DEFTEMPLATE ATRIBUTI

- CLIPS pruža brojne slot atribute kojima se postavljaju ograničenja na tipove i vrednosti koje pojedini slotovi mogu da poprime.

Svakom slotu ili multislotu, u šablonu, mogu se dodeliti različiti ograničavajući atributi (*constraint attributes*) koji se odnose na vrednosti koje on može da prihvati. Tako se mogu definisati:

- atribut za tip
- atribut za default vrednost
- atributi za liste dozvoljenih konstanti
- atribut za opseg dozvoljenih brojnih vrednosti
- atribut za broj polja u multislotu

Atribut za tip (*type attribute*) definiše koji se tip vrednosti može dodeliti slotu. Može biti:

- (**type** **SYMBOL**),
- (**type** **STRING**),
- (**type** **LEXEME**),
- (**type** **INTEGER**),
- (**type** **FLOAT**),
- (**type** **NUMBER**).

```
CLIPS> (deftemplate osoba
                    (multislot ime (type SYMBOL))
                    (slot godine (type INTEGER)))

CLIPS> (assert (osoba (ime Pera Peric)
                      (godine deset)))

[CSTRNCHK1] A literal slot value found in the assert
   command does not match the allowed types for slot
   godine.

CLIPS>
```

Atributi za dozvoljene vrednosti

- Pored restrikcije dozvoljenog tipa za slot, može se definisati i lista dozvoljenih vrednosti koje slot određenog tipa sme da poprimi

```
(deftemplate osoba
  (multislot ime (type SYMBOL))
  (slot godine (type INTEGER)))
 (slot pol
    (allowed-symbols zenski muski)))
```

Atributi za dozvoljene vrednosti:

`allowed-symbols`, **`allowed-strings`**, **`allowed-lexemes`**, **`allowed-integers`**, **`allowed-floats`**, **`allowed-numbers`**, **`allowed-values`**. Iza svakog od ovih atributa treba da se navede lista vrednosti odgovarajućeg tipa, ili ključna reč ?VARIABLE koja označava da su sve vrednosti naznačenog tipa dozvoljene.

Atributi za dozvoljene vrednosti

- Ne ograničavaju tip vrednosti koje slot može poprimiti.
- U prethodnom primeru, **`(allowed-symbols zenski muski)`** samo govori da ako je vrednost koju slot poprima tipa **`symbol`**, onda ona mora biti ili zenski ili muski, dok ako nije tipa symbol, može biti bilo koja.

```
(deftemplate osoba
  (multislot ime (type SYMBOL))
  (slot godine (type INTEGER)))
  (slot pol (type SYMBOL)
     (allowed-symbols zenski muski)))
```

ILI

```
(deftemplate osoba
  (multislot ime (type SYMBOL))
  (slot godine (type INTEGER)))
  (slot pol (allowed-values zenski muski)))
```

Razlika između allowed-symbols i allowed-values

- `(allowed-symbols red green blue)`
- `(allowed-values red green blue)`

Allowed-symbols atribut znači da ako je vrednost slota tipa symbol onda ona mora biti sa date liste simbola, u suprotnom može biti bilo koja vrednost. Allowed-values atribut potpuno ograničava dozvoljene vrednosti slota na one sa liste bez obzira na tip.

Atribut za opseg dozvoljenih brojnih vrednosti (*range attribute*)

- Određuje granice intervala iz kog se mogu uzeti brojne vrednosti koje se dodeljuju slotu tipa NUMBER, INTEGER ili FLOAT, na koji se atribut odnosi.

Atribut za opseg dozvoljenih brojnih vrednosti (*range attribute*)

(range <donja-granica> <gornja-granica>)

- donja-granica i gornja-granica mogu biti celi ili realni brojevi.
- Ako se umesto broja za donju granicu upotrebi ključna reč ?VARIABLE, onda je donja granica - `
- Ako se umesto broja za gornju granicu upotrebi ključna reč ?VARIABLE, onda je gornja granica + `

- Ne može se primenjivati sa sledećim atributima:
  allowed-values, allowed-numbers, allowed-integers, allowed-floats.

Atribut za broj polja u multislotu (*cardinality attribute*)

- Ograničava broj polja multifield slota.
- Ne može se upotrebiti kao atribut single field slota.

```
(cardinality <minimalan-broj-polja>
             <maksimalan-broj-polja>)
```

- Za minimalan i maksimalan broj polja mogu se upotrebiti samo celi brojevi.

