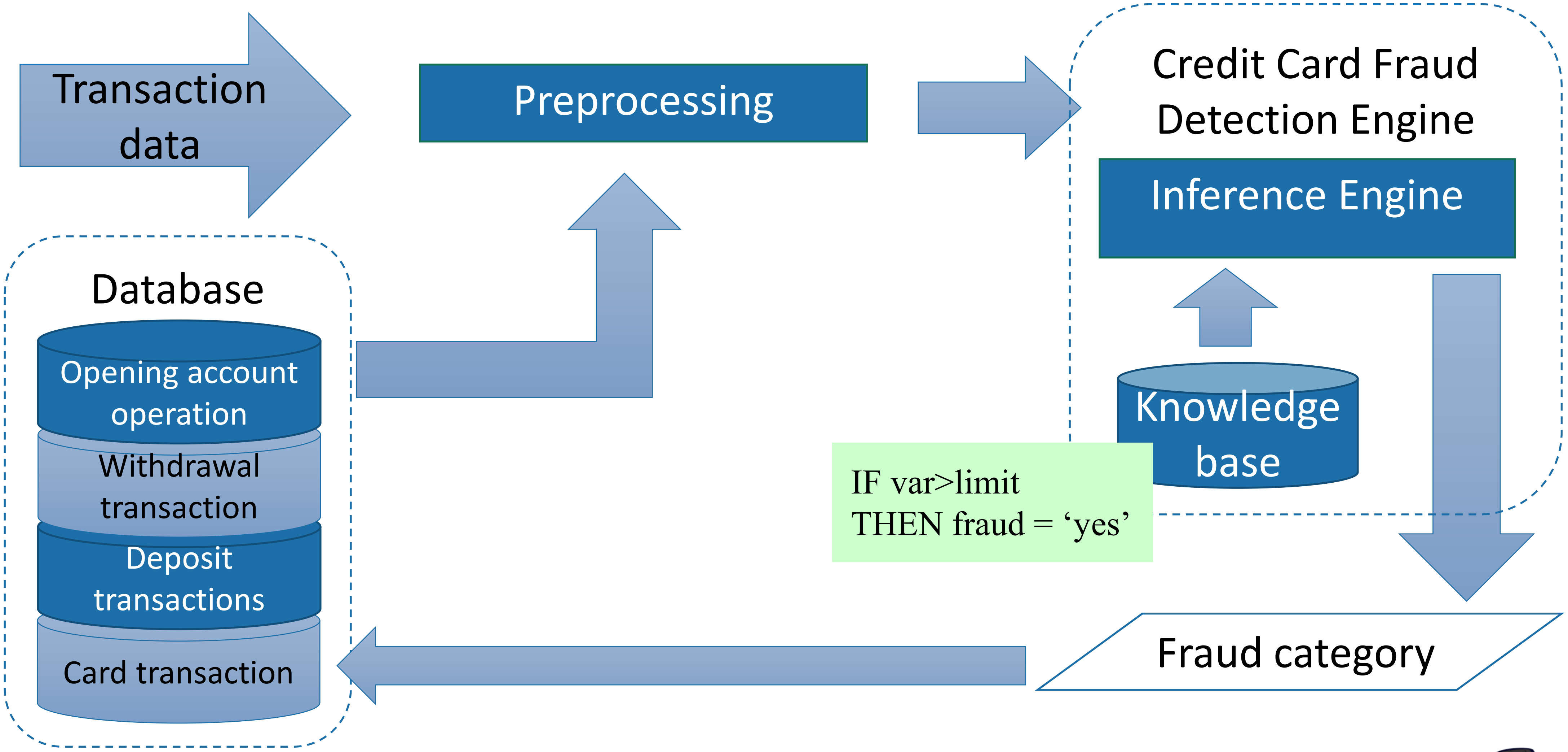
- Kesehatan: BAL2000, LISA, ISABEL, CTSHIV, DxPlain, MedWeaver, The Analyst, FuzzyFluid, Casnet, PUFF, Centaur, EasyDiagnosis, CLEM, VIE-PNN
- Lingkungan: ESS-WWTP, CREWS, CORMIX, HITERM, GCES, Oncologic
- Jaringan: NIDES, AudES, eXpert-BSM, Expert Advisor, Online ES (listrik)
- ITS: ActiveMath, TEST, ELM-ART, SID2002 Math ES, Chest
- Komputer/HW: DART, PEARL, PDAmum
- Manajemen: DXMAS, CESA, FINEVA
- Permainan: FRES, Rogomatic
- Geologi: PROSPECTOR II, DAS
- Pertanian: EXSEL, HABES, DSS4Ag
- Biologi: RIH, PSORTb
- NASA: Weather ES, SHINE
- Lainnya: TTA (teroris), ACAS-PRO (kartu kredit), USLIMITS 2, CATD-RT, HWYCON, SHYSTER (hukum)

EasyDiagnosis Medical Expert System





Credit Card Fraud Detection



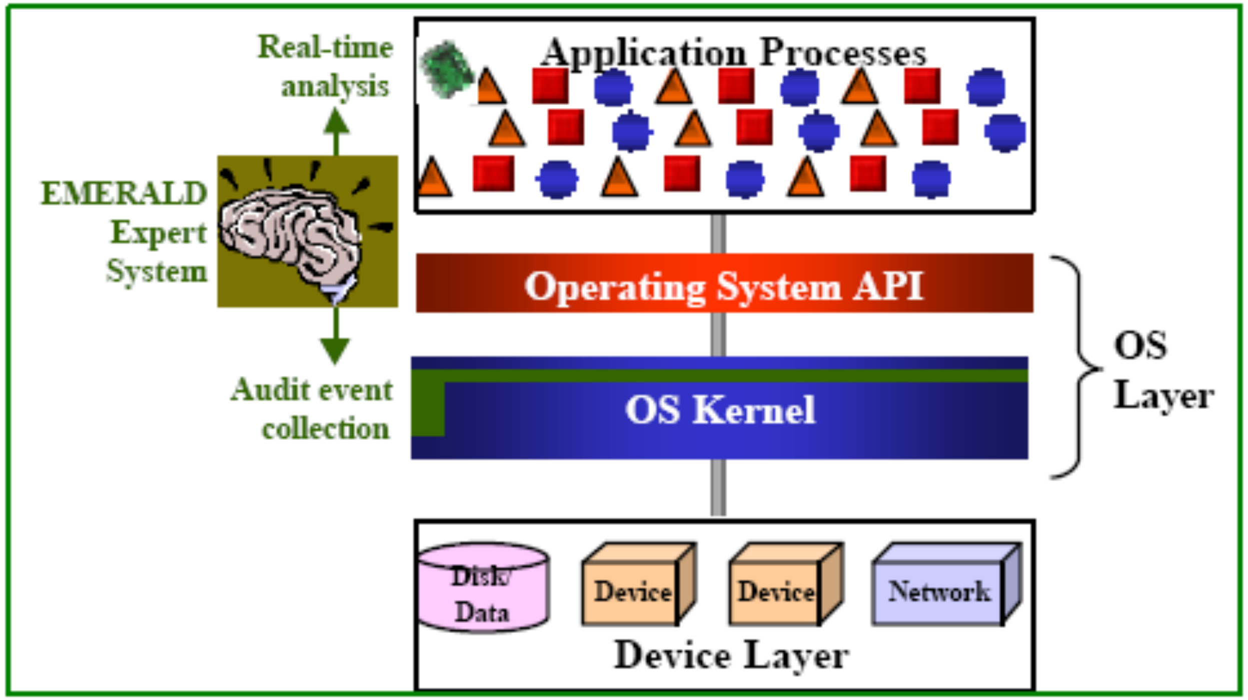
Ogwueleka, F. N. (2011). Data mining application in credit card fraud detection system. *Journal of Engineering Science and Technology*, *6*(3), 311-322.

Green Chemistry Expert System (GCES)

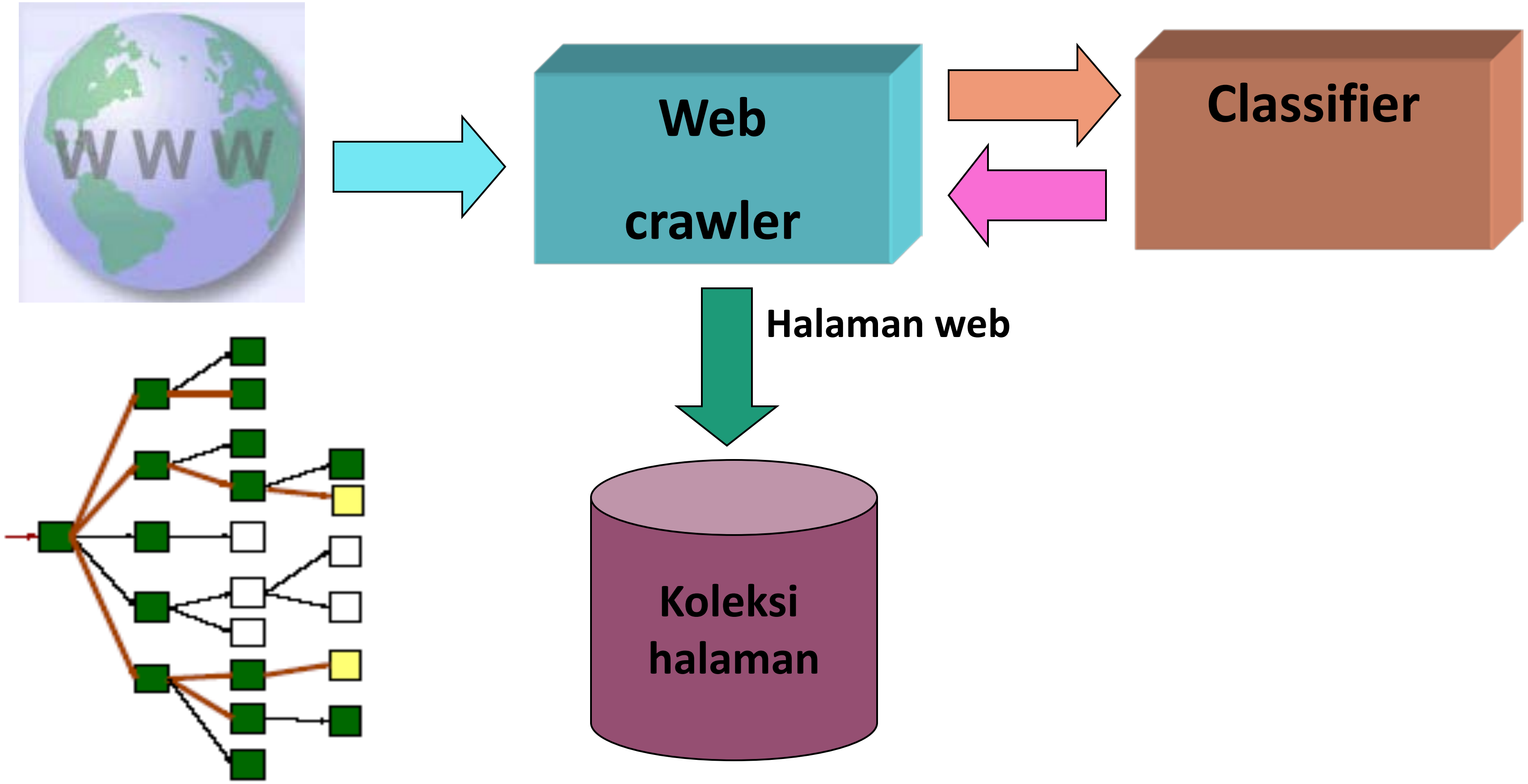
- Developer: EPA (*Evironmental Protection Protection Agency*) Amerika Serikat
  - MS Access, DBMS
- untuk menilai substansi yang berbahaya dalam reaksi kimia sehingga polusi dapat dicegah
- http://www.epa.gov/greenchemistry/pubs/gces.html

eXpert-BSM

- Intrusion Detection Solution for Sun Solaris
- Output: hasil analisis dan alert adanya intrusi pada audit trail dari Sun Solaris
- Sub sistem Emerald ES

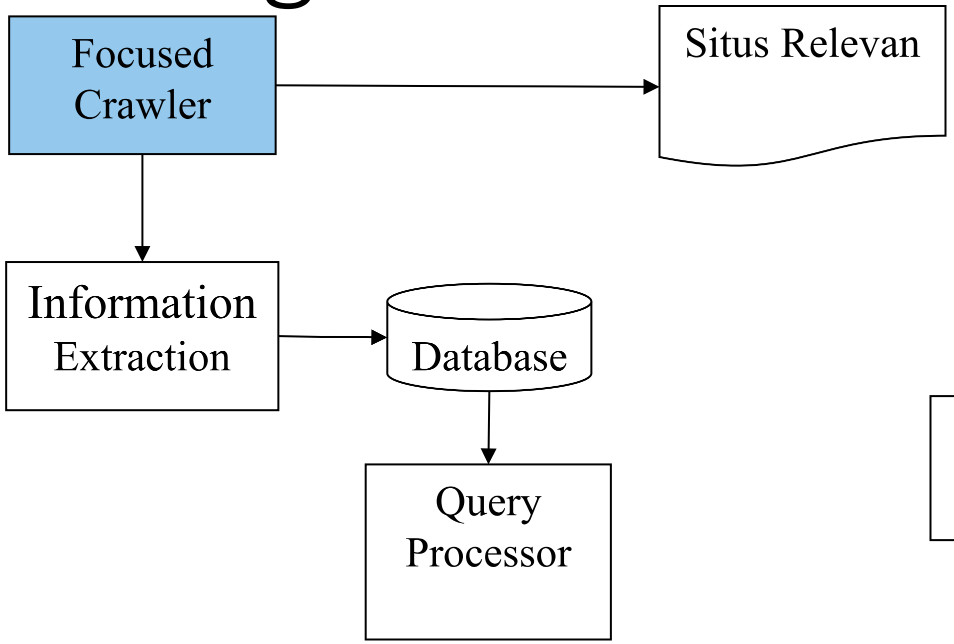


Focused Crawler Domain X



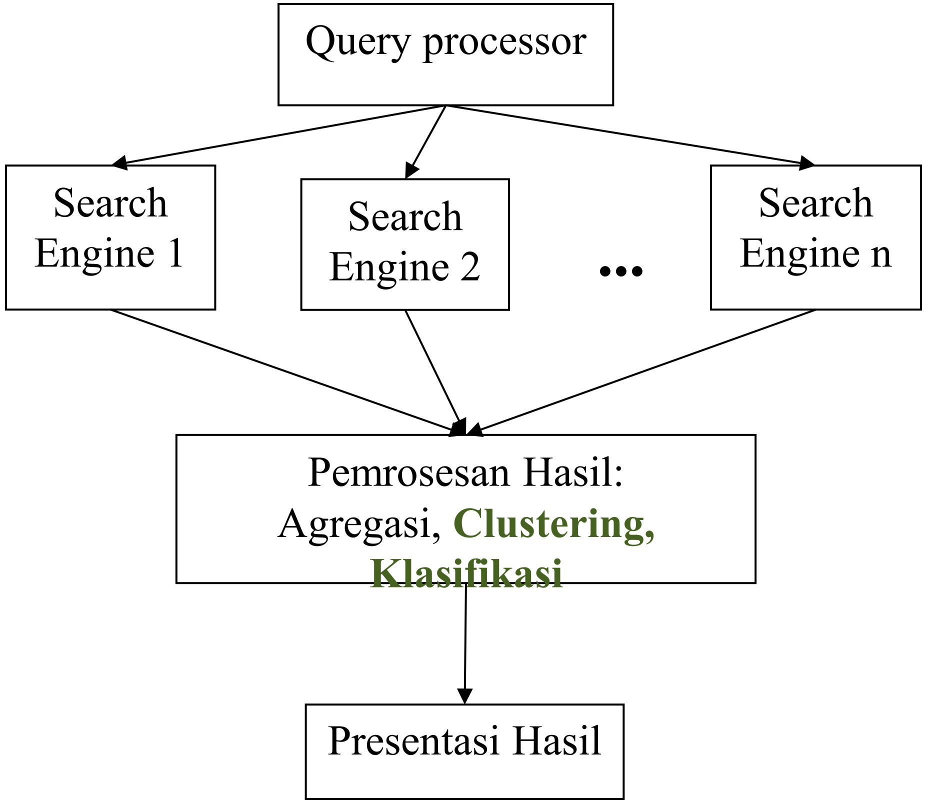
Vertical Search Engine

Search Engine: Architecture



Vertical search engine

Meta search engine



Modul 5: Knowledge-based System

03 Architecture

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Knowledge-based System

Problem-solving method

+

knowledge

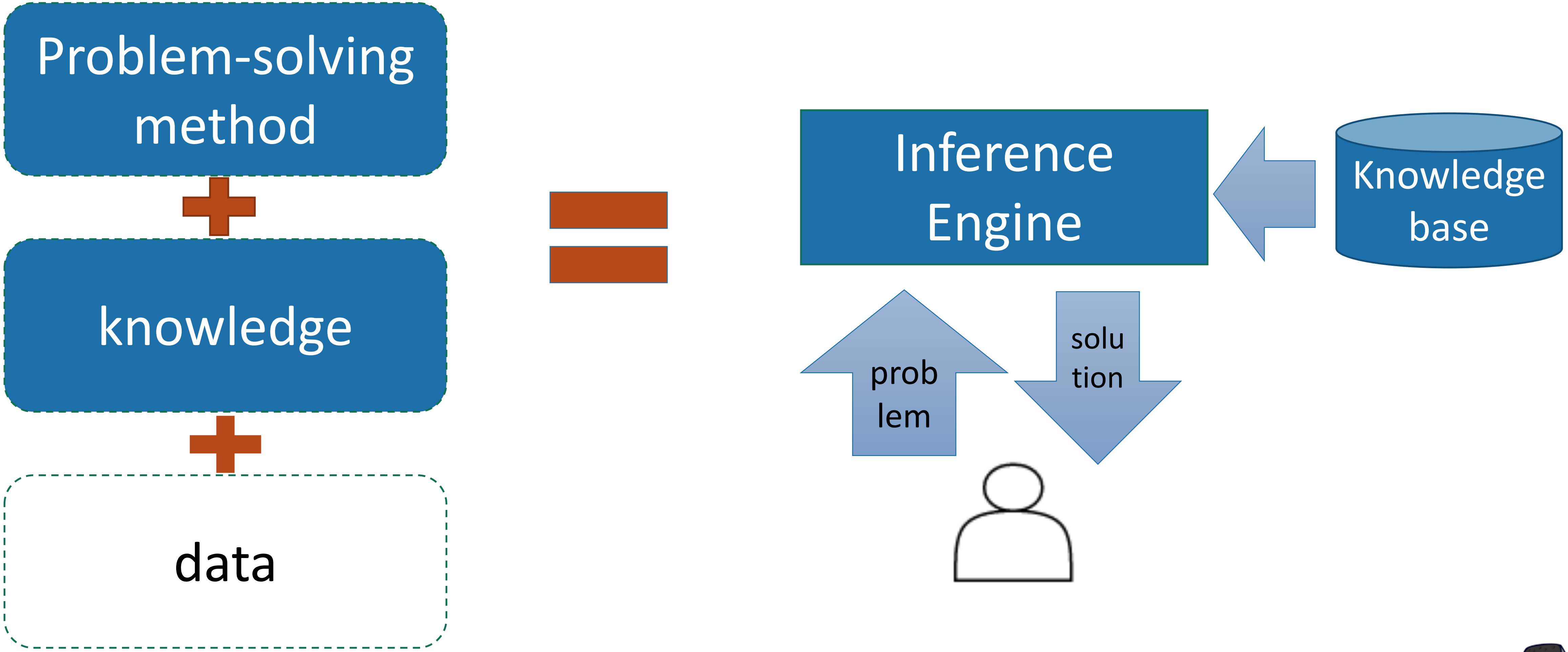
+

data

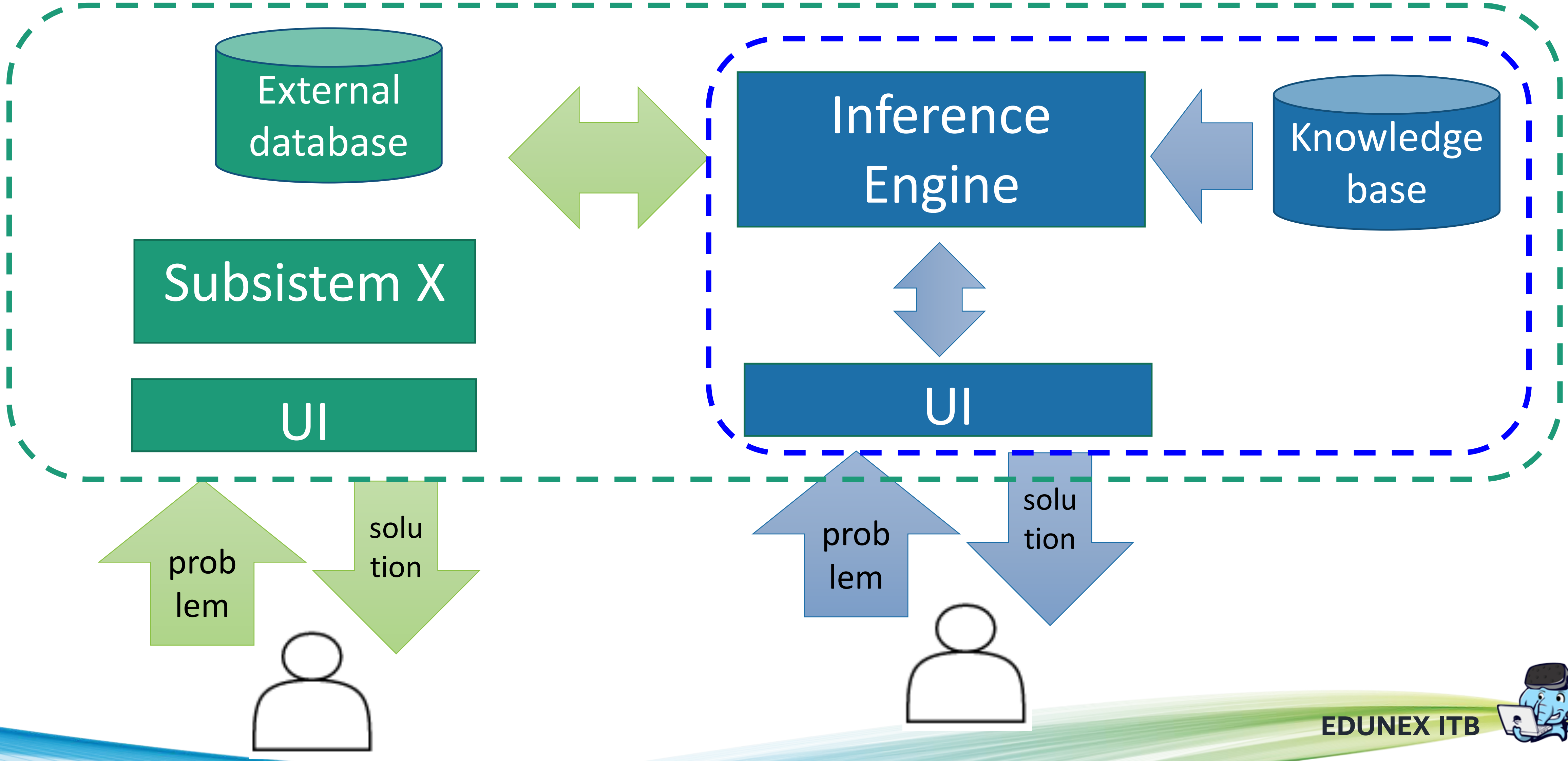
Problem-solving method is an algorithm which determines how domain-specific knowledge is used for solving problems (Puppe, 1993)

Example: knowledge reasoning methods (e.g. forward chaining for rule), general procedures (e.g. partially order plan)

