Paradigme de programare

2010-2011, semestrul 2

Curs 10

Cuprins – Mecanisme de control al fluxului si eficienta in CLIPS

- Controlul fluxului
  - Agenda
  - Strategii de rezolvare a conflictelor
  - Fapte de control
  - Prioritati
  - Reguli reactivate
  - Module
- Eficienta
  - Algoritmul de pattern match
  - Importanta ordinii patternurilor
  - Designul regulilor

Controlul fluxului in CLIPS

- Prin mecanismul de aplicare a regulilor existent in limbaj
  - Agenda
- Prin mecanisme la nivel de utilizator
  - Fapte de control
  - Prioritati (asociate regulilor)
  - Strategii de rezolvare a conflictelor intre reguli cu aceeasi prioritate
  - Reguli reactivate
  - Module

Mecanisme la nivel de utilizator - Prioritati

- Regulilor li se pot asigna prioritati:

```
(defrule myRule
   (declare (salience 10))
   <patternuri>
=>
   <actiuni>)
```

- salience ia valori intre -10000 si 10000
- salience-ul default este 0
- Cu cat salience-ul are o valoare mai mare, cu atat mai mare este prioritatea regulii
- Observatie: regula trebuie sa fie aplicabila, altfel nu se pune problema de salience

Strategii de rezolvare a conflictelor intre reguli cu salience egal

- Depth (default) = au prioritate cele mai nou create inregistrari de activare (LIFO)
- Breadth = … vechi … (FIFO)
- Random = se alege una la intamplare
- Simplicity
- Complexity
- Lex
- Mea

Problema cu strategiile de rezolvare a conflictelor

- Executia depinde de ordinea introducerii datelor

Fapte de control

- Idee: procesul de rezolvare este organizat a.i. daca o anumita conditie este indeplinita sa aplicam un anumit grup de reguli, iar daca nu – un alt grup de reguli
- Schema generala:

```
(defrule am-conditie
   (test conditie)
=>
   (assert (cond true)))
```

Fapte de control - continuare

```
(defrule regula-grup-1
    (cond true)
=>
    …)

(defrule regula-grup-2
    (not (cond true))
=>
    …)
```

Problema cu faptele de control

- Pentru multe conditii => foarte multe fapte de control
- Solutia: modularizarea programului

Reguli reactivate

- (refresh $regula_1$ $regula_2$ … $regula_n$) este o actiune care poate aparea intr-o regula oarecare si are ca efect sustragerea regulilor $regula_1$ … $regula_n$ de sub imperiul principiului refractiei

Module de program

- Baza de fapte si baza de reguli pot fi impartasite selectiv de procese separate de rezolvare a problemei (= module)
- Rezolvarea decurge prin cedarea controlului de la un modul la altul
- Modul activ = modul ale carui reguli sunt in curs de executie
- Modul inactiv = modul ale carui reguli nu se executa chiar daca sunt aplicabile, intrucat modulul nu are controlul
- Fiecare modul are agenda proprie – in permanenta activa si sensibila la modificarile din exterior (chiar cand modulul e inactiv)

Gestiunea modulelor

- focus-stack: modulul din varful stivei este cel activ
- Controlul se cedeaza intr-unul din urmatoarele feluri:
  - O regula cu auto-focus din alt modul devine aplicabila
  - Modulul nu mai are reguli aplicabile, caz in care cedeaza controlul urmatorului modul din stiva (face return)
  - Modulul cedeaza voluntar controlul unui alt modul

Operatii cu module

- (focus M1 M2 … Mn)
  - Pune M1 … Mn in varful stivei de module
  - M1 devine modulul activ dupa ce regula care a initiat comanda focus isi incheie executia
- (return)
  - Modulul curent cedeaza controlul urmatorului modul din stiva
  - Se apeleaza implicit atunci cand un modul nu mai are reguli aplicabile
- (clear-focus-stack)
  - Scoate toate modulele din stiva

