Pravila

Primer:

Pseudokod jednog pravila ekspertnog sistema za monioring u industrijskom postrojenju

IF nepogoda je pozar

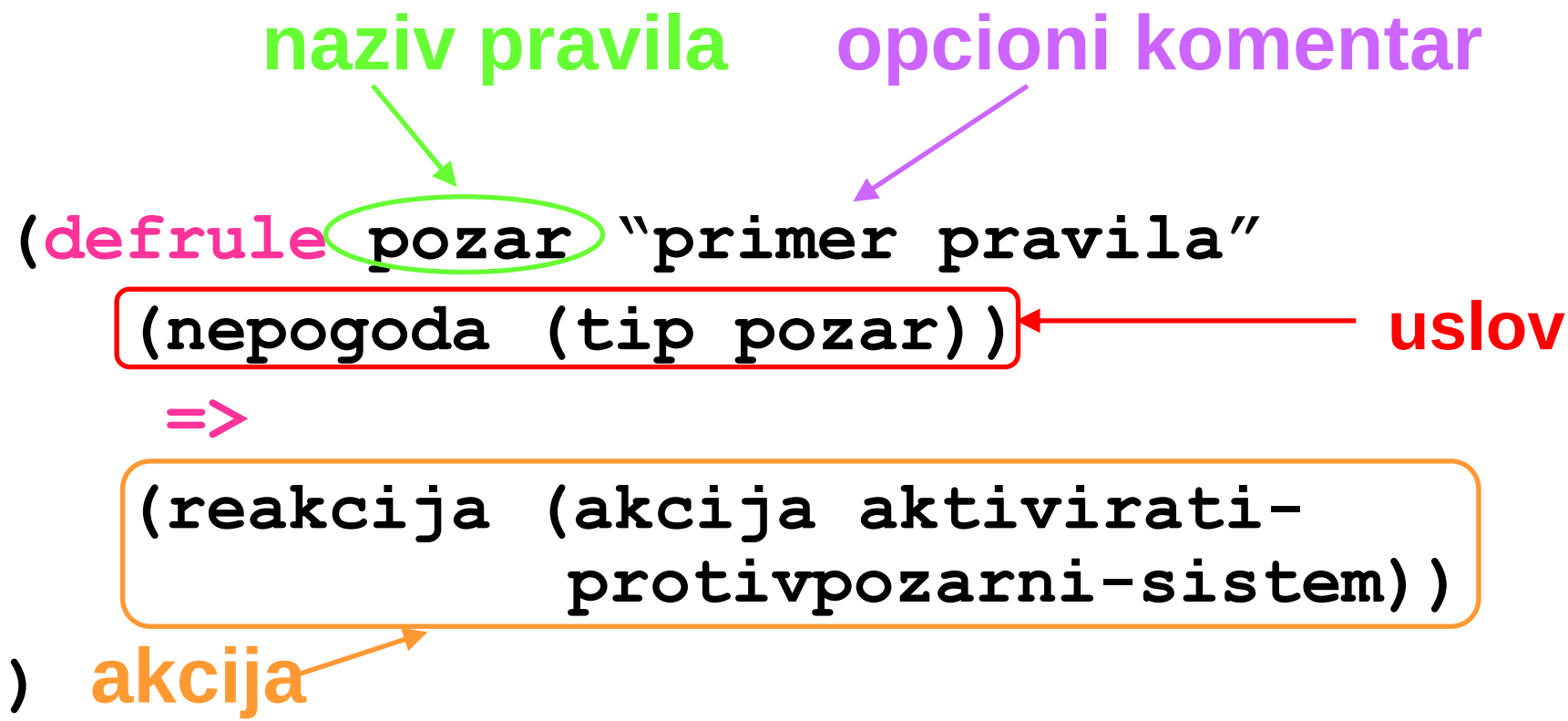
THEN reakcija je aktivirati protivpozarni sistem