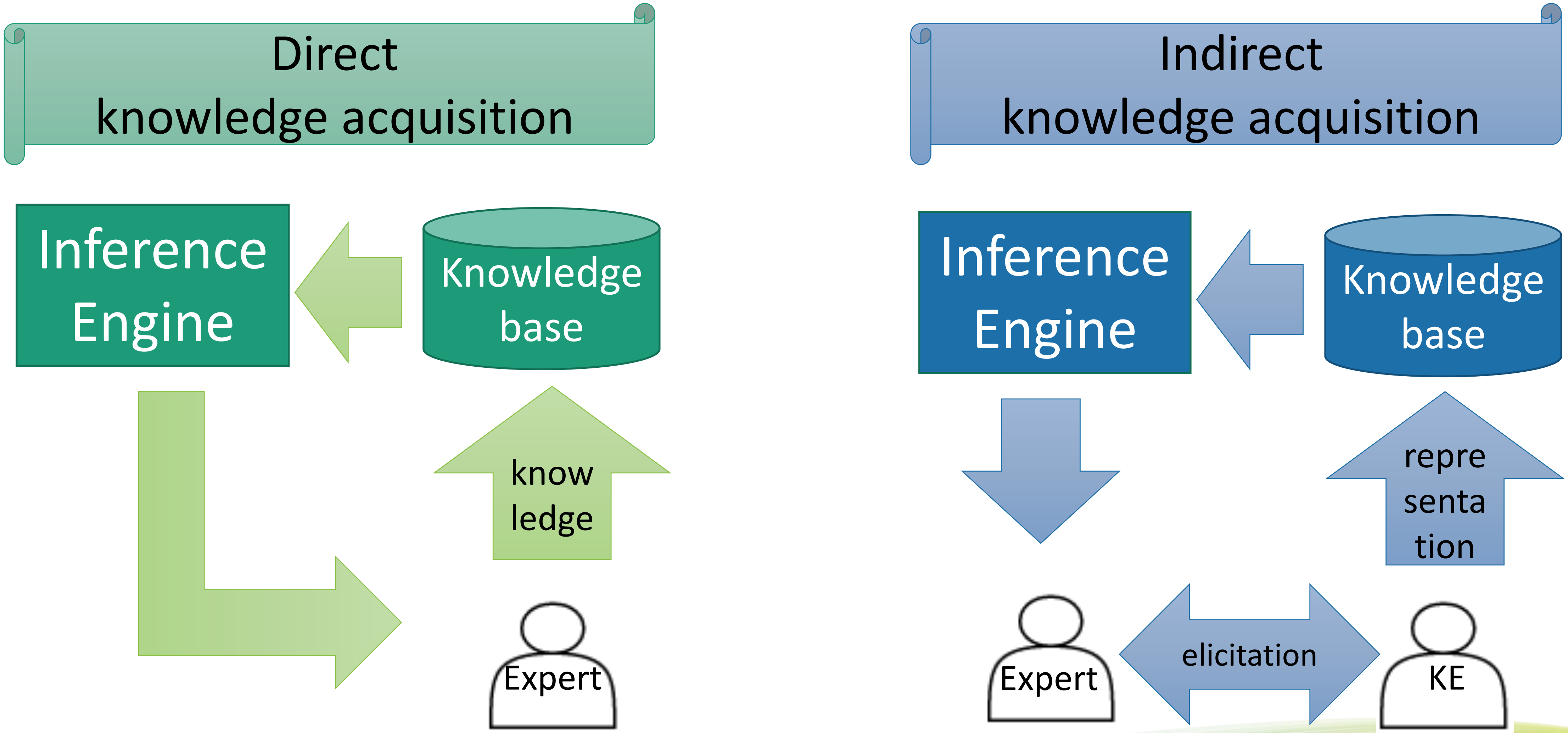
Knowledge-based System: Terminology



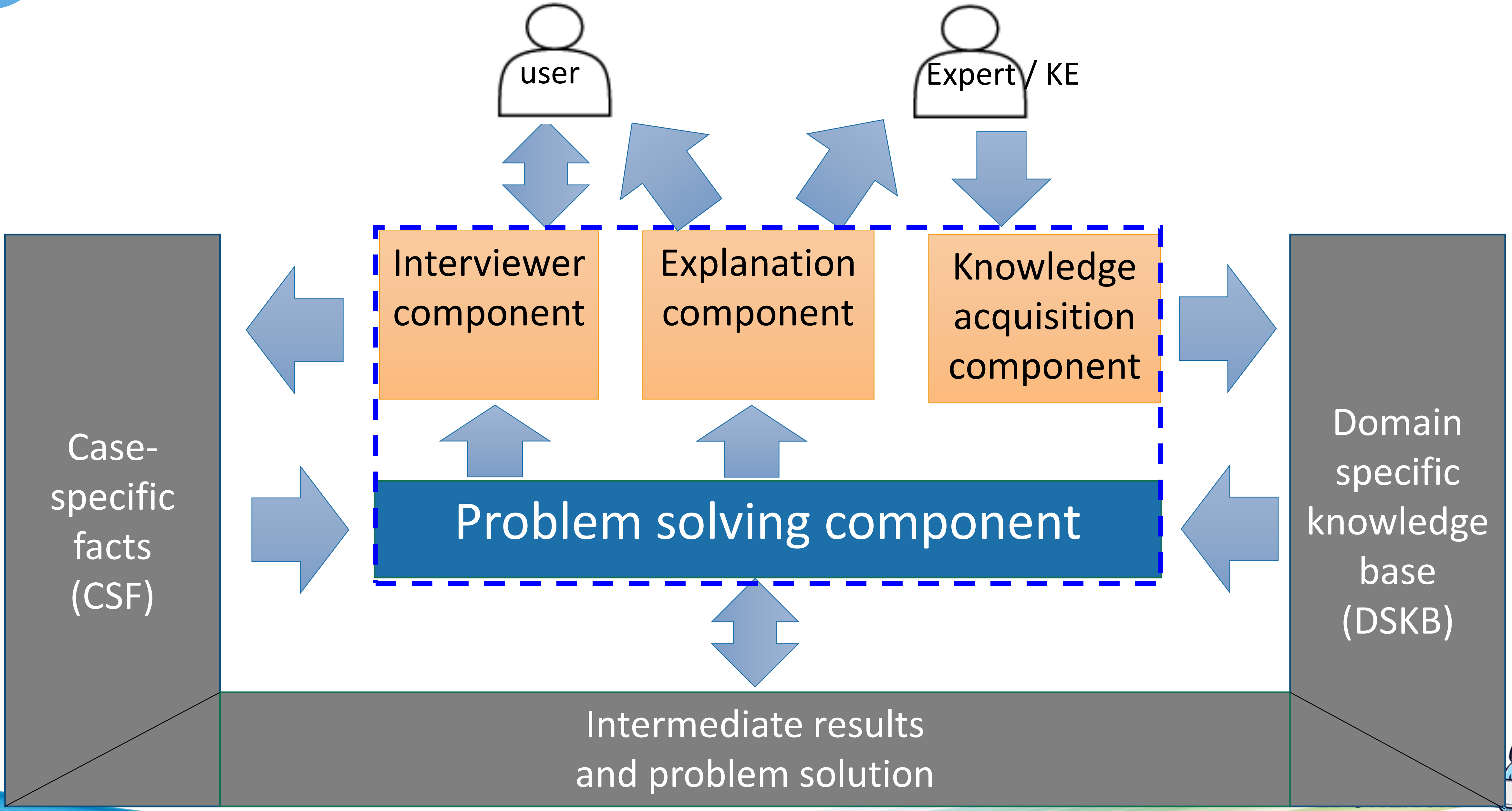
Interactive vs Embedded KBS



Knowledge Acquisition

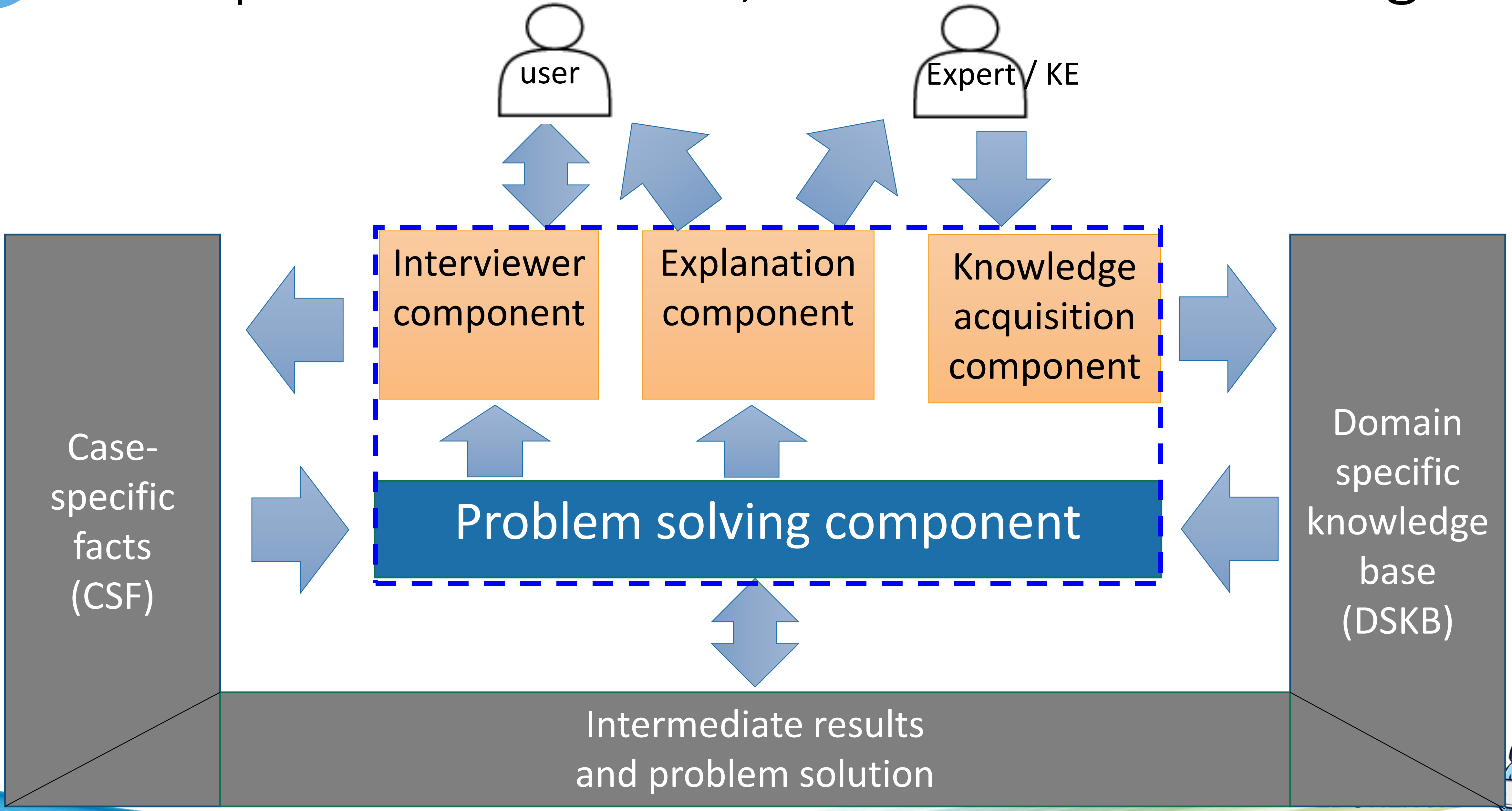


General Architecture of KBS



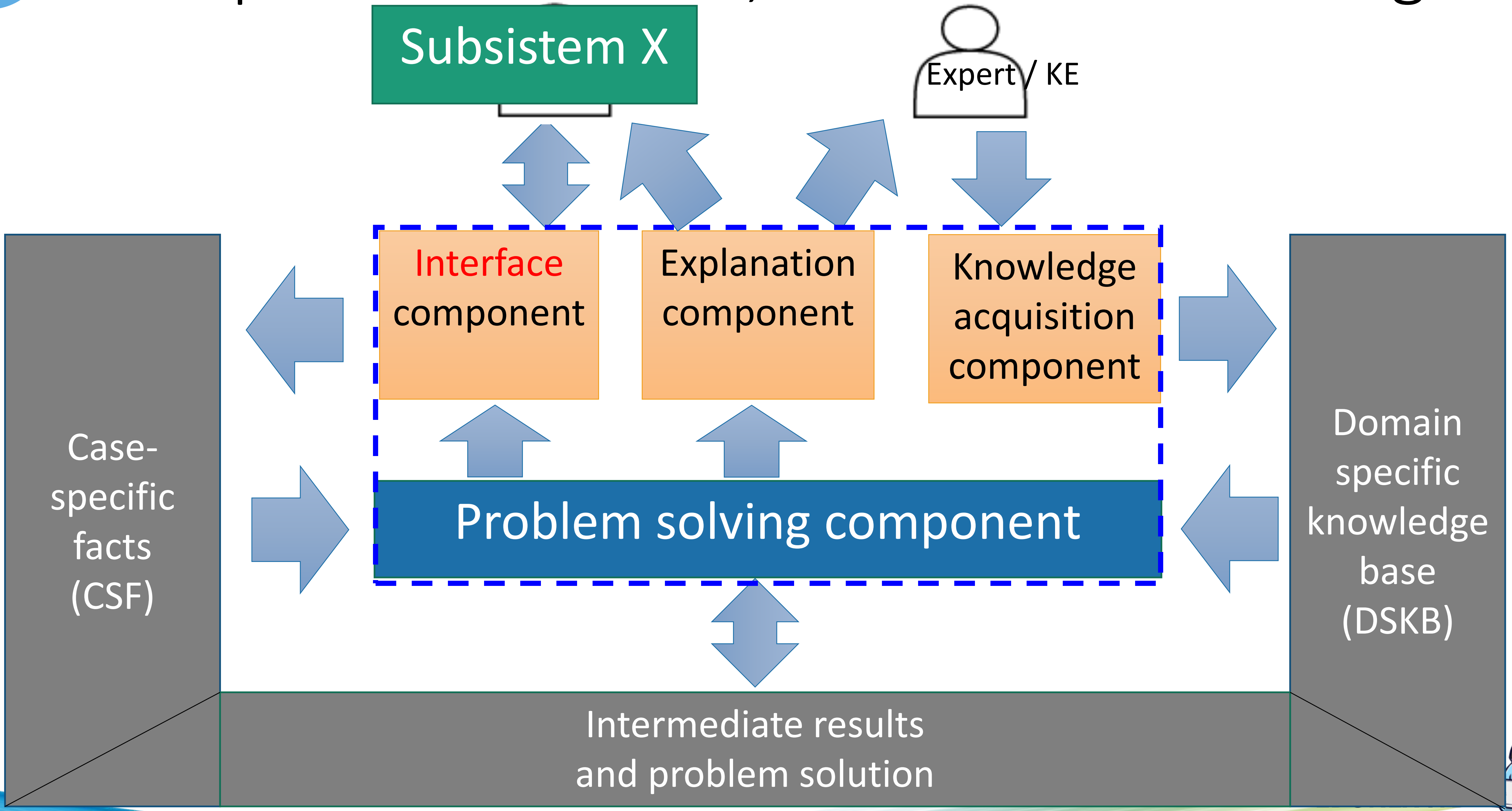
Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Example 1: Interactive, No Update Knowledge



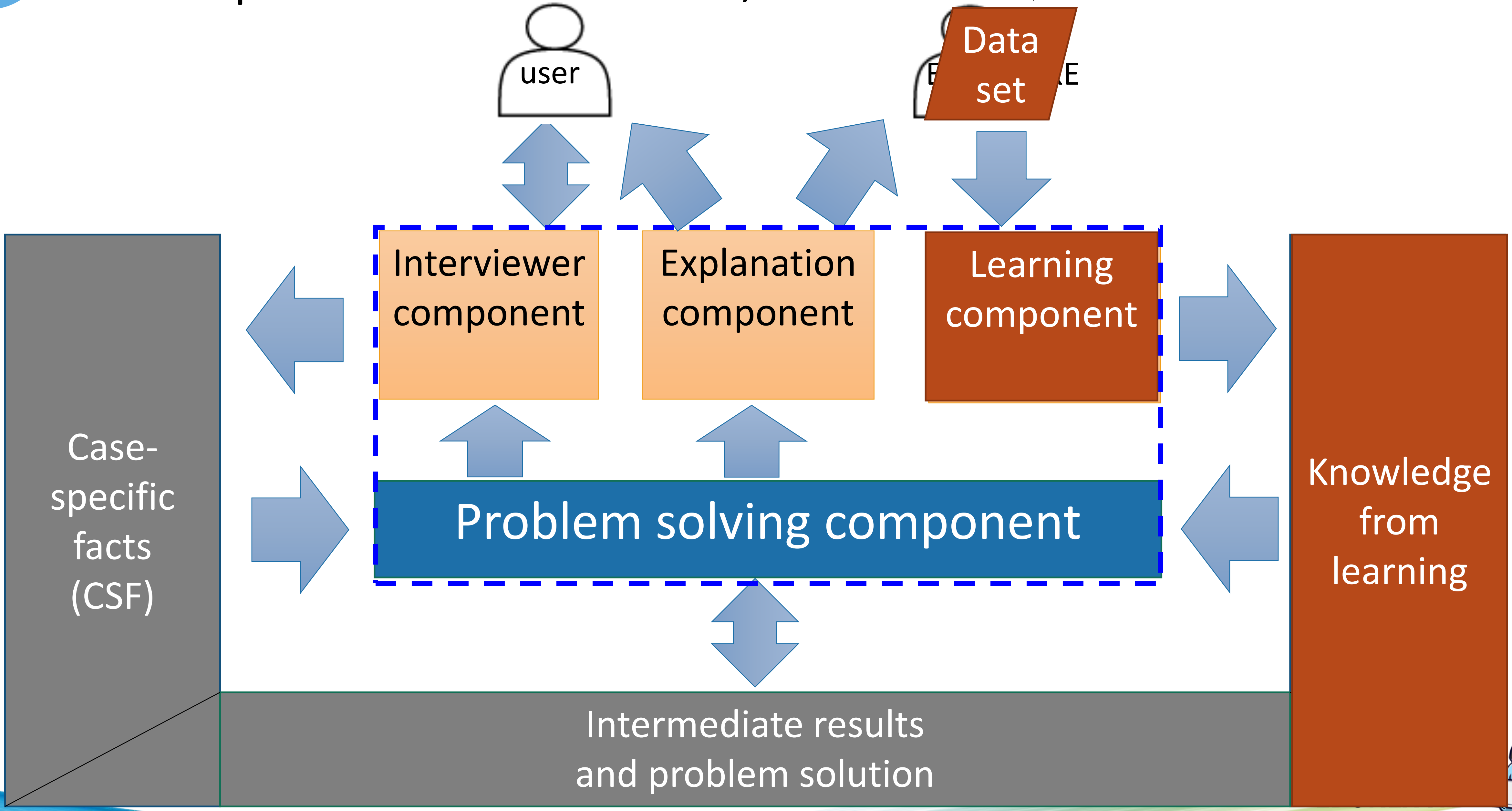
Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Example 2: Embedded, No Update Knowledge



Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Example 3: Interactive, Learning



Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Summary

KBS=PSM+ knowledge+ data

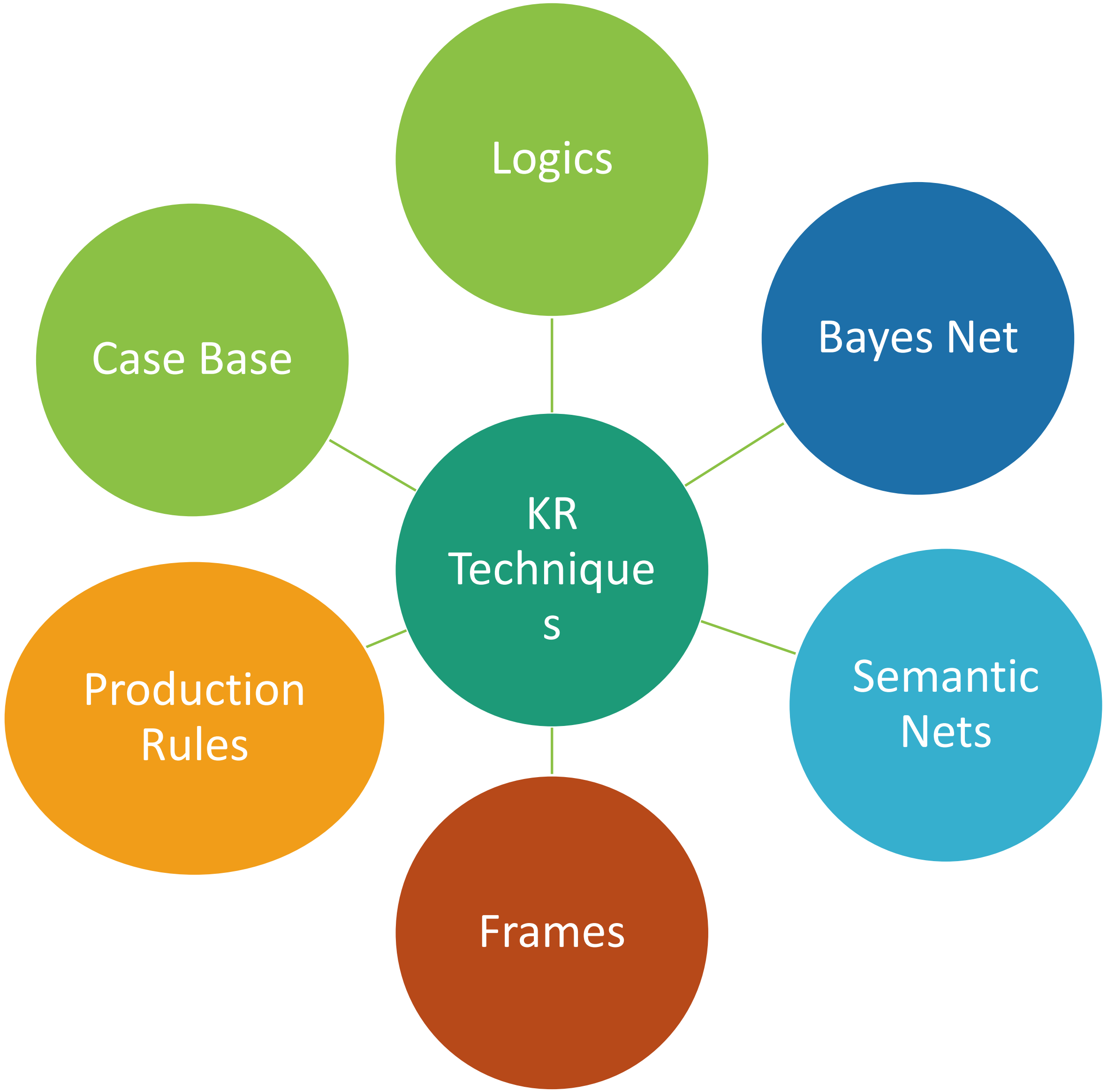
Inference Engine, Knowledge base

Knowledge Acquisition

General Architecture of KBS

Examples of Architecture

Knowledge Representation


Logics
Bayes Net
Case Base
KR
Technique
s
Production
Rules
Semantic
Nets
Frames

Propositional and Predicate Logic

- Sistem berbasis pengetahuan yang menggunakan **logika proposisional atau predikat** untuk merepresentasikan pengetahuan umumnya melibatkan aturan IF-THEN, hubungan antar objek dan kuantifikasi objek.
- Contoh:

a. Smart Building Contol System: menggunakan sensor untuk mendeteksi kondisi lingkungan dan membuat keputusan otomatis berdasarkan aturan logika proposisional. Jika sensor mendeteksi bahwa tidak ada orang di ruangan, lampu akan dimatikan secara otomatis.

b. Automated Theorem Proving: Menggunakan logika predikat untuk membuktikan validitas teorema matematika dengan cara otomatis, seperti dalam program **Prolog** yang dapat menjawab pertanyaan-pertanyaan logis berbasis fakta dan aturan yang sudah ada.

Semantic Networks (Ontology)

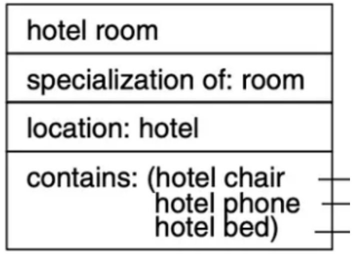
- **Semantic Networks** merepresentasikan pengetahuan dalam bentuk graf, di mana konsep atau objek diwakili sebagai node, dan hubungan di antara konsep tersebut diwakili sebagai edge.
- Contoh:

a. Google Knowledge Graph: untuk pencarian berbasis semantik,

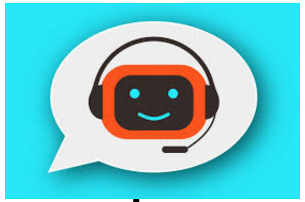
b. PayPal Fraud Detection System: memanfaatkan ontologi untuk memahami pola transaksi yang berpotensi fraud. Sistem ini menganalisis relasi antara akun, lokasi, jenis transaksi, dan sejarah pembayaran untuk mendeteksi aktivitas yang mencurigakan.

c. Penelitian medis: menemukan hubungan baru antara penyakit, gen, dan obat-obatan, yang dapat mendukung penelitian pengobatan baru.

Frames



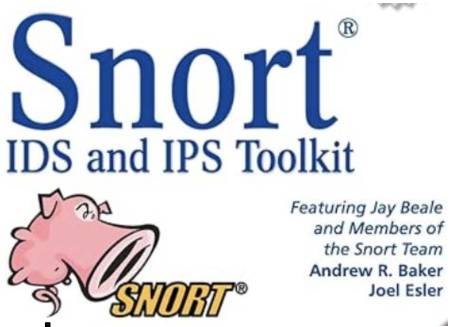
- **Frames** adalah struktur data yang digunakan untuk merepresentasikan pengetahuan dalam bentuk slot (tempat penyimpanan informasi) dan filler (nilai atau informasi yang mengisi slot).
- Contoh:



a. Chatbot dan Asisten Virtual: menggunakan frame untuk memahami konteks pertanyaan pengguna dan menjawab sesuai konteks.

b. Robotic Navigation Systems: frames digunakan untuk merepresentasikan pengetahuan tentang lingkungan robot dan tindakan yang bisa diambil oleh robot, misalnya robot pembersih, robot industri manufaktur.