- Ako se kao minimalan (maksimalan) broj polja upotrebi ključna reč ?VARIABLE onda je minimalna (maksimalna) kardinalnost nula (+ `).

Atribut za default vrednost slota definiše vrednost koja će biti upotrebljena u slotu, ako mu se ona ne dodeli **`assert`** komandom. Može biti:

- (**`default <vrednost>`**)
- (**`default ?DERIVE`**) – sam CLIPS izvodi default vrednost na osnovu ograničenja nametnutih datom slotu.
- (**`default ?NONE`**) – obavezuje korisnika da slotu mora dodeliti vrednost pri obavljanju assert-a.

PRIMER:

```
(deftemplate osoba
  (multislot ime
    (type STRING)
    (default ?NONE))
  (slot godine
    (type INTEGER)
    (default ?DERIVE)
    (range 0 100))
  (slot boja_ociju
    (allowed-values
              smedja plava zelena crna siva))
  (multislot krvni_pritisak
    (type INTEGER)
    (default 120 80)
    (cardinality 2 2)))
```

PRIMER:

```
CLIPS> (assert (osoba (ime "Pera Peric")
                      (godine 45)
                      (boja_ociju plava)))

CLIPS> (facts)
f-0     (osoba (ime "Pera Peric") (godine 45)
  (boja_ociju plava) (krvni_pritisak 120 80))
For a total of 1 fact.

CLIPS>
```

PRIMER:

```
CLIPS> (assert (osoba))

[TMPLTRHS1] Slot ime requires a value because of its
   (default ?NONE) attribute.

CLIPS> (assert (osoba (ime "Laza Lazic")))
==> f-1     (osoba (ime "Laza Lazic") (godine 0)
   (boja_ociju smedja) (krvni_pritisak 120 80))
<Fact-1>

CLIPS> (assert (osoba (ime "Mika Mikic") (godine 234)))

[CSTRNCHK1] A literal slot value found in the assert
   command does not fall in the allowed range 0 to 100
   for slot godine.
```

Eksplicitna kontrola izvršavanja programa

- salience
- moduli

Salience

- Agenda je kao stek – LAST IN FIRST OUT. Komanda salience obezbeđuje da se prioritet pravila eksplicitno odredi bez obzira na redosled aktivacije pravila.

```
(declare (salience <brojna-vrednost>))
```

- Salience može imati vrednosti iz opsega od -10,000 do 10,000. Ako salience pravila nije eksplicitno deklarisan, CLIPS mu dodeljuje default vrednost 0.

```
CLIPS>(defrule r1
         =>
       (printout t "Prvo pravilo" crlf))

CLIPS>(defrule r2
         =>
       (printout t "Drugo pravilo" crlf))

CLIPS>(defrule r3
         =>
       (printout t "Trece pravilo" crlf))

CLIPS> (run)
Trece pravilo
Drugo pravilo
Prvo pravilo
```

```
CLIPS>(defrule r1
      (declare (salience 30))
       =>
      (printout t "Prvo pravilo" crlf))

CLIPS>(defrule r2
      (declare (salience 20))
       =>
      (printout t "Drugo pravilo" crlf))

CLIPS>(defrule r3
      (declare (salience 10))
       =>
      (printout t "Trece pravilo" crlf))

CLIPS>
Prvo pravilo
Drugo pravilo
Trece pravilo
```

`salience`

- Preterana uporeba **`salience`** naredbe rezultira loše kodiranim programom .
- **`salience`** ne treba koristiti kao metodu odabira jednog pravila iz grupe pravila kad god je moguće upotrebiti uslove kojima se opisuje kriterijum selekcije određenog pravila

Primer:

```
(defrule pobeda
(declare (salience 10))
?faza <- (izbor-poteza)
(otvoreno-polje pobeda)
=>
(retract ?faza)
(assert (potez za-pobedu)))

(defrule blokada
(declare (salience 5))
?faza <- (izbor-poteza)
(otvoreno-polje blokada)
=>
(retract ?faza)
(assert (potez za-blokadu)))

(defrule bilo-koje-polje
?faza <- (izbor-poteza)
(otvoreno-polje ?a&ugao|sredina|strana)
=>
(retract ?faza)
(assert (potez ?a)))
```

```
CLIPS> (assert (izbor-poteza))
==> f-1     (izbor-poteza)
<Fact-1>

CLIPS> (assert (otvoreno-polje sredina))
==> Activation 0      bilo-koje-polje: f-1,f-2
<Fact-2>

CLIPS> (assert (otvoreno-polje pobeda))
==> Activation 10     pobeda: f-1,f-3
<Fact-3>

CLIPS> (agenda)
10     pobeda: f-2,f-4
0      bilo-koje-polje: f-2,f-3
For a total of 2 activations.
```