Primer

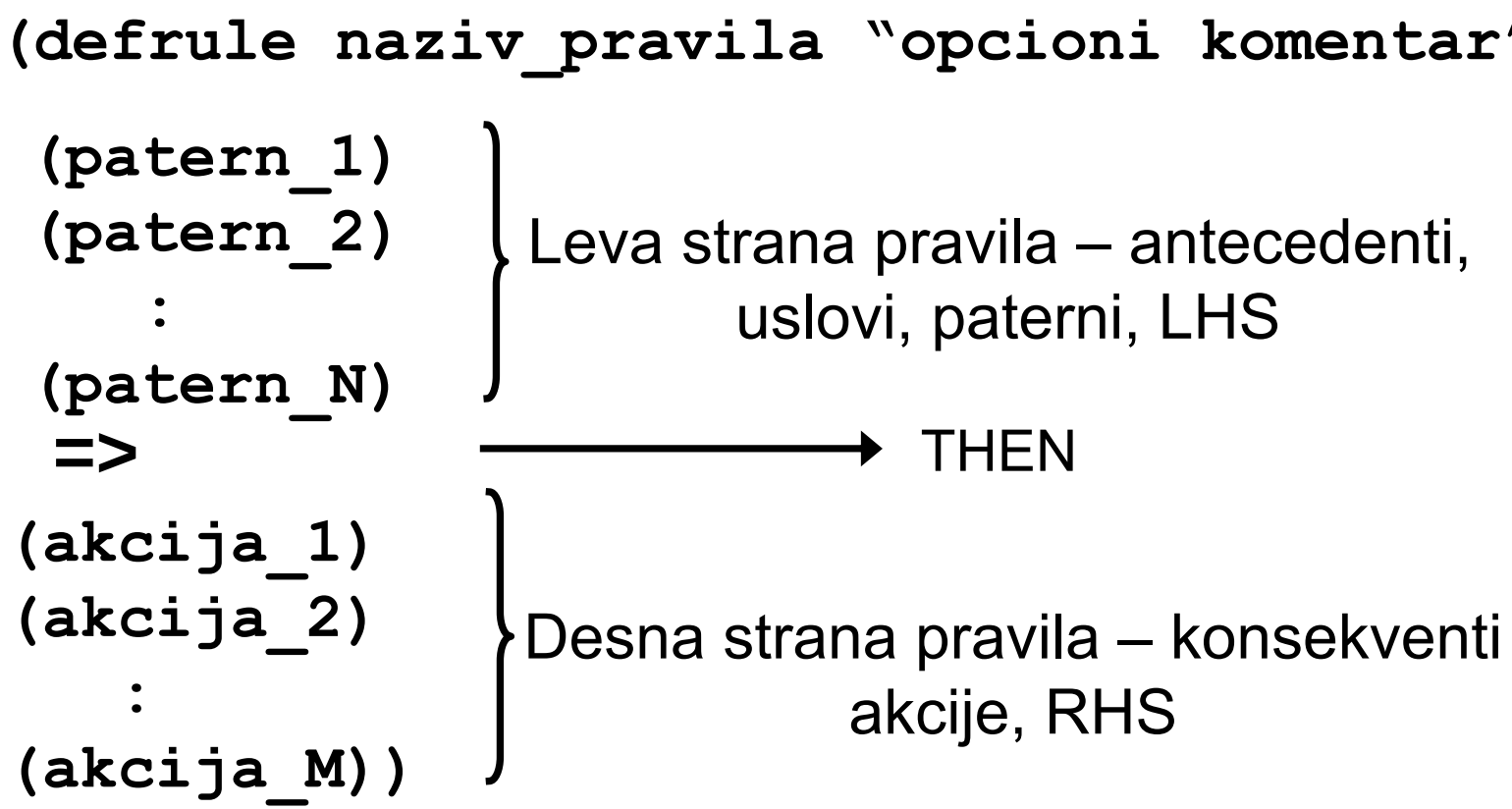
```
(deftemplate nepogoda (slot tip))
(deftemplate reakcija (slot akcija))

(defrule pozar "primer pravila"
   (nepogoda (tip pozar))
    =>
   (assert (reakcija (akcija aktivirati-
              protivpozarni-sistem)))
)
```

Delovi pravila



Kako se pravi pravilo u CLIPSu?



- CLIPS pokušava da spoji uslove pravila sa činjenicama u listi.
- Pravilo se aktivira ako svi njegovi uslovi odgovaraju nekim činjenicama iz liste.
- Aktivirano pravilo se smešta u agendu.
- Agenda je lista pravila čiji su uslovi zadovoljeni (tzv. aktivirana pravila), a koja još uvek nisu izvršena.

- Pravila u agendi CLIPS uređuje u opadajući poredak po prioritetu i prvo se izvršava pravilo sa najvišim prioritetom.

- Izvršeno pravilo se uklanja iz agende i neće biti ponovo aktivirano istim skupom činjenica iz liste kojim je bilo aktivirano prvi put.

- Ako pravilo nema nijedan uslov CLIPS će automatski dodati **`(initial-fact)`** kao uslov pravila. To pravilo će se aktivirati uvek nakon **`reset`** komande.

Pokretanje programa

Naredba za pokretanje programa je run. Sintaksa:

```
(run [<granica>])
```

Granica je maksimalni broj pravila koja će se izvršiti. Ako nije navedena pravila će se izvršavati sve dok se ne isprazni agenda.

