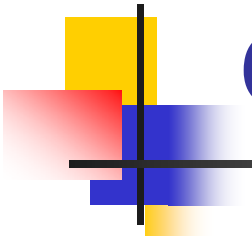
Inteligenta Artificiala

Universitatea Politehnica Bucuresti
Anul universitar 2010-2011

Adina Magda Florea

http://turing.cs.pub.ro/ia_10 si
curs.cs.pub.ro



Curs nr. 6

Sisteme bazate pe reguli

- Reprezentarea prin reguli
- Structura SBR
- Ciclul de inferenta al unui sistem bazat pe reguli
- Strategia de control
- SBR cu inlantuire inainte
- Strategia familiei OPS5
- SBR cu inlantuire inapoi
- Sisteme bazate pe agenda

1. Reprezentarea prin reguli

- Reprezentare modulara a cunostintelor procedurale
- Cunostinte procedurale in maniera declarativa
- Permit atasare de actiuni (functii, proceduri)
- Se pot combina cu reprezentari structurate ale cunostintelor, de ex. ontologii
- Modelelul formal pe care se bazeaza
- Limbaje

Exemple de reguli

Temperatura(p,v)

R1: **daca** pacientul are temperatura mare

si tipul organismului este gram-pozitiv — Tip(o,f)

si pacientul are gitul uscat — GitUscat(p)

atunci organismul este streptococ — Identitate(o,i)

R2: **daca** masina nu porneste

si farurile nu se aprind

atunci bateria este consumata

sau bornele bateriei nu fac contact

R3: **daca** temperatura > 95^{o} C

atunci deschide valva de protectie

Reprezentarea prin reguli - cont

R1

$$(\forall p)(\forall o)(\text{Temperatura}(p, \text{mare}) \wedge \text{Tip}(o, \text{gram-pozitiv}) \wedge \text{GitUscat}(p)) \rightarrow \text{Identitate}(o, \text{streptococ})$$

R2

$$(\forall m)(\exists b)(\text{NuPorneste}(m) \wedge \text{NuAprinde}(m, \text{far})) \rightarrow (\text{Stare}(b, \text{consumata}) \vee \text{Borne}(b, \text{fara\_contact}))$$

R3: **daca** temperatura > 95^{o} C

atunci deschide valva de protectie

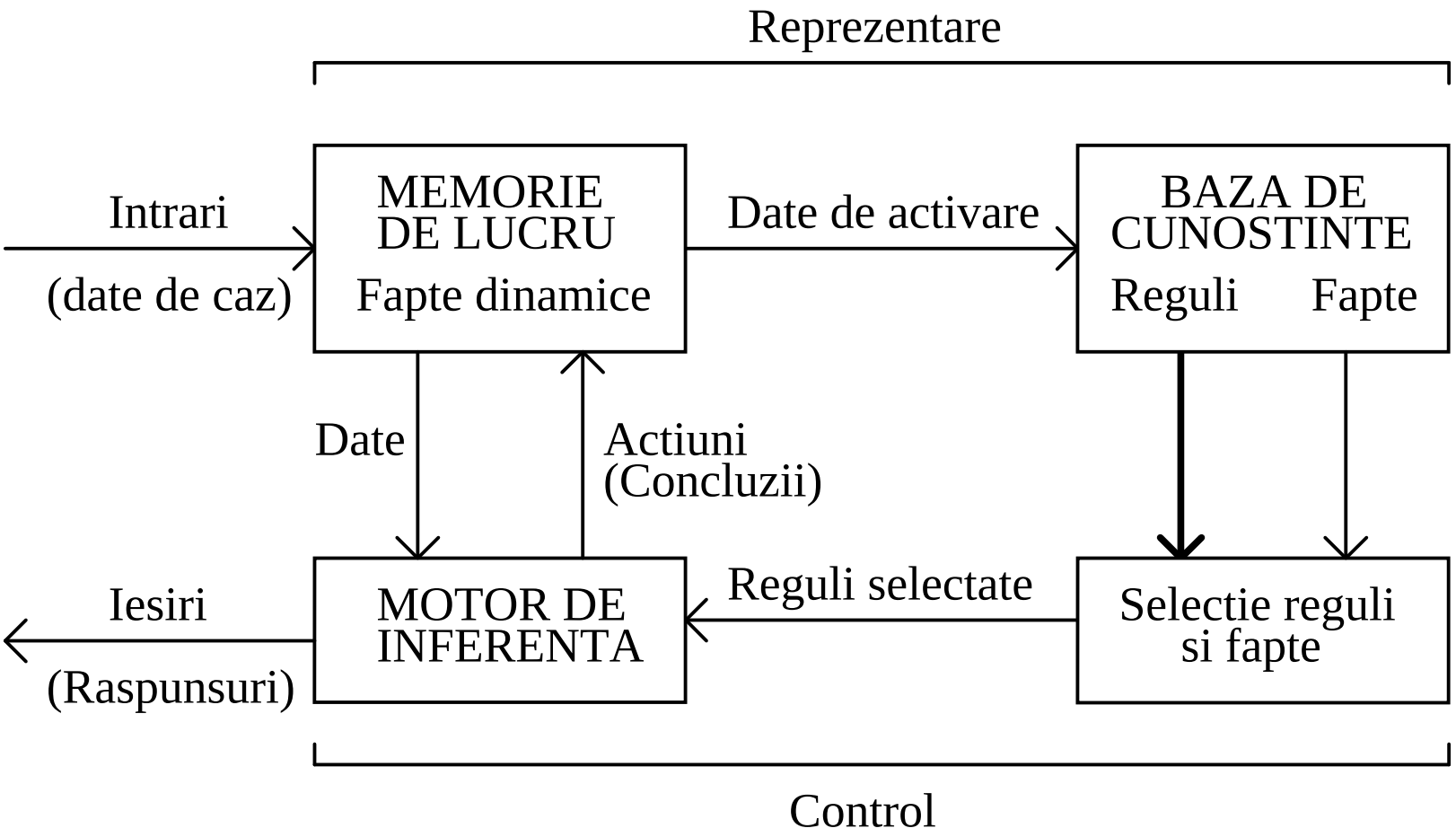
- R3? Reprezentare actiuni in LP ?