Rešenje bez upotrebe salience komande

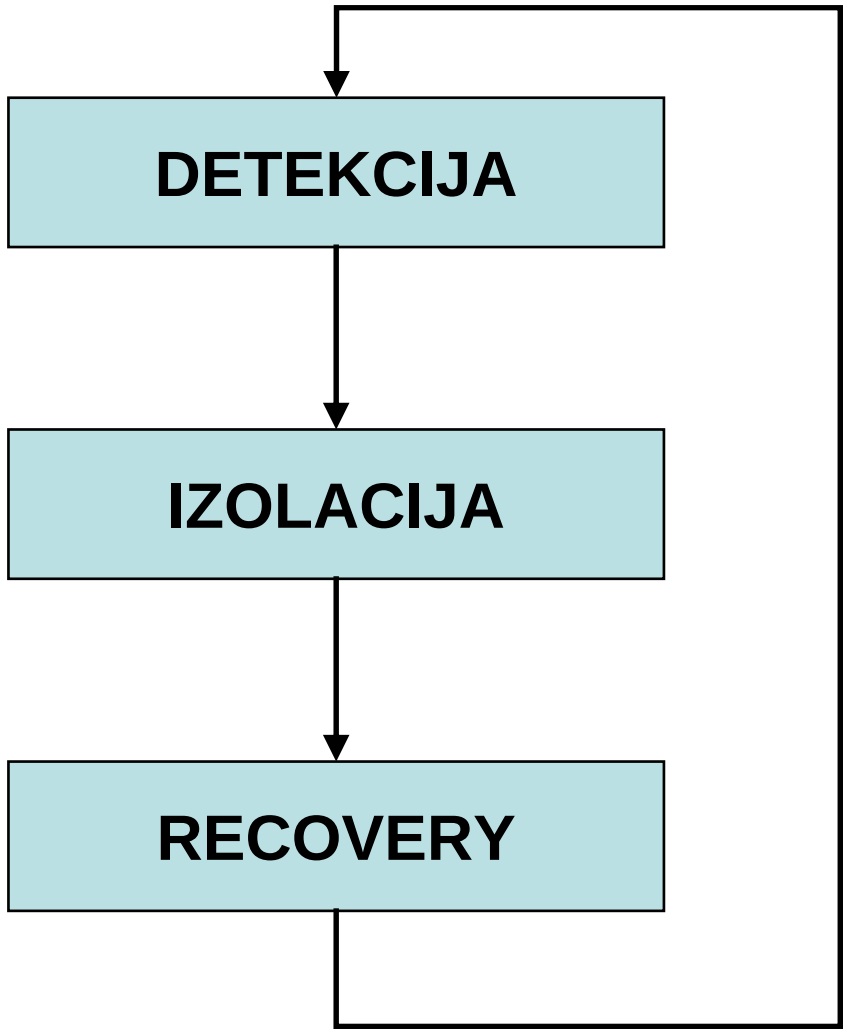
```
(defrule pobeda
?faza <- (izbor-poteza)
(otvoreno-polje pobeda)
=>
(retract ?faza)
(assert (potez za-pobedu)))

(defrule blokada
?faza <- (izbor-poteza)
(otvoreno-polje blokada)
(not (otvoreno-polje pobeda))
=>
(retract ?faza)
(assert (potez za-blokadu)))

(defrule bilo-koje-polje
?faza <- (izbor-poteza)
(otvoreno-polje ?a&ugao|sredina|strana)
(not (otvoreno-polje pobeda))
(not (otvoreno-polje blokada))
=>
(retract ?faza)
(assert (potez ?a)))
```

```
CLIPS> (assert (izbor-poteza))
==> f-1     (izbor-poteza)
<Fact-1>

CLIPS> (assert (otvoreno-polje sredina))
==> f-2     (otvoreno-polje sredina)
==> Activation 0      bilo-koje-polje: f-1,f-2,*,*
<Fact-2>

CLIPS> (assert (otvoreno-polje pobeda))
==> f-3     (otvoreno-polje pobeda)
<== Activation 0      bilo-koje-polje: f-1,f-2,*,*
==> Activation 0      pobeda: f-1,f-3
<Fact-3>

CLIPS> (agenda)
0      pobeda: f-1,f-3
For a total of 1 activation.
```

Primer faznog izvršavanja ES-a

Neka treba napraviti ES koji vrši detekciju, izolaciju i recovery sistema koa što je neki elektronski uređaj.

- Detekcija: ES prepoznaje da uređaj ne funkcioniše ispravno.
- Izolacija: ES određuje koje komponente su uzrokovale problem
- Recovery: ES utvrđuje korake neophodne za ispravku greške, ako je to moguće


DETEKCIJA
IZOLACIJA
RECOVERY

Primer faznog izvršavanja ES-a

- Kontrola toka izvršavanja u ovakvom ESu se može vršiti na više načina, primenom **`salience`**-a ili modula (o kojima će biti reči kasnije)

Prvi pristup

- Znanje o kontroli toka je u pravilima (npr. Pravila iz oblasti detekcije će uključivati i ona koja će određivati kada treba ući u fazu izolacije)
- Svaka grupa pravila (za detekciju, izolaciju i recovery) će imati uslov kojim se određuje u kojoj fazi se ta pravila primenjuju.

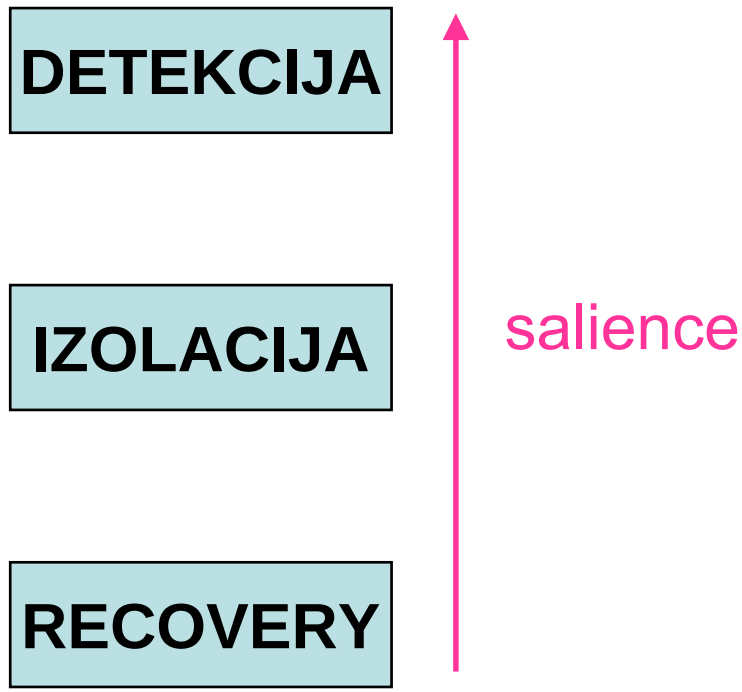
Mane prvog pristupa:

- Uključivanje znanja o kontoli toka u pravila, čini pravila teško razumljivim.

- Nije uvek lako utvrditi kada je nastupio kraj jedne faze, te zato uvek mora postojati pravilo koje je primenljivo samo nakon što su sva pravila iz jedne faze okinula.

Drugi pristup

- Upotreba salience naredbe za organizaciju pravila u agendi



Mane drugog pristupa

- Znanje o kontroli toka je i dalje u pravilima
- Nije garantovan korektan redosled izvršavanja pravila

Treći pristup

- Odvajanje znanja o kontroli toka od ostalog znanja u ESu

Treći pristup

- Za svako pravilo postoji uslov koji određuje u kojoj se fazi to pravilo primenjuje

  npr. **`(defrule za-detekciju`**
  `(faza detekcija)...=>...)`

- Kontrolna pravila prebacuju kontrolu sa jedne faze na drugu.

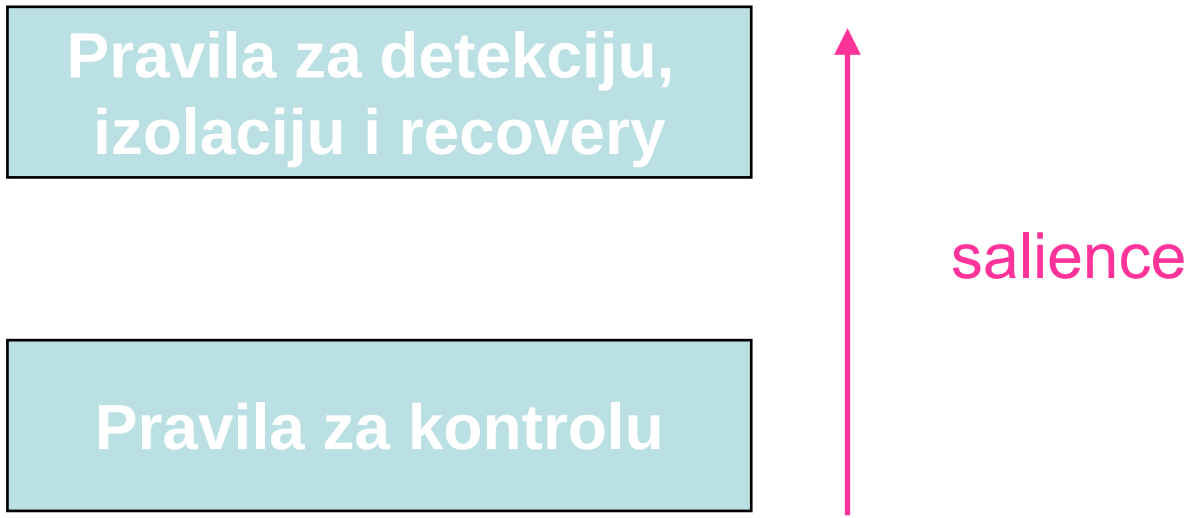
```
(defrule predji-sa-detekcije-na-izolaciju
(declare (salience -10)
?faza <- (faza detekcija)
=>
(retract ?faza)
(assert (faza izolacija)))

(defrule predji-sa-izolacije-na-recovery
(declare (salience -10)
?faza <- (faza izolacija)
=>
(retract ?faza)
(assert (faza recovery)))

(defrule predji-sa-recovery-na-detekciju
(declare (salience -10)
?faza <- (faza recovery)
=>
(retract ?faza)
(assert (faza detekcija)))
```

Hijerarhija salience-a

- Hijerarhija salience-a je opis salience vrednosti koje koristi ES.

- Svaki nivo u hijerarhiji salience-a odgovara određenom skupu pravila čiji članovi imaju jednak salience.

Ako pravila za detekciju, izolaciju i recovery imaju salienc
0, a kontrolna pravila izgledaju kao u primeru, tada je
hijerarhija salience-a sledeća:



Na primer, sve dok je činjenica **`(faza detekcija)`** u listi činjenica pravilo **`predi-sa-detekcije-na-izolaciju`** će biti u agendi. Kako ima niži salience od svih pravila za detekciju, neće okinuti sve dok sva pravila za detekciju ne dobiju priliku da se izvrše.

`defmodule` konstrukcija

- Koristi se u CLIPSu kako bi se izvršila podela baze znanja na delove – module.
- Sintaksa

```
(defmodule <naziv-modula> [<komentar>])
```

CLIPS po default-u definiše modul MAIN

```
CLIPS> (deftemplate senzor (slot naziv))
CLIPS> (ppdeftemplate senzor)
(deftemplate MAIN::senzor
   (slot naziv))
CLIPS>
```

`::` je modul separator. Sa njegove desne strane je naziv konstrukcije, a sa leve je naziv modula u kome se konstrukcija nalazi.

```
CLIPS> (defmodule detekcija)
CLIPS> (defmodule izolacija)
CLIPS> (defmodule recovery)

CLIPS> (defrule primer =>)
CLIPS> (ppdefrule primer)
(defrule recovery::primer
   =>)
CLIPS> (defrule izolacija::primer2 =>)
CLIPS> (ppdefrule primer2)
(defrule izolacija::primer2
   =>)
CLIPS> (get-current-module)
izolacija
CLIPS> (set-current-module detekcija)
izolacija
CLIPS> (get-current-module)
detekcija
```

naredba za prikaz trenutnog modula

naredba za promenu trenutnog modula

```
CLIPS> (list-defrules)
CLIPS>
CLIPS>(set-current-module izolacija)
detekcija
CLIPS> (list-defrules)
primer2
For a total of 1 defrule.
CLIPS> (list-defrules recovery)
primer
For a total of 1 defrule.

CLIPS> (list-defrules *)
MAIN:
detekcija:
izolacija:
   primer2
recovery:
   primer
For a total of 2 defrules.
```

```
CLIPS> (ppdefrule primer2)
(defrule izolacija::primer2
   =>)
CLIPS> (ppdefrule primer)
[PRNTUTIL1] Unable to find defrule primer.
CLIPS> (ppdefrule recovery::primer)
(defrule recovery::primer
   =>)

CLIPS> (defrule detekcija::primer =>)
CLIPS> (list-defrules *)
MAIN:
detekcija:
   primer
izolacija:
   primer2
recovery:
   primer
For a total of 3 defrules
```