Reguli cu auto-focus

- Asteapta terminarea regulii curente
- Abia apoi modulul regulii cu auto-focus preia controlul
- Pot fi folosite la implementarea de constrangeri

Eficienta in CLIPS

- Operatia critica a sistemului: pattern match – se produce pentru fiecare iteratie din algoritmul agendei (pe masura ce faptele sunt introduse/retrase, agenda trebuie mentinuta “la zi”)

- Abordari:
  - Pattern match condus de reguli: ineficient
  - Pattern match condus de fapte: mai eficient, datorita redundantei temporale (baza de fapte se schimba foarte lent)

Pattern match condus de reguli

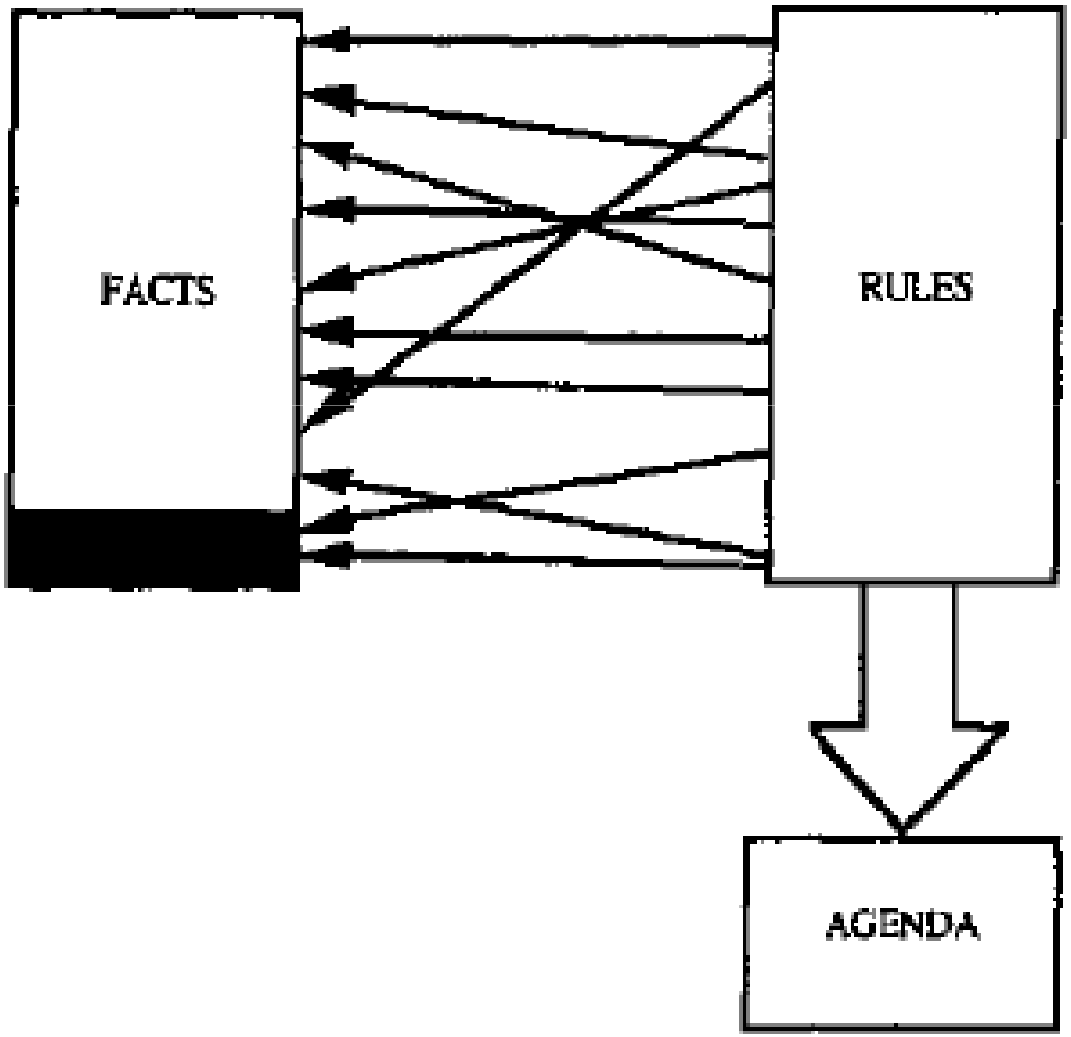
Pentru:

- n patternuri
- N_k potriviri pentru fiecare pattern k
- Variabile multicamp care pot fi potrivite in q feluri diferite in fiecare pattern

=>

Timpul necesar construirii iregistrarilor de activare este de ordinul $q^{n*} (\Pi N_k)$

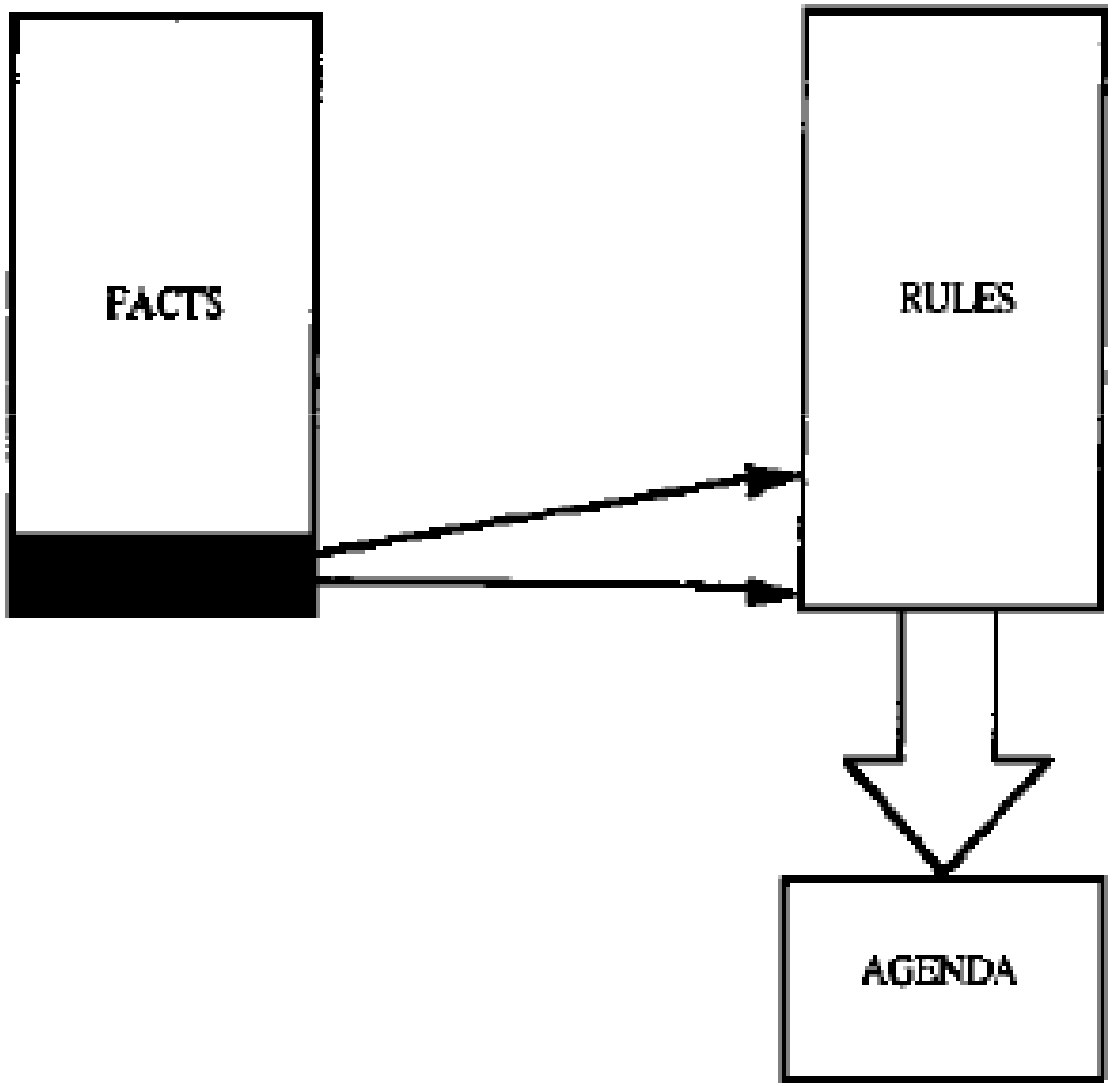
Pattern match condus de reguli (2)