Production Rule



- **Production rule** adalah representasi pengetahuan dalam bentuk aturan berbasis kondisi dan aksi, atau sering disebut dengan **if-then rules**.
- Contoh: Medical Expert Systems

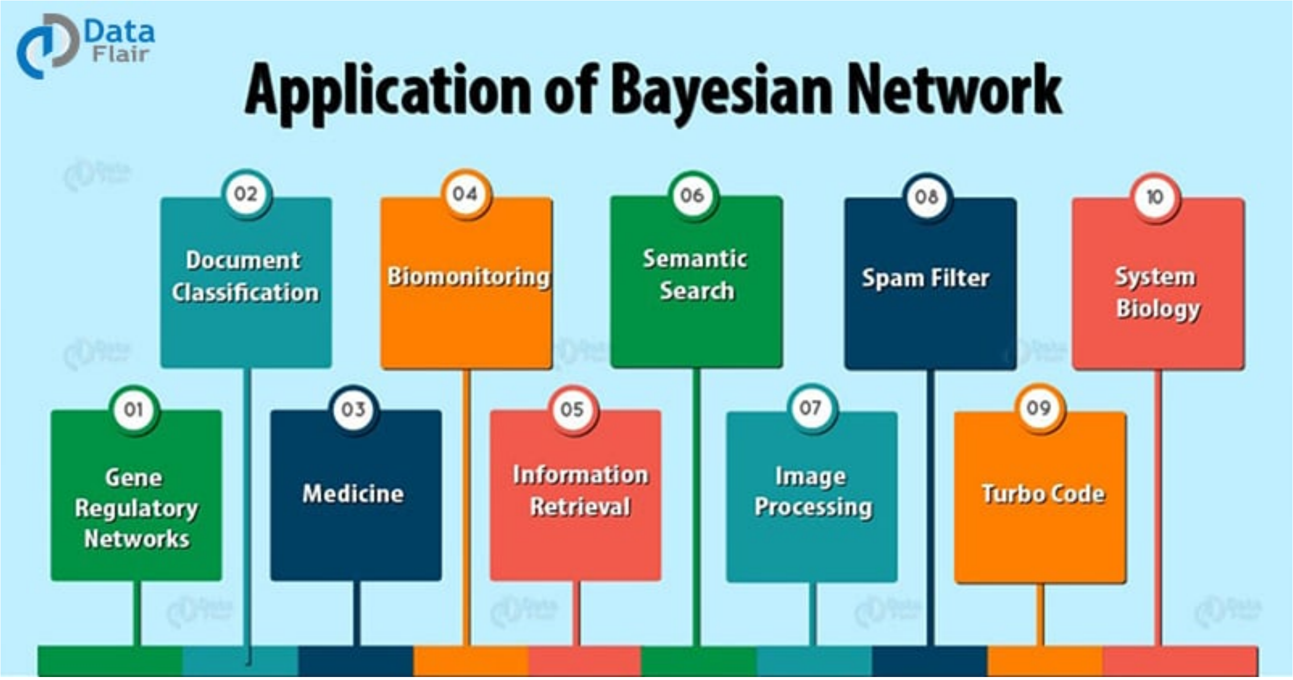
a. **MYCIN:** Sistem pakar medis MYCIN yang dikembangkan pada tahun 1970-an untuk mendiagnosis infeksi bakteri menggunakan production rules. MYCIN bekerja dengan aturan-aturan seperti "If pasien memiliki gejala X dan hasil tes Y, then infeksi bakteri Z terindikasi.“

b. **Snort:** Snort adalah salah satu sistem deteksi intrusi open-source yang populer dan menggunakan production rules untuk mendeteksi aktivitas mencurigakan atau serangan dalam jaringan komputer. Aturan-aturan ini dapat disesuaikan untuk mendeteksi berbagai jenis ancaman seperti serangan DDoS, brute force, atau malware.

Case-Based Reasoning

- **Case-Based Reasoning (CBR)** adalah metode representasi pengetahuan di mana solusi dari masalah-masalah sebelumnya (kasus) digunakan kembali untuk memecahkan masalah baru yang serupa. Penalaran ini bergantung pada basis data kasus yang mencakup solusi atau keputusan yang diambil dalam skenario sebelumnya.
- Contoh:
- **Case-Based Radiology System:** Beberapa sistem radiologi berbasis CBR membantu ahli radiologi untuk mendiagnosis gambar medis berdasarkan kasus-kasus gambar yang telah didiagnosis sebelumnya, seperti mendeteksi kanker atau kerusakan organ.
- **HelpDesk Systems:** Banyak perusahaan teknologi menggunakan sistem CBR untuk memberikan dukungan teknis kepada pelanggan. Saat pelanggan melaporkan masalah teknis, sistem CBR mengacu pada kasus-kasus teknis yang telah dipecahkan sebelumnya dan menyarankan solusi.

Bayes Network



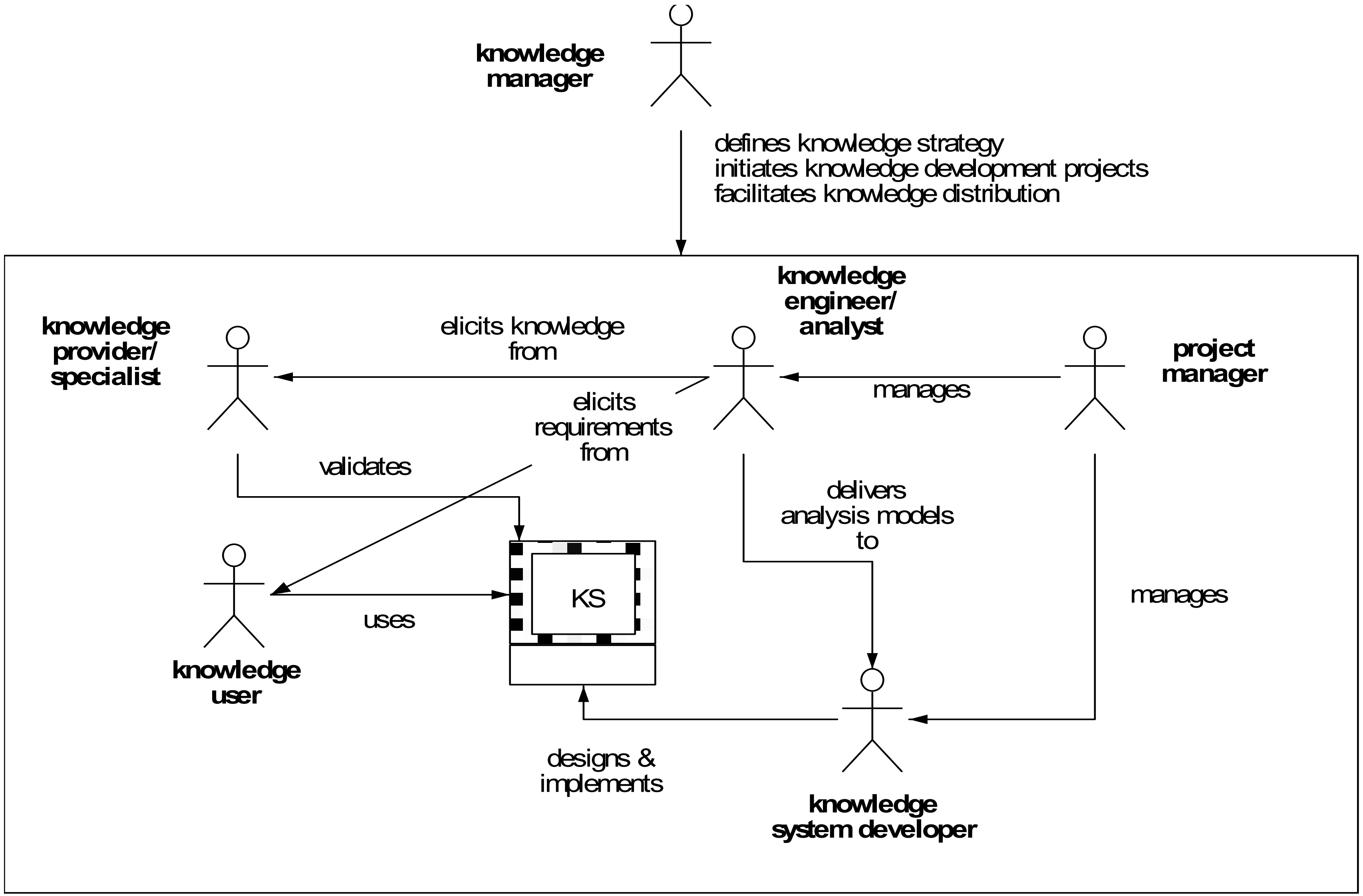
Source: https://data-flair.training/blogs/bayesian-network-applications/

- Reasoning under uncertainty
- Represent uncertainty not only by disjunction (Logic), but also likelihood (probability)

→ Next Course

Knowledge Engineering

KBS Developer



Rekayasa Pengetahuan

- Akuisisi pengetahuan dalam suatu domain dari satu atau lebih sumber non-elektronik dan konversinya ke dalam suatu bentuk yang dapat digunakan oleh komputer untuk memecahkan persoalan yang umumnya hanya dapat dipecahkan oleh pakar domain tersebut.

Akuisisi Pengetahuan (KA)

- KA=knowledge ***elicitation*** + ***representation***
- knowledge elicitation
  - Proses ekstraksi pengetahuan domain dan strategik dari pakar
  - Interview antara KE dan pakar
  - a cyclical process
- Knowledge representation
  - Proses merepresentasikan pengetahuan hasil ekstraksi ke suatu bentuk formal

Task dalam Knowledge Elicitation

- Pada setiap iterasi:
  - ***collect*** knowledge (e.g. from expert)
  - determine ***key concepts*** in problem domain
  - establish ***relationships*** between various concepts in problem domain
  - decide h***ow knowledge is represented*** in KBS
  - determine what knowledge needs to be collected in the next cycle

Tahapan Akuisisi Pengetahuan

- Identification
  - Identifikasi karakteristik masalah
- Conceptualization
  - Menemukan konsep2 untuk merepresentasikan pengetahuan
- Formalization
  - Design struktur untuk mengorganisasikan pengetahuan
- Implementation
  - Formulasi pengetahuan ke bentuk runnable program
- Testing
  - Validasi pengetahuan

Teknik Akuisisi Pengetahuan

- Manual:
  1. Interview
  2. Observasi
  3. Intuitive: tukar peran Knowledge Engineer dan pakar
- Otomatis:
  - Menggunakan tools untuk memfasilitasi akuisisi
  - Tools untuk pakar
  - Tools machine learning

INSTITUT TEKNOLOGI BANDUNG

1920

Modul 12: Probabilistic Reasoning System

Introduction to Probabilistic Reasoning System

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(Artificial Intelligence)

Probabilistic Reasoning System (PRS)

Supervised Learning with Uncertainty (Non Deterministic)

Probability theory provides a quantitative way of encoding likelihood

Probability is a model of degree of belief

Given state(s) e, what is the probabilty that x happens → P(x|e)

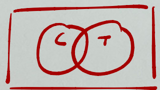
- Joint Probability Distribution
- Bayesian/ Belief Network

Joint Probability Distribution

- Random variables
  - Function: discrete domain → [0, 1]
  - Sums to 1 over the domain
    - Raining is a propositional random variable
    - Raining(true) = 0.2
      - P(Raining = true) = 0.2
    - Raining(false) = 0.8
      - P(Raining = false) = 0.8
- Joint distribution
  - Probability assignment to all combinations of values of random variables

Inference using Joint Probability Distribution

	toothache		¬ toothache	
	catch	¬ catch	catch	¬ catch
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

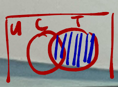


$P(C \cup T)$
$= P(C) + P(T) - P(C \cap T)$.

- For any proposition ϕ, sum the atomic events where it is true: $P(\phi) = \Sigma_{\omega:\omega \models \phi} P(\omega)$
- $P(toothache) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$
- $P(cavity) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2$
- P(cavity U toothache) = ?
- $P(\neg cavity \mid toothache)$ $= P(\neg cavity \cap toothache) / P(toothache)$
  $= (0.016+0.064) / (0.108 + 0.012 + 0.016 + 0.064)$
  $= 0.4$

given

↳ kalau udah gini, domainnya udah bukan U lagi. tapi T aja



domainnya yg biru aja.

Inference using Joint Probability Distribution

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

- If you have n binary propositional variables □ requires 2^n numbers to build Joint Probability Distribution

□ Bayesian Network (We want to exploit independences in the domain)

Bayes' Rule:

$P(A \mid B) = P(A \cap B) / P(B)$

$\quad = P(B \mid A)\, P(A) / P(B)$

Conditioning:

$P(A) \quad = P(A \mid B)\, P(B) + P(A \mid \neg B)\, P(\neg B)$

$\quad = P(A \cap B) + P(A \cap \neg B)$

$P(A|B) = \dfrac{P(A \cap B)}{P(B)}$

$P(A \cap B) = P(A|B) \cdot P(B)$

$= P(B|A) \cdot P(A)$

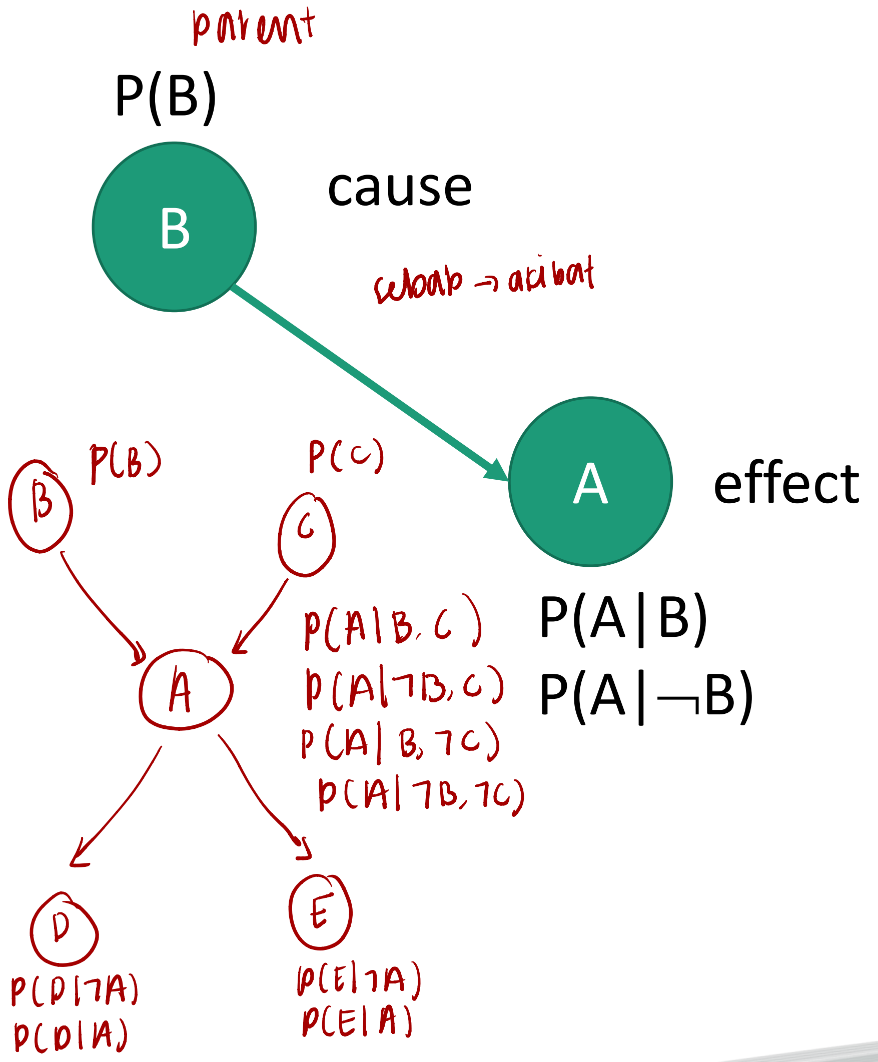
$P(\neg A|B) = 1 - P(A|B)$

$P(\neg A|\neg B) = 1 - P(A|\neg B)$

Structure of Bayesian Network

Nodes (variable)

Directed arc



Should be
Directed Acyclic Graph (DAG)

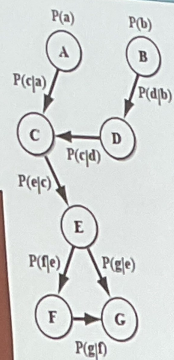
karena nanti muter²
dan ga tau penyebabnya
(masa A → B, terus B → A ?)

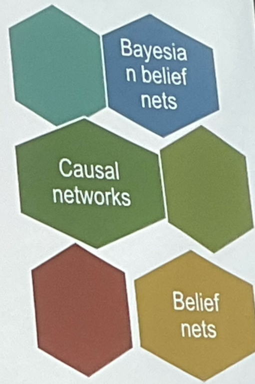
Structure of Bayesian Network

Representation of causal dependencies graphically (Hart et al.,2001)

BN have capability probabilistic reasoning like full joint probability distribution. It can answer any question about the domain.

-- How we exploit Independence?



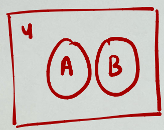


Independence

- A and B are independent iff
  - $P(A \cap B) = P(A) \cdot P(B)$
  - $P(A \mid B) = P(A)$
  - $P(B \mid A) = P(B)$

  } ga ada effect satu sama lain

- Independence is essential for efficient probabilistic reasoning
- A and B are conditionally independent given C iff
  - $P(A \mid B, C) = P(A \mid C)$
  - $P(B \mid A, C) = P(B \mid C)$
  - $P(A \cap B \mid C) = P(A \mid C) \cdot P(B \mid C)$



Example of Independence

- X is late (X)
- Traffic Jam (T)
- Y is late (Y)
- None of these propositions are independent of one other
- X and Y are conditionally independent given T

Types of Connections in Bayesian Network

	Forward	Backward
Serial	$C \rightarrow B \rightarrow A$	$C \leftarrow B \leftarrow A$
Diverging	$C \leftarrow B \rightarrow A$	
Converging	$C \rightarrow B \leftarrow A$	

Why? To exploit Independence

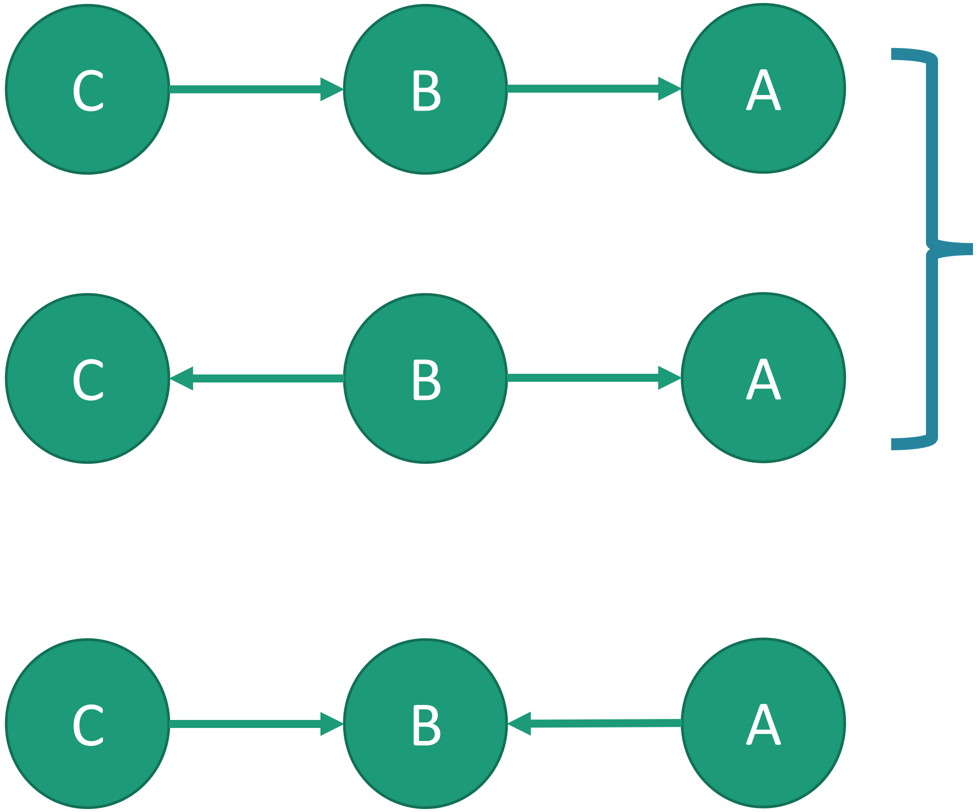
↳ diawal tidak berhubungan, pas ada informasi dari B atau informasi dari variabel lain ke B, C dan A bisa aja berhubungan

Independence in Connection

Serial

Diverging

Converging



Knowing C will tell us about A, but if we know B, knowing C will tell us nothing about A (C and A conditionally independent or d-separated)

Knowing C will tell us nothing about A without knowing B, but if we see evidence about B, C and A becomes dependent

Example of Bayesian Network Structure

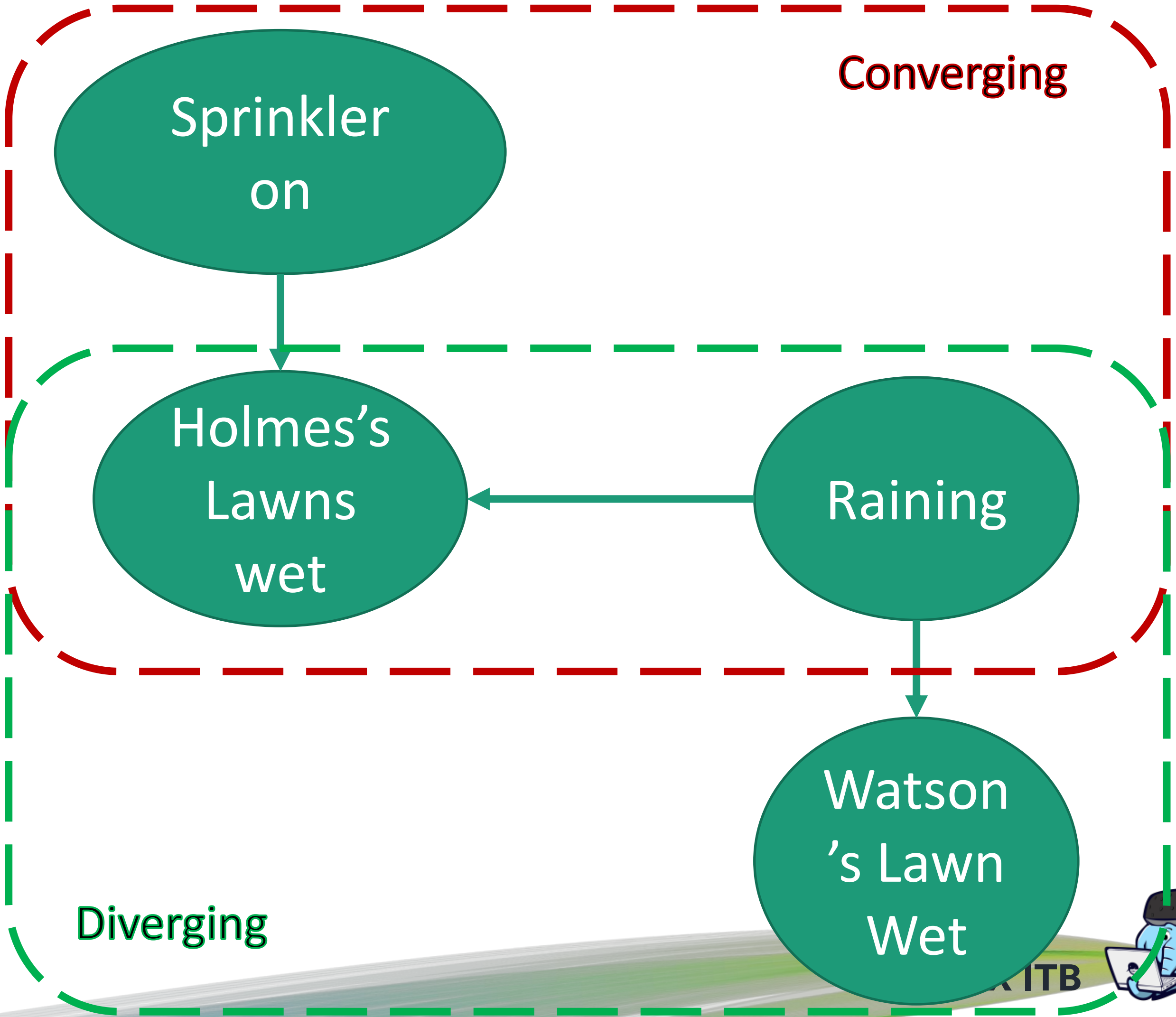
(from Kaelbling of MIT)

Holmes and Watson are neighbor

Holmes wakes up to find his lawn wet

Sprinkler on?

Raining?



Example of Bayesian Network with Likelihood

(from Kaelbling of MIT)

Holmes and Watson are neighbor

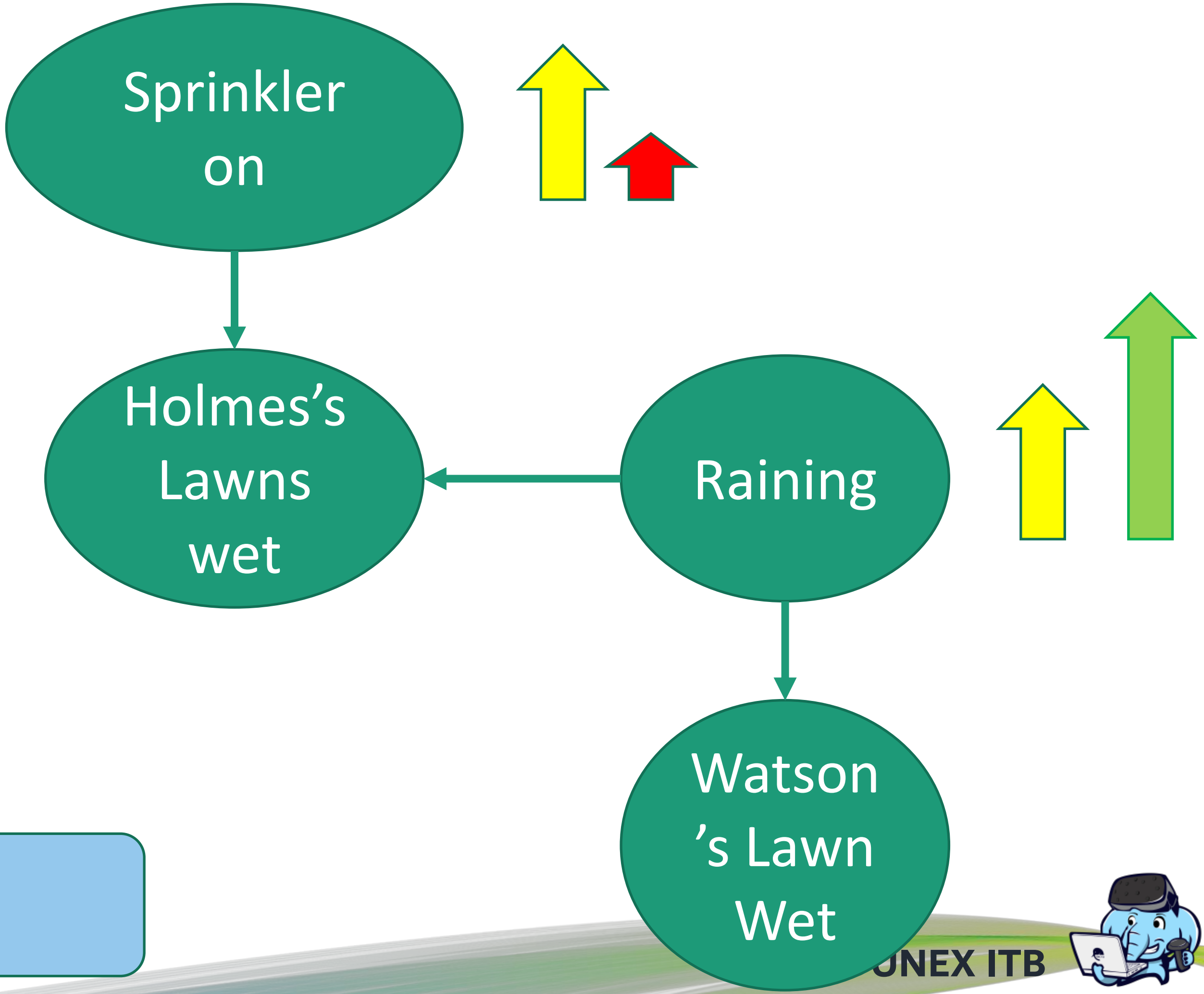
Holmes wakes up to find his lawn wet

Sprinkler on?

Raining?

Watson’s lawn and he sees it is wet too

Likelihood → Probability



Modul 12: Probabilistic Reasoning System

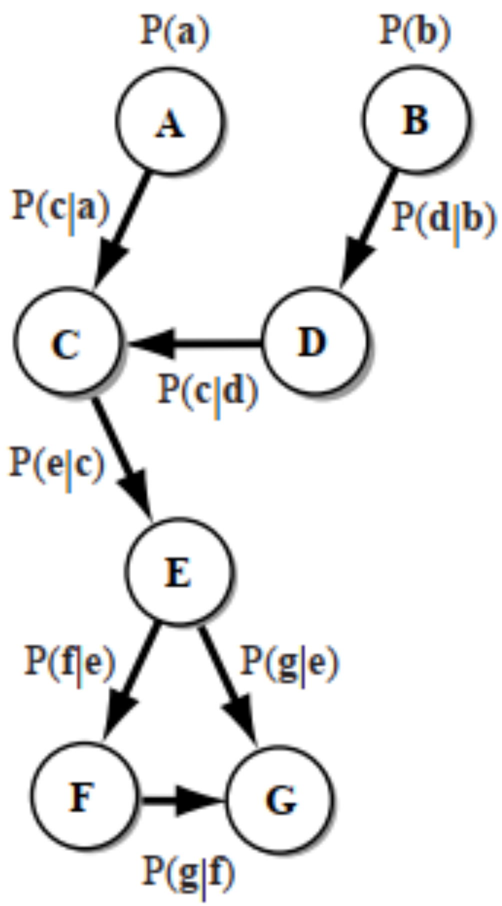
Bayesian Network: What & Why

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Bayesian Networks: What

Representation of causal dependencies graphically (Hart et al.,2001)



Bayesian belief nets

Causal networks

Belief nets

Why Bayesian Networks ?

BN have capability probabilistic reasoning like full joint probability distribution. It can answer any question about the domain.

Full joint probability distribution can become intractably large as the number of variables grows.

BN: Independence and conditional independence relationships can greatly reduce the number of probabilities

lightness	width	category	Prob
1.5	14.6	salmon	0.3
…	…	…	…
8.3	15	Sea bass	0.25
…	…	…	…

In full joint probability distribution, each combination of variable values has information how probable it is.

BN Components

Structure

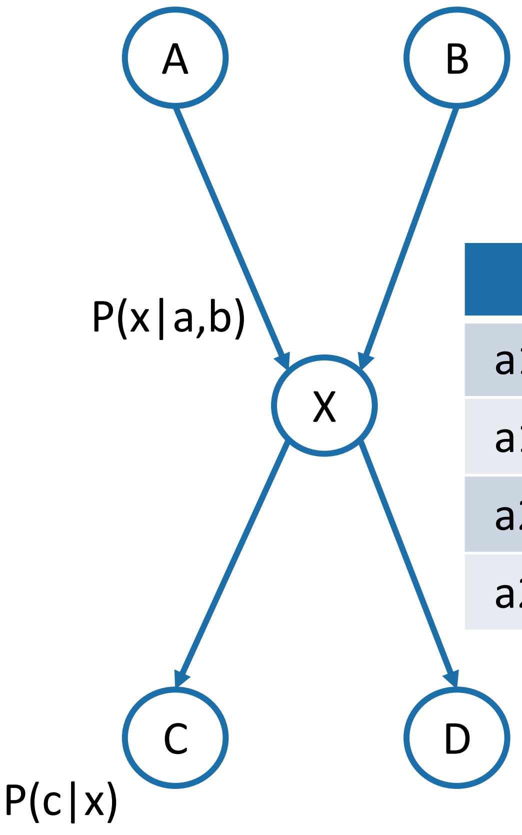
- Node (variables)
- Directed arcs (acyclic graph)

Numerical parameters

- Prior probability
- Probability conditional tables

P(a1)=0.25

P(b1)=0.6



	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8

	P(c1|x)
x1	0.3
x2	0.6

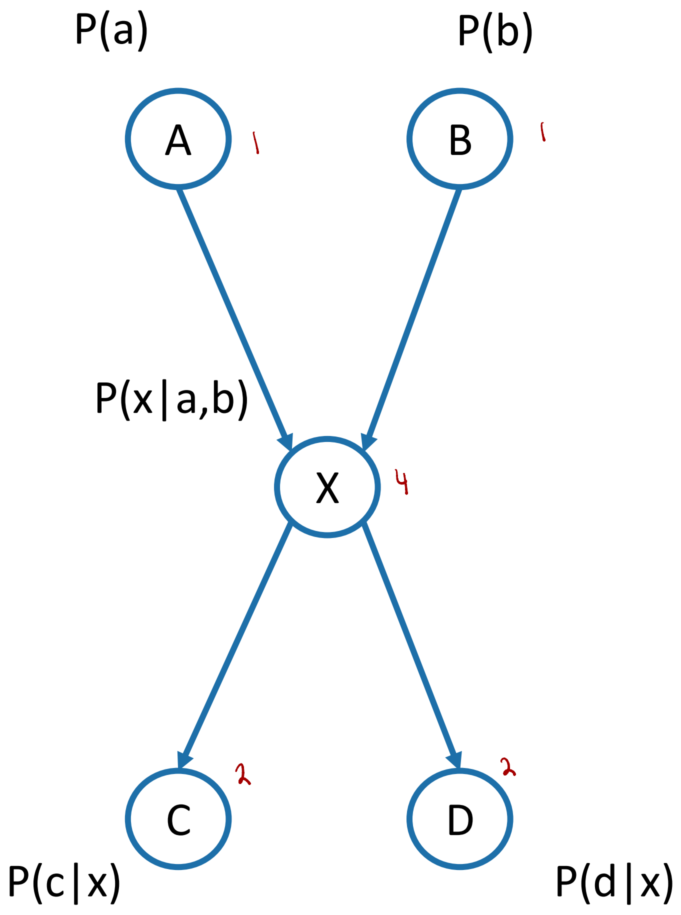
	P(d1|x)
x1	0.3
x2	0.6

Reduction Number of Probabilities

In a domain with N binary propositional variables (2 possibilities value), one needs 2^N numbers to specify the joint probability distribution. N=5: need 32 probabilities

Independence and conditional independence relationships among variables can greatly reduce the number of probabilities that need to be specified in order to define the full joint distribution (Russel & Norvig, 2013)

For 5 binary variables with casual networks: need 2+2+8+4+4=20 probabilities (or 10 with complements).

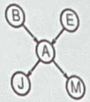


Bayesian Network as Joint Probability Distribution (chain rule)

The full joint distribution is defined as the product of the local conditional distributions:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Parents(X_i))$$

$\boxed{P(A|B) = 1 - P(A|\neg B)}$



e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$$= P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$$

J ketika dikasih A, jadi ga perlu lagi berhubungan dengan B. dst. $\Rightarrow$ jadi J dan B conditionally independent.

$P(\neg J, M, \neg A, B, E)$

$= P(\neg J \mid \neg A) \cdot P(M \mid \neg A) \cdot P(\neg A \mid B, E) \cdot P(B) \cdot P(E)$

$P(J, M, A, B) = P(J|A) \cdot P(M|A) \cdot P(A|B,E) \cdot P(B) \cdot P(E)$

$+ P(J|A) \cdot P(M|A) \cdot P(A|\underbrace{B, \neg E}) \cdot \underbrace{P(B) \cdot P(\neg E)}$

harus sama

$P(M, A, B, E) = P(J|A) \cdot P(M|A) \cdot P(A|B,E) \cdot P(B) \cdot P(E)$

$+ P(\neg J|A) \cdot P(M|A) \cdot P(A|B,E) \cdot P(B) \cdot P(E)$

$= \underbrace{(P(J|A) + P(\neg J|A))}_{1} \cdot P(M|A) \cdot P(A|B,E) \cdot P(B) \cdot P(E)$

P(A) = hanya perlu B dan E saja soalnya yang J dan M nanti jadi 1 juga

$= P(A|B,E) \cdot P(B) \cdot P(E) +$

$P(A|\neg B,E) \cdot P(\neg B) \cdot P(E) +$

$P(A|B,\neg E) \cdot P(B) \cdot P(\neg E) +$

$P(A|\neg B,\neg E) \cdot P(\neg B) \cdot P(\neg E)$

Bayesian Network as Joint Probability

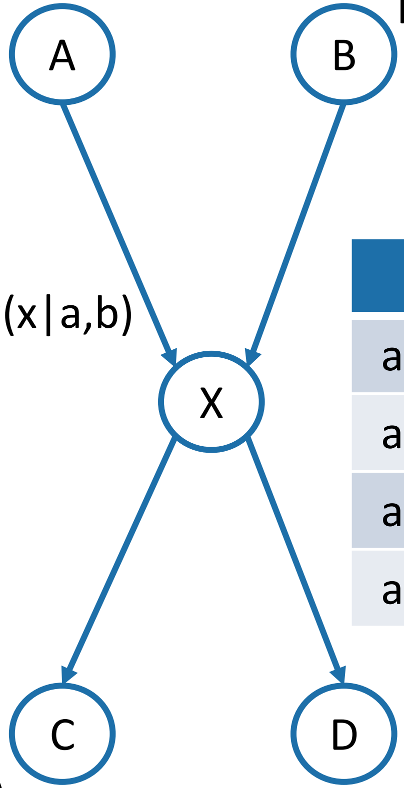
- We can determine the value of any entry in the joint probability.
- P(a2,b1,x2,c2,d1)
  =P(a2)P(b1)P(x2|a2,b1)P(c2|x2)P(d1|x2)
  =0.75*0.6*0.4*0.4*0.6
  =0.0432

P(a1)=0.25

p(a2) = 1-0.25
= 0.75

P(b1)=0.6

p(b2) = 1-0.6
= 0.4



P(x|a,b)

	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8

P(c|x)

	P(c1|x)
x1	0.3
x2	0.6

p(c2|x)

x1	1-0.3 = 0.7
x2	1-0.6 = 0.4

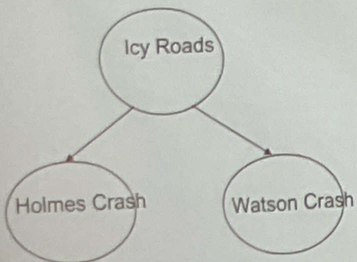
P(d|x)

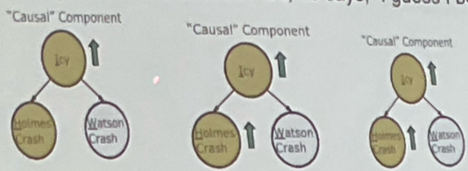
	P(d1|x)
x1	0.3
x2	0.6

Example of Bayesian Network

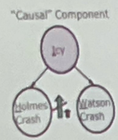
Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, "It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch." But, his secretary says, "No, the roads are not icy, look at the window." So, he says, "I guess I better wait for Holmes."



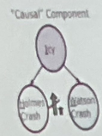


H and W are dependent,



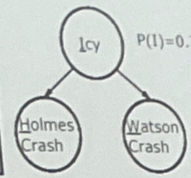
H and W are dependent,

Holmes mungkin kecelakaan tapi informasi watson kecelakaan udah ga pengaruh



H and W are dependent, but conditionally independent given I

Icy Roads with Numbers



$P(I)=0.7$

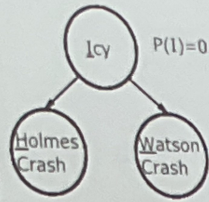
	$P(H \mid I)$
I	0.8
$\neg I$	0.1

	$P(W \mid I)$
I	0.8
$\neg I$	0.1

Probability that Watson Crashes:

$$
\begin{aligned}
P(W) &= P(W \mid I)\,P(I) + P(W \mid \neg I)\,P(\neg I) \\
&= 0.8 \cdot 0.7 + 0.1 \cdot 0.3 \\
&= 0.56 + 0.03 \\
&= 0.59
\end{aligned}
$$

Icy Roads with Numbers



	P(H| I)
I	0.8
¬I	0.1

	P(W| I)
I	0.8
¬I	0.1

Probability of Icy given Watson (Bayes' Rule):

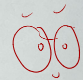
$$P(I \mid W) = P(W \mid I)\, P(I) / P(W)$$

→ universenya dah terbatas bukan U lagi. → watson sudah kecelakaan

$$= 0.8 \cdot 0.7 / 0.59$$

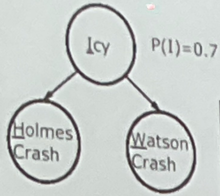
$$= 0.95$$

We started with P(I) = 0.7; knowing that Watson crashed raised the probability to 0.95



Icy Roads with Numbers

	P(H| I)
I	0.8
¬I	0.1



	P(W| I)
I	0.8
¬I	0.1

Probability of Holmes given Watson :

$P(H|W) = P(H|W,I)P(I|W) + P(H|W,\neg I)\, P(\neg I|\, W)$

$= P(H|I)P(I|W) + P(H|\neg I)\, P(\neg I|\, W)$

$= 0.8 \cdot 0.95 + 0.1 \cdot 0.05$

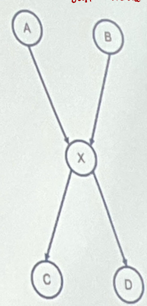
$= 0.765$

We started with P(H) = 0.59; knowing that Watson crashed raised the probability to 0.765

- X represents the fish : x1=salmon and x2=sea bass.
- X is influenced by A and B.
- A represents time of year: a1 = winter, a2 = spring, a3 = summer and a4 = autumn. Probability distribution on A in uniform.
- B represents geographical area where the fish was caught: b1 = north Atlantic and b2 = south Atlantic. The probabilities that any fish came from those areas are 0.6 and 0.4.
- C represents lightness with c1 = light, c2 = medium and c3 = dark
- D represents thickness with d1 = wide and d2 = thin.

kalo ditanya
berapa angka
↳ tulis aja semua

if binangkan 20
(10 + complement)
but if case kyk gini
just write all.



The probability that the fish was caught in the summer in the north Atlantic and is a sea bass that is dark and thin.

The probability that the fish was caught in the summer (a3) in the north Atlantic (b1) and is a sea bass (x2) that is dark (c3) and thin (d2).

$\Downarrow$

$p(a3, b1, x2, c3, d2)$

Inference: Example

$$P(a3,b1,x2,c3,d2) = P(a3)P(b1)P(x2|a3,b1)P(c3|x2)P(d2|x2)=0.012$$

P(a)

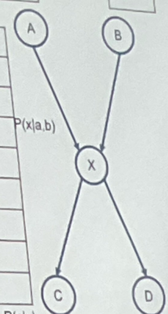
P(a1)	P(a2)	P(a3)	P(a4)
0.25	0.25	0.25	0.25

P(b)

P(b1)
0.6

P(x|a,b)

	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8
a3,b1	0.4
a3,b2	0.1
a4,b1	0.2
a4,b2	0.3



P(c|x)

	P(c1|x)	P(c2|x)	P(c3|x)
x1	0.6	0.2	0.2
x2	0.2	0.3	0.5

P(d|x)

	P(d1|x)
x1	0.3
x2	0.6

Classify the fish that is light (c1) and caught in the south Atlantic (b2), but we do not know what time of year the fish was caught nor its thickness.

Maximum a posterior probability: $P(x1|c1,b2)$ vs $P(x2|c1,b2)$

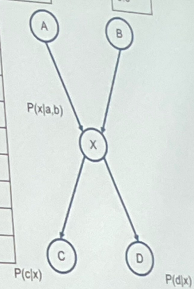
P(a)

P(a1)	P(a2)	P(a3)	P(a4)
0.25	0.25	0.25	0.25

P(b)

P(b1)
0.6

	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8
a3,b1	0.4
a3,b2	0.1
a4,b1	0.2
a4,b2	0.3



	P(c1|x)	P(c2|x)	P(c3|x)
x1	0.6	0.2	0.2
x2	0.2	0.3	0.5

	P(d1|x)
x1	0.3
x2	0.6

Classification (2)

Q: query

e: evidence of all variables

$P(Q|\mathbf{e})=P(q,e)/P(e)=\alpha P(Q,e)$ $\frac{1}{p(e)}$ dianggap α

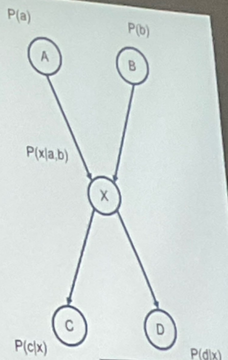
$P(x1|c1,b2)=P(x1,c1,b2)/P(c1,b2)$

$=\alpha \sum P(x1,a,b2,c1,d)$

$=\alpha\sum P(a).P(b2).P(x1|a,b2).\ P(c1|x1).\ P(d|x1)$

$=\alpha P(b2).P(c1|x1)\ \sum P(a).\ P(x1|a,b2).\ P(d|x1)$

$=\alpha P(b2).P(c1|x1)\ [\sum P(a).\ P(x1|a,b2)][\sum P(d|x1)]=\alpha\ 0.114$



	P(d1|x)
x1	0.3
x2	0.6

Classification (3)

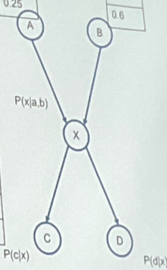
P(a)

P(a1)	P(a2)	P(a3)	P(a4)
0.25	0.25	0.25	0.25

P(b)

P(b1)
0.6

	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8
a3,b1	0.4
a3,b2	0.1
a4,b1	0.2
a4,b2	0.3



	P(c1|x)	P(c2|x)	P(c3|x)
x1	0.6	0.2	0.2
x2	0.2	0.3	0.5

	P(d1|x)
x1	0.3
x2	0.6

$P(x1|c1,b2)=P(x1,c1,b2)/P(c1,b2)$
$=\alpha P(b2).P(c1|x1).[\sum P(a).P(x1|a,b2)]. [\sum P(d|x1)]= \alpha\ 0.114$

$P(x2|c1,b2)=P(x2,c1,b2)/P(c1,b2)$
$=\alpha P(b2).P(c1|x2)\ [\sum P(a). P(x2|a,b2)][\sum P(d|x2)]= \alpha\ 0.042$

Normalize:

$P(x1|c1,b2)=0.73$, $\frac{0.114\alpha}{0.114\alpha+0.042\alpha}$

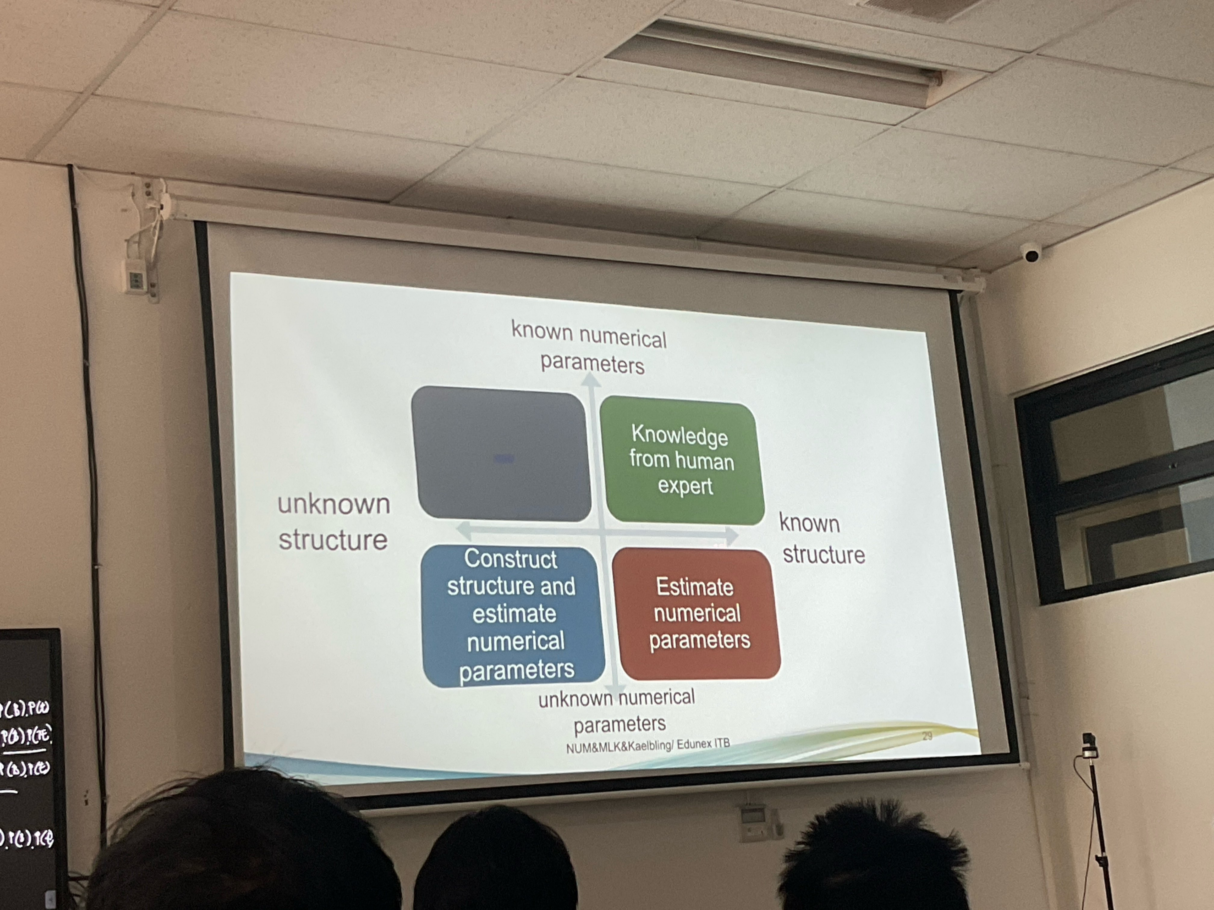
$P(x2|c1,b2)=0.27$, $\frac{0.0.42\alpha}{0.114\alpha+0.042\alpha}$

Decision: x1=salmon

Problems in BN constructed by human expert

Knowledge acquisition bottleneck problem

Knowledge elicitation problem: slow speed, and inability of expert to express the knowledge they posses.


known numerical parameters
Knowledge from human expert
unknown structure
known structure
Construct structure and estimate numerical parameters
Estimate numerical parameters
unknown numerical parameters
NUM&MLK&Kaelbling/ Edunex ITB
29

Parameter Estimation

Given structure with m nodes

Given a data set

$D = \{<v_1^1,..,v_m^1>,..,<v_1^k,..,v_m^k>\}$

Count $\#(V_i{=}T)$, $\#(V_i{=}F)$, $\#(V_i{=}T, V_j{=}T)$, $\#(V_i{=}T, V_j{=}F)$

Variable V_i with no parent

$$P(V_i) \approx \frac{\#(V_i = T)}{k}$$

Variable V_i with parent V_j

$$P(V_i|V_j) \approx \frac{\#(V_i = T, V_j = T)}{\#(V_j = T)}$$

$$P(V_i|\neg V_j) \approx \frac{\#(V_i = T, V_j = F)}{\#(V_j = F)}$$

Smoothing: Avoid Probability = 0

Variable V_i with no parent

$$P(V_i) \approx \frac{\#(V_i = T) + 1}{k + 2}$$

Variable V_i with parent V_j

$$P(V_i|V_j) \approx \frac{\#(V_i = T, V_j = T) + 1}{\#(V_j = T) + 2}$$

$$P(V_i|\neg V_j) \approx \frac{\#(V_i = T, V_j = F) + 1}{\#(V_j = F) + 2}$$

Summary

Bayesian Network

BN vs Joint Probability

Classification using BN

THANK YOU

Knowledge Representation and Reasoning

Introduction to Probabilistic Reasoning System

Nur Ulfa Maulidevi
(ulfa@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Artifisial
(Artificial Intelligence)

Probabilistic Reasoning System (PRS)

Agent in real world need to handle uncertainty

Logical Agents handle uncertainty by disjunction → cannot tell us how likely the difference conditions are

Probability theory provides a quantitative way of encoding likelihood

Given state(s) e, what is the probabilty that x happens → $P(x|e)$

- Joint Probability Distribution
- Bayesian/ Belief Network

Joint Probability Distribution

- Random variables
  - Function: discrete domain → [0, 1]
  - Sums to 1 over the domain
    - Raining is a propositional random variable
    - Raining(true) = 0.2
      - P(Raining = true) = 0.2
    - Raining(false) = 0.8
      - P(Raining = false) = 0.8
- Joint distribution
  - Probability assignment to all combinations of values of random variables

Inference using Joint Probability Distribution

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

- For any proposition ϕ, sum the atomic events where it is true: $P(\phi) = \Sigma_{\omega:\omega \models \phi} P(\omega)$
- P(*toothache*) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2
- P(*cavity*) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2
- P(cavity U toothache) = ?
- P(¬*cavity* | *toothache*) = P(¬*cavity* ∩ *toothache*) / P(*toothache*)

  = (0.016+0.064) / (0.108 + 0.012 + 0.016 + 0.064)

  = 0.4

Inference using Joint Probability Distribution

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

- If you have n binary propositional variables → requires 2^n numbers to build Joint Probability Distribution

→ Bayesian Network (We want to exploit independences in the domain)

Bayes' Rule:

$$P(A \mid B) = P(A \cap B) / P(B)$$
$$= P(B \mid A)\, P(A) / P(B)$$

Conditioning:

$$P(A) = P(A \mid B)\, P(B) + P(A \mid \neg B)\, P(\neg B)$$
$$= P(A \cap B) + P(A \cap \neg B)$$

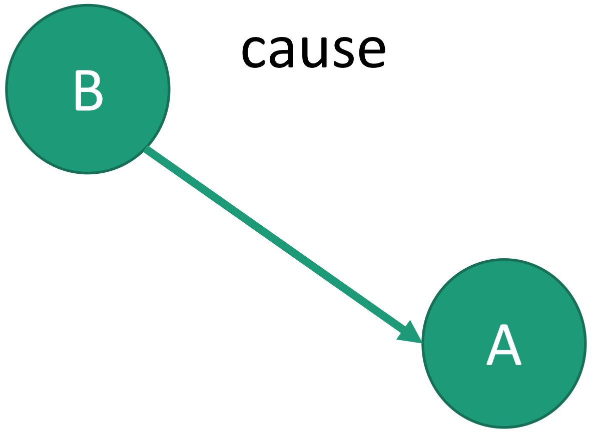
Structure of Bayesian Network

Nodes (variable)

Directed arc

P(B)

cause



effect

$P(A|B)$

$P(A|\neg B)$

Should be
Directed Acyclic Graph (DAG)

Numerical Parameters

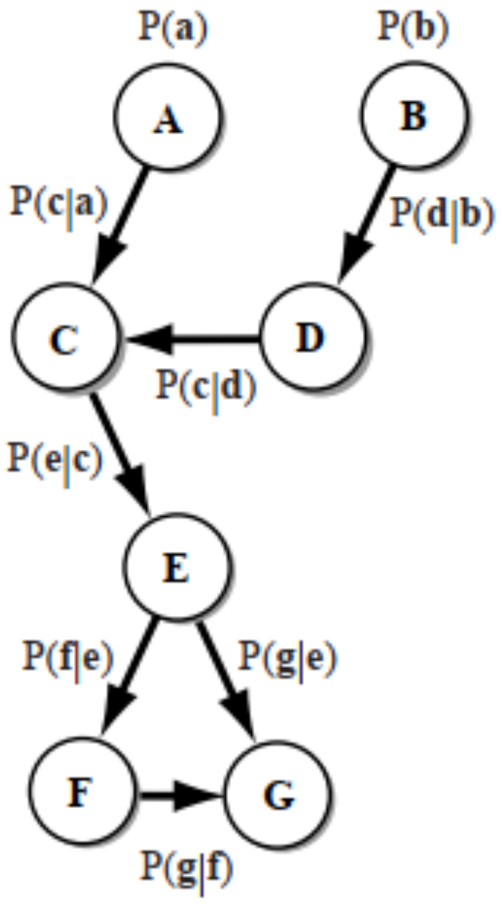
- Prior probability
- Probability conditional tables

Structure of Bayesian Network

Representation of causal dependencies graphically (Hart et al.,2001)

BN have capability probabilistic reasoning like full joint probability distribution. It can answer any question about the domain.

-- How we exploit Independence?



Bayesian belief nets

Causal networks

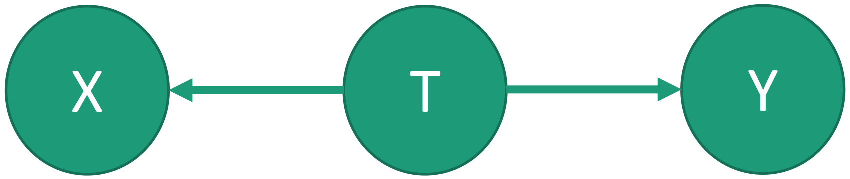
Belief nets

Independence

- A and B are independent iff
  - $P(A \cap B) = P(A) \cdot P(B)$
  - $P(A \mid B) = P(A)$
  - $P(B \mid A) = P(B)$
- Independence is essential for efficient probabilistic reasoning
- A and B are conditionally independent given C iff
  - $P(A \mid B, C) = P(A \mid C)$
  - $P(B \mid A, C) = P(B \mid C)$
  - $P(A \cap B \mid C) = P(A \mid C) \cdot P(B \mid C)$

Example of Independence

- X is late (X)
- Traffic Jam (T)
- Y is late (Y)
- None of these propositions are independent of one other
- X and Y are conditionally independent given T

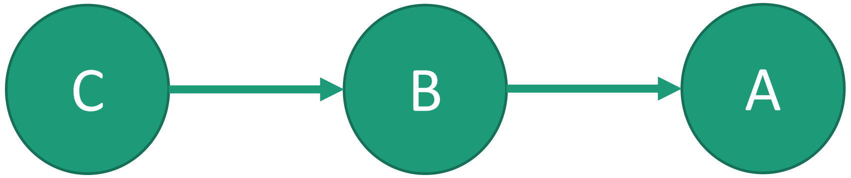


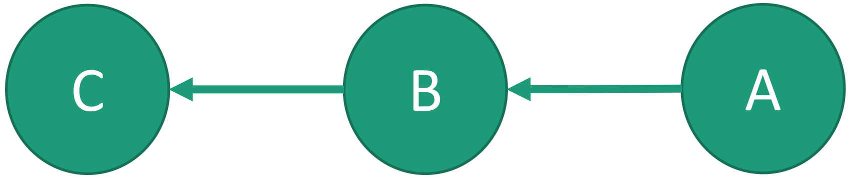
Types of Connections in Bayesian Network

Forward

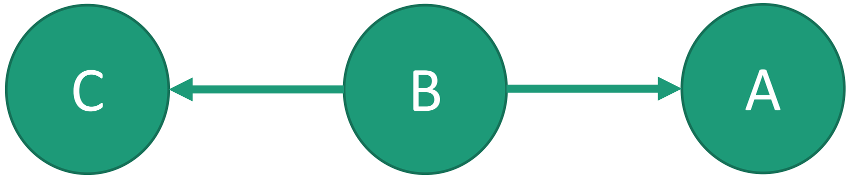
Backward

Serial



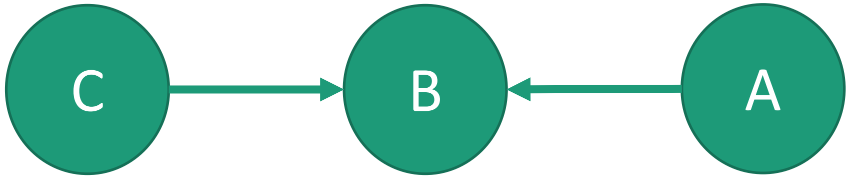


Diverging



Why? To exploit Independence

Converging



Independence in Connection

Serial

$C \rightarrow B \rightarrow A$

Diverging

$C \leftarrow B \rightarrow A$

Knowing C will tell us about A, but if we know B, knowing C will tell us nothing about A (C and A conditionally independent or d-separated)

Converging

$C \rightarrow B \leftarrow A$

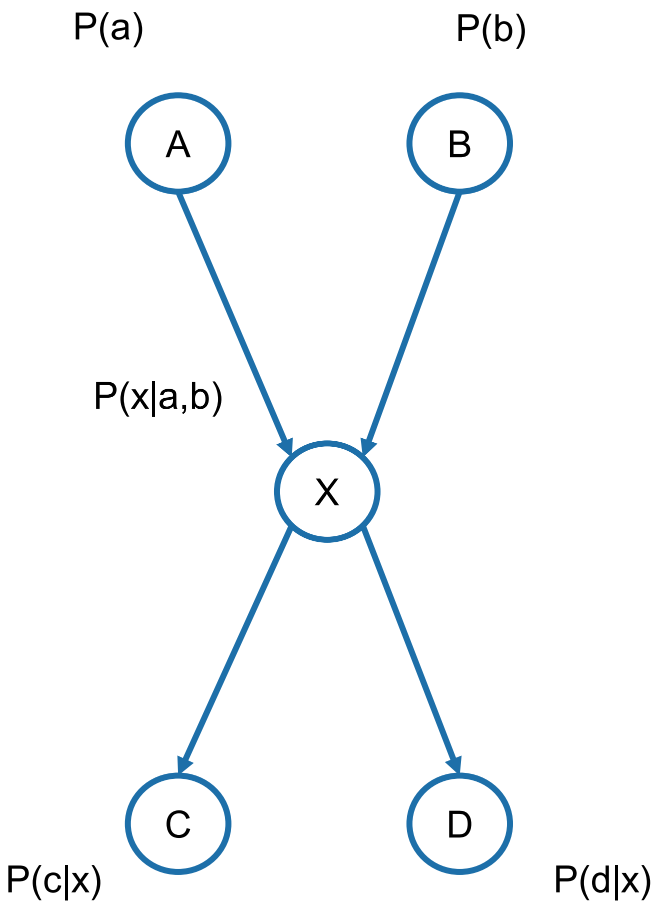
Knowing C will tell us nothing about A without knowing B, but if we see evidence about B, C and A becomes dependent

Reduction Number of Probabilities

In a domain with N binary propositional variables (2 possibilities value), one needs 2^N numbers to specify the joint probability distribution. N=5: need 32 probabilities

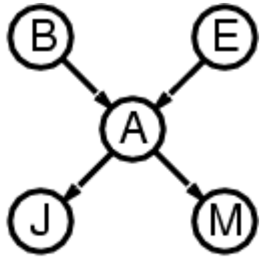
Independence and conditional independence relationships among variables can greatly reduce the number of probabilities that need to be specified in order to define the full joint distribution (Russel & Norvig, 2013)

For 5 binary variables with causal networks: need 2+2+8+4+4=20 probabilities (or 10 with complements).



Bayesian Network as Joint Probability Distribution (chain rule)

The full joint distribution is defined as the product of the local conditional distributions:



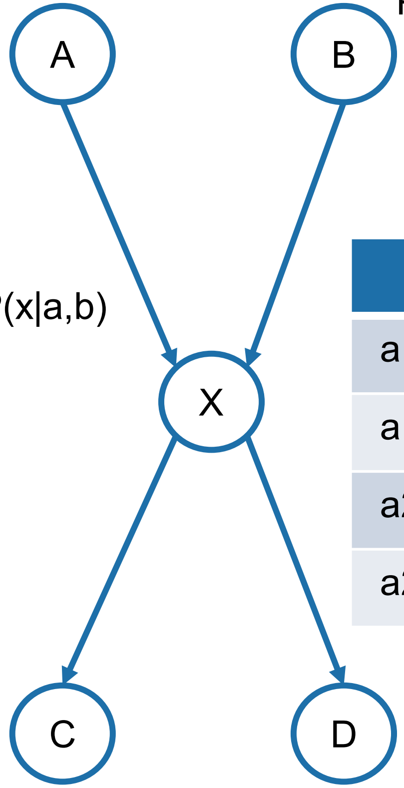
$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | Parents(X_i))$$

e.g., $\mathbf{P}(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$$= \mathbf{P}(j \mid a)\, \mathbf{P}(m \mid a)\, \mathbf{P}(a \mid \neg b, \neg e)\, \mathbf{P}(\neg b)\, \mathbf{P}(\neg e)$$

Example

- We can determine the value of any entry in the joint probability.
- P(a2,b1,x2,c2,d1)
=P(a2)P(b1)P(x2|a2,b1)P(c2|x2)P(d1|x2)
=0.75*0.6*0.4*0.4*0.6
=0.0432



	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8

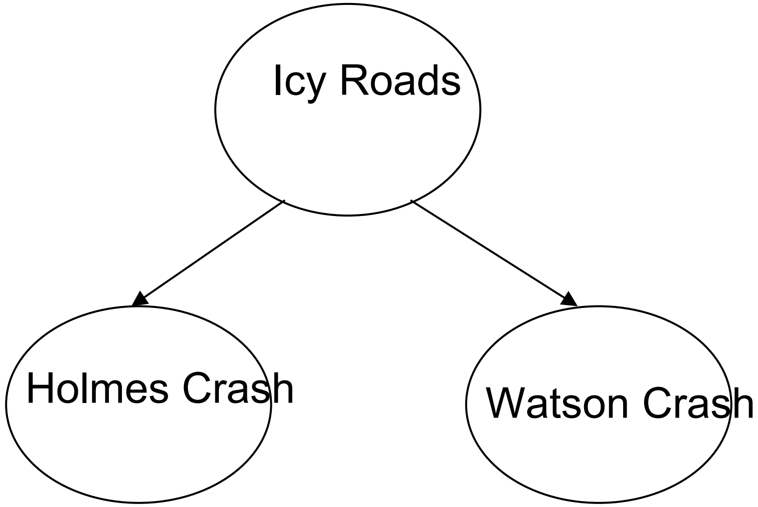
	P(c1|x)
x1	0.3
x2	0.6

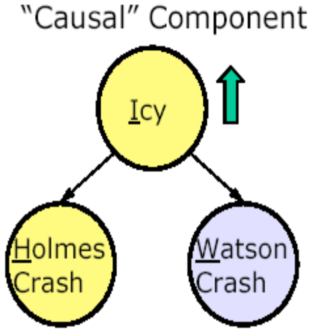
	P(d1|x)
x1	0.3
x2	0.6

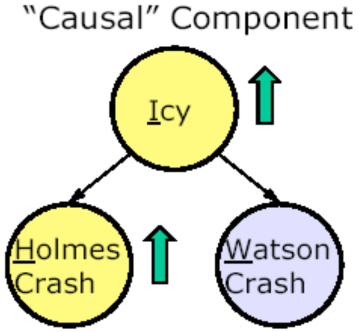
Example of Bayesian Network

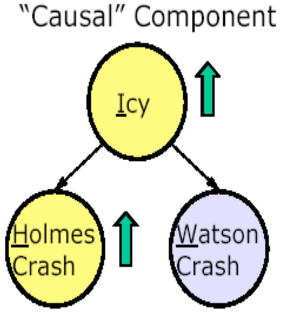
Icy Roads

Inspector Smith is waiting for Holmes and Watson, who are driving (separately) to meet him. It is winter. His secretary tells him that Watson has had an accident. He says, “It must be that the roads are icy. I bet that Holmes will have an accident too. I should go to lunch.” But, his secretary says, “No, the roads are not icy, look at the window.” So, he says, “I guess I better wait for Holmes.”

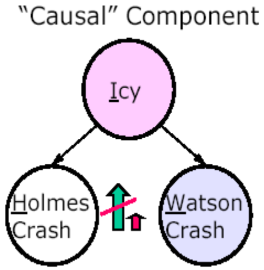




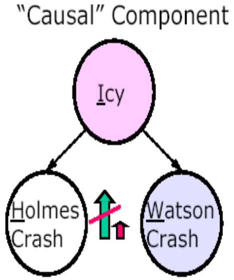




H and W are dependent,



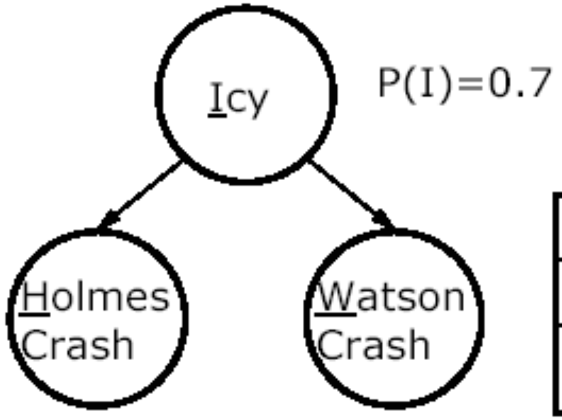
H and W are dependent,



H and W are dependent, but conditionally independent given I

Icy Roads with Numbers

	P(H| I)
I	0.8
¬I	0.1

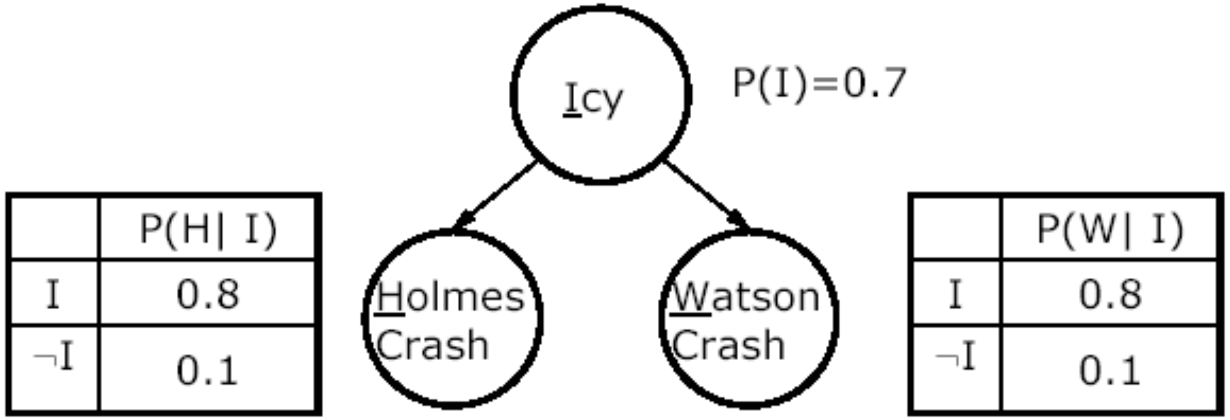


	P(W| I)
I	0.8
¬I	0.1

Probability that Watson Crashes:

$$P(W) = P(W|\ I)\ P(I) + P(W|\ \neg I)\ P(\neg I)$$
$$= 0.8 \cdot 0.7 + 0.1 \cdot 0.3$$
$$= 0.56 + 0.03$$
$$= 0.59$$

Icy Roads with Numbers



Probability of Icy given Watson (Bayes' Rule):

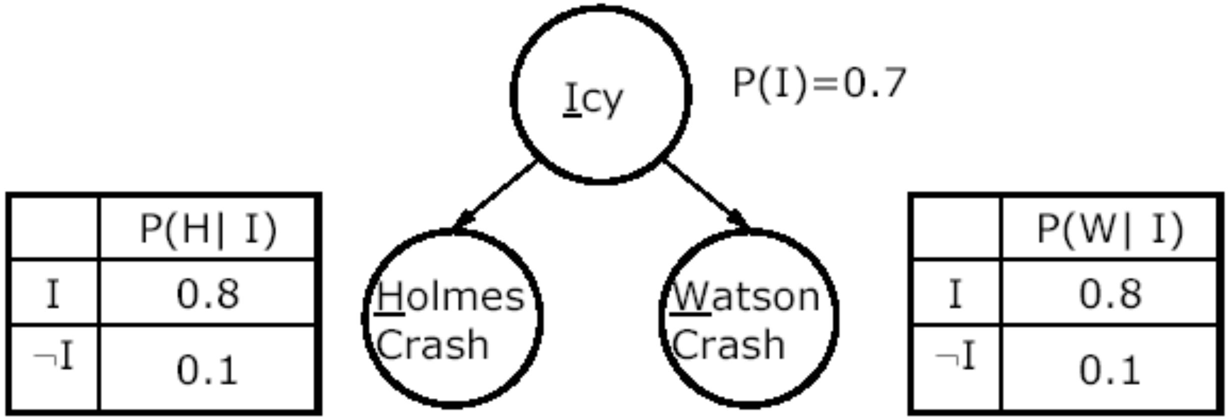
P(I | W) = P(W | I) P(I) / P(W)

= 0.8 · 0.7 / 0.59

= 0.95

We started with P(I) = 0.7; knowing that Watson crashed raised the probability to 0.95

Icy Roads with Numbers



Probability of Holmes given Watson :

$P(H|W) = P(H|W,I)P(I|W) + P(H|W,\neg I)\, P(\neg I|\, W)$

$= P(H|I)P(I|W) + P(H|\neg I)\, P(\neg I|\, W)$

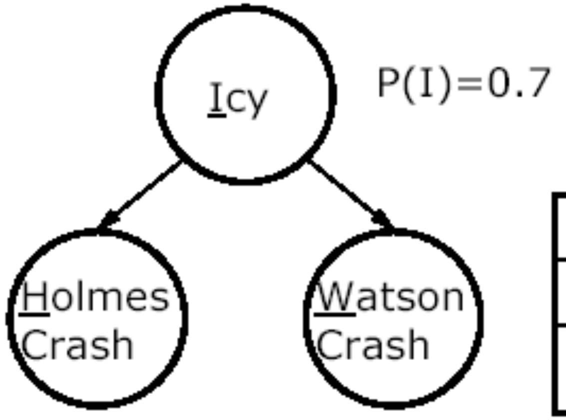
$= 0.8 \cdot 0.95 + 0.1 \cdot 0.05$

$= 0.765$

We started with P(H) = 0.59; knowing that Watson crashed raised the probability to 0.765

Icy Roads with Numbers

	P(H| I)
I	0.8
¬I	0.1



	P(W| I)
I	0.8
¬I	0.1

Probability of Holmes given Icy and Watson :
$P(H|W, \neg I) = P(H|\neg I) = 0.1$

H and W are d-separated given I, so H and W are conditionally independent given I

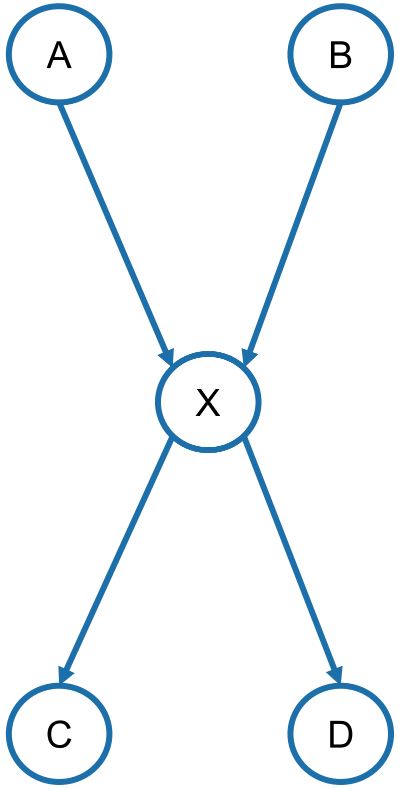
Classification using Bayesian Network

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Artifisial
(*Artificial Intelligence*)

- X represents the fish : x1=salmon and x2=sea bass.
- X is influenced by A and B.
- A represents time of year: a1 = winter, a2 = spring, a3 = summer and a4 = autumn. Probability distribution on A in uniform.
- B represents geographical area where the fish was caught: b1 = north Atlantic and b2 = south Atlantic. The probabilities that any fish came from those areas are 0.6 and 0.4.
- C represents lightness with c1 = light, c2 = medium and c3 = dark
- D represents thickness with d1 = wide and d2 = thin.



The probability that the fish was caught in the summer in the north Atlantic and is a sea bass that is dark and thin.

The probability that the fish was caught in the summer (a3) in the north Atlantic (b1) and is a sea bass (x2) that is dark (c3) and thin (d2).

P(a3,b1,x2,c3,d2)

Inference: Example

P(a)

P(a1)	P(a2)	P(a3)	P(a4)
0.25	0.25	0.25	0.25

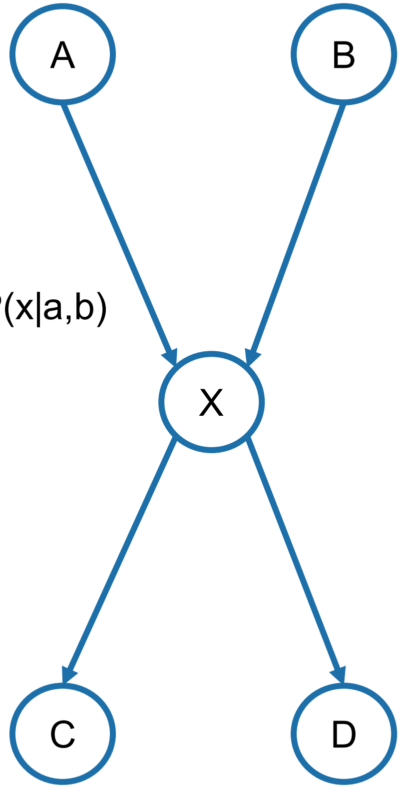
P(b)

P(b1)
0.6

P(a3,b1,x2,c3,d2)
=P(a3)P(b1)P(x2|a3,b1)P(c3|x2)P(d2|x2)=0.012

P(x|a,b)

	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8
a3,b1	0.4
a3,b2	0.1
a4,b1	0.2
a4,b2	0.3



P(c|x)

	P(c1|x)	P(c2|x)	P(c3|x)
x1	0.6	0.2	0.2
x2	0.2	0.3	0.5

P(d|x)

	P(d1|x)
x1	0.3
x2	0.6

Classification

Classify the fish that is light (c1) and caught in the south Atlantic (b2), but we do not know what time of year the fish was caught nor its thickness.

Maximum a posterior probability:
P(x1|c1,b2) vs P(x2|c1,b2)

P(a)

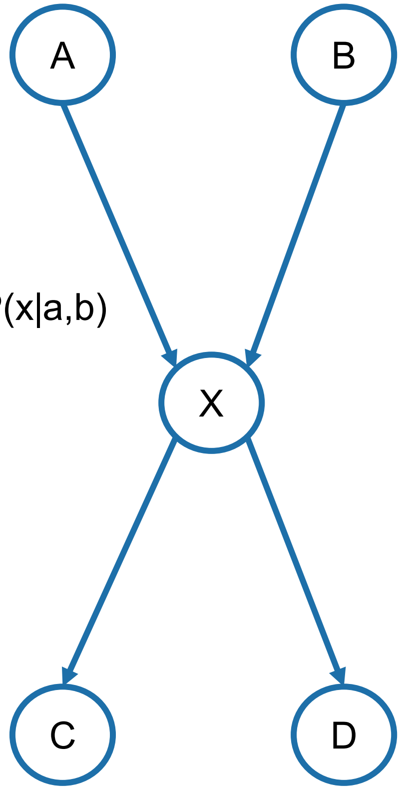
P(a1)	P(a2)	P(a3)	P(a4)
0.25	0.25	0.25	0.25

P(b)

P(b1)
0.6

	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8
a3,b1	0.4
a3,b2	0.1
a4,b1	0.2
a4,b2	0.3

P(x|a,b)



P(c|x)

	P(c1|x)	P(c2|x)	P(c3|x)
x1	0.6	0.2	0.2
x2	0.2	0.3	0.5

P(d|x)

	P(d1|x)
x1	0.3
x2	0.6

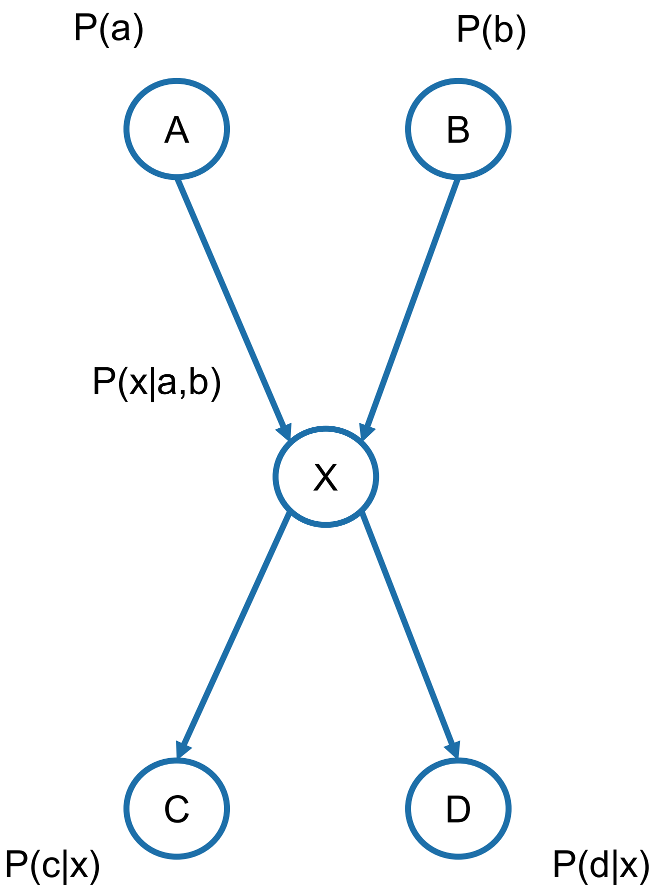
Classification (2)

Q: query
e: evidence of all variables
$P(Q|\mathbf{e})=P(q,e)/P(e)=\alpha P(Q,e)$

$P(x1|c1,b2)=P(x1,c1,b2)/P(c1,b2)$
$=\alpha \sum P(x1,\mathbf{a},b2,c1,\mathbf{d})$
$=\alpha\sum P(\mathbf{a}).P(b2).P(x1|\mathbf{a},b2).\ P(c1|x1).\ P(\mathbf{d}|x1)$
$=\alpha P(b2).P(c1|x1)\ \sum P(\mathbf{a}).\ P(x1|\mathbf{a},b2).\ P(\mathbf{d}|x1)$
$=\alpha P(b2).P(c1|x1)\ [\sum P(\mathbf{a}).\ P(x1|\mathbf{a},b2)][\sum P(\mathbf{d}|x1)]=\alpha\ 0.114$

$P(a1).\ P(x1|a1,b2)+$
$P(a2).\ P(x1|a2,b2)+$
$P(a3).\ P(x1|a3,b2)+$
$P(a4).\ P(x1|a4,b2)$

$P(d1|x1)+P(d2|x1)=1.0$



	P(d1|x)
x1	0.3
x2	0.6

Classification (3)

$P(x1|c1,b2)=P(x1,c1,b2)/P(c1,b2)$
$=\alpha P(b2).P(c1|x1).[\sum P(a).P(x1|a,b2)]. [\sum P(d|x1)]= \alpha\ 0.114$

$P(x2|c1,b2)=P(x2,c1,b2)/P(c1,b2)$
$=\alpha P(b2).P(c1|x2)\ [\sum P(a).P(x2|a,b2)][\sum P(d|x2)]= \alpha\ 0.042$

Normalize:
$P(x1|c1,b2)=0.73$
$P(x2|c1,b2)=0.27$
Decision: x1=salmon

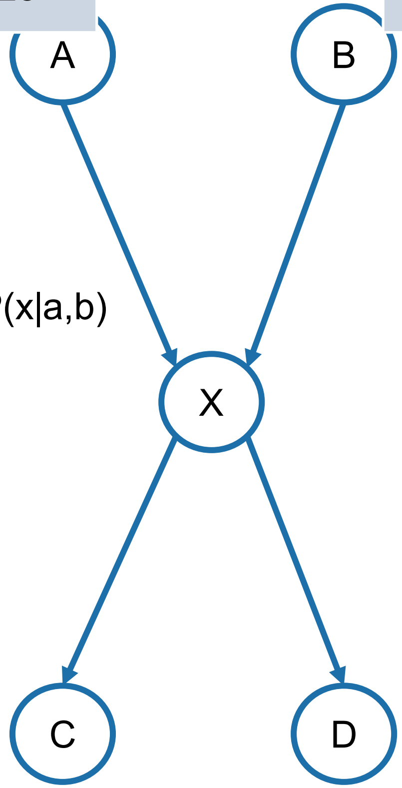
P(a)

P(a1)	P(a2)	P(a3)	P(a4)
0.25	0.25	0.25	0.25

P(b)

P(b1)
0.6

	P(x1|a,b)
a1,b1	0.3
a1,b2	0.7
a2,b1	0.6
a2,b2	0.8
a3,b1	0.4
a3,b2	0.1
a4,b1	0.2
a4,b2	0.3



P(c|x)

	P(c1|x)	P(c2|x)	P(c3|x)
x1	0.6	0.2	0.2
x2	0.2	0.3	0.5

P(d|x)

	P(d1|x)
x1	0.3
x2	0.6

Knowledge Representation and Reasoning

Learning from Data

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Problems in BN constructed by human expert

Knowledge acquisition bottleneck problem

Knowledge elicitation problem: slow speed, and inability of expert to express the knowledge they posses.

Very difficult in getting reliable probability estimates.

Ideally, determine probability by data distribution, experience, and assumption (subjective).

What is difference between P(a)=0.5 and P(a)=0.6 ?

known numerical parameters

unknown structure

known structure

Knowledge from human expert

Construct structure and estimate numerical parameters

Estimate numerical parameters

unknown numerical parameters

Humans are good at providing structure, data is good at providing numbers

Given structure with m nodes

Given a data set

$D = \{<v_1^1,..,v_m^1>,..,<v_1^k,..,v_m^k>\}$

Count $\#(V_i=T)$, $\#(V_i=F)$, $\#(V_i=T, V_j=T)$, $\#(V_i=T, V_j=F)$

Variable V_i with no parent

$$P(V_i) \approx \frac{\#(V_i = T)}{k}$$

Variable V_i with parent V_j

$$P(V_i|V_j) \approx \frac{\#(V_i = T, V_j = T)}{\#(V_j = T)}$$

$$P(V_i|\neg V_j) \approx \frac{\#(V_i = T, V_j = F)}{\#(V_j = F)}$$

Smoothing: Avoid Probability = 0

Variable V_i with no parent

$$P(V_i) \approx \frac{\#(V_i = T) + 1}{k + 2}$$

kemungkinan nilainya True/False => ada 2

Variable V_i with parent V_j

$$P(V_i|V_j) \approx \frac{\#(V_i = T, V_j = T) + 1}{\#(V_j = T) + 2}$$

$$P(V_i|\neg V_j) \approx \frac{\#(V_i = T, V_j = F) + 1}{\#(V_j = F) + 2}$$

Construct BN Structure

1. Determine set of variables
2. Determine ordering of variables $X_1,..X_m$
3. Loop: Add X_i, and select set of parents for X_i from $X_1,..,X_{i-1}$

Construct BN Structure: Example

Based on causal knowledge:

- Set of variables: A,B,C,D,X
- Ordering of variables: A,B,C,D,X
- Add A: no parent.
- Add B: is A parent of B ?
- Add C: parent of C ?
- Add D: parent of D ?
- Add X: parent of X ?

Based on data:

- Set of variables: A,B,C,D,X
- Ordering of variables: A,B,C,D,X
- Add A: no parent.
- Add B: $P(B|A)=P(B)$?
- Add C:
  - $P(C|A)=P(C)$?
  - $P(C|B)=P(C)$?
  - $P(C|A,B)=P(C|A)$?
  - $P(C|A,B)=P(C|B)$?
  - $P(C|A,B)=P(C)$?
- Etc.

is A the parent of B ??

cara ceknya computer akan hitung $P(B|A) = P(B)$
kalau iya berarti mereka saling independen dan A bukan parent dari B

$P(C|A,B) = P(C,B)$
A tidak mempengaruhi C,
A bukan parent C.
$P(C|B) \neq P(C)$.
berarti parent C adalah B.

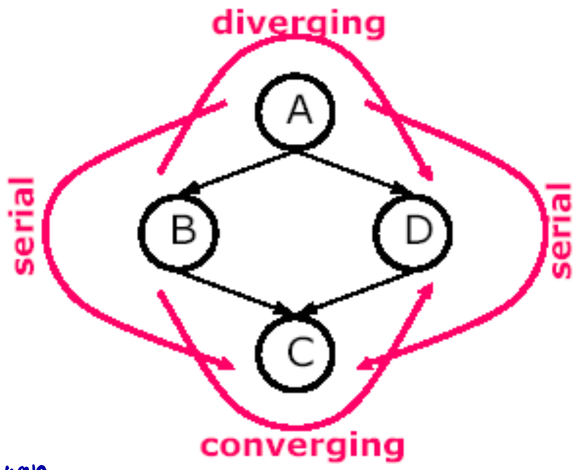
D Separation

Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either

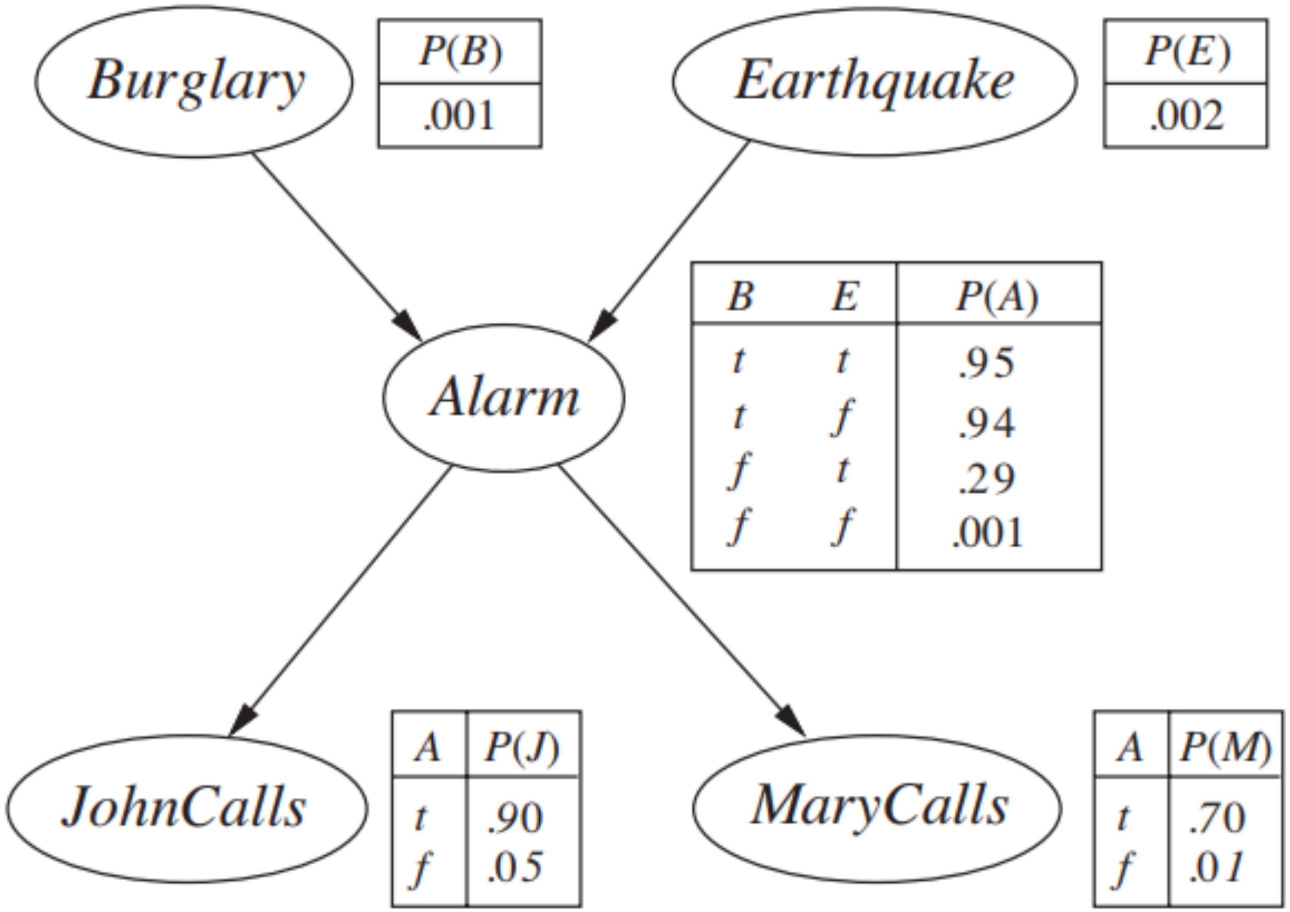
- The connection is serial or diverging and V is known
- The connection is converging and neither V nor any descendant is instantiated

Two variables are d-connected iff they are not d-separated

dalam BN,
boleh ubah² panah asal
ga bikin converging
baru + ga menghilangkan
converging lama



- A-B-C: serial, blocked when B is known, connected otherwise
- A-D-C: serial, blocked when D is known, connected otherwise
- B-A-D: diverging, blocked when A is known, connected otherwise
- B-C-D: converging, blocked when C has no evidence, connected otherwise



$P(B)$
.001

$P(E)$
.002

B	E	$P(A)$
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	$P(J)$
t	.90
f	.05

A	$P(M)$
t	.70
f	.01

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

Variable Ordering Influences Structure

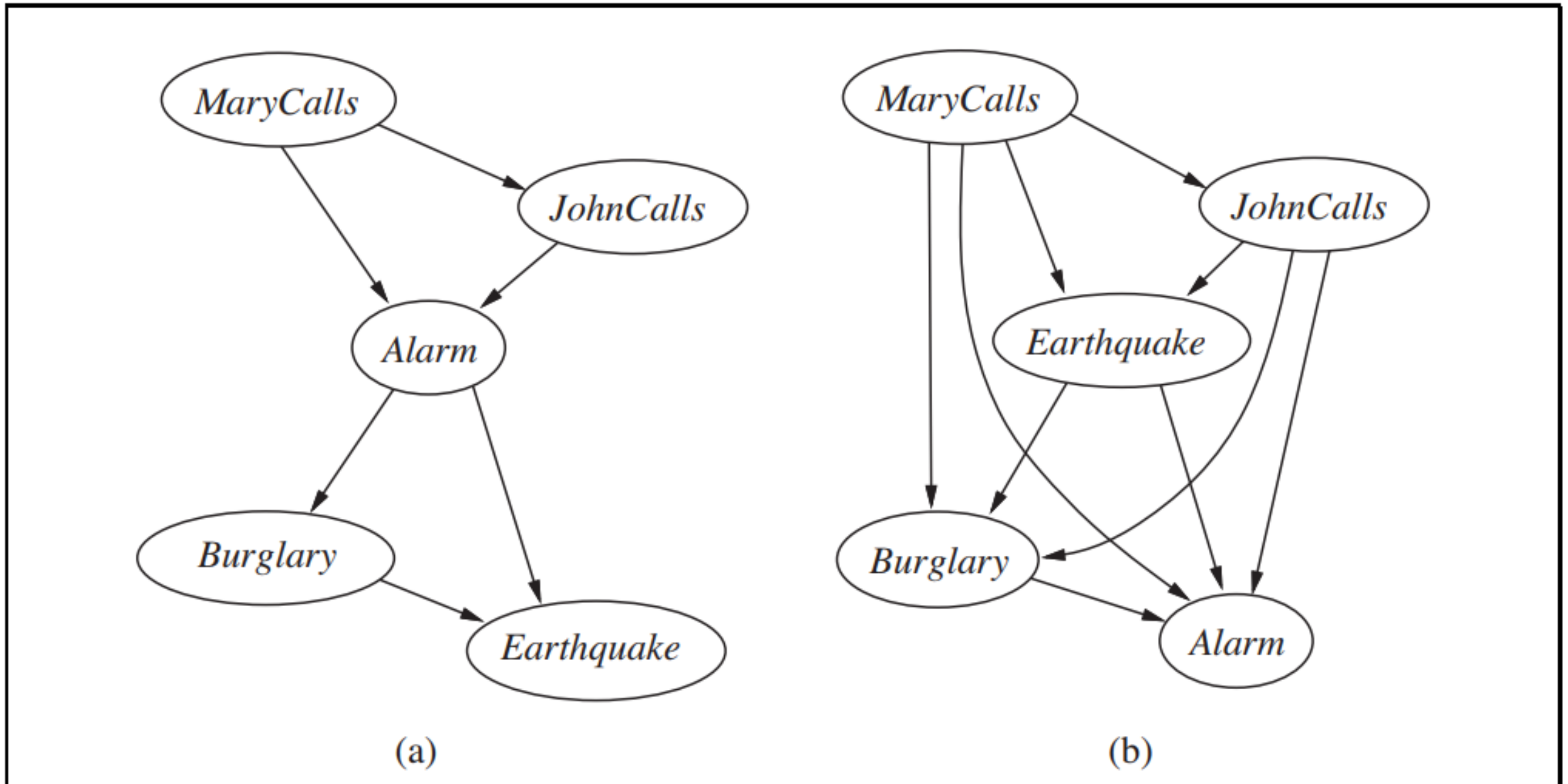


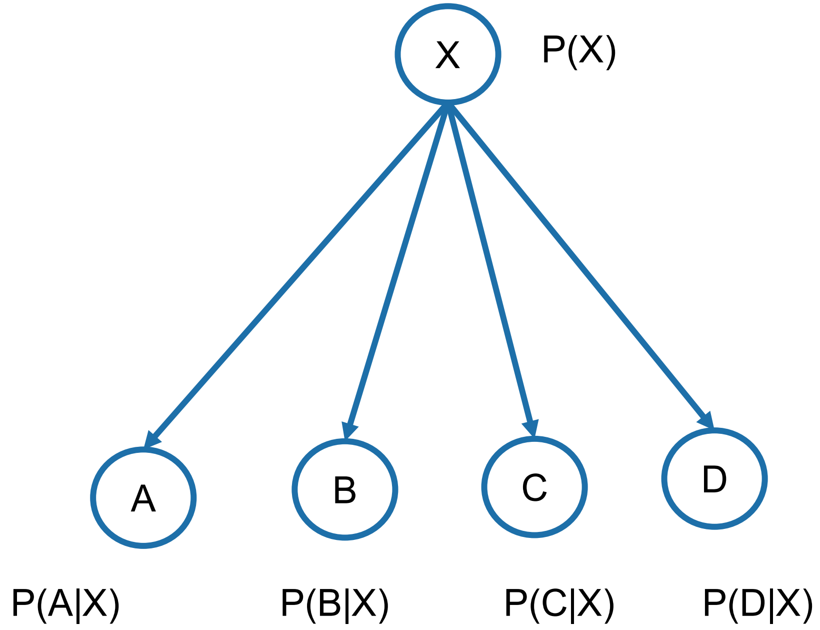
Figure 14.3 Network structure depends on order of introduction. In each network, we have introduced nodes in top-to-bottom order.

Russel & Norvig (2010). AI.

Construct BN Structure: Example 2

Based on causal knowledge:

- Set of variables: X,A,B,C,D
- Ordering of variables: X,A,B,C,D
- Add X: no parent.
- Add A: parent(A)=X.
- Add B: parent(B)=X. P(B|A,X)=P(B|X)
- Add C: parent(C)=X. P(C|A,B,X)=P(C|X)
- Add D: parent(D)=X. P(D|A,B,C,X)=P(D|X)

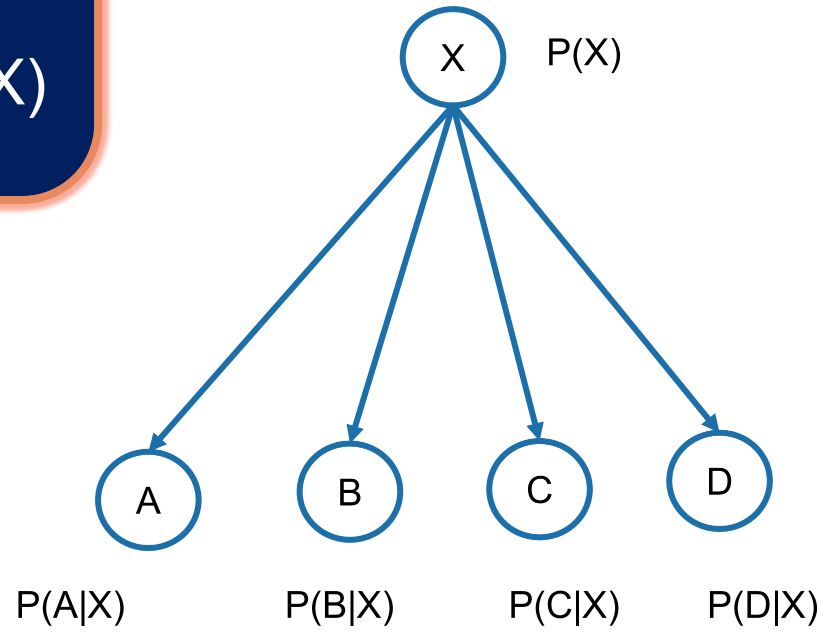


Naive Bayes as "Special" Bayesian Network

What is the Bayes Network for X1,….Xn with NO assumed conditional independencies?

Probability Model of P(X) and $P(a_i|X)$
$P(X, A, B, C, D) = P(X). P(A|X). P(B|X). P(C|X). P(D|X)$

Classification:
Find the maximum $P(X \mid A,B,C,D)$



Example: Play Tennis Dataset

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes

outlook	temp.	humidity	windy	play
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Frequency of (sunny|yes) → 2

Frequency of (overcast|yes) → 4

Frequency of (hot|yes) → 2

...

Frequency of class 'yes'→ 9

Frequency of (sunny|no) → 3

Frequency of (overcast|no) → 0

Frequency of (hot|no) → 2

Frequency of class 'no' → 5

Example: Play Tennis Dataset

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Frequency o
(sunny|yes) →

Frequency o
(sunny|no) →

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								

Frequency of
class 'yes'→ 9

Frequency of
class 'no' → 5

Example: Play Tennis Dataset

Frequency

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								

Probability Model

$P(a_i|v_j)$

outlook			temperature			humidity			windy		
	yes	no		yes	no		yes	no		yes	no
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5
rainy	3/9	2/5	cool	3/9	1/5						

$P(v_j)$

play	
yes	no
9/14	5/14

Classify New Instance: <Sunny, Cool, High, True>

$P(a_i|v_j)$

outlook	yes	no	temperature	yes	no	humidity	yes	no	windy	yes	no
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5
rainy	3/9	2/5	cool	3/9	1/5						

$P(v_j)$

play	
yes	no
9/14	5/14

$$P(v_j \mid a_1, a_2, \ldots, a_n) = P(v_j) \cdot \prod_i P(a_i \mid v_j)$$

P(yes|sunny, cool, high, true)

= P(yes). P(sunny|yes).P(cool|yes).P(high|yes).P(true|yes)

= 9/14 . 2/9 . 3/9 . 3/9 . 3/9 = 0.0053

P(no |sunny, cool, high, true)

= P(no). P(sunny|no).P(cool| no).P(high| no).P(true| no)

= 5/14 . 3/5 . 1/5 . 4/5 . 3/5 = $\boxed{0.0206}$

THANK YOU

INSTITUT TEKNOLOGI BANDUNG

1920

Modul 6: Rule-based System

01 What & Why

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

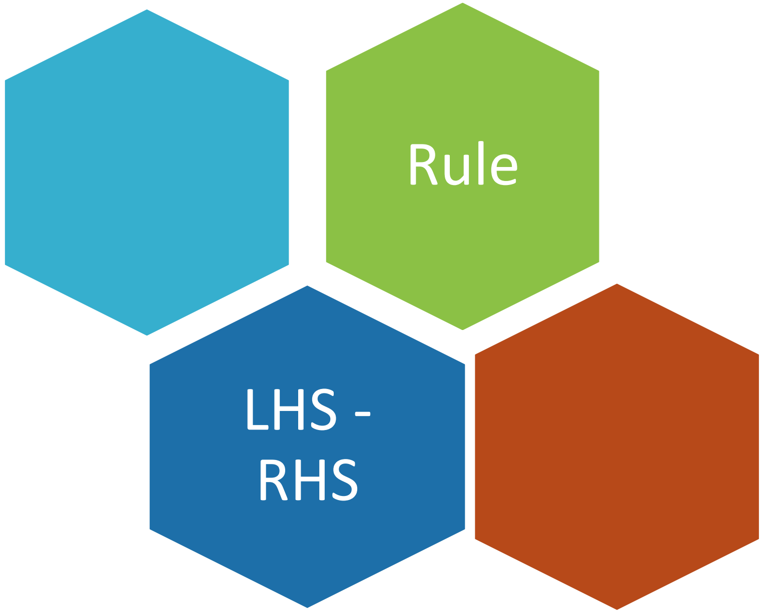
Rule-based System

- What & Why RBS
- Forward Chaining
- Backward Chaining

Rule-based System (RBS): What

KBS with rule as knowledge representation

Rule =
precondition - action

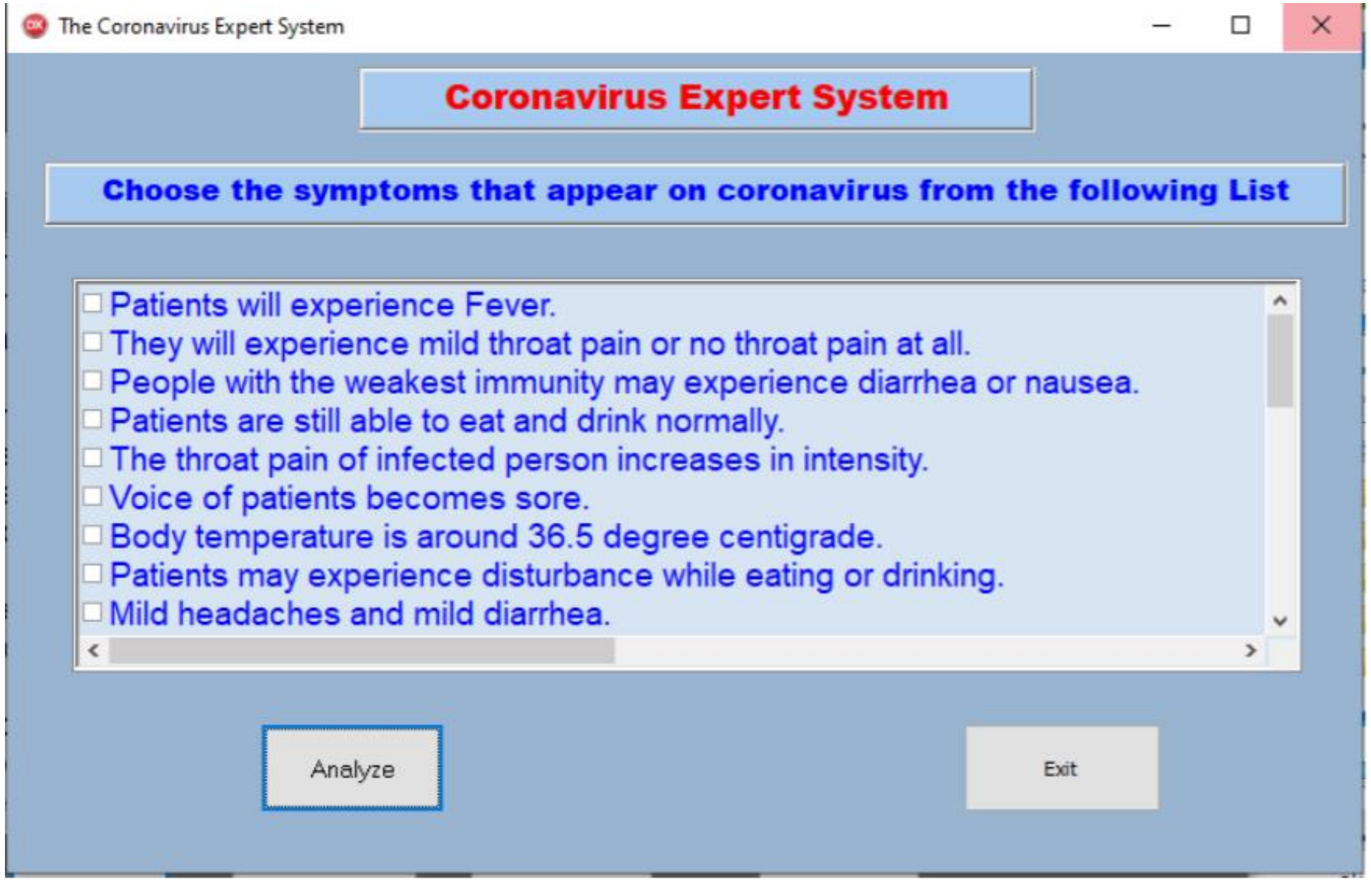


Rule-based System: Why

Rule-based system: the simplest and most widespread solution in the real world

Rule: the simplest and most common knowledge representation

Rule-based ES shell: CLIPS



Salman, F. M., & Abu-Naser, S. S. (2020). Expert System for COVID-19 Diagnosis. International Journal of Academic Information Systems Research (IJAISR)

RBS: Why

Hybrid Approach: RBS+ML

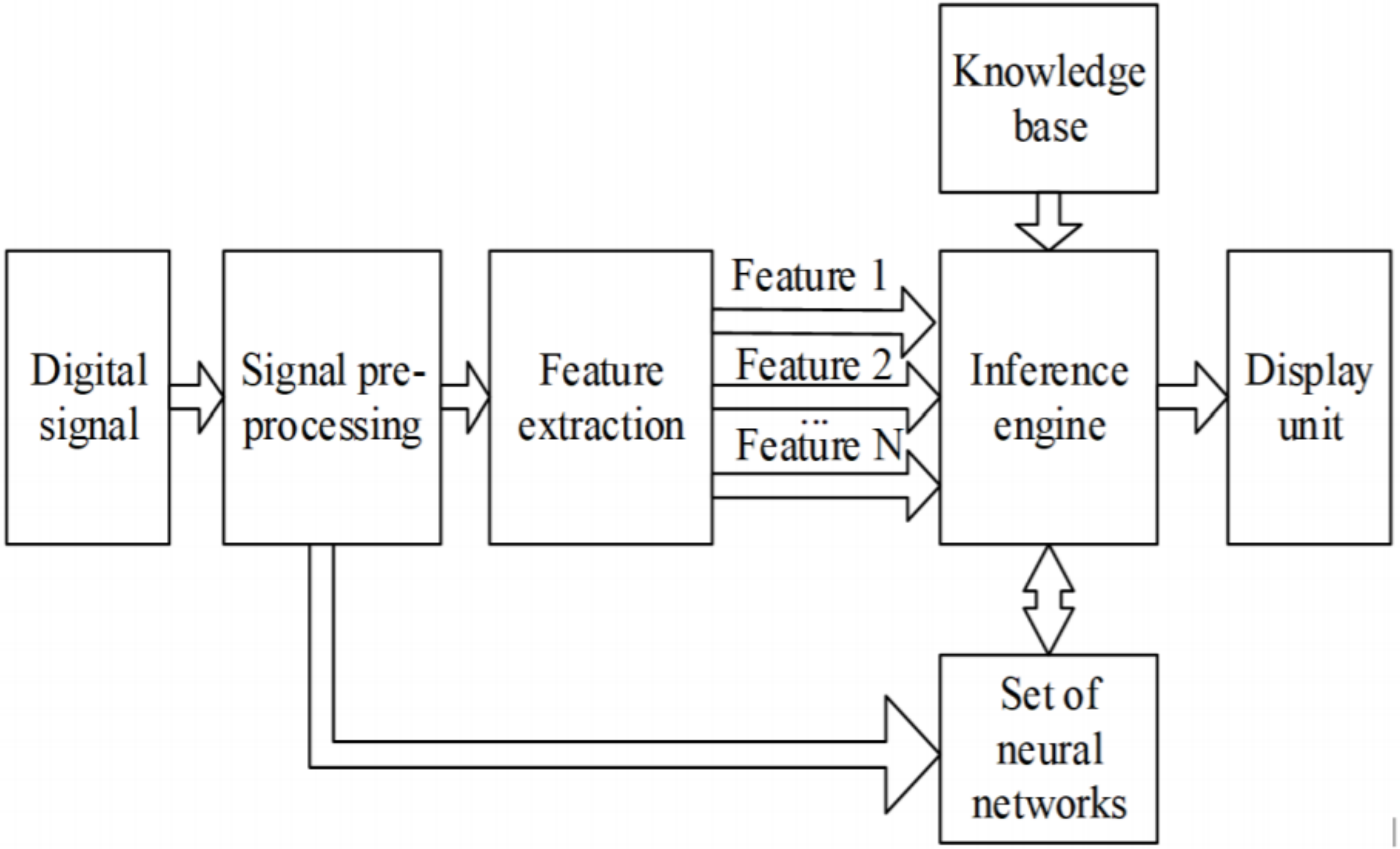


Figure 1. Structure diagram of the software for signal classification.

Donskih, D. N., & Barabanov, V. F. (2020, March). Usage of production-based expert system and neural network for signal recognition. In *Journal of Physics: Conference Series* (Vol. 1479, No. 1, p. 012060). IOP Publishing.

Rule: Logical Implication

```
IF    certain conditions are true
THEN  execute the following actions
```

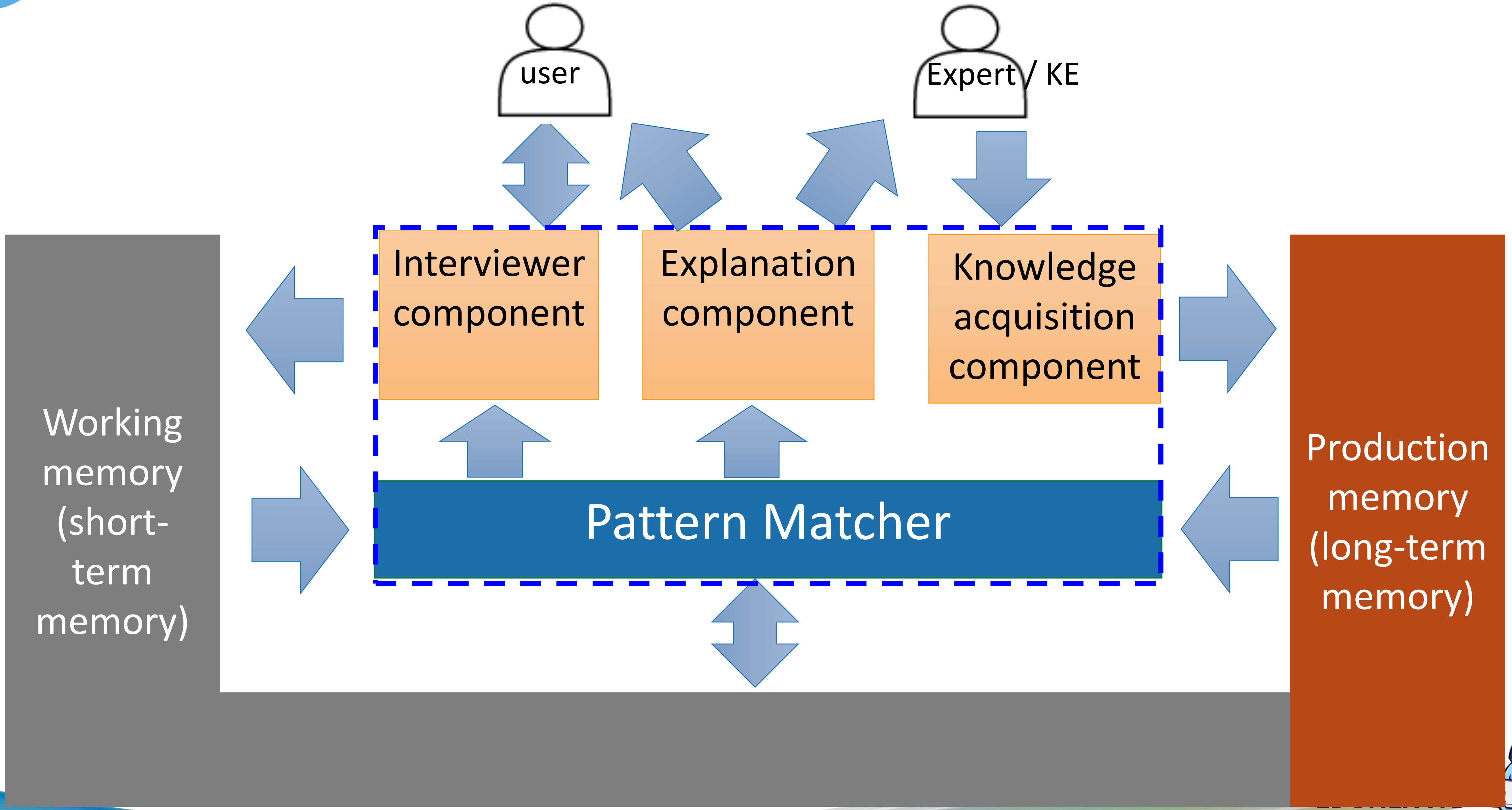
Preconditions, premises, LHS,

Actions, conclusion, RHS

CLIPS: C Language Integrated Production System

```
(defrule R
   (is-a ?x horse)
   (is-parent-of ?x ?y)
   (is-fast ?y)
=>
   (assert (is-valuable ?x))
)
```

General Architecture of RBS



Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Pattern Matching: Example

IF: is-a (x, horse),
is-parent-of(x, *y),*
is-fast(y)
THEN: *x* is valuable

Facts

Comet	is-a	horse
Prancer	is-a	horse
Comet	is-parent-of	Dasher
Comet	is-parent-of	Prancer
Prancer	is	fast
Dasher	is-parent-of	Thunder
Thunder	is	fast
Thunder	is-a	horse
Dasher	is-a	horse

Example: Rule in CLIPS

```
(defrule R
   (is-a ?x horse)
   (is-parent-of ?x ?y)
   (is-fast ?y)
=>
   (assert (is-valuable ?x))
)
(defrule output
   (is-valuable ?x)
=>
   (printout t ?x " is valuable" crlf)
)
```

IF: is-a (x, horse),
is-parent-of(x, *y),*
is-fast(y)
THEN: *x* is valuable

Example: Facts in CLIPS

```
(deffacts horse
   (is-a Comet horse)
   (is-a Prancer horse)
   (is-a Thunder horse)
   (is-a Dasher horse)
)

(deffacts parent
   (is-parent-of Comet Dasher)
   (is-parent-of Comet Prancer)
   (is-parent-of Dasher Thunder)
)

(deffacts fast
   (is-fast Prancer)
   (is-fast Thunder)
)
```

Facts

Comet	is-a	horse
Prancer	is-a	horse
Comet	is-parent-of	Dasher
Comet	is-parent-of	Prancer
Prancer	is	fast
Dasher	is-parent-of	Thunder
Thunder	is	fast
Thunder	is-a	horse
Dasher	is-a	horse

Example in CLIPS: Run

```
CLIPS> (load "horse.clp")
Defining deffacts: horse
Defining deffacts: parent
Defining deffacts: fast
Defining defrule: R +j+j+j+j
Defining defrule: output +j+j
TRUE
CLIPS> (reset)

CLIPS> (run)
Dasher is valuable
Comet is valuable
CLIPS>
```

```
CLIPS> (facts)
f-0      (initial-fact)
f-1      (is-a Comet horse)
f-2      (is-a Prancer horse)
f-3      (is-a Thunder horse)
f-4      (is-a Dasher horse)
f-5      (is-parent-of Comet Dasher)
f-6      (is-parent-of Comet Prancer)
f-7      (is-parent-of Dasher Thunder)
f-8      (is-fast Prancer)
f-9      (is-fast Thunder)
f-10     (is-valuable Dasher)
f-11     (is-valuable Comet)
For a total of 12 facts.
```

CLIPS: Watch

```
FIRE    1 R: f-4,f-7,f-9
f-4     (is-a Dasher horse)
f-7     (is-parent-of Dasher Thunder)
f-9     (is-fast Thunder)
==> f-10    (is-valuable Dasher)
```

```
(defrule R
    (is-a ?x horse)
    (is-parent-of ?x ?y)
    (is-fast ?y)
=>
    (assert (is-valuable ?x))
)
```

```
FIRE    3 R: f-1,f-6,f-8
==> f-11    (is-valuable Comet)
```

```
CLIPS> (reset)
<== f-0     (initial-fact)
==> f-0     (initial-fact)
==> f-1     (is-a Comet horse)
==> f-2     (is-a Prancer horse)
==> f-3     (is-a Thunder horse)
==> f-4     (is-a Dasher horse)
==> f-5     (is-parent-of Comet Dasher)
==> f-6     (is-parent-of Comet Prancer)
==> f-7     (is-parent-of Dasher Thunder)
==> f-8     (is-fast Prancer)
==> f-9     (is-fast Thunder)
CLIPS> (run)
FIRE    1 R: f-4,f-7,f-9
==> f-10    (is-valuable Dasher)
FIRE    2 output: f-10
Dasher is valuable
FIRE    3 R: f-1,f-6,f-8
==> f-11    (is-valuable Comet)
FIRE    4 output: f-11
Comet is valuable
CLIPS>
```

Rule Inference Methods

Forward chaining

- Data driven
- Match LHS

Backward chaining

- Goal driven
- Match RHS

Summary

- What & Why RBS
- Rule syntax
- RBS Architecture
- Inference: Forward vs Backward Chaining

Forward Chaining

Modul 6: Rule-based System

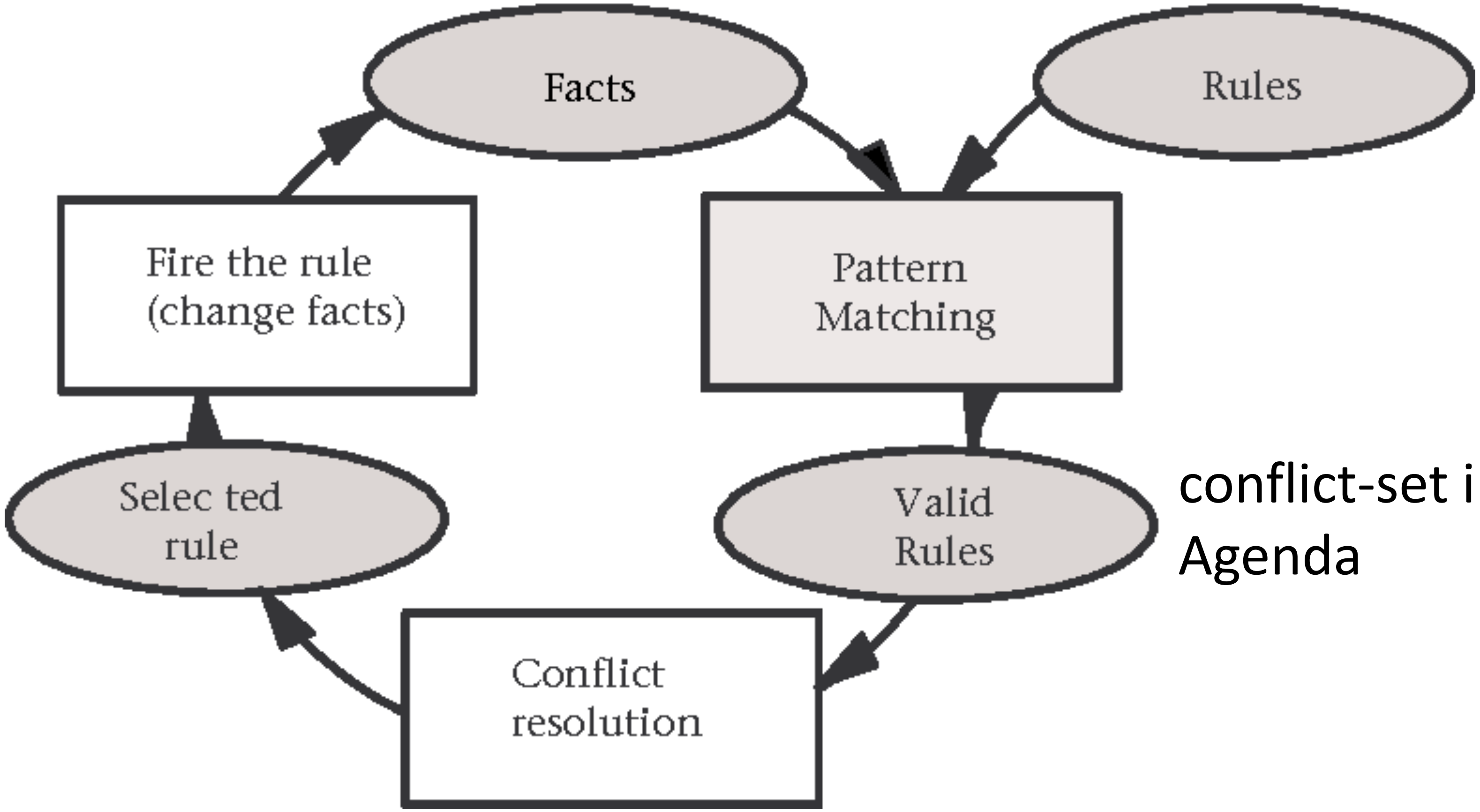
02 Forward Chaining

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Forward Chaining: Recognize-Act Cycle



conflict-set in
Agenda

Barachini, F. (1994). Frontiers in run-time prediction for the production-system paradigm. *AI Magazine*, *15*(3), 47-47.

Forward Chaining: Pseudo code

```
data ← initial facts
repeat
      conflictSet ← determine set of rules whose
                    preconditions are satisfied by data
                    //preselection
      R ← select a rule from conflictSet by conflict-
          resolving strategy
      data ← result of applying action part of R to data
until data satisfied termination condition
```

RBS Example

Rule-base:
R1: IF (lecturing X) AND (marking-practicals X) THEN ADD (overworked X)
R2: IF (month february) THEN ADD (lecturing alison)
R3: IF (month february) THEN ADD (marking-practicals alison)
R4: IF (overworked X) OR (slept-badly X) THEN ADD (bad-mood X)
R5: IF (bad-mood X) THEN DELETE (happy X)
R6: IF (lecturing X) THEN DELETE (researching X)

Facts:
(month february)
(happy alison)
(researching alison)

CS={R2, R3} → R=?

Conflict-resolution Strategy

Global control

- Selection by order: rule order vs fact recency
- Refractoriness: once only
- Specificity: by syntactic structure of the rule

Local control

- Selection by priority
- Selection by meta rules

Refractoriness

→ gk boleh pakai rules itu lg utk data fakta yg sama (ga boleh ngulang pakai rulesnya)

Do not select a rule that has just been applied with the same values of its variables (Brachman, 2004).

Rule-base:

R1: IF (lecturing X) AND (marking-practicals X) THEN ADD (overworked X)

R2: IF (month february) THEN ADD (lecturing alison)

R3: IF (month february) THEN ADD (marking-practicals alison)

R4: IF (overworked X) OR (slept-badly X) THEN ADD (bad-mood X)

R5: IF (bad-mood X) THEN DELETE (happy X)

R6: IF (lecturing X) THEN DELETE (researching X)

Facts:
(month february)
(happy alison)
(researching alison)

Iteration	CS	R
1	{R2, R3}	R2
2	{R2, R3, R6}	R3
3	etc….	

Selection by Order (with Refractoriness)

Knowledge-base:

```
R1: if (priority second)
then out("print second")
R2: if (priority first)
then out("print first")
R3: if (priority third)
then out("print third")
```

Facts:

```
(priority first)
(priority second)
(priority third)
```

- Selection by rule order (FIFO):

print second
print first
print third

iterasi selesai saat tidak ada rules yg bisa dipilih lagi

- Selection by fact (recency) order (LIFO):

print third
print second
print first

Selection by Syntactic Structure of the Rule

- Specificity: select **most specific rule** first
- Example:
  - Conflict set: {R1,R2}
    R1: if A, B, C then <aksi R1>
    R2: if A,C then <aksi R2>
  - A and B and C is more specific than A and C → select R1

Selection by Supplementary Knowledge

Select high priority rule

Example:

R1: If (burung ?X)
then (terbang ya)

R2: If (burung penguin)
(declare salience 100)
then (terbang tidak)

Fakta: (burung penguin)

Meta rules

Pruning rules:

If the culture was not obtained from a sterile source,

there are rules which mention in their premise a previous organism

then each of them is not going to be useful

Forward Chaining: Exercise

What action to take to get to a theatre by using conflict resolution strategy refractoriness, specificity ?

Facts: Distance is about 6 miles; Weather is “bad”; Location is downtown; Time is about 20 minutes

R	IF	THEN
1	Distance > 5 miles	Means is “drive”
2	Distance > 1 mile, time < 15 minutes	Means is “drive”
3	Distance > 1 mile, time > 15 minutes	Means is “walk”
4	Means is “drive”, location is “downtown”	Action is “take a cab”
5	Means is “drive”, location is not “downtown”	Action is “drive your car”
6	Means is “walk”, weather is “bad”	Action is “take a coat and walk”
7	Means is “walk”, weather is “good”	Action is “walk”

Facts: Distance is about 6 miles; Weather is "bad"; Location is downtown; Time is about 20 minutes

R	IF	THEN
1	Distance > 5 miles	Means is "drive"
2	Distance > 1 mile, time < 15 minutes	Means is "drive"
3	Distance > 1 mile, time > 15 minutes	Means is "walk"
4	Means is "drive", location is "downtown"	Action is "take a cab"
5	Means is "drive", location is not "downtown"	Action is "drive your car"
6	Means is "walk", weather is "bad"	Action is "take a coat and walk"
7	Means is "walk", weather is "good"	Action is "walk"

conflict resolution strategy refractoriness, specificity , fact recency

Iteration	CS	R	WM
1	{R1, R3}	R3	+ Means is "walk"
2	{R1, R3, R6}	R6	+ Action is "take a coat and walk"
3	{R1, R3, R6}	R1	+ Means is "drive"
4	{R1, R3, R6, R4}	R4	+ Action is "take a cab"
5	{R1, R3, R6, R4}	-	stop

Conclusion:

+ Action is "take a coat and walk"

+ Action is "take a cab"

conflict resolution strategy refractoriness, fact recency, specificity

Iteration	CS	R	WM
1	{R1, R3}	R3	+ Means is "walk"
2	{R1, R3, R6}	R6	+ Action is "take a coat and walk"
3	{R1, R3, R6}	R1	+ Means is "drive"
4	{R1, R3, R6, R4}	R4	+ Action is "take a cab"
	{R1, R3, R6, R4}	-	stop

EDUNEX ITB

Summary

Forward Chaining

Conflict resolution strategy

Global control: refractoriness, rule order, recency, specificity

Local control: priority, meta rules

Backward Chaining

INSTITUT TEKNOLOGI BANDUNG

1920

INSTITUT TEKNOLOGI BANDUNG

1920

Modul 6: Rule-based System

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

03 Backward Chaining

Inteligensi Buatan
(*Artificial Intelligence*)

WM

Fact_1

rules
$$\begin{cases} Fact\_1 \rightarrow Fact\_2 \\ Fact\_2 \rightarrow Fact\_3 \\ Fact\_3 \rightarrow Fact\_4 \\ Fact\_4 \rightarrow Fact\_5 \end{cases}$$

Action=Fact_5

Forward Chaining	Backward Chaining
Match LHS	Match RHS
Data-driven reasoning	Goal-driven reasoning

$$\left.\begin{matrix} Goal\_1 \rightarrow Goal\_2 \\ Goal\_2 \rightarrow Goal\_3 \\ Goal\_3 \rightarrow Goal\_4 \\ Goal\_4 \rightarrow Goal\_5 \end{matrix}\right\}$$ rules

Question

Forward Chaining: Is Z true ?

Rule-base:

R1: Y, D → Z

R2: X, B, E → Y

R3: A → X

R4: C → L

R5: L, M → N

Facts:

A,B,C,D,E

Conflict resolution strategy:

refractoriness > fact recency > specificity > rule order

Iteration	Conflict set	Selected Rule	Working memory
1	{R3, R4}	R4	+ L
2	{R3, R4}	R3	+ X
3	{R2, R3, R4}	R2	+ Y
4	{R1, R2, R3, R4}	R1	+Z
5	{R1, R2, R3, R4}	-	stop

Answer: Yes

cek dulu sampai ga ada yg bisa dipilih baru stop, jgn langsung walau udah dapat z

Backward Chaining: Is Z true ?

Rule-base:

R1: Y, D → Z

R2: X, B, E → Y

R3: A → X

R4: C → L

R5: L, M → N

Facts:

A,B,C,D,E

Question: Z?

R1: Y, D → Z

R2: X, B, E → Y

D? Yes / True

R3: A → X

B? Yes / True

E? Yes / True

A? Yes / True

Answer: Yes

(1) cari rule yg sisi kanannya Z

(2) cari rule yg RHS nya mengandung Y atau D.

* ga ada rule yg RHS nya D, tapi masih bisa lanjut karena ga ada yg nilainya false.

(3) periksa rule yg ada RHS X, B, E

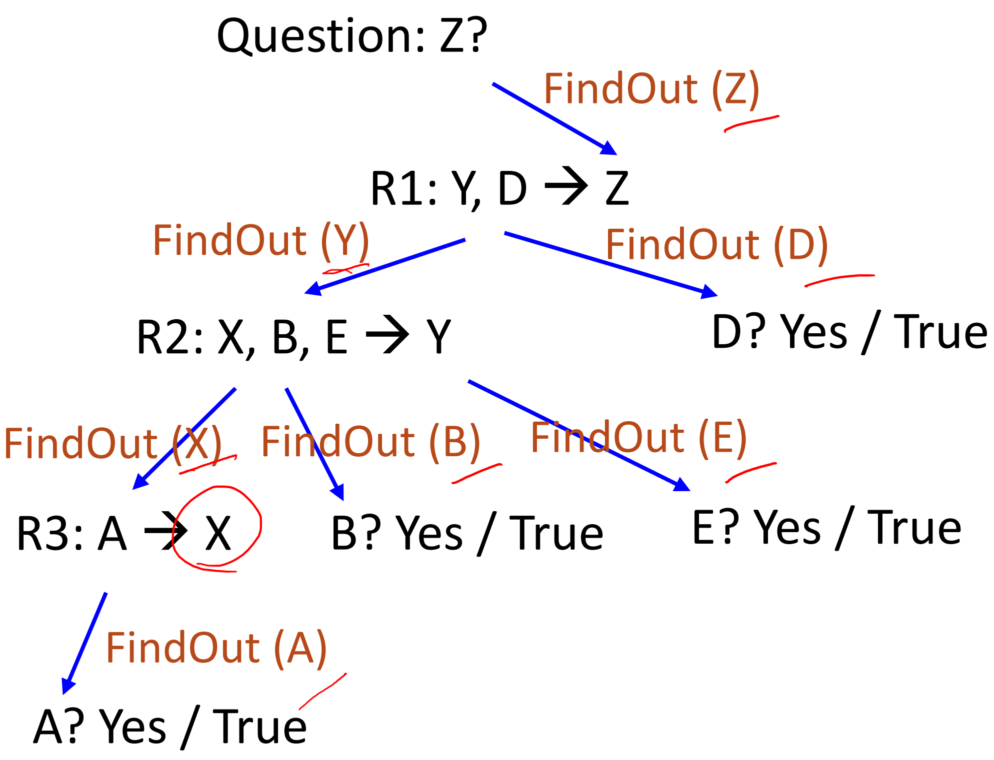
A ← X

B sama E ga ada

(4) cari yg RHS nya A → ga ada. jd True!

↳ rulelist empty.

Interpreter: FindOut (Goal)



<u>Procedure</u> FINDOUT (GOAL)

<u>If</u> (GOAL can be inferred)

<u>then</u>

  set RULE-LIST = list all rules whose action part fulfills GOAL

  until (RULE-LIST = empty) or (GOAL inferred) do

    MONITOR(first or next rule from RULE-LIST)

    delete this rule from RULE-LIST

<u>else</u> (request GOAL)

Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Backward Chaining Process

Rule-base:

R1: Y, D → Z

R2: X, B, E → Y

R3: A → X

R4: C → L

R5: L, M → N

Question: Z?

FindOut (Z)

R1: Y, D → Z

FindOut (Y)

FindOut (D)

R2: X, B, E → Y

D? Yes / True

FindOut (X)

FindOut (B)

FindOut (E)

R3: A → X

B? Yes / True

E? Yes / True

FindOut (A)

A? Yes / True

FindOut (Z) {R1}

Monitor(R1)

<u>Procedure</u> FINDOUT (GOAL)

<u>If</u> (GOAL can be inferred)

<u>then</u>

set RULE-LIST = list all rules whose action part fulfills GOAL

until (RULE-LIST = empty) or (GOAL inferred) do

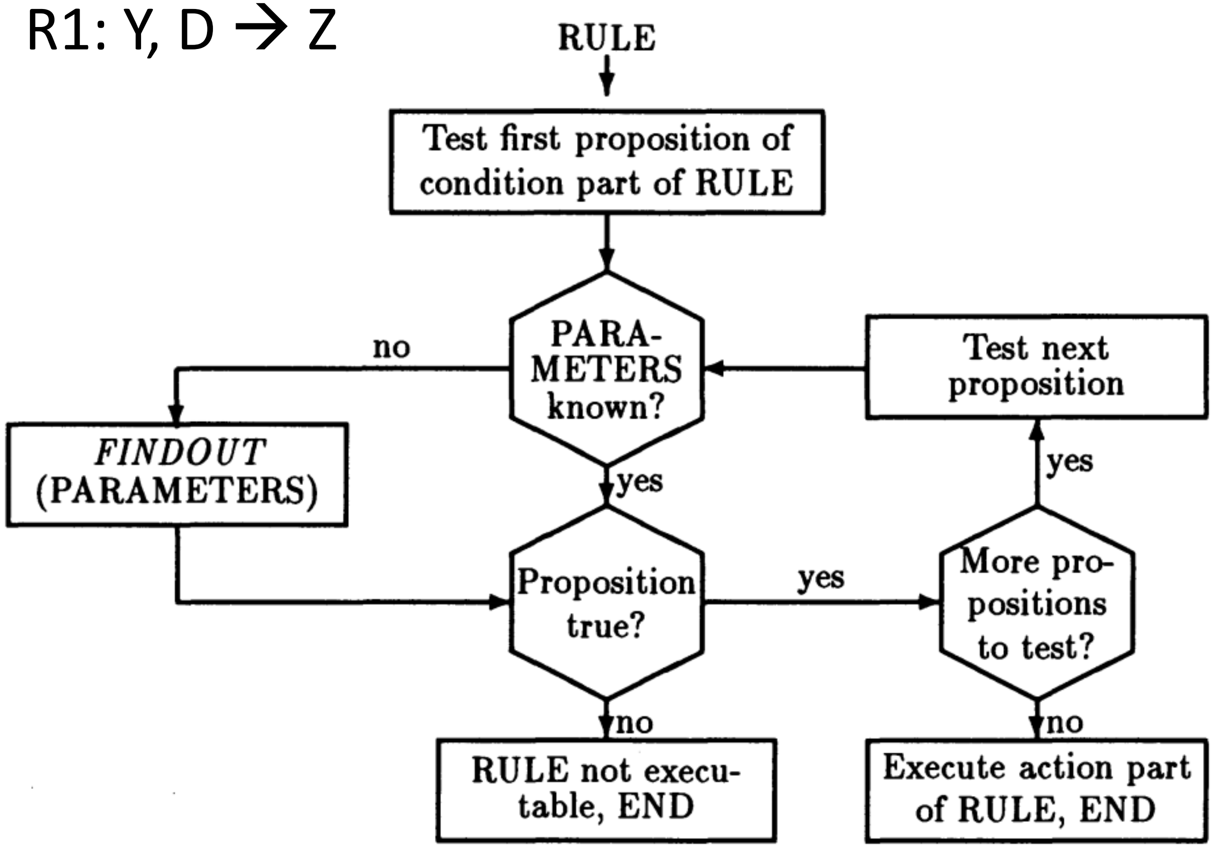
MONITOR(first or next rule from RULE-LIST)

delete this rule from RULE-LIST

<u>else</u> (request GOAL)

Interpreter: Monitor (Rule)

R1: Y, D → Z



Procedure MONITOR (RULE)

Test first proposition of condition part of RULE

repeat

If parameters known then

if proposition true then

proposition ← next proposition

else RULE not executable

else FINDOUT(PARAMETERS)

Until (no more propositions to test) or (RULE not executable)

If (no more propositions to test) then

execute action part of RULE

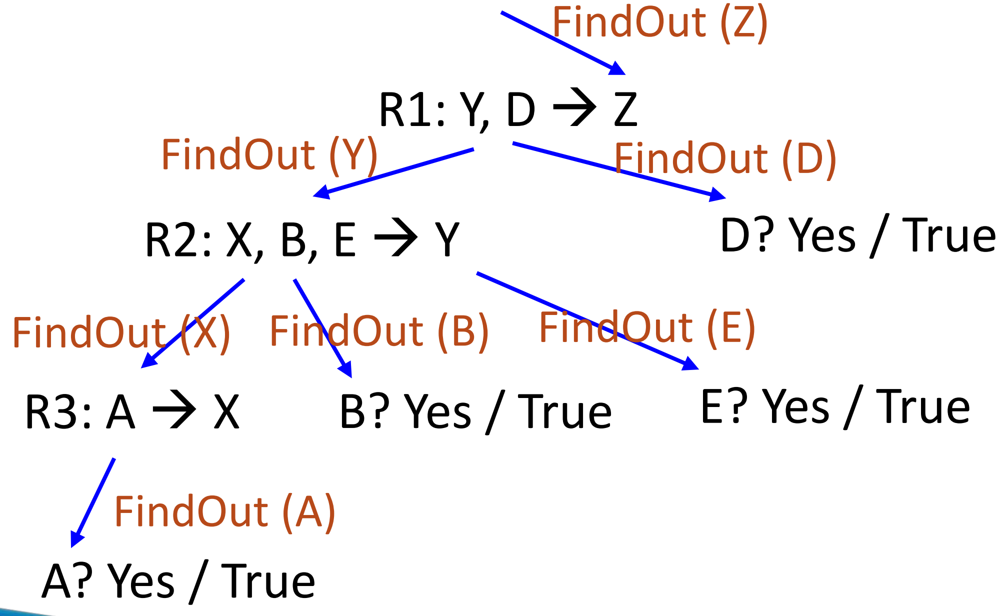
Puppe, F. (1993). *Systematic Introduction to Expert Systems* . Springer, Berlin, Heidelberg.

Backward Chaining Process

Rule-base:
R1: Y, D → Z
R2: X, B, E → Y
R3: A → X
R4: C → L
R5: L, M → N

Facts:
A,B,C, E

Question: Z?



FindOut (Z)	{R1}
Monitor(R1)	{Y,D}
FindOut(Y)	{R2}
Monitor(R2)	{X,B,E}
FindOut(X)	{R3}
Monitor(R3)	{A}
FindOut(A)	True
Execute(R3)	+X
Delete(R3)	
FindOut(B)	True
FindOut(E)	True
Execute(R2)	+Y
Delete (R2)	
FindOut(D)	request(D): True
Execute(R1)	+Z
Delete(R1)	

Answer: Yes

Infenence pakai backward chaining

↳ cari rule yg RHSnya action.

What action to take to get to a theatre

↳ di findoutnya dari rule yg mulai/munculnya lebih awal

Inference using Backward Chaining to decide what action to take to get to a theatre. Start the process by FindOut(Action) until first action is inferred, and working memory is empty.

R	IF	THEN
1	Distance > 5 miles	Means is "~~drive~~" walk
2	Distance > 1 mile, time < 15 minutes	Means is "drive"
3	Distance > 1 mile, time > 15 minutes	Means is "walk"
4	Means is "drive", location is "downtown"	Action is "take a cab"
5	Means is "drive", location is not "downtown"	Action is "drive your car"
6	Means is "walk", weather is "bad"	Action is "take a coat and walk"
7	Means is "walk", weather is "good"	Action is "walk"

Request facts:
Distance is about 6 miles;
Weather is "bad";
Location is downtown;
Time is about 20 minutes

What action to take to get to a theatre

Inference using Backward Chaining to decide what action to take to get to a theatre. Working memory is empty. Start the process by FindOut(Action) until first action is inferred.

R	IF	THEN
1	Distance > 5 miles	Means is "drive"
2	Distance > 1 mile, time < 15 minutes	Means is "drive"
3	Distance > 1 mile, time > 15 minutes	Means is "walk"
4	Means is "drive", location is "downtown"	Action is "take a cab"
5	Means is "drive", location is not "downtown"	Action is "drive your car"
6	Means is "walk", weather is "bad"	Action is "take a coat and walk"
7	Means is "walk", weather is "good"	Action is "walk"

Request facts:
Distance is about 6 miles;
Weather is "bad";
Location is downtown;
Time is about 20 minutes

```
Procedure MONITOR (RULE)
Test first proposition of condition part of RULE
repeat
    If parameters known then
        if proposition true then
            proposition ← next proposition
        else RULE not executable
    else FINDOUT(PARAMETERS)
Until (no more propositions to test) or (RULE
    not executable)
If (no more propositions to test) then
    execute action part of RULE
```

Kayaknya ini slidenya salah

R	IF	THEN
1	Distance > 5 miles	Means is "walk"
2	Distance > 1 mile, time < 15 minutes	Means is "drive"
3	Distance > 1 mile, time > 15 minutes	Means is "walk"
4	Means is "drive", location is "downtown"	Action is "take a cab"
5	Means is "drive", location is not "downtown"	Action is "drive your car"
6	Means is "walk", weather is "bad"	Action is "take a coat and walk"
7	Means is "walk", weather is "good"	Action is "walk"

```
FindOut(Action)                      (R4,R5,R6,R7)
    Monitor(R4)                      (means is drive ? F,
        FindOut(means)               (R1,R2,R3)
            Monitor(R1)              (distance > 5 miles ?
                FindOut(distance)        request(distance): 6 miles
                Execute(R1)          + means is walk
            Delete(R1)
        R4 not executable
    Delete(R4)
    Monitor(R5)
Stop
```

```
Procedure FINDOUT (GOAL)
If (GOAL can be inferred)
then
    set RULE-LIST = list all rules whose action part fulfills GOAL
    until (RULE-LIST = empty) or (GOAL inferred) do
        MONITOR(first or next rule from RULE-LIST)
        delete this rule from RULE-LIST
else (request GOAL)
```

```
Procedure MONITOR (RULE)
Test first proposition of condition part of RULE
repeat
    If parameters known then
        If proposition true then
            proposition ← next proposition
        else RULE not executable
    else FINDOUT(PARAMETERS)
Until (no more propositions to test) or (RULE not executable)
If (no more propositions to test) then
    execute action part of RULE
```

EDUNEX ITB

kan means is drive, kok rule yg masuk ga R2 aja?
krn yg masuk itu parameternya => means
drive itu cuman value doang.
makanya semua yg masuk tu semua yg mengandung
means => makanya R1, R2, R3 masuk.

R	IF	THEN
1	Distance > 5 miles	Means is "drive"
2	Distance > 1 mile, time < 15 minutes	Means is "drive"
3	Distance > 1 mile, time > 15 minutes	Means is "walk"
4	Means is "drive", location is "downtown"	Action is "take a cab"
5	Means is "drive", location is not "downtown"	Action is "drive your car"
6	Means is "walk", weather is "bad"	Action is "take a coat and walk"
7	Means is "walk", weather is "good"	Action is "walk"

Inference using Backward Chaining to decide what action to take to get to a theatre. Start the process by FindOut(Action) until first action is inferred, and working memory is empty.

Request facts:
Distance is about 6 miles;
Weather is "bad";
Location is downtown;
Time is about 20 minutes

findout(action)	{R4,R5,R6,R7}
Monitor(R4)	{means=drive, location=downtown}
findout(means)	{R1, R2, R3}
monitor(R1)	{distance > 5 miles}
FindOut(distance)	request(distance): 6 miles
execute(R1)	+ means=drive
delete(R1)	
stop	
findout(location)	request(location): downtown
execute(R4)	+ action=take a cab
delete(R4)	
stop	

R7 gpp gausah di cek soalnya goalnya sudah berhasil di inference.
(jd ga kosong rule list isokei).

Answer: action=take a cab

berhasil menginferensi action.

EDUNEX ITB

mau ga mau utk soal ini
semua rules harus di cek.

Latihan

Basis pengetahuan dari sistem yang menentukan *resort* bagi *skier*:

R1: <u>if</u> Rating = beginner, Purpose = fun **<u>then</u>** Resort = St.Sartre
R2: <u>if</u> Rating = beginner, Purpose = serious **then** Resort = Schloss Heidegger
R3: <u>if</u> Rating = advanced, Purpose = serious **<u>then</u>** Resort = Chateau Derrida
R4: <u>if</u> Rating = advanced, Purpose = fun **<u>then</u>** Resort = Wittgenstein Gladbach
R5: <u>if</u> Lessons < 30 hours **<u>then</u>** Rating = beginner
R6: <u>if</u> Lessons >= 30 hours, Fitness = poor **<u>then</u>** Rating = beginner
R7: <u>if</u> Lessons >= 30 hours, Fitness = good **<u>then</u>** Rating = advanced
R8: <u>if</u> Pressups < 10 **<u>then</u>** Fitness = poor
R9: <u>if</u> Pressups >= 10 **<u>then</u>** Fitness = good

BC: WM kosong, jawaban saat request: purpose = fun, lesson = 178, pressups = 15

↳ goal

R1: If Rating = beginner, Purpose = fun then Resort = St. Sartre
R2: If Rating = beginner, Purpose = serious then Resort = Schloss Heidegger
R3: If Rating = advanced, Purpose = serious then Resort = Chateau Derrida
R4: If Rating = advanced, Purpose = fun then Resort = Wittgenstein Gladbach
R5: If Lessons < 30 hours then Rating = beginner
R6: If Lessons >= 30 hours, Fitness = poor then Rating = beginner
R7: If Lessons >= 30 hours, Fitness = good then Rating = advanced
R8: If Pressups < 10 then Fitness = poor
R9: If Pressups >= 10 then Fitness = good

Backward Chaining

FindOut(Resort)	{R1,R2,R3,R4}
Monitor(R1)	Rating = beginner, Purpose = fun
FindOut(rating)	{R5,R6,R7}
Monitor(R5)	Lessons < 30 hours
FindOut(lessons)	request(lesson): 178
R5 not executable	
Delete R5	
Monitor(R6)	Lessons >= 30 hours, Fitness = poor
FindOut(fitness)	{R8,R9}
Monitor(R8)	Pressups < 10
FindOut(pressups)	request(pressups): 15
R8 not executable	
Delete R8	
Monitor(R9)	Pressups >= 10
Execute R9	+ Fitness=good
Delete R9	
R6 not executable	
Delete R6	
Monitor(R7)	Lessons >= 30 hours, Fitness = good
Execute R7	+ Rating=advanced
Delete R7	
R1 not executable	
Delete R1	
Monitor(R2)	Rating = beginner, Purpose = serious
R2 not executable	
Delete R2	
Monitor (R3)	Rating = advanced, Purpose = serious
FindOut(purpose)	request(purpose): fun
R3 not executable	
Delete R3	
Monitor(R4)	Rating = advanced, Purpose = fun
Execute R4	+ Resort=Wittgenstein Gladbach
Delete R4	
Terminate	

Fakta pada WM
purpose = fun,
lesson = 178,
pressups = 15

Rule-based System Features

Modularity

- Each rule defines a small, relatively independent piece of knowledge

Incrementability

- New rules can be added to the knowledge base relatively independently of other rules

Modifiability

- Old rules can be changed relatively independently of other rules

Support systems transparency

Summary

Forward vs Backward Chaining

Goal-driven reasoning; Match RHS

FindOut(Goal) & Monitor(Rule)

Referensi

1. Frank Puppe, Systematic Introduction to Expert Systems: Knowledge Representations and Problem-Solving Methods, Springer, 1st ed. 1993
2. Peter Jackson, Introduction To Expert Systems, Addison-Wesley 3rd Edition, 1999,

INSTITUT TEKNOLOGI BANDUNG

1920

RBS

- R1: If temperature<37 then no fever
- R2: If temperature>37 and temperature<38 then low fever
- R3: If temperature>38 then high fever
- R4: If light nasal breathing then nasal discharge
- R5: If heavy nasal breathing then sinus membranes swelling
- R6: If low fever and headache and nasal discharge and cough then cold
- R7: If cold and not(soar throat) then don't treat
- R8: If cold and soar throat then treat
- R9: If don't treat then action: don't give medication
- R10: If treat then give medication
- R11: If give medication and antibiotics allergy then action: give Tylenol
- R12: If give medication and not(antibiotics allergy) then action: give antibiotics

Aksi apa yang harus dilakukan?

WM kosong, ketika request berikut fakta yang dimasukkan:

Fakta: headache; light nasal breathing; temperature=37,5; cough;
not(antibiotics allergy); soar throat

findout(Action) R9, R11, R12

→ lihat yg RHSnya don't treat R7, R8.

→ lihat yg RHSnya cold R6

→ RHSnya fever R1, R2, R3

monitor(R1) false delete R1
monitor(R2) cocok, masuk ke R2.

→ R6 udah true yg fever, lanjut cek headache
headache ga ada, lanjut check nasal R4
R6 → True.

R7 → salah karena not(sore throat) salah
jadi lanjut cek R8.

R8 → R10.
give medication
R11, R12
false
masuk ke.