Rule

Masayu Leylia Khodra

Informatics Engineering Study Program
School of Electrical Engineering and Informatics

Institute of Technology Bandung

Review

- Knowledge-based system = knowledge + inference
  - Act humanly
  - Expert quality
- Knowledge representation
  - Rule: the simplest, most common for KBS
  - Rule based system: the simplest and most widespread solution in the real world

Sintaks Rule

- Ekspresi logika implikasi
  if P1,P2,..,Pm then Q1, Q2,…, Qn
  if <prekondisi> then <aksi>

  Bentuk ekivalen dari implikasi pada logika.
  <prekondisi> → <aksi>

- P1,P2,..,Pm: premis, prekondisi, LHS
- Q1, Q2,…, Qn: aksi, konklusi, RHS

Premis / Prekondisi / LHS

- Pola fakta, representasi:
  - Logika (proposisi/predikat)
  - attribute-value
    - Contoh: (location RoomA)
  - object-attribute-value
    - Entitas fisik seperti box, robot, bunga
      Contoh: (bunga (warna merah) (jenis mawar))
    - Entitas konsep seperti pinjaman bank, in, waktu
      Contoh:
      (in (object robot) (location RoomA))
      (in (object ?X) (location ?Y))
      (pinjaman (waktu 2) (bunga 5))

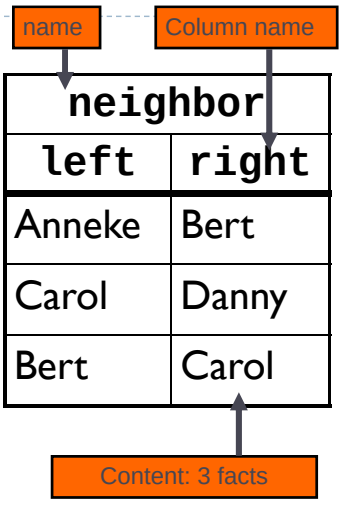
Object-Attribute-Value (2)

- Define structure:

```
(deftemplate neighbor
  (field left  (type SYMBOL))
  (field right (type SYMBOL))
)
```

- Provide facts:

```
(deffacts inputs
  (neighbor (left Anneke)
            (right Bert))
  (neighbor (left Bert)
            (right Carol))
  (neighbor (left Carol)
            (right Danny))
)
```



Aksi Rule

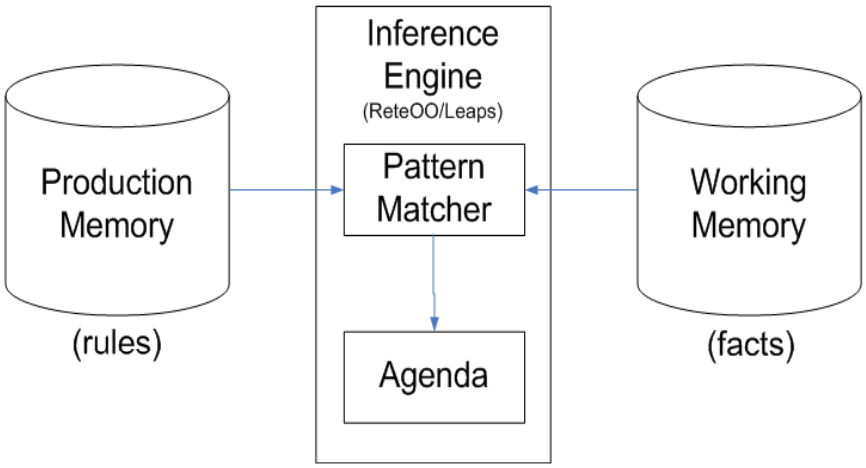
- Implikasi: penurunan fakta baru

if leher kaku and
suhu badan tinggi and
kehilangan kesadaran
then penyakit meningitis