Pattern match condus de fapte

- Tine cont de redundanta temporala: tipic, de la o iteratie la alta nu apar/dispar multe fapte => o mica parte din reguli este afectata
- Regulile sunt statice, faptele sunt dinamice => faptele ar trebui sa caute regulile pe care le afecteaza, nu invers
- Se evita repetarea multor calcule retinand potrivirile din ciclul anterior care nu au fost afectate de ultima aplicare de regula

Pattern match condus de fapte (2)



Algoritmul RETE pentru pattern matching

- 2 pasi:
  - Reteaua de pattern determina ce fapte se potrivesc cu ce patternuri
  - Reteaua de join compara valorile variabilelor din patternuri pentru a se asigura ca acestea sunt consistente (atunci cand aceeasi variabila este folosita in mai multe locuri)

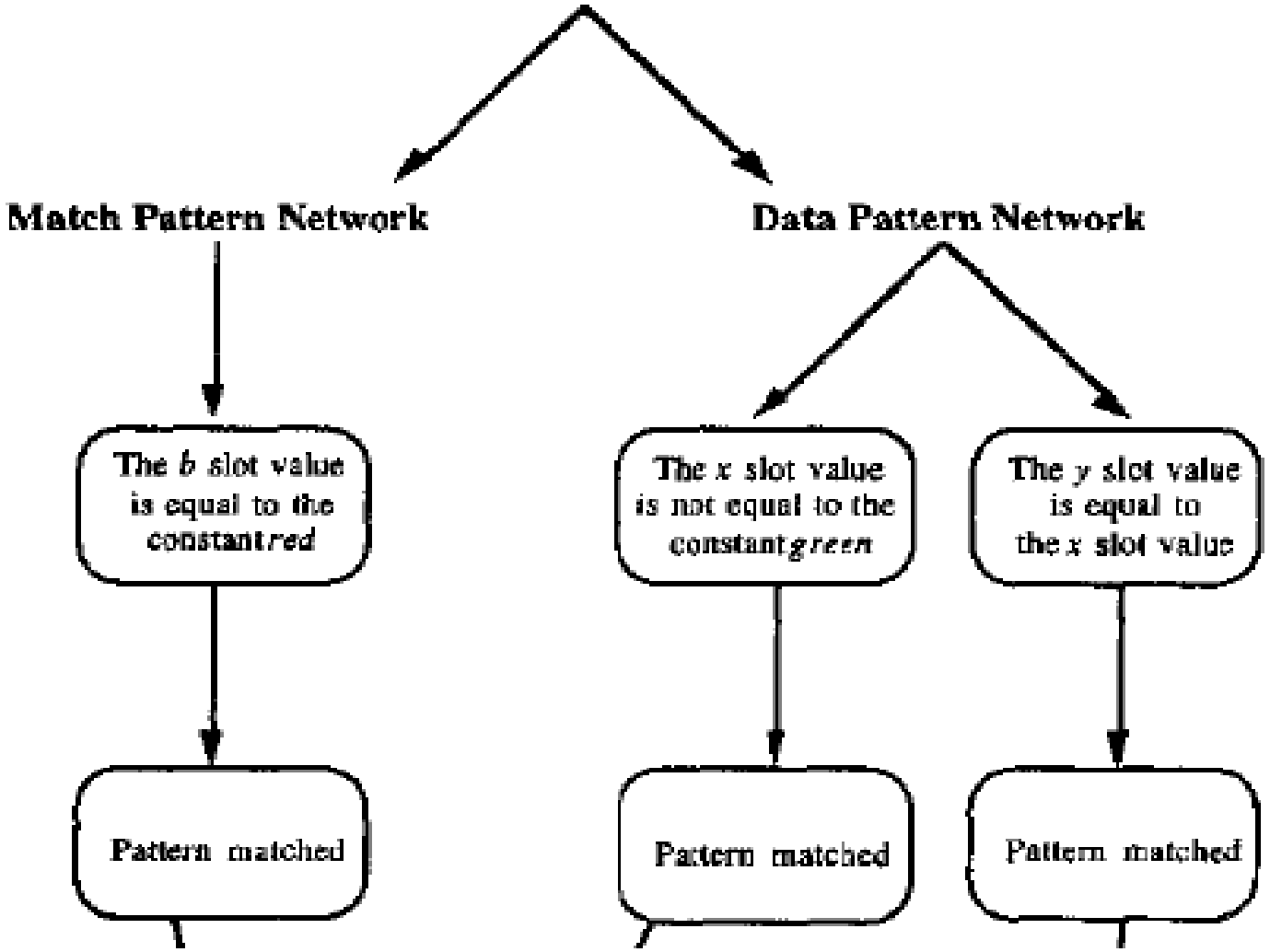
Reteaua de pattern

- Organizata ca un arbore (cate un arbore pentru fiecare template)
- Prima constrangere a unui pattern se afla pe primul nivel
- A doua – pe al doilea nivel… etc
- Tine cont de similaritatea structurala a regulilor (acelasi pattern poate sa apara in mai multe reguli => nodurile corespunzatoare in arbore pot fi partajate)

Reteaua de pattern - exemplu

```
(defrule rete-rule
    (match (a ?x) (b red))
    (data (x ~green) (y ?x))
    (data (x ?x) (y ?x))
=>
    …)
```

