

# Džokeri (*wildcards*)

Džokeri se koriste kada je potrebno utvrditi postojanje nekih polja u slotu, a da pri tome nije potrebno utvrditi i konkretne vrednosti tih polja.

- ✓ singlefield wildcards ?
- ✓ multiframe wildcards \$?

# Primer za singlefield wildcards

Dat je deftemplate:

```
(deftemplate osoba
  (multislot ime)
  (slot JMBG))
```

Neka je potrebno prikazati JMBG svih osoba sa zadatim prezimenom, pri čemu multislot ime podrazumeva ime, srednje slovo i prezime.

```
(defrule prikazi-JMBG
  (prikazi-JMBG-za ?prezime)
  (osoba (ime ? ? ?prezime)(JMBG ?br))
  =>
  (printout t ?br crlf))
```

Koristimo džokere ? jer nam nije potrebno da pamtimo vrednosti polja ime i srednje slovo u multislottu ime.

```
(deftemplate osoba
  (multislot ime)
  (slot JMBG))
```

```
(def facts ljudi
  (osoba (ime Pera S. Peric)(JMBG 1203976-456922))
  (osoba (ime Laza M. Lazic)(JMBG 2311967-345673))
  (osoba (ime Mika R. Peric)(JMBG 0506965-425077))
  (osoba (ime Lana B. Peric)(JMBG 1607977-725022))
  (osoba (ime Mika S. Mikic)(JMBG 2304956-234567))
  )
```

```
(defrule prikazi-JMBG
  (prikazi-JMBG-za ?prezime)
  (osoba (ime ? ? ?prezime)(JMBG ?br))
  =>
  (printout t ?br crlf))
```

```
CLIPS> (reset)

==> f-0      (initial-fact)
==> f-1      (osoba (ime Pera S. Peric) (JMBG 1203976-456922))
==> f-2      (osoba (ime Laza M. Lazic) (JMBG 2311967-345673))
==> f-3      (osoba (ime Mika R. Peric) (JMBG 0506965-425077))
==> f-4      (osoba (ime Lana B. Peric) (JMBG 1607977-725022))
==> f-5      (osoba (ime Mika S. Mikic) (JMBG 2304956-234567))
```

```
CLIPS> (assert (prikazi-JMBG-za Peric))
```

```
==> f-6      (prikazi-JMBG-za Peric)
==> Activation 0      prikazi-JMBG: f-6,f-1
==> Activation 0      prikazi-JMBG: f-6,f-3
==> Activation 0      prikazi-JMBG: f-6,f-4
<Fact-6>
```

```
CLIPS> (run)
```

```
FIRE      1 prikazi-JMBG: f-6,f-4
1607977-725022
FIRE      2 prikazi-JMBG: f-6,f-3
0506965-425077
FIRE      3 prikazi-JMBG: f-6,f-1
1203976-456922
```



- **\$?** (*multifield wildcard*) je džoker koji može da zameni nijedno ili više polja u slotu u uslovu pravila.

U prethodnom primeru je bolje upotrebiti jedan multifield wildcard, nego dva singlefield.

```
(defrule prikazi-JMBG
  (prikazi-JMBG-za ?prezime)
  (osoba (ime $? ?prezime) (JMBG ?br))
  =>
  (printout t ?br crlf))
```

# Primer:

```
CLIPS> (defrule nadji_boje
  (boje crvena $?)
  =>
  (printout t "Boja je nadjena" crlf))
```

```
CLIPS> (deffacts f1
  (boje crvena plava zelena)
  (boje plava zuta)
  (boje crvena)
  (boje crvena zelena))
```



CLIPS> (reset)

```
==> f-0      (initial-fact)
==> f-1      (boje crvena plava zelena)
==> Activation 0      nadji_boje: f-1
==> f-2      (boje plava zuta)
==> f-3      (boje crvena)
==> Activation 0      nadji_boje: f-3
==> f-4      (boje crvena zelena)
==> Activation 0      nadji_boje: f-4
```

CLIPS> (run)

```
FIRE      1 nadji_boje: f-1
Boja je nadjena
FIRE      2 nadji_boje: f-3
Boja je nadjena
FIRE      3 nadji_boje: f-4
Boja je nadjena
```

CLIPS>

# Primeri:

- Ako bi na levoj strani pravila imali patern (boje ? plava \$? ), onda bi on odgovarao, na primer, sledećim paternima:
  - ✓ (boje crvena plava)
  - ✓ (boje crvena plava zelena)
  - ✓ (boje zuta plava crvena zelena)
  - ✓ ...

- Ako bi na levoj strani pravila imali patern (boje \$? plava \$?), onda bi on odgovarao, na primer, sledećim paternima:
  - ✓ (boje crvena plava)
  - ✓ (boje crvena zuta plava zelena roze)
  - ✓ (boje plava)
  - ✓ (boje plava boje plava)

Poslednja činjenica će aktivirati pravilo koje sadrži dati patern dva puta

U zavisnosti od toga da li se navode iza single-field ili multifold džokera promenljive takođe mogu biti:

✓ *single-field*

(mogu da prihvate tačno jednu vrednost, ?**x**)

✓ *multifold*

(mogu da prihvate nijednu ili više vrednosti, \$?**y**)