Forme de reguli

- **Reguli de inferenta**
  - **if** conditie(i) **then** concluzie/actiune
- Utilizare
  - obtinerea de noi cunostinte
  - executarea actiunilor in functie de conditii

- **Reguli reactive**
  - **on** eveniment **if** conditie(i) **then** (executa) actiune
- Utilizare
  - triggere in SMBD
  - detectarea exceptiilor
  - reguli de integritate
- Se pot combina

2. Structura SBR



3. Ciclul de inferenta al unui sistem bazat pe reguli

- Identificare
- Selectie
- Executie

Ciclul de inferenta al unui sistem bazat pe reguli - cont

Algoritm: Functionarea unui sistem bazat pe reguli

1. ML ← Date de caz
2. **repeta**

   2.1. Executa identificare intre ML si BC (partea stinga sau partea dreapta a regulilor si fapte) si creeaza multimea de conflicte a regulilor aplicabile (MC)

   2.2. Selecteaza o regula dupa un criteriu de selectie

   2.3. Aplica regula prin executia partii drepte a regulii

   pana nu mai sunt reguli aplicabile **sau**

   memoria de lucru satisface conditia de stare scop **sau**

   o cantitate predefinita de efort a fost epuizata

sfarsit.

4. Strategia de control

- Criteriile de selectie din MC
- Directia de aplicare a regulilor

4.1 Criteriile de selectie din MC

- Selectia primei reguli aplicabile
- Alegerea unei reguli din multimea de conflicte
  - Preferinte bazate pe natura regulilor

    Specificitate

    Momentului folosirii anterioare
  - Preferinte bazate pe obiectele identificate
  - Preferinte bazate pe natura starilor

Criteriile de selectie din MC - cont

- Utilizarea metaregulilor

daca o regula are conditiile A si B **si**
regula refera {nu refera} X
{ de loc/
numai in partea stinga/
numai in partea dreapta }
atunci regula va fi in special utila
{ probabil utila/
probabil inutila/
sigur inutila }

- Aplicarea tuturor regulilor din multimea de conflicte

4.2 Directia de aplicare a regulilor

INLANTUIRE INAINTE

daca A **atunci** B
daca B **atunci** C
A (data)

C (concluzie)

INLANTUIRE INAPOI

determina C
daca B **atunci** C
daca A **atunci** B

(**daca** A **atunci** C, implicit)
Este A adevarata? (data)

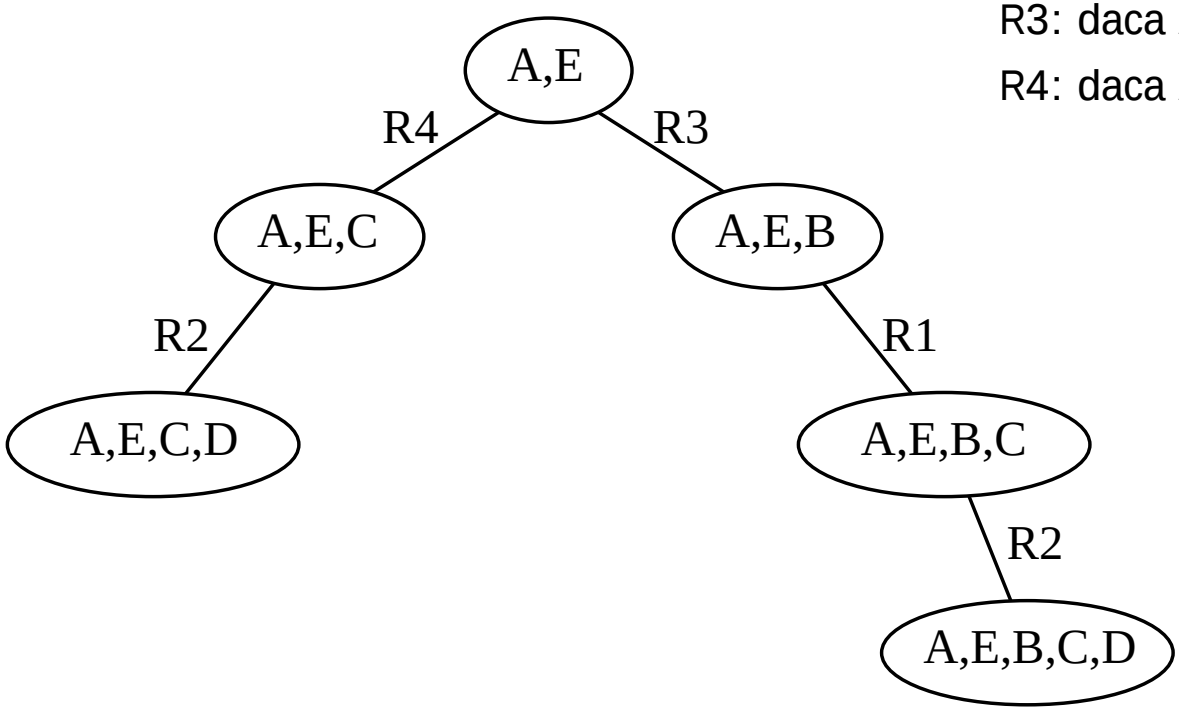
Continut initial al memoriei de lucru: A,E
Stare scop: D