Prikazivanje agende

Lista pravila koja su u agendi može biti prikazana navođenjem komande **`(agenda)`**

Sadržaj agende se može i neprestano pratiti uključivanjem prozora Agenda u meniju Window

```
CLIPS> (assert (nepogoda (tip pozar)))
<Fact-0>
CLIPS> (agenda)
0      pozar: f-0
For a total of 1 activation.
CLIPS> (run)
CLIPS> (facts)
f-0     (nepogoda (tip pozar))
f-1     (reakcija (akcija aktivirati-
                          protivpozarni-
  sistem))
For a total of 2 facts.
CLIPS>
```

Komande za prikaz lista

- `(list-defrules)`
- `(list-deftemplates)`
- `(list-deffacts)`

Komande za prikaz pojedinačnih pravila, template-a i činjenica

- **(ppdefrule <naziv-pravila>)**
- **(ppdeftemplate <naziv-templatea>)**
- **(ppdeffacts <naziv-cinjenice>)**

Brisanje

- **(undefrule <naziv-pravila>)**
- **(undeftemplate <naziv-templatea>)**
- **(undeffacts <naziv-cinjenice>)**

Praćenje dešavanja

- Uključivanjem odgovarajućih prozora u meniju *Windows* (Facts, Agenda,...)
- Komadom *Watch* koja se može pokrenuti iz menija *Execution*, nakon čega se vrši odabir onoga što će se pratiti (facts, rules, activations,...)
- Komandom *Turn dribble on* započinje se sa “snimanjem” trenutne sesije u neki tekstualni fajl.

Primer:

```
CLIPS> (defrule duck_quack
         (animal-is duck)
          =>
         (assert (sound-is quack))
         (printout t "Quack, quack" crlf))

CLIPS> (assert (animal-is duck))
<Fact-0>
CLIPS> (run)
Quack, quack
CLIPS> (facts)
f-0     (animal-is duck)
f-1     (sound-is quack)
For a total of 2 facts.
CLIPS>
```

Kada pokrenemo **(watch facts)** i **(watch rules)**:

```
CLIPS> (defrule duck_quack
         (animal-is duck)
         =>
         (assert (sound-is quack))
         (printout t "Quack, quack" crlf))
CLIPS> (assert (animal-is duck))
==> f-0     (animal-is duck)
<Fact-0>
==> Activation 0      duck_quack: f-0
CLIPS> (run)
FIRE    1 duck_quack: f-0
==> f-1     (sound-is quack)
Quack, quack
CLIPS>
```

Važne komande

`(exit)`	Zatvara CLIPS
`(clear)`	Uklanja sva pravila i sve činjenice iz radne memorije. Ekvivalentno zatvaranju i ponovnom pokretanju CLIPS-a
`(reset)`	Uklanja činjenice iz memorije i resetuje agendu
`(run)`	Pokreće CLIPS program

Zadatak 3

Na osnovu pravila datog u pseudokodu napravite CLIPS pravilo:

IF nepogoda je poplava

THEN isključiti električne uređaje.

Uputstvo: koristite deftemplate nepogoda i deftemplate reakcija.

SET-BREAK komanda

- Komanda za debagovanje
- Prekida izvršenje programa na određenom pravilu
- Pravilo kod koga se prekida program (to pravilo se ne izvršava kada na njega dođe red) naziva se *breakpoint*.
- Sintaksa

```
(set-break <naziv-pravila>)
```

Primer

```
(defrule prvo-pravilo
 =>
(assert (izvrsi drugo-pravilo)))

(defrule drugo-pravilo
(izvrsi drugo-pravilo)
 =>
(assert (izvrsi trece-pravilo)))

(defrule trece-pravilo
(izvrsi trece-pravilo)
 =>)
```

```
CLIPS> (watch rules)
CLIPS> (reset)
CLIPS> (run)
FIRE    1 prvo-pravilo: f-0
FIRE    2 drugo-pravilo: f-1
FIRE    3 trece-pravilo: f-2


CLIPS> (set-break drugo-pravilo)
CLIPS> (set-break trece-pravilo)
CLIPS> (reset)
CLIPS> (run)
FIRE    1 prvo-pravilo: f-0
Breaking on rule drugo-pravilo.
CLIPS> (run)
FIRE    1 drugo-pravilo: f-1
Breaking on rule trece-pravilo.
CLIPS> (run)
FIRE    1 trece-pravilo: f-2
CLIPS>
```