Sve promenljive koji smo do sada koristili su singlefield.

```
deftemplate osoba
  (multifield ime)
  (multifield deca))
(deffacts neki-ljudi
  (osoba (ime Pera Peric)(deca Ana Milan Jelena))
  (osoba (ime Laza Lazic)(deca Marko Ivana))
  (osoba (ime Mika Mikic)(deca Milan Aleksa Lena))
)
(defrule prikazi-decu
  (prikazi-decu-od $?ime)
  (osoba (ime $?ime)(deca $?deca))
  =>
  (printout t ?ime " ima decu: " ?deca crlf))
```

Kada se multifield promenljive koriste na RHS ne mora se navoditi znak \$.

```
CLIPS> (reset)
==> f-0 (initial-fact)
==> f-1 (osoba (ime Pera Peric)(deca Ana Milan Jelena))
==> f-2 (osoba (ime Laza Lazic)(deca Marko Ivana))
==> f-3 (osoba (ime Mika Mikic)(deca Milan Aleksa Lena))
```

```
CLIPS> (assert (prikazi-decu-od Pera Peric))
==> f-3      (prikazi-decu-od Pera Peric)
==> Activation 0      prikazi-decu: f-3,f-1
<Fact-3>
```

```
CLIPS> (run)
FIRE      1 prikazi-decu: f-3,f-1
(Pera Peric) ima decu koja se zovu: (Ana Milan Jelena)
```

```
CLIPS>
```

```
(defrule prikazi-osobu
  (dete-se-zove ?dete)
  (osoba (ime $?ime)(deca $? ?dete $?))
  =>
  (printout t ?ime " ima dete: " ?dete crlf))
```

```
CLIPS> (reset)
==> f-0      (initial-fact)
==> f-1      (osoba (ime Pera Peric)(deca Ana Milan Jelena))
==> f-2      (osoba (ime Laza Lazic)(deca Marko Ivana))
==> f-3      (osoba (ime Mika Mikic)(deca Milan Aleksa Lena))
```

```
CLIPS> (assert (dete-se-zove Milan))
==> f-4      (dete-se-zove Milan)
==> Activation 0      prikazi-osobu: f-4,f-1
==> Activation 0      prikazi-osobu: f-4,f-3
<Fact-4>
```

```
CLIPS> (run)
FIRE      1 prikazi-osobu: f-4,f-3
(Mika Mikic) ima dete: Milan
FIRE      2 prikazi-osobu: f-4,f-1
(Pera Peric) ima dete: Milan
```

```
CLIPS>
```



## Primer:

```
(deffacts f
  (data 1 blue)
  (data 1 blue red)
  (data 1 blue red 6.9))

(defrule find-data-1
  (data ?x $?y ?z)
  =>
  (printout t "?x = " ?x crlf
             "?y = " ?y crlf
             "?z = " ?z crlf
             "-----" crlf))
```

```
CLIPS> (reset)
==> f-0      (initial-fact)
==> f-1      (data 1 blue)
==> Activation 0      find-data-1: f-1
==> f-2      (data 1 blue red)
==> Activation 0      find-data-1: f-2
==> f-3      (data 1 blue red 6.9)
==> Activation 0      find-data-1: f-3
```

```

CLIPS> (run)
FIRE    1 find-data-1:
    f-3
?x = 1
?y = (blue red)
?z = 6.9
-----
FIRE    2 find-data-1:
    f-2
?x = 1
?y = (blue)
?z = red
-----
FIRE    3 find-data-1:
    f-1
?x = 1
?y = ( )
?z = blue
-----

```

## Primer 1:

### Objašnjenje:

Promenljive ?x i ?z su single-field, pa mogu prihvatiti tačno po jednu vrednost i to drugog i poslednjeg polja u fact-u. ?y je multifield, pa ona prihvata 0 ili više vrednosti polja koja se nalaze između drugog i poslednjeg polja u fact-u

Promenljive koje se koriste u uslovima pravila imaju sledeće svojstvo:

- Kada se promenljivoj prvi put dodeli neka vrednost, ona tu vrednost zadržava samo u okviru trenutnog pravila, kako na levoj tako i na desnoj strani pravila, osim ako se ta vrednost ne promeni nekom akcijom na desnoj strani pravila.

# Primer:

```
CLIPS> (defrule r1
        (broj1 ?x)
        (broj2 ?x)
        =>)
```

Ako želimo samo da  
ispitamo na koji način  
će se povezivati uslovi  
nekog pravila sa nekim  
činjenicama iz liste  
RHS se ne mora  
navoditi

```
CLIPS> (deffacts f
        (broj1 12)
        (broj2 12)
        (broj1 45)
        (broj2 45))
```

```
CLIPS> (reset)
```

```
==> f-0      (initial-fact)
```

```
==> f-1      (broj1 12)
```

```
==> f-2      (broj2 12)
```

```
==> Activation 0      r1: f-1,f-2
```

```
==> f-3      (broj1 45)
```

```
==> f-4      (broj2 45)
```

```
==> Activation 0      r1: f-3,f-4
```

```
CLIPS>
```

# Objašnjenje primera:

- Pravilo se aktivira samo parovima činjenica f-1, f-2 i f-3, f-4. Dakle, pravilo se ne može aktivirati, na primer, parom f-1, f-3 jer kada ?x u prvom paternu veže za broj 12 iz f-1, onda vrednost promenljive ?x i u drugom paternu pravila mora biti 12, pa drugi patern ne može odgovarati činjenici f-3.