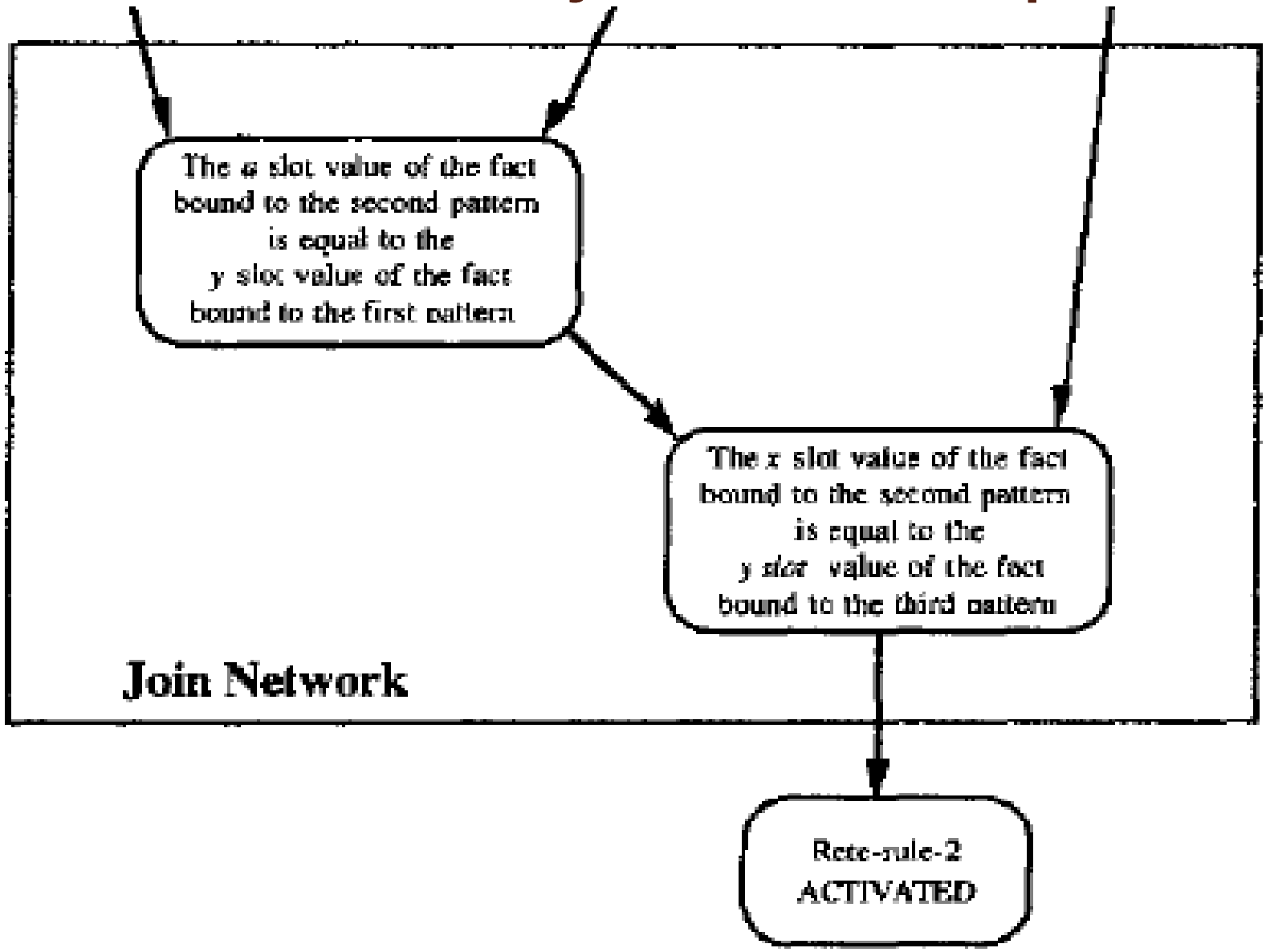
Reteaua de pattern - exemplu



Reteaua de join

- Input = nodurile terminale din reteaua de pattern
- Primul join compara primele 2 patternuri
- Al n-lea join compara patternul n+1 cu potrivirile partiale generate de joinul anterior
- 2 reguli pot partaja o parte din reteaua de join daca incep cu un numar de patternuri si comparatii identice

Reteaua de join - exemplu



Eficienta in CLIPS

Influentata de:

- Ordinea patternurilor
- Gradul de folosire a variabilelor multicamp
- Modul si locul de efectuare a testelor
- Optiunea pentru reguli generale/specifice
- Optiunea pentru reguli simple/complexe
- Folosirea programarii procedurale, atunci cand se impune
- Gradul de incarcare al bazei de fapte

Importanta ordinii patternurilor

- La inceput
  - Patternuri specifice
  - Patternuri care se potrivesc cu putine fapte
- La sfarsit
  - Patternuri care se potrivesc cu fapte volatile (care sunt introduse/retrase frecvent)

Folosirea variabilelor multicamp

- Numai atunci cand este necesar
- Mare grija la a nu folosi mai multe variabile multicamp in acelasi slot

Teste

- Trebuie plasate cat mai devreme (astfel se reduce numarul de potriviri partiale)
- Folosirea constrangerilor built-in este intotdeauna mai eficienta decat efectuarea unor teste cu semnificatie echivalenta

Reguli generale versus reguli specifice

- Reguli specifice
  - Mai multe
  - Reduc lucrul in retelele de pattern si de join
- Reguli generale
  - Mai putine
  - Mai mari sanse de a partaja noduri in retelele de pattern si de join
  - Mai usor de mentinut