Prikazivanje i uklanjanje breakpoint-a

```
CLIPS> (show-breaks)
drugo-pravilo
trece-pravilo

CLIPS> (remove-break drugo-pravilo)
CLIPS> (remove-break trece-pravilo)
CLIPS> (show-breaks)
CLIPS>
```

Promenljive (variables)

- Promenljive čuvaju dodeljene vrednosti
- Naziv promenljive mora počinjati znakom pitanja:

  ?x

  ?ime

  ?boja

Promenljive

Promenljive se upotrebljavaju tako da im se tokom procesa uparivanja činjenica na levoj strani pravila sa činjenicama u listi dodeli vrednost, a zatim se ta vrednost koristi na desnoj strani pravila

```
(deftemplate osoba
        (slot ime)
        (slot godine)
        (slot boja-ociju)
        (slot boja-kose))


(defrule nadji-plave-oci
(osoba (ime ?ime)(boja-ociju plava))
=>
(printout t ?ime " ima plave oci." crlf))
```

```
(deffacts ljudi
  (osoba (ime "Mika Mikic")
         (godine 45)
         (boja-ociju plava)
         (boja-kose plava))
  (osoba (ime "Laza Lazic")
         (godine 37)
         (boja-ociju zelena)
         (boja-kose crna))
  (osoba (ime "Mara Maric")
         (godine 28)
         (boja-ociju plava)
         (boja-kose smedja)))
```

```
CLIPS> (reset)
CLIPS> (run)
FIRE    1 nadji-plave-oci: f-3
Mara Maric ima plave oci.
FIRE    2 nadji-plave-oci: f-1
Mika Mikic ima plave oci.
CLIPS>
```

Višestruka upotreba promenljive

- Kada se promenljiva prvi put upotrebi na LHS nekog pravila, biva joj dodeljena vrednost koju zadržava kroz celo pravilo.

- Ostala pojavljivanja promenljive u pravilu moraju imati vrednost koja joj je dodeljena pri prvom pojavljivanju.

Primer

Umesto da pravimo posebna pravila za traženje svake boje očiju, možemo, višestrukom upotrebom jedne promenljive u pravilu, koristiti jedno pravilo za razne boje. Pri tome će korisnik unosom jedne činjenice izabrati o kojoj boji se radi.

```
(defrule nadji-oci
(nadji ?boja)
(osoba (ime ?ime)(boja-ociju ?boja))
=>
(printout t ?ime " ima " ?boja " oci." crlf))

CLIPS> (assert (nadji plava))
<Fact-4>
CLIPS> (run)
FIRE    1 nadji-oci: f-4,f-3
Mara Maric ima plava oci.
FIRE    2 nadji-oci: f-4,f-1
Mika Mikic ima plava oci.

CLIPS> (assert (nadji zelena))
<Fact-5>
CLIPS> (run)
FIRE    1 nadji-oci: f-5,f-2
Laza Lazic ima zelena oci.
CLIPS>
```

Dodeljivanje adrese činjenice promenljivoj:

```
?naziv_prom <- (neka činjenica)
```

Koristi se kada je na RHS strani pravila potrebno izvršiti modifikaciju ili duplikaciju činjenice ili njeno izbacivanje iz liste.

```
(deftemplate osoba
        (slot ime)
        (slot adresa))

(deftemplate preseljen
        (slot ime)
        (slot adresa))

(deffacts primer
  (osoba (ime "Mika Mikic")
         (adresa "Nusiceva 56"))
  (preseljen (ime "Mika Mikic")
         (adresa "Karadjordjeva 108")))
```

```
(defrule promena-adrese
?f1 <-(preseljen (ime ?ime)
                 (adresa ?adresa))
?f2 <-(osoba (ime ?ime))
=>
(retract ?f1)
(modify ?f2 (adresa ?adresa)))
```

```
CLIPS> (reset)
==> f-0     (initial-fact)
==> f-1     (osoba (ime "Mika Mikic") (adresa
"Nusiceva 56"))
==> f-2     (preseljen (ime "Mika Mikic")
(adresa "Karadjordjeva 108"))
CLIPS> (run)
FIRE    1 promena-adrese: f-2,f-1
<== f-2     (preseljen (ime "Mika Mikic")
(adresa "Karadjordjeva 108"))
<== f-1     (osoba (ime "Mika Mikic") (adresa
"Nusiceva 56"))
==> f-3     (osoba (ime "Mika Mikic") (adresa
"Karadjordjeva 108"))
CLIPS>
```

Zadatak 4

Definisati šablon student koji sadrži ime, broj indeksa, koliko ispita mu je ostalo do kraja studija.

Napisati pravilo koje proverava da li je student završio fakultet i ako jeste, ukloniti njegove podatke.