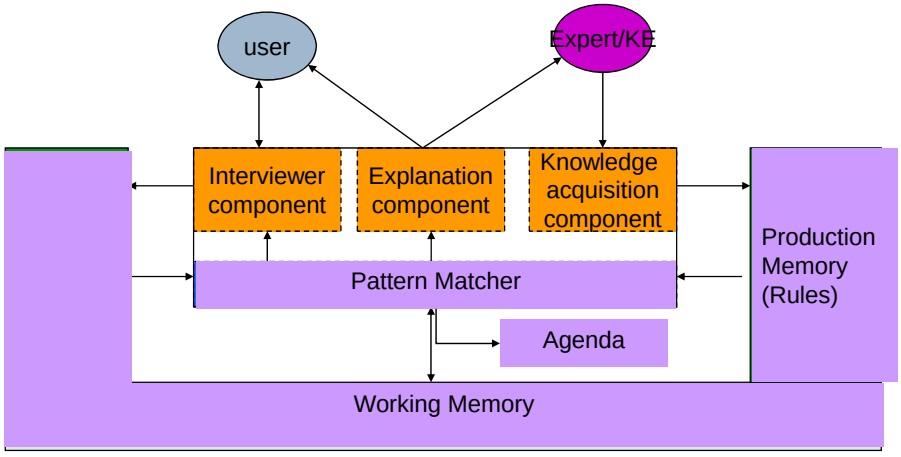
- Instruksi: mengubah state sistem

if clear(box1) and
put(box2)
then on(box2,box1)

Arsitektur Rule-based system

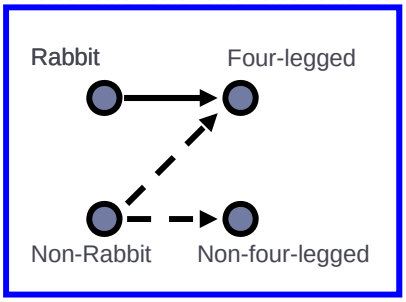


Arsitektur Umum KBS: Rule-BS



Rule Matching

- Rule: Z-knowledge
- Example of a declarative rule:
  IF X is a rabbit
  THEN X is four-legged



- Procedural interpretation:
  IF the memory state has X stored as a rabbit
  THEN update the state to make X four-legged

- We don't know the situation when X is not a rabbit

Contoh Matching

IF: *x* is a horse,
x is the parent of *y*,
y is fast
THEN: *x* is valuable

"*x* is-a horse"

x = Comet
x = Prancer
x = Thunder
x = Dasher

"*x* is a parent of *y*"

x = Comet , *y* = Dasher
x = Comet , *y* = Prancer
x = Dasher , *y* = Thunder

"*y* is fast"

y = Prancer
y = Thunder

Comet is valuable
Dasher is valuable

Facts

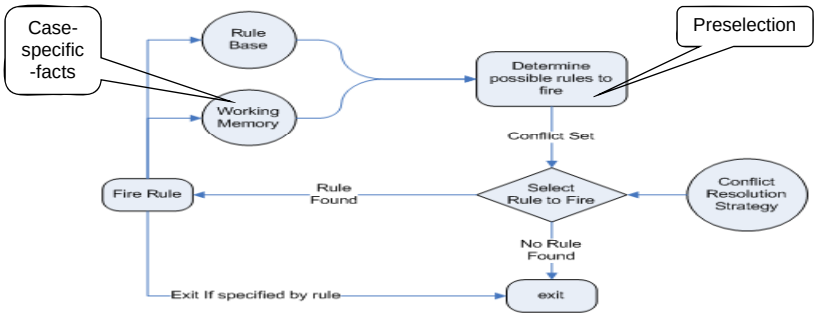
Comet	is-a	horse
Prancer	is-a	horse
Comet	is-parent-of	Dasher
Comet	is-parent-of	Prancer
Prancer	is	fast
Dasher	is-parent-of	Thunder
Thunder	is	fast
Thunder	is-a	horse
Dasher	is-a	horse

Pemrosesan Rules

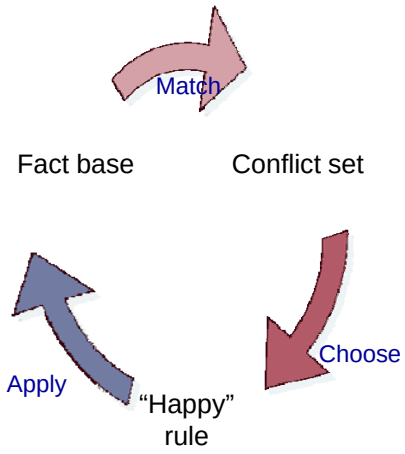
- Forward chaining: fakta → konklusi
  - Data driven
  - Rangkaian Modus Ponen
- Backward chaining: given goal ← fakta
  - Goal driven: diberikan goal → trigger rules yg aksinya mengandung goal
  - Rangkaian abduksi
  - Cocok untuk menelusuri fakta yang masih belum lengkap

Forward Chaining

Data ← case-specific-facts
Repeat
  CS ← Preselection {membentuk conflict set}
  R ← Selection {memilih satu rule dgn conflict-resolving strategy}
  Data ← aplikasi aksi R terhadap Data
Until kondisi terminasi terpenuhi



The Recognize-Act Cycle



1. **Match**:
   Conflict-set ← Preselection
2. **Choose:**
   "Happy" rule ← Selection by Conflict Resolution strategy.
3. **Apply**

Contoh Kasus

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)

- Fakta

(month february)
(happy alison)
(researching alison)

R2 atau R3 ?

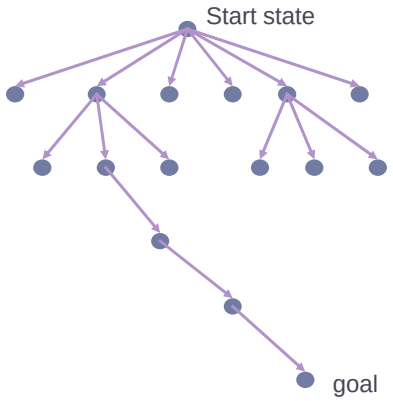
Tergantung
conflict resolution strategy

Conflict Resolving/Resolution Strategy

- Controlling *Rule-based System*
- Global control
  - Domain independent
  - Hard coded pada interpreter/mesin inferensi
  - Selection:
    - by order: rule order vs fact order, depth (recency) vs breadth (old)
    - Refractoriness: once only
    - by syntactic structure of the rule: specificity/complexity
    - LEX (fact recency + complexity), MEA (old fact in first condition)
- Local control
  - Domain dependent
  - Selection by means of supplementary knowledge: priority, meta rules, goal variable

Selection by Order

- Rule order strategy
- Fact order strategy
  - Depth strategy (recency): Prefer instantiations by new facts
  - Breadth strategy: Prefer instantiations by old fact



Selection by Order

- Knowledge-base:
  if (priority second) then out("print second")
  if (priority first) then out("print first")
  if (priority third) then out("print third")
- Facts:
  (priority first)
  (priority second)
  (priority third)

By rule order FIFO	By fact order (recency) LIFO	By fact order (breadth) FIFO
print second print first print third	print third print second print first	print first print second print third

Refractoriness

- Suatu rule tidak diperbolehkan diaktivasi lebih dari satu kali dengan data yang sama

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)

- Fakta

(month february)
(happy alison)
(researching alison)

- Iterasi CS R
  1 {R2, R3} R2
  2 {R2, R3, R6} R6

Refractoriness Example

- KB:
  R1: IF (priority X)
  THEN ADD (print X)
- Facts:
  (priority first)
  (priority second)
  (priority third)
- Conflict resolution strategy: refractoriness, recency
- Inference results:
  print third
  print second
  print first

Selection by Syntactic Structure of the Rule

- Specificity / complexity
- 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

Specificity / Complexity Example

```
(defrule swap
  (goal sort)
  (neighbor ?X ?Y)
  ?Xs <- (holds ?X ?a)
  ?Ys <- (holds ?Y ?b>?a)
  =>
  (modify ?Xs (number ?b))
  (modify ?Ys (number ?a))
  )

(defrule endsort
  ?c <- (goal sort)
  =>
  (modify ?c indexing)
)
```

- Conditions of `endsort` less complex than `swap`
- Goal is changed only when no `swap` applies

Selection by Supplementary Knowledge: Priority

- 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)
  Apakah ada yang salah pada kasus ini ?

Selection by Supplementary Knowledge: Goal Variable

- Goal variable: fine control over subtasks

```
(deftemplate goal
    (field phase (type SYMBOL)) )

(defrule endindexing
      ?c <- (goal (phase indexing))
      (index (place 2))
      =>
      (modify ?c (phase output)) )
```


Meta Rules

- Rules tidak bisa dipanggil seperti prosedur
  - Aktivasi rule hanya bisa dilakukan jika premis dipenuhi
- Knowledge and meta-knowledge tidak berbeda pd RBS
- Peran: mengarahkan reasoning
  - Pruning rule pada conflict set

    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
  - Reorder relevant domain rules: try this before trying that

    If the infection is pelvic-abscess,
    there are rules which mention in their premise enterobacteriaceae,
    there are rules which mention in their premise gram-positive rods,
    then the former should not be done before the latter

Backward Chaining

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)

Backward Chaining (2)

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

Contoh Backward Chaining

```
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
```

- R5: if Z and L then S
- R1: if A and N then E
- R3: if D or M then Z
- R2: if A then M
- R4: if Q and (not W) and (not Z) then N
- R6: if L and M then E
- R7: if B and C then Q
- Fakta: A,L
- Goal: E ?

Inferensi dgn Backward Chaining

- R5: if Z and L then S
- R1: if A and N then E
- R3: if D or M then Z
- R2: if A then M
- R4: if Q and (not W) and (not Z) then N
- R6: if L and M then E
- R7: if B and C then Q
- Fakta: A,L
- Goal: E ?

FindOut(E)	{R1,R6}
Monitor(R1)	A, FindOut(N)
FindOut(N)	{R4}
Monitor(R4)	FindOut(Q), FindOut(~W), FindOut(~Z)
FindOut(Q)	{R7}
Monitor (R7)	FindOut(B), FindOut(C)
FindOut(B)	request B
FindOut(C)	request C
FindOut(~W)	request ~W
FindOut(~Z)	request ~Z
Monitor(R6)	L, FindOut(M)
FindOut(M)	{R2}
Monitor(R2)	A
Yes	

PR: What action to take to get to a theatre

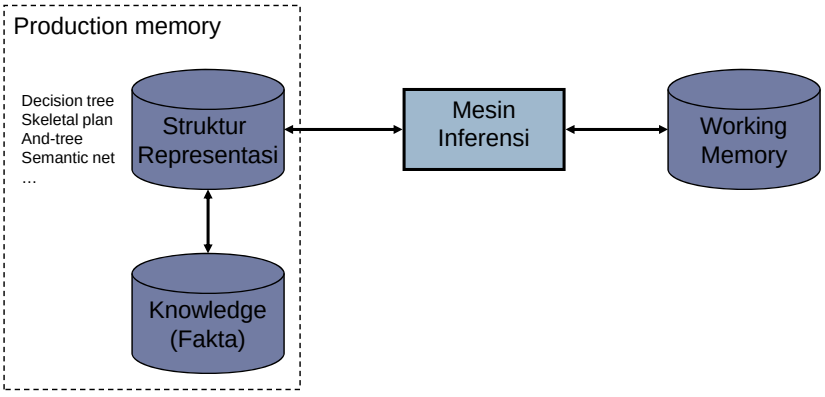
R	IF	THEN
1	Distance > 5 miles	Means is "drive"
2	Distance > 1 mile, time < 15 minutes	Means is "drive"
3	Distance > 1 mile, time > 15 minutes	Means is "walk"
4	Means is "drive", location is "downtown"	Action is "take a cab"
5	Means is "drive", location is not "downtown"	Action is "drive your car"
6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

Lakukan inferensi :

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

1. Forward chaining dgn conflict resolution: rule order
2. Forward chaining dgn conflict resolution: refractoriness, specificity
3. Backward chaining

Implementasi Rule-BS untuk Representasi Lain



Struktur Decision Tree pada Rule-BS

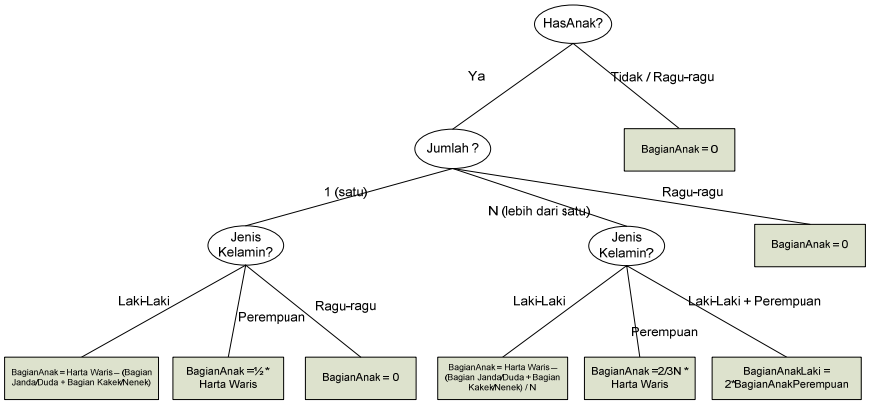
```
(deftemplate simpul
    (slot nama)
    (slot tipe)
    (slot pertanyaan)
    (slot jawaban-1)
    (slot jawaban-2)
    (slot jawaban-3)
    (multislot solusi))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Definisi Rule Mesin Inferensi;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defrule inisialisasi
    (declare (salience 90))
    (not (simpul (nama akar)))
    =>
    (load-facts "ketentuan-anak.dat")
    (assert (simpul-sekarang akar)))
```

```
(defrule simpul-tanya-pilihan
    ? simpul <- (simpul-sekarang ?nama)
    (simpul(nama ?nama)
            (tipe pilihan)
            (pertanyaan ?pertanyaan))
    (not (jwb ?))
    =>
    (printout t ?pertanyaan " (jawaban 1/2/3) ")
    (assert (jwb (read))))

(defrule proses-jawaban1
    ? simpul <- (simpul-sekarang ?nama)
    (simpul(nama ?nama) (tipe pilihan)
    (jawaban-1 ?cabang-jawaban1))
    ?jwb <- (jwb jawaban1)
    =>
    (retract ? simpul ?jwb)
    (assert (simpul-sekarang ?cabang-jawaban1))
    (format t "%s," ?cabang-jawaban1))
```


Contoh Decision Tree



Implementasi Decision Tree sbg Fakta

```
(simpul(nama akar)
        (tipe pilihan)
        (pertanyaan PertanyaanAnak1)
        (jawaban-1 jumlah-anak)
        (jawaban-2 Goal-Ketentuan-Anak1)
        (jawaban-3 NONE))

(simpul(nama jumlah-anak)
        (tipe pilihan)
        (pertanyaan PertanyaanAnak2)
        (jawaban-1 jenis-kelamin-anak1)
        (jawaban-2 jenis-kelamin-anak2)
        (jawaban-3 NONE))

(simpul(nama jenis-kelamin-anak1)
        (tipe pilihan)
        (pertanyaan PertanyaanAnak3)
        (jawaban-1 Goal-Ketentuan-Anak2)
        (jawaban-2 Goal-Ketentuan-Anak3)
        (jawaban-3 nil))

(simpul(nama jenis-kelamin-anak2)
        (tipe pilihan)
        (pertanyaan PertanyaanAnak4)
        (jawaban-1 Goal-Ketentuan-Anak4)
        (jawaban-2 Goal-Ketentuan-Anak5)
        (jawaban-3 Goal-Ketentuan-Anak6))

(simpul(nama Goal-Ketentuan-Anak1)
    (tipe solusi) (solusi TidakMendapatHP))
(simpul(nama Goal-Ketentuan-Anak2)
    (tipe solusi) (solusi SolusiAnak1 ))
(simpul(nama Goal-Ketentuan-Anak3)
    (tipe solusi) (solusi SolusiAnak2 ))
(simpul(nama Goal-Ketentuan-Anak4)
    (tipe solusi) (solusi SolusiAnak3 ))
(simpul(nama Goal-Ketentuan-Anak5)
    (tipe solusi) (solusi SolusiAnak4 ))
(simpul(nama Goal-Ketentuan-Anak6)
    (tipe solusi) (solusi SolusiAnak5 ))
```


Fitur Rule

- 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