A,E $\xrightarrow{R3}$ A,E,B $\xrightarrow{R1}$ A,E,B,C $\xrightarrow{R2}$ A,E,B,C,D

(a) Inlantuire inainte

(a)

R1: daca A&B atunci C
R2: daca C atunci D
R3: daca A atunci B
R4: daca A&E atunci C

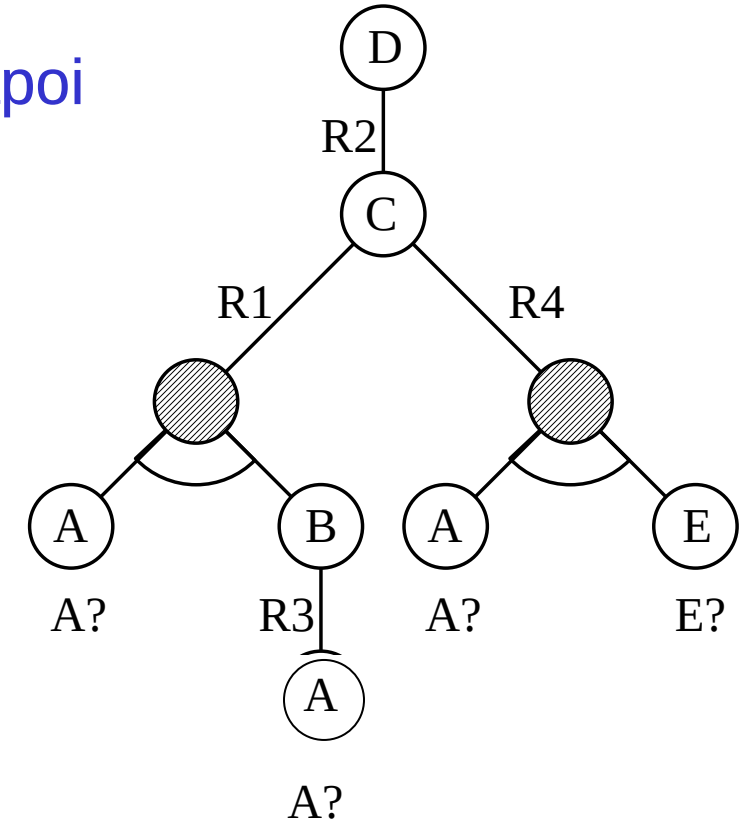


(b)

Continut initial al memoriei de lucru: A,E
Stare scop: D

D? $\xrightarrow{R2}$ C? $\xrightarrow{R1}$ A?, B? $\xrightarrow{R3}$ A?, E?

(a)



R1: daca A&B atunci C

R2: daca C atunci D

R3: daca A atunci B

R4: daca A&E atunci C

(b) Inlantuire inapoi

- Descriere problema

Nod SAU - sau(NumeProb, ListaSuccesori)

Nod SI - si(NumeNod, ListaSubprob)

Nod elementar - elementar(NumeProb)

Nod neelementar - neelementar(NumeProb)

- Solutie

frunza(NumeNod)

nod SAU – unic succesor = nodul SI ales pentru descompunere,

marcat prin structura succ(NumeProb, ArboreSolutie)

descriere(sau(a, [si(b, [elementar(d), sau(e, [elementar(h)])]),
si(c, [sau(f, [elementar(h), neelementar(i)]), elementar(g)])])).

```
% solutie(-ArboreSolutie)
solutie(ArboreSolutie) :- descriere(Problema),
                          rezolv(Problema, ArboreSolutie).

% rezolv(+Problema, -ArboreSolutie)
rezolv(elementar(Problema), frunza(Problema)).
rezolv(sau(Problema, Succesori), succ(Problema, ArboreSolutie)) :-
        member(S, Succesori),
        rezolv(S, ArboreSolutie).
rezolv(si(Problema, Succesori), si(Problema, ArboriSolutie)) :-
        rezolv_toti(Succesori, ArboriSolutie).

rezolv_toti([], []).
rezolv_toti([Problema | Rest], [Arbore | Arbori]) :-
        rezolv(Problema, Arbore), rezolv_toti(Rest, Arbori).
```

? - solutie(ArbSol).

ArbSol = succ(a, si(b, [frunza(d), succ(e, frunza(h))])) ;

ArbSol = succ(a, si(c, [succ(f, frunza(h)), frunza(g)])) ;

No

Algoritm: **Determinarea unei valori cu inlantuire inainte a regulilor (a)**

DetValoareD(A)

1. **daca** A are valoare
   atunci intoarce SUCCES
2. Construieste multimea de conflicte, MC, si initializeaza Efort
3. **daca** MC = {}
   atunci intoarce INSUCCES
4. Alege o regula R$\in$MC dupa un criteriu de selectie
5. Aplica regula R
   Adauga faptul din concluzia regulii R la ML, executa actiune
6. **daca** A are valoare (A $\in$ML)
   atunci intoarce SUCCES
7. MC $\leftarrow$ MC - R
8. MC $\leftarrow$ MC $\cup$ Noile reguli aplicabile
9. Actualizeaza Efort
10. Daca Efort = {} atunci **intoarce** INSUCCES
11. **repeta** de la 3

sfarsit.

