SYSTEMY EKSPERTOWE

ZAJĘCIA 3-4

DOKUMENTACJA: http://www.jessrules.com/jess/docs/71/rules.html

Tworzenie reguł

1. Reguła składa się z dwóch części, oddzielonych symbolem =>

2. Pierwsza część (przesłanka) opisuje warunki jakie muszą być spełnione, aby reguła została uruchomiona. Zwykle są w niej opisane fakty z jakich korzysta reguła i warunki jakie są na te fakty nałożone.

3. Druga część (konkluzja) opisuje działanie jakie będzie zostać podjęte, jeśli reguła jest uruchamiana , np. wypisanie komunikatu, stworzenie, usunięcie lub zmