INSTITUT TEKNOLOGI BANDUNG

1920

Modul 6: Rule-based System

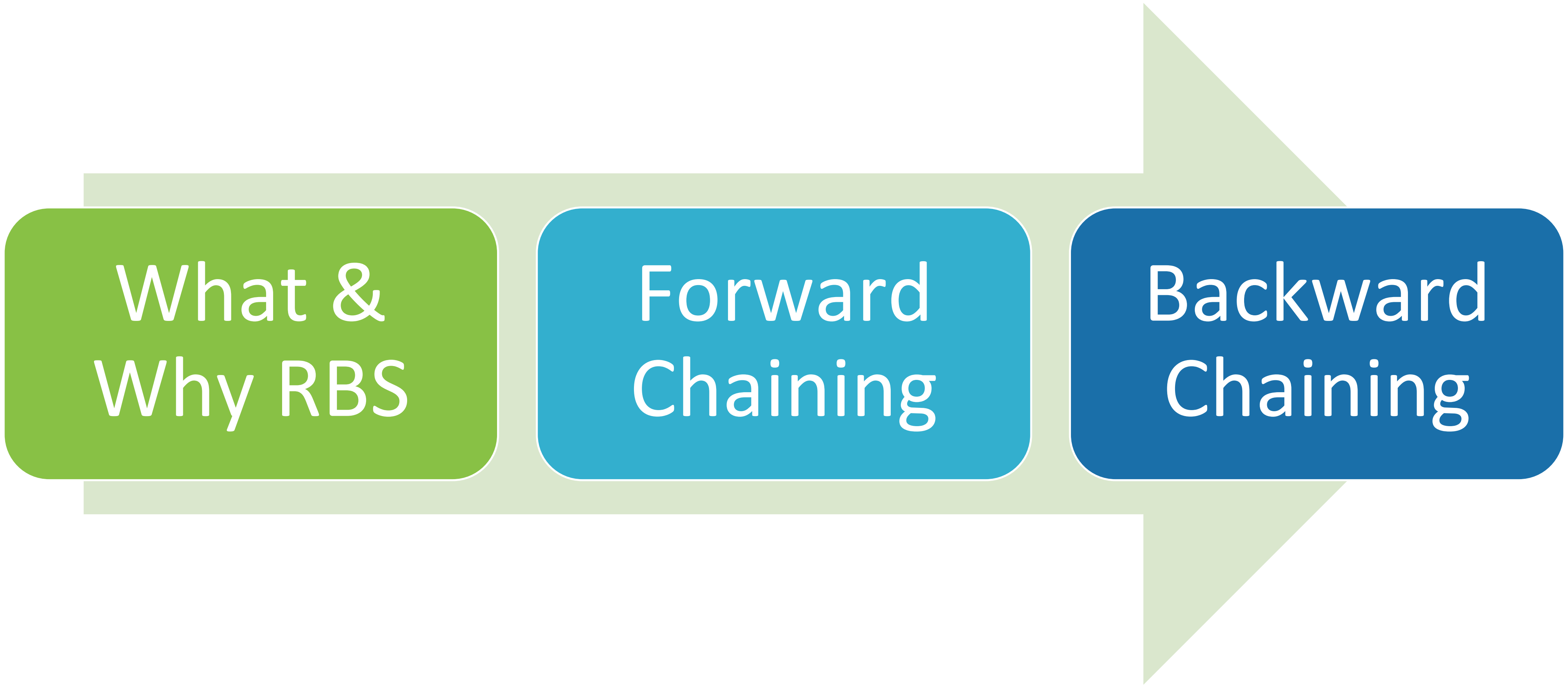
Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

01 What & Why

Inteligensi Buatan
(*Artificial Intelligence*)

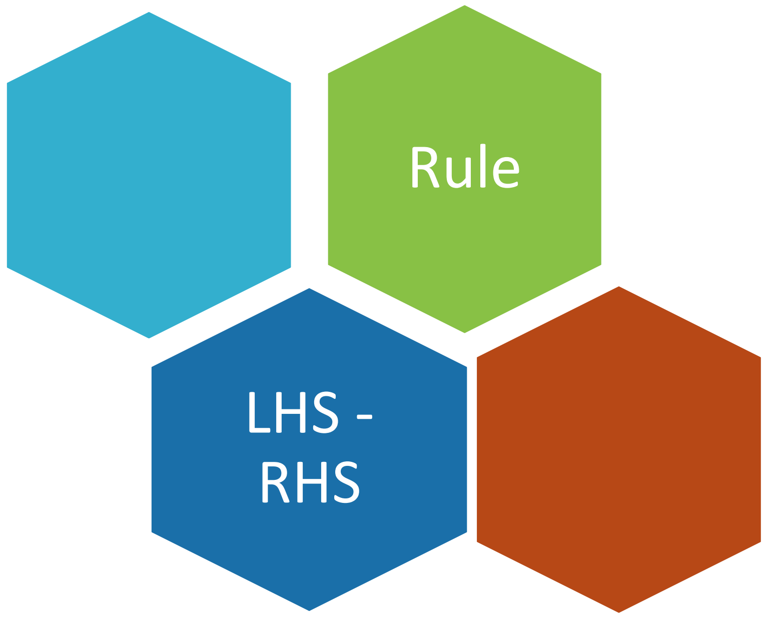
Rule-based System



Rule-based System (RBS): What

KBS with rule as knowledge representation

Rule =
precondition - action

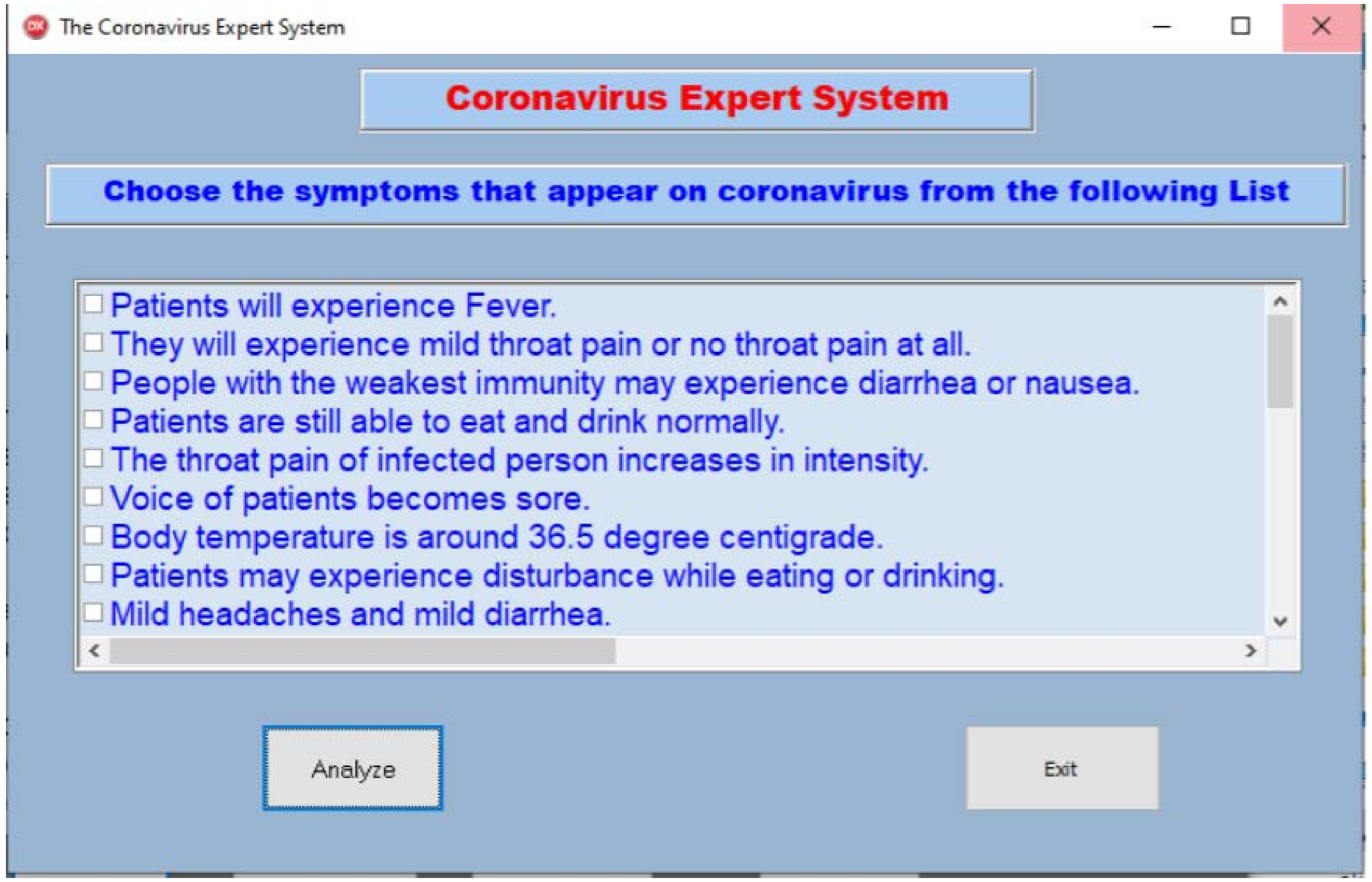


Rule-based System: Why

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


The Coronavirus Expert System
Coronavirus Expert System
Choose the symptoms that appear on coronavirus from the following List
Analyze
Exit

☐ Patients will experience Fever.
☐ They will experience mild throat pain or no throat pain at all.
☐ People with the weakest immunity may experience diarrhea or nausea.
☐ Patients are still able to eat and drink normally.
☐ The throat pain of infected person increases in intensity.
☐ Voice of patients becomes sore.
☐ Body temperature is around 36.5 degree centigrade.
☐ Patients may experience disturbance while eating or drinking.
☐ Mild headaches and mild diarrhea.

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

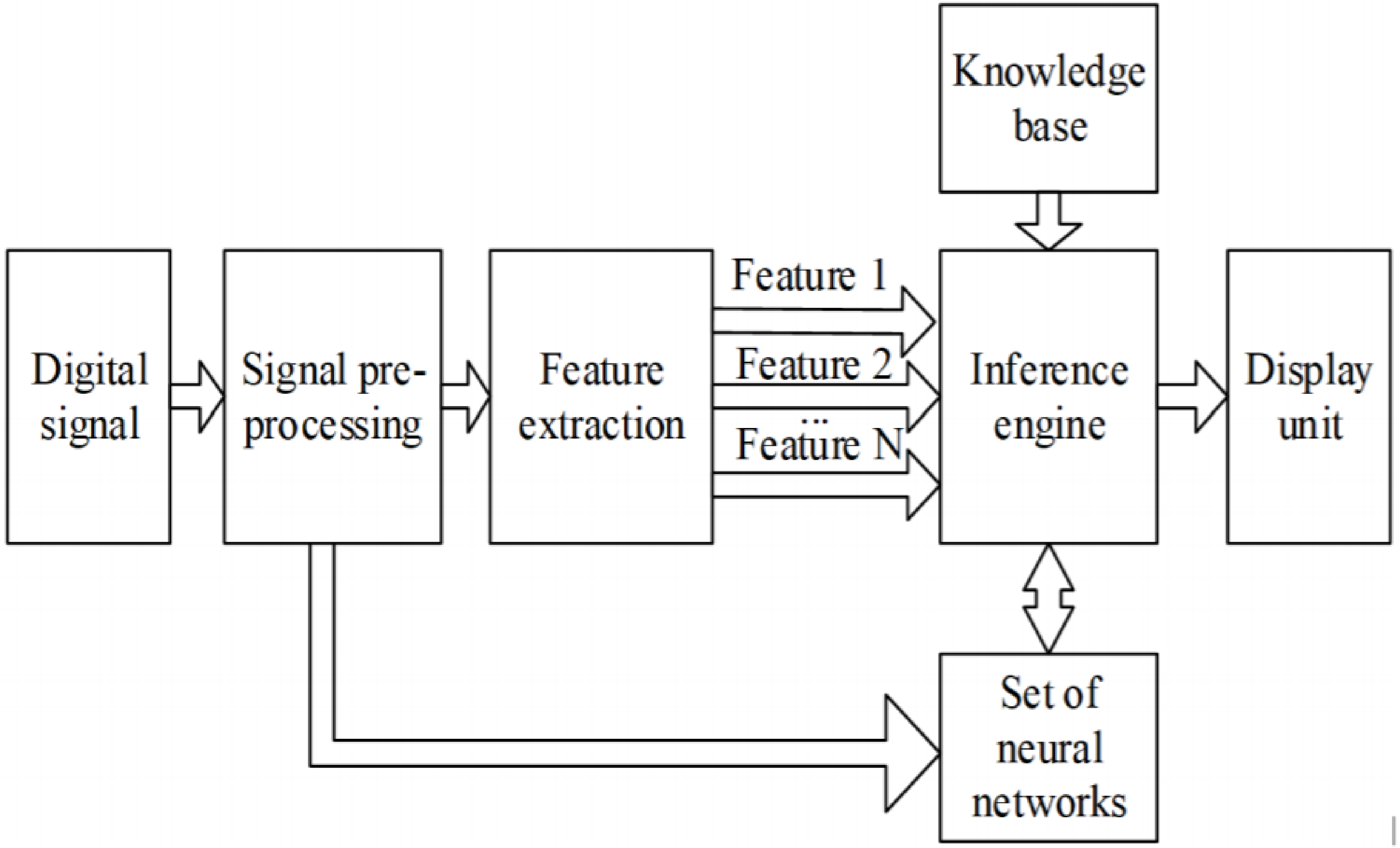


Figure 1. Structure diagram of the software for signal classification.

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
```

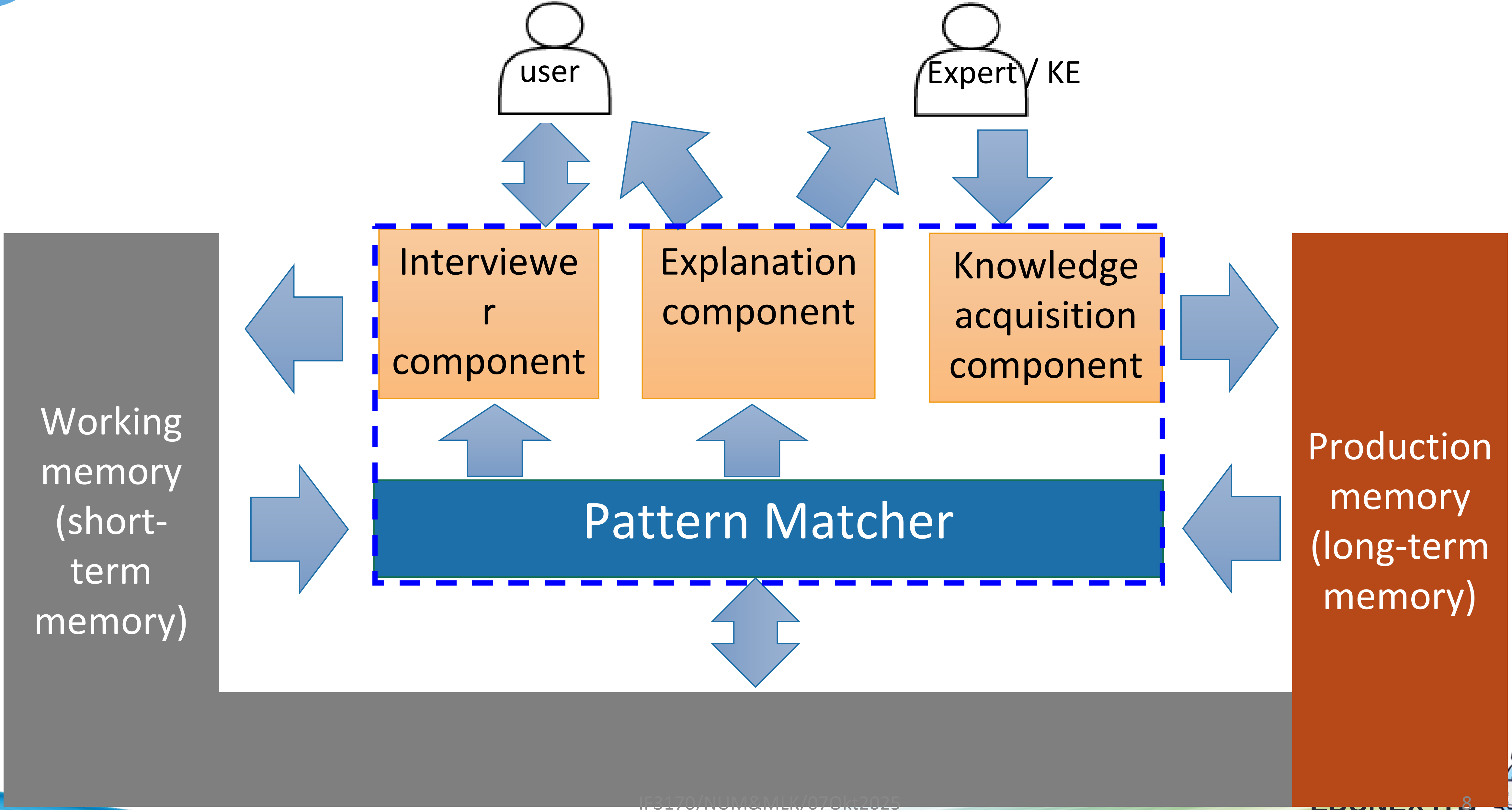
Preconditions, premises, LHS,

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

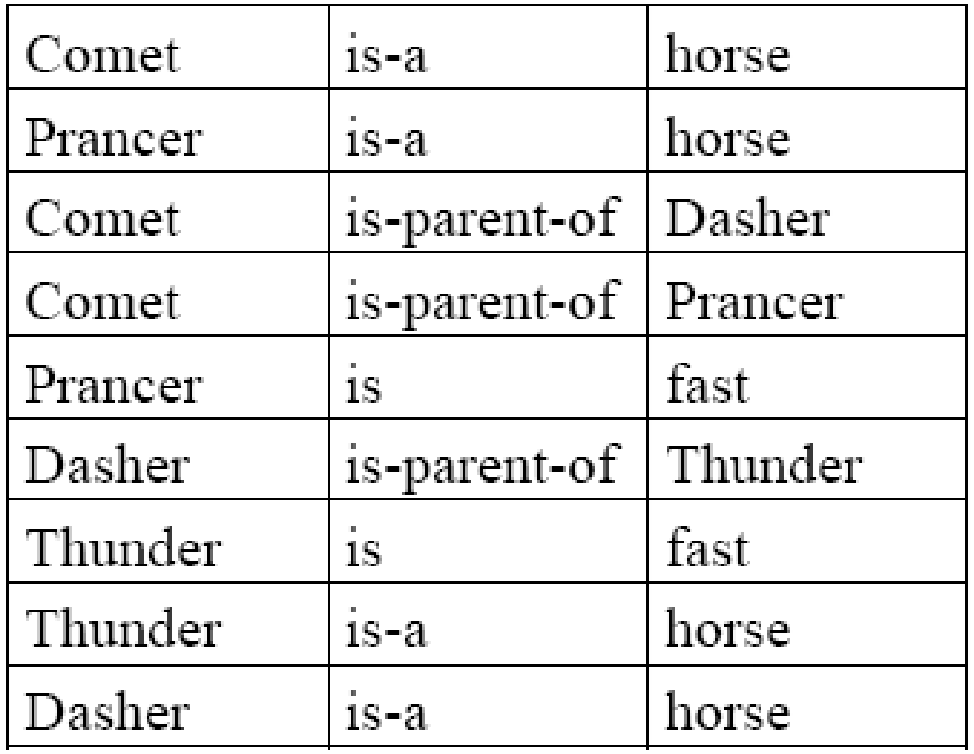

user
Expert / KE
Interviewe
r
component
Explanation component
Knowledge acquisition component
Pattern Matcher
Working memory (short-term memory)
Production memory (long-term memory)

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)
)
```

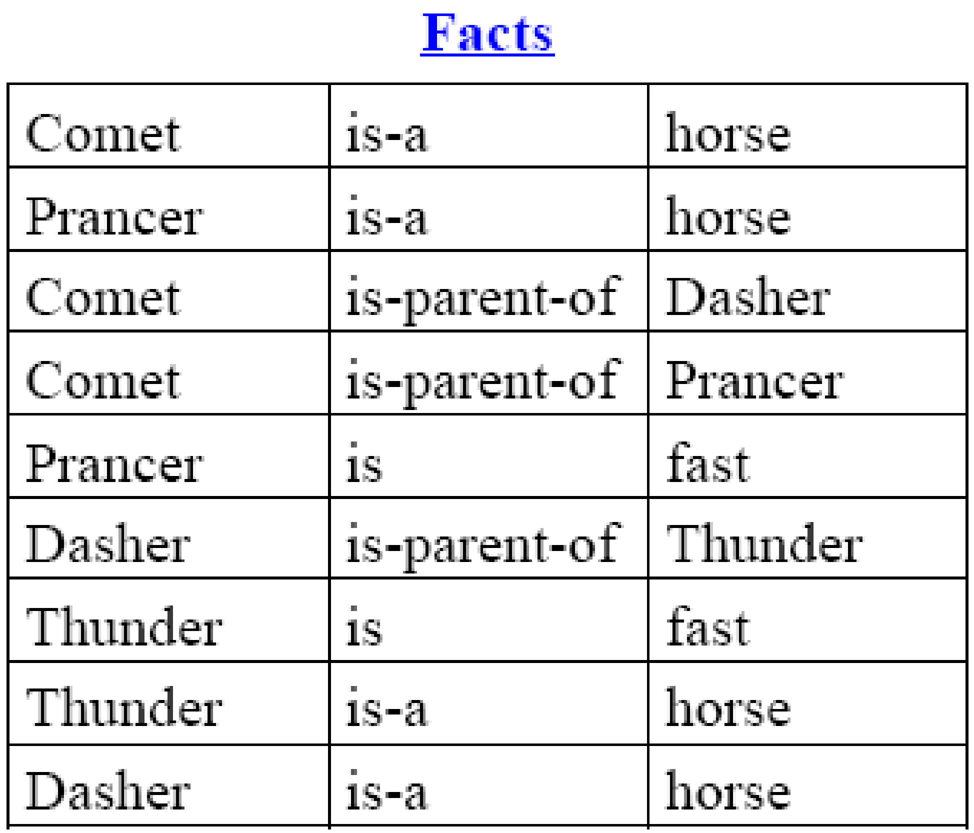
IF: is-a (x, horse),
is-parent-of(x, *y),*
is-fast(y)
THEN: *x* is valuable

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

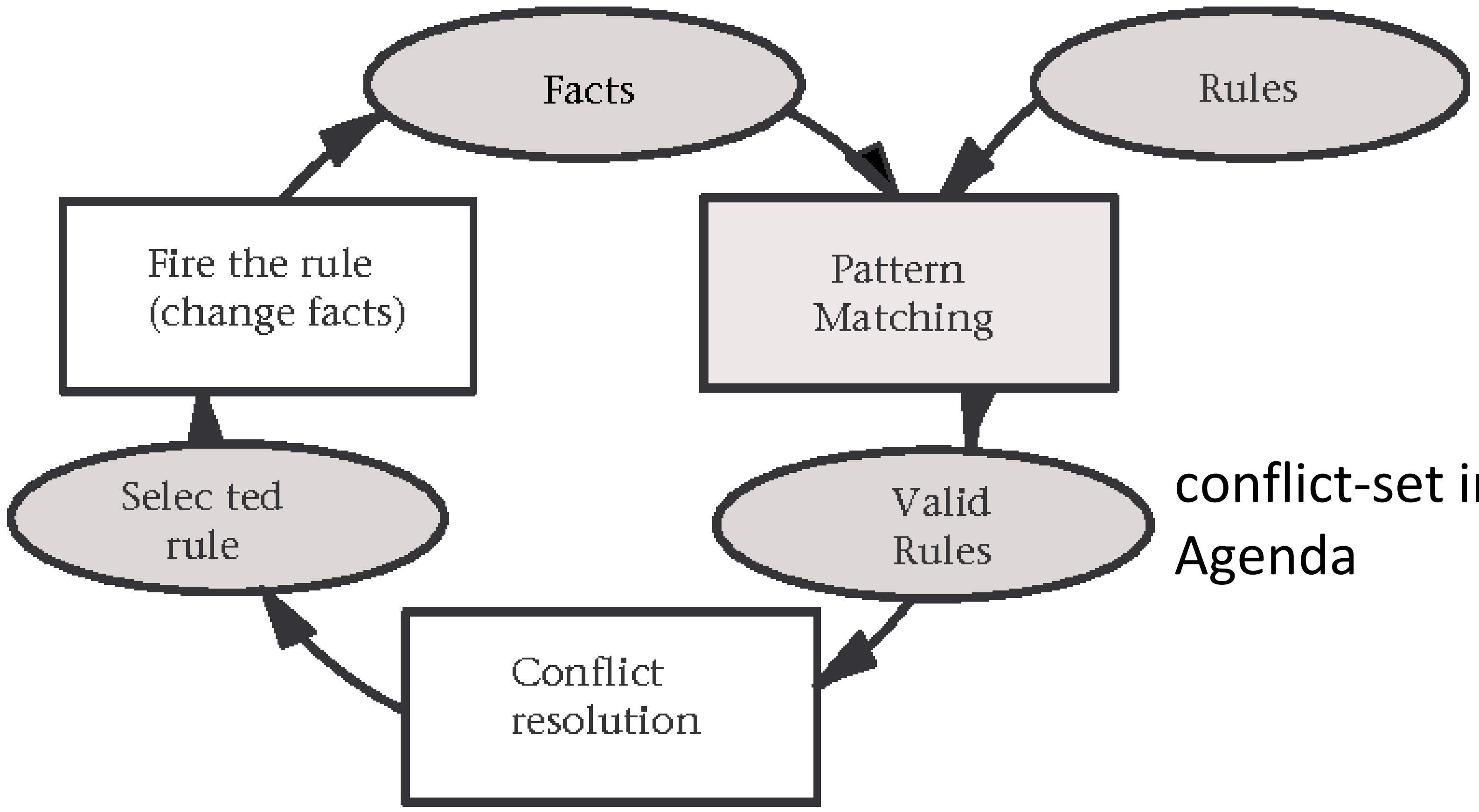
02 Forward Chaining

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

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

- 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”
5	{R1, R3, R6, R4}	-	stop

Summary

- Forward Chaining
- Conflict resolution strategy
- Global control: refractoriness, rule order, recency, specificity
- Local control: priority, meta rules

Backward Chaining

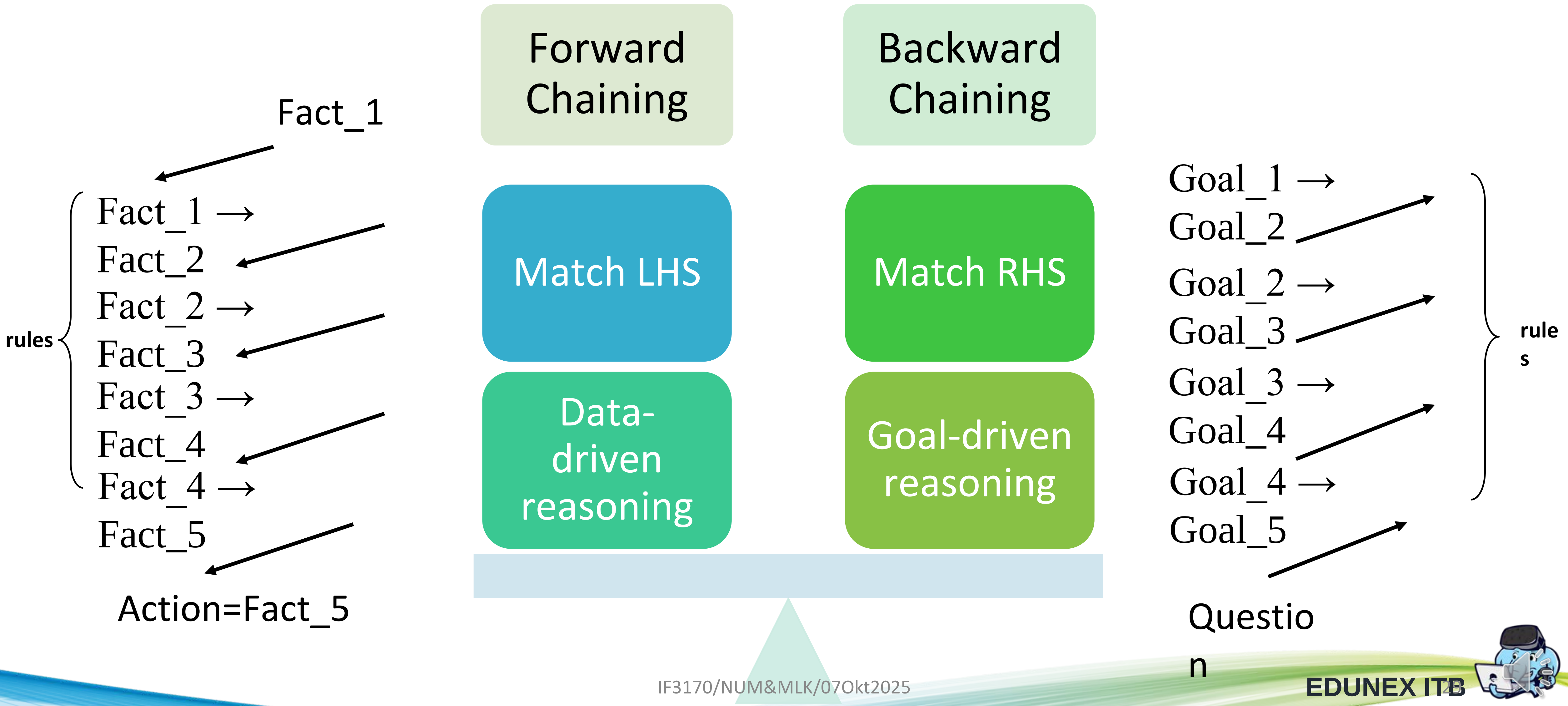
Modul 6: Rule-based System

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

03 Backward Chaining

Inteligensi Buatan
(*Artificial Intelligence*)


Forward Chaining
Backward Chaining
Match LHS
Match RHS
Data-driven reasoning
Goal-driven reasoning
Fact_1
rules
Fact_1 →
Fact_2
Fact_2 →
Fact_3
Fact_3 →
Fact_4
Fact_4 →
Fact_5
Action=Fact_5
Goal_1 →
Goal_2
Goal_2 →
Goal_3
Goal_3 →
Goal_4
Goal_4 →
Goal_5
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

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

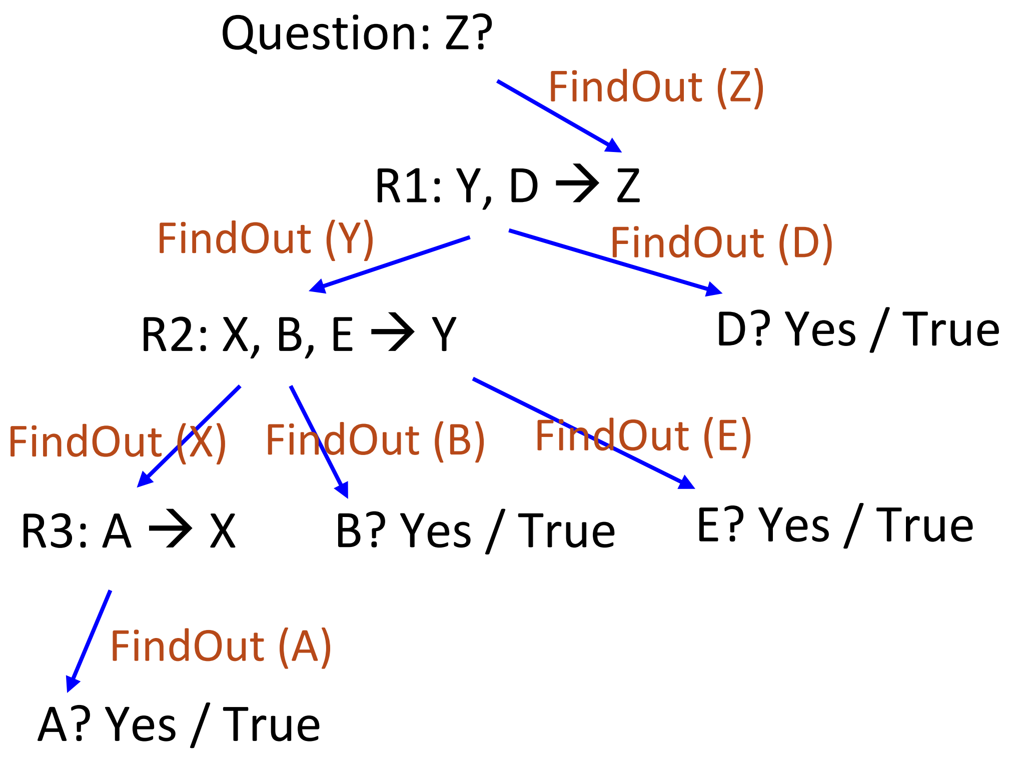
Facts:

A,B,C,D,E

$$\begin{tikzcd}
 & & \text{Question: Z?} \arrow[dr] & & \\
 & & & \text{R1: Y, D} \rightarrow \text{Z} \arrow[dl] \arrow[dr] & \\
 & & \text{R2: X, B, E} \rightarrow \text{Y} \arrow[dl] \arrow[d] \arrow[dr] & & \text{D? Yes / True} \\
 & \text{R3: A} \rightarrow \text{X} \arrow[dl] & \text{B? Yes / True} & \text{E? Yes / True} & \\
\text{A? Yes / True} & & & &
\end{tikzcd}$$

Answer: Yes

Interpreter: FindOut (Goal)

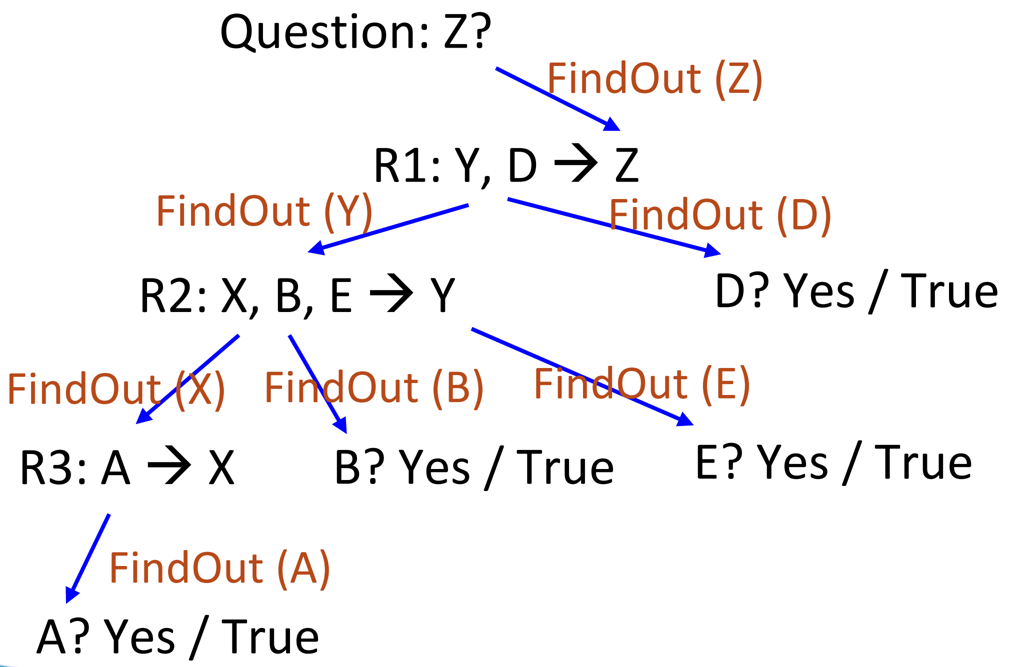


```
Procedure FINDOUT (GOAL)
If (GOAL can be inferred)
then
    set RULE-LIST = list all rules whose action part fulfills GOAL
    until (RULE-LIST = empty) or (GOAL inferred) do
        MONITOR(first or next rule from RULE-LIST)
        delete this rule from RULE-LIST
else (request GOAL)
```

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

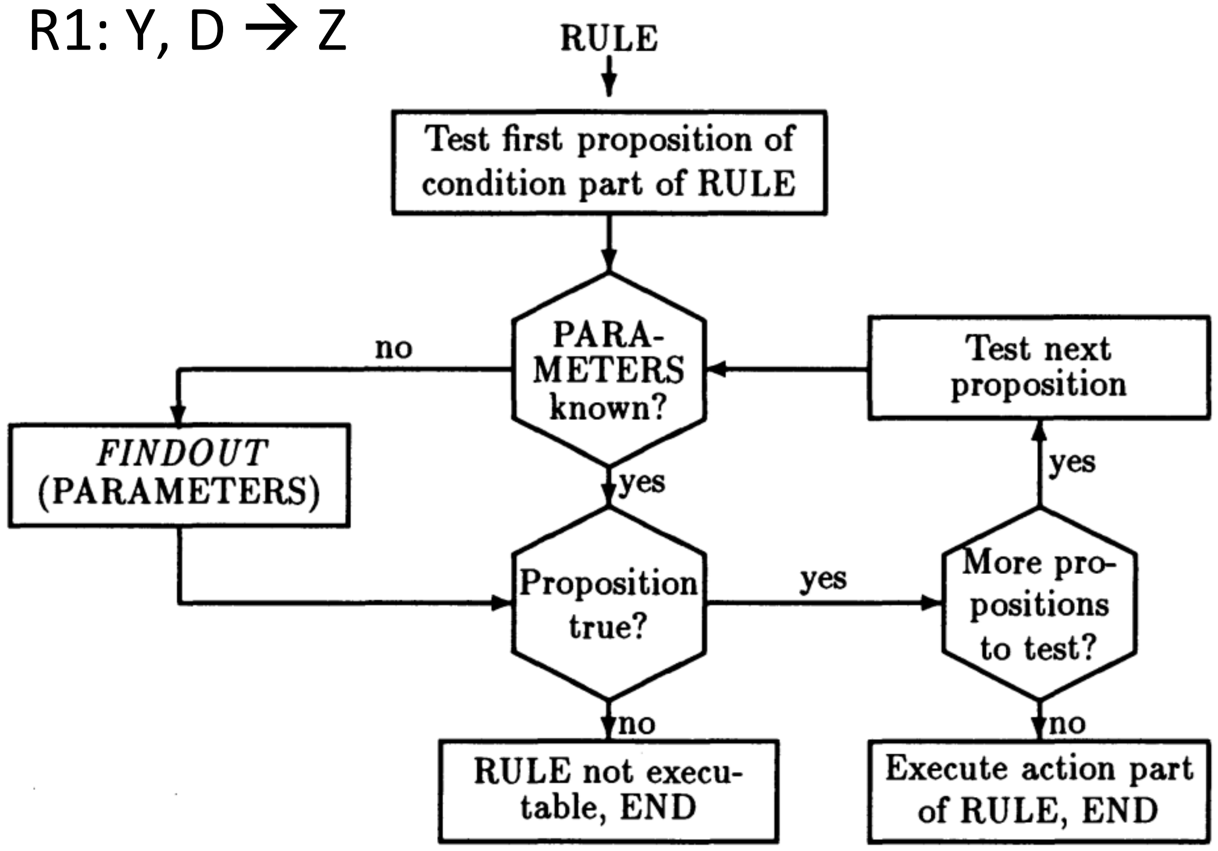


FindOut (Z)
    {R1}
    Monitor(R1)

```
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)
```

Interpreter: Monitor (Rule)

R1: Y, D → Z



RULE
Test first proposition of condition part of RULE
PARA-METERS known?
no
FINDOUT (PARAMETERS)
yes
Proposition true?
yes
More pro-positions to test?
yes
Test next proposition
no
RULE not execu-table, END
no
Execute action part of RULE, END

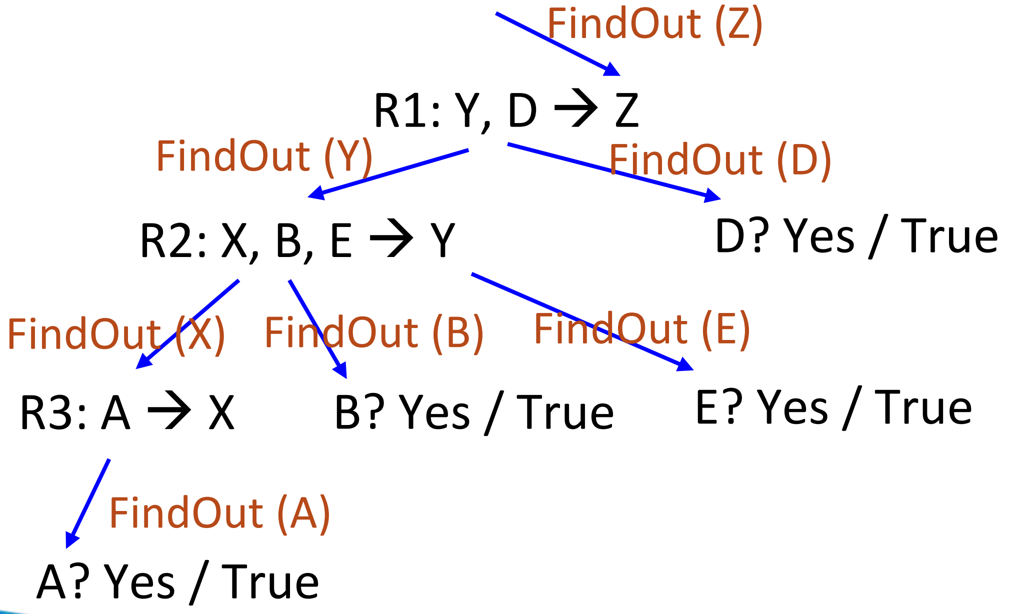
```
Procedure MONITOR (RULE)
Test first proposition of condition part of RULE
repeat
    If parameters known then
        if proposition true then
            proposition ⃞  next proposition
        else RULE not executable
    else FINDOUT(PARAMETERS)
Until (no more propositions to test) or (RULE not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

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

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

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
```

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)
     findout(location)            request(location): downtown
     execute(R4)                  + action=take a cab
   delete(R4)
stop
```

Answer: action=take a cab

Latihan

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

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

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

Backward Chaining

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

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

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	
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	

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