Algoritm: Determinarea unei valori cu inlantuirea inapoi a regulilor (b)

DetValoare(A)

1. Construieste multimea regulilor care refera A in concluzie, MC
2. **daca** MC = {}
   atunci
   - 2.1. Intreaba utilizatorul valoarea lui A
   - 2.2. **daca** se cunoaste valoarea lui A
     atunci depune in ML aceasta valoare
     altfel intoarce INSUCCES
3. **altfel**
   - 3.1. Alege o regula dupa un criteriu de selectie
   - 3.2. **pentru** fiecare ipoteza Ij, j=1,N, a premisei lui R **executa**
     - 3.2.1. Fie Aj atributul referit de ipoteza Ij
     - 3.2.2. **daca** Aj are valoare
       atunci atunci evalueaza adevarul ipotezei Ij
     - 3.2.3. **altfel**
       - i. **daca** DetValoare(Aj) = SUCCES
         atunci evalueaza adevarul ipotezei Ij
       - ii. **altfel** considera ipoteza Ij nesatisfacuta

3.3. **daca** toate ipotezele Ij, j=1,N sunt satisfacute

atunci depune in ML valoarea lui A dedusa pe baza concluziei R

4. **daca** A are valoare (in ML)

atunci intoarce SUCCES

5. **altfel**

5.1 MC ← MC – R

5.2 **repeta** de la 2

sfarsit.

1'. **daca** A este data primara

atunci

1'.1. Intreaba utilizatorul valoarea lui A

1'.2. **daca** se cunoaste valoarea lui A

atunci **-** depune in ML valoarea lui A

- intoarce SUCCES

1’.3. **altfel intoarce** INSUCCES

Exemplu SBR cu inlantuire inainte (a)

Nume	Greutate	Dimensiune	Fragil	Impachetat
Statuie	mediu	mare	nu	nu
Tablou	usor	medie	nu	nu
Vaza	usor	mica	da	nu
Soclu	greu	mare	nu	nu

Etapa : Verifica - Decor
Cutie 1 : { }
Obiecte– Neimpachetate: Statuie, Tablou, Vaza, Soclu

R11: **daca** Etapa este Verifica-Decor
si exista o statuie
si nu exista soclu
atunci adauga soclu la Obiecte-Neimpachetate …..

R17: **daca** Etapa este Verifica-Decor
atunci termina Etapa Verifica-Decor
si incepe Etapa Obiecte-Mari

Exemplu SBR cu inlantuire inainte - cont

R21: **daca** Etapa este Obiecte-Mari
si exista un obiect mare de ambalat
si exista un obiect greu de ambalat
si exista o cutie cu mai putin de trei obiecte mari
atunci pune obiectul greu in cutie

R22: **daca** Etapa este Obiecte-Mari
si exista un obiect mare de ambalat
si exista o cutie cu mai putin de trei obiecte mari …..
atunci pune obiectul mare in cutie

R28: **daca** Etapa este Obiecte-Mari
si exista un obiect mare de pus
atunci foloseste o cutie noua

R29: **daca** Etapa este Obiecte-Mari
atunci termina Etapa Obiecte-Mari
si incepe Etapa Obiecte-Medii

Etapa : Obiecte - Medii
Cutie 1 : Soclu, Statuie
Obiecte – Neimpachetate : Tablou, Vaza

Exemplu OPS5

WME

(literalize student nume plasat_in sex_stud)

(literalize camera numar capacitate
libere sex_cam ocupanti)

(vector-attribute ocupanti)

camera 111 4 2 F (Maria Elena) time tag

(make camera ^numar 112 ^capacitate 4 ^libere 4
^sex_cam nil ^ocupanti nil)

(make student ^nume Mihai ^plasat_in nil ^sex-stud M)

```
(p atrib_stud_cam_libera
(<Student_neplasat>
   (student ^nume <nume_stud> ^plasat_in nil ^sex_stud <gen>))
(<Camera_libera>
    (camera ^numar <nr_camera>
            ^capacitate <capacitate_max>
            ^libere   <capacitate_max>))
-->
(modify <Student_neplasat> ^plasat_in <nr_camera>)
(modify <Camera_libera> ^ocupanti <nume_stud>
                        ^sex_camera <gen>
                        ^libere (compute <capacitate_max>-1)).
```

Exemplu Jess

```
(deftemplate    persoana
    (slot prenume)
    (slot nume)
    (slot varsta (type INTEGER))
)
```

```
(defrule  adult
    (persoana    (prenume  ?p_first)
                 (varsta      ?p_age))
    (test  (>=   ?p_age    18)))
=>
    (assert (adult  (prenume  ?p_first)))
)
```

Stud/cam in Jess

```
(deftemplate room
        (slot number)
        (slot capacity (type INTEGER)(default 4))
        (slot sexes_are)
        (slot vacancies (type INTEGER))
        (multislot occupants))
```

```
(deftemplate      student
    (slot name)
    (slot sex)
    (slot placed_in)
    (slot special_considerations (default no))
```

```
(defrule  assign-student-empty-room
    ?unplaced_student ← (student (name  ?stud_name)
                                 (placed_in nil)
                                 (sex  ?gender))
    ?empty_room ← (room (number ?room_no)
                        (capacity ?room_cap)
                        (vacancies  ?max_size))
    (test  (=   ?room_cap    ?max_size))
=>
    (modify ?unplaced_student (placed_in ?room_no))
    (modify ?empty_room (occupants  ?stud_name) (sexes_are ?gender)
                        (vacancies (-- ?max_size))
)
```

5. Strategii SBR familia OPS5

- Ciclul recunoastere-actiune:
  - Match
  - Select
  - Act
- WME = working memory element
  - Identificat printr-un "time tag"
- Instantiere: multime de WME care satisface o regula
- Multime de conflicte (MC)
- Rezolvarea conflictelor

Ciclul recunoastere-actiune

Match

- O regula se poate aplica daca conditiile din antecedent sunt satisfacute de WMEs din ML
- Procesul are 2 aspecte:
  - match intra-element
  - match inter-element

```
(defrule teenager
   (person (firstName ?name) (age ?age))
=> (printout t ?name " is " ?age " years old." crlf))
```

Ciclul recunoastere-actiune

- match inter-element

```
(defrule   assign-private-room
    (student  (name ?stud_name)
              (placed_in nil)
              (special_consideration yes))
    (room     (number ?room_no)
              (capacity 1)
              (vacancies 1)
=> …
```

Instantieri (MC)
assign-private-room 41 9
assign-private-room 41 17
assign-private-room 52 9
assign-private-room 52 17

WMEs
41 (student name Mary sex F placed_in nil special_consideration yes)
52 (student name Peter sex M placed_in nil special_consideration yes)
9 (room number 221 capacity 1 vacancies 1)
12 (room number 346 capacity 2 vacancies 1)
17 (room number 761 capacity 1 vacancies 1)

Rezolvarea conflictelor

- Diferite strategii
  - Refractie = o aceeasi instantiere nu este executata de mai multe ori (2 instantieri sunt la fel daca au acelasi nume de regula si aceleasi time tags, in aceeasi ordine)
  - Momentul utilizarii = Se prefera instantierile care au identificat cu WMEs cu cele mai recente time tags (sau invers)
  - Specificitate = Au prioritate instantierile cu LHS specifice = nr de valori testate in LHS
    - teste asupra: nume clasa, predicat cu 1 arg constanta, predicat cu un operator variabila legata

Rezolvarea conflictelor

- **Strategia LEX**
  - Elimina din MC instantierile care au fost deja executate
  - Ordoneaza instantierile pe baza momentului utilizarii
  - Daca mai multe instantieri au aceleasi time tags, utilizeaza specificitate
  - Daca mai multe instantieri au aceeasi specificitate alege arbitrar
- **Strategia MEA** – aceeasi dar utilizeaza primul time tag

Eficienta executiei

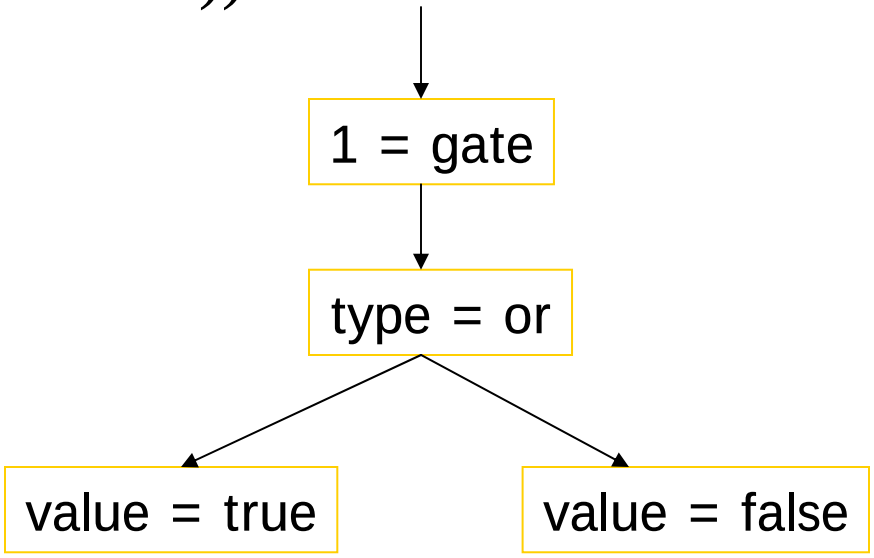
- Match – cam 80% din timpul ciclului recunoastere-actiune
- Fiecare WME este comparat cu fiecare conditie din fiecare regula
  - $O(R*F^P)$, R – nr de reguli, F – nr WME, P nr mediu de conditii in LHS regula
- RETE match algorithm (1982 Forgy)
  - Salveaza starea de match intre 2 cicluri recunoastere-actiune
  - La crearea unui WME, acesta este comparat cu toate elementele conditie din program si este memorat impreuna cu fiecare element conditie cu care a identificat =>
  - Numai schimbarile incrementale din WM sunt identificate in fiecare ciclu
  - O(R*F*P) aprox

Compilarea regulilor

- Se compileaza conditiile regulilor intr-o retea de noduri de test
- Un test este un predicat cu o constanta sau variabila legata sau o disjunctie
- Procesul de match are loc numai la adaugarea sau la eliminarea unui WME (modify este retract urmat de assert)

  (gate (type or) (value true))

  (gate (type or) (value false))
- **Node sharing**



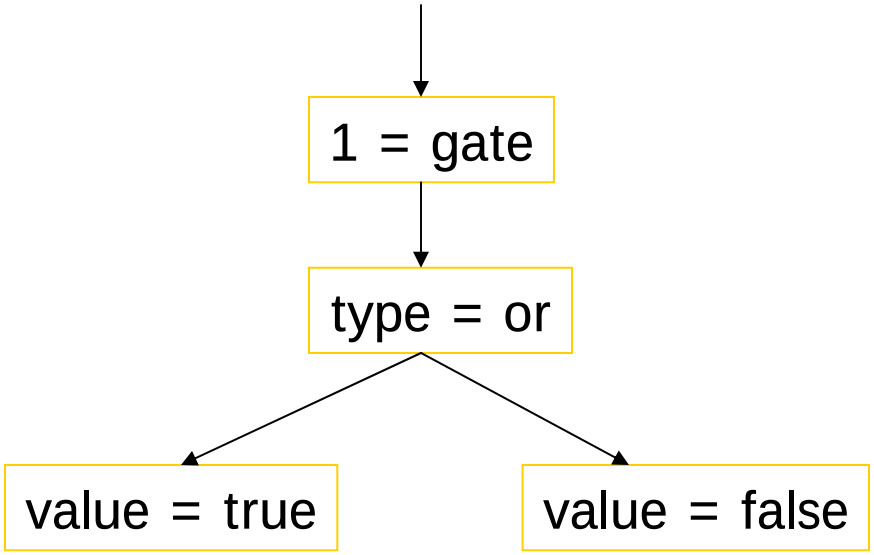
Compilarea regulilor

- Un pointer la noul WME este trecut prin retea, incepand de la nodul radacina
- Fiecare nod actioneaza ca un switch
- Cand un nod primeste un pointer la un WME, testeaza WME asociat. Daca testul reuseste, nodul se deschide si WME trece mai departe
- Altfel nu se intampla nimic
- Daca un WME va trece prin retea, va fi combinat cu alte WME care trec pentru a forma o instantiere in MC
- Pointerii la WME sunt trimisi prin reteaua RETE ca tokens = pointer + stare (assert sau retract)

Tipuri de noduri in retea

Nod cu 1 intrare = test intra-element

- realizat de noduri cu 1 intrare
- fiecare nod efectueaza un test corespunzator unei conditii
- testarea aceleiasi variabile – tot noduri cu 1 intrare



Sisteme bazate pe reguli

- Foarte multe
- Cele mai influente

- OPS5
- ART
- CLIPS
- Jess
- Familia Web Rule languages
  - In special RuleML si SWRL
  - Interoperabilitatea regulilor

RuleML

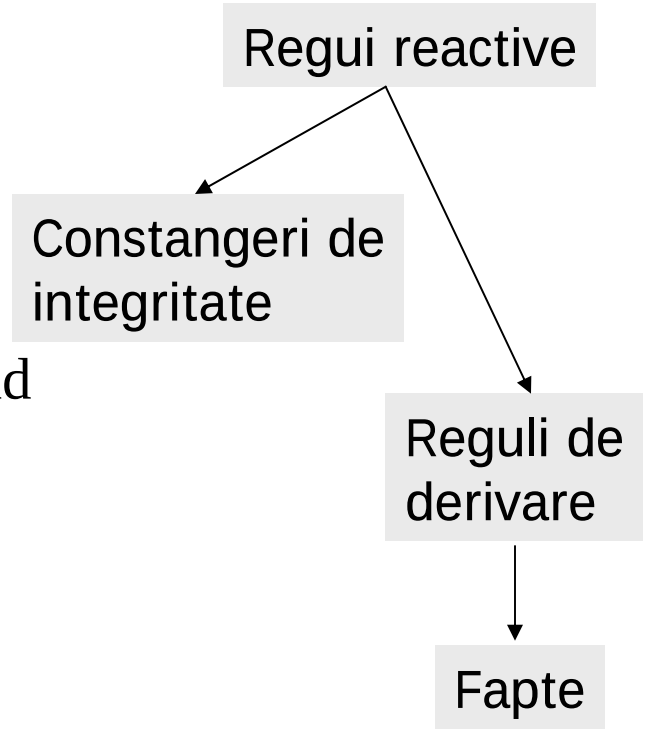
- RuleML Initiative - August 2000 Pacific Rim International Conference on Artificial Intelligence.
- RuleML Initiative dezvolta limbaje bazate pe reguli deschise XML/RDF
- Aceasta permite schimbul de reguli intre diferite sisteme, inclusiv componete software distribuite pe Web si sisteme client-server eterogene
- Limbajul RuleML – sintaxa XML pentru reprezentarea cunostintelor sub forma de reguli
- Integrarea cu ontologii: sistemul de reguli trebuie sa deriveze/utilizeze cunostinte din ontologie

RuleML

- RuleML – se bazeaza pe Datalog
- Datalog = limbaj de interogare si reguli pentru baze de date deductive
- Subset al Prolog cu restrictii:
  - argumente ne-functionale (constante sau variabile)
  - limitari in nivelul de apeluri recursive
  - variabilele din concluzie trebuie sa apara in predicate ne-negate din ipoteza
- Hornlog – Datalog extins cu variabile functionale (termeni generali)

RuleML

- **Reguli reactive** = Observa/verifica anumite evenimente/conditii si executa o actiune – numai forward
- **Constrangeri de integritate** = reguli speciale care semnaleaza inconistente cand se indeplinesc anumite conditii – numai forward
- **Reguli de inferenta** (derivare) = reguli reactive speciale cu actiuni care adauga o concluzie daca conditiile (premisele sunt adevarate) - Se pot aplica atat forward cat si backward
- **Fapte** = reguli de inferenta particulare



RuleML

- **Reguli reactive**

<rule> **<_body>** **<and>** *prem1 ... premN* **</and>** **</_body>**
<_head> *action* **</_head>**
</rule>

- **Constrangeri de integritate**

<ic> **<_head>** *inconsistency* **</_head>**
<_body> **<and>** *prem1 ... premN* **</and>** **</_body>**
</ic>

implementate ca

<rule> **<_body>** **<and>** *prem1* **...** *premN* **</and>** **</_body>**
<_head> **<signal>** *inconsistency* **</signal>** **</_head>**
</rule>

RuleML

- **Reguli de inferenta/derivare**

<imp> <_head> *conc* **</_head>**
<_body> <and> *prem1 ... premN* **</and> </_body>**
</imp >

implementate prin
<rule> <_body> <and> *prem1 ... premN* **</and> </_body>**
<_head> <assert> *conc* **</assert> </_head>**
</rule>

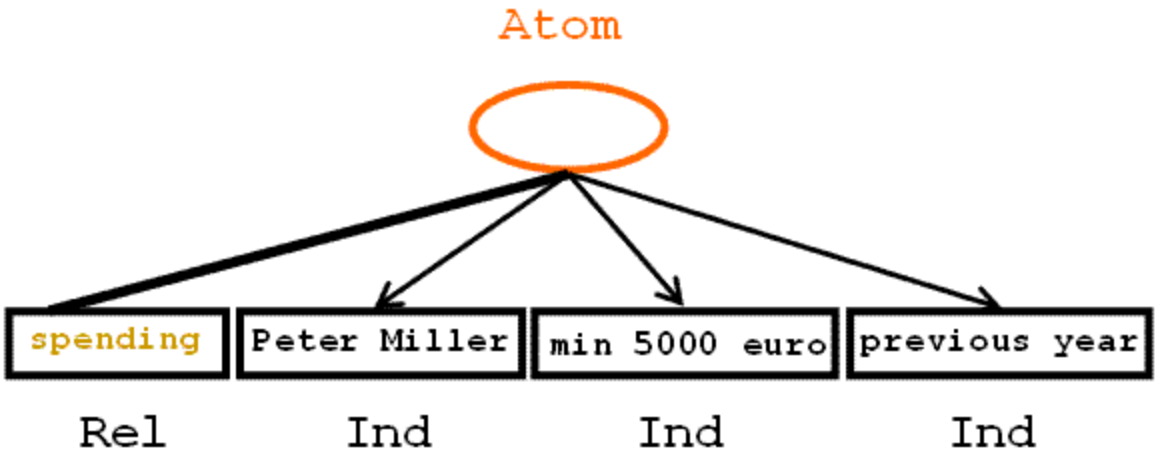
- **Fapte**

<atom> <_head> *conc* **</_head> </atom>**
implementate prin
<imp> <_head> *conc* **</_head>**
<_body> <and> </and> </_body> </imp>

RuleML

Fapte

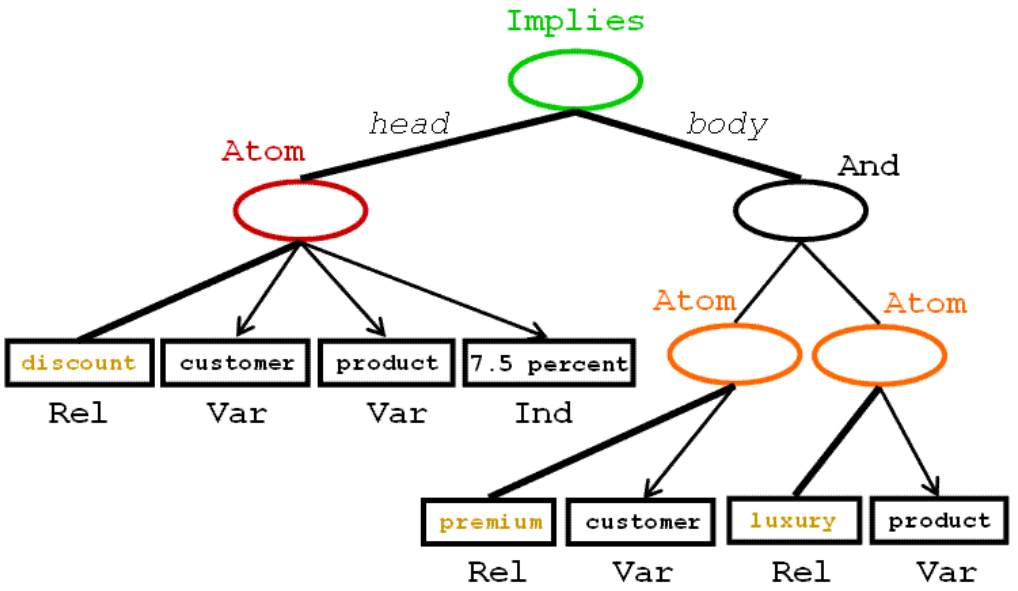
```
<atom> <rel>spending</rel>
        <ind>Peter Miller</ind>
        <ind>min 5000 euro</ind>
        <ind>previous year</ind>
</atom>
```



spending(petterMiller, min5000euro, previousYear).

Reguli de inferenta/derivare

```
<imp>
  <head> <atom>
            <rel>discount</rel>
            <var>customer</var>
            <var>product</var>
            <ind>7.5 percent</ind>
           </atom>
  </head>
  <body>
    <and>
           <atom>
            <rel>premium</rel>
            <var>customer</var>
           </atom>
           <atom>
             <rel>luxury</rel>
             <var>product</var>
           </atom>
     </and>
  </body>
</imp>
```



discount(Customer, Product, 7.5_percent):-
premium(Customer), luxury(Product).

6. SBR cu inlantuire inapoi

R1: **daca** X are par
    atunci X este mamifer
R2: **daca** X hraneste puii cu lapte
    atunci X este mamifer
R3: **daca** X este mamifer
    si X are dinti ascutiti
    si X are falci
    atunci X este carnivor
R4: **daca** X este carnivor
    si X este maroniu
    si X are pete
    atunci X este leopard
R5: **daca** X este carnivor
    si X este maroniu
    si X are dungi
    atunci X este tigru

Ce este X?

Exemplu SBR cu inlantuire inapoi si calcul incert (CF)

- Exemplu: o baza de cunostinte pentru alegerea vinului adecvat unui meniu
- Valoarea fiecarui atribut este memorata impreuna cu coeficientul de certitudine asociat.
- Coeficientii de certitudine sunt valori pozitive in intervalul [0,1].

  (vin chardonney 0.8 riesling 0.6)
- O regula poate avea asociat un coeficient de certitudine

  daca sos-meniu = sos-alb

  atunci culoare-vin = alba 0.6

R11: daca componenta-meniu = curcan
atunci culoare-vin = rosie 0.7
si culoare-vin = alba 0.2

R12: daca componenta-meniu = peste
atunci culoare-vin = alba

R13: daca sos-meniu = sos-alb
atunci culoare-vin = alba 0.6

R14: daca componenta-meniu = porc
atunci culoare-vin = rosie

R21: daca sos-meniu = sos-alb
atunci tip-vin = sec 0.8
si tip-vin = demisec 0.6

R22: daca sos-meniu = sos-tomat
atunci tip-vin = dulce 0.8
si tip-vin = demisec 0.5

R23: daca sos-meniu = necunoscut
atunci tip-vin = demisec

R24: daca componenta-meniu = curcan
atunci tip-vin = dulce 0.6
si tip-vin = demisec 0.4

R31: daca culoare-vin = rosie
si tip-vin = dulce
atunci vin = gamay

R32: daca culoare-vin = rosie
si tip-vin = sec
atunci vin = cabernet-sauvignon

R33: daca culoare-vin = rosie
si tip-vin = demisec
atunci vin = pinot-noir

R34: daca culoare-vin = alba
si tip-vin = dulce
atunci vin = chenin-blanc

R35: daca culoare-vin = alba
si tip-vin = sec
atunci vin = chardonnay

R36: daca culoare-vin = alba
si tip-vin = demisec
atunci vin = riesling

scop(vin)

monovaloare(componenta-meniu)

monovaloare(culoare-vin)

monovaloare(sos-meniu)

multivaloare(tip-vin)

multivaloare(vin)

valori-legale(componenta-meniu) = [curcan, peste, porc]

valori-legale(sos-meniu) = [sos-alb, sos-tomat]

valori-legale(tip-vin) = [sec, demisec, dulce]

valori-legale(vin) = [gamay, cabernet-sauvignon, pinot-noir,
chenin-blanc,chardonnay, riesling]

valori-legale(culoare-vin) = [rosie, alba]

7. Sisteme bazate pe agenda

- Structura de control a anumitor sisteme bazate pe reguli nu foloseste nici inlantuirea inainte nici inlantuirea inapoi a regulilor, ci o ***strategie de control de tip agenda*.**
- Strategie utila in cazul in care se foloseste un *criteriu de selectie* a regulilor bazat pe **preferinta starilor**, deci o functie euristica de evaluare.
- O ***agenda*** este o lista de sarcini pe care sistemul trebuie sa le execute.
- O sarcina are asociata una sau mai multe **justificari -** prioritate

Algoritm: Strategia de control de tip "agenda"

1. Initializeaza agenda cu sarcina de executat
2. **repeta**
   - 2.1. Selecteaza sarcina cu prioritate maxima, T
   - 2.2. Executa sarcina T in limitele unor resurse de timp si spatiu determinate de importanta lui T
   - 2.3. **pentru** fiecare noua sarcina T_i generata de T **executa**
     - 2.3.1. **daca** T_i este deja in agenda **atunci**
       - i. Fie T'_i copia lui T_i din agenda
       - ii. **daca** justificarile lui T'_i nu includ justificarea lui T_i **atunci** adauga justificarea lui T_i justificarilor lui T'_i
       - iii. Inlocuieste T'_i cu T_i

2.3.2. **altfel** adauga T_i si justificarea asociata in agenda

2.3.3. Calculeaza prioritatea sarcinii T_i pe baza justificarilor asociate

pana agenda satisface conditia de stare finala **sau**

agenda este vida

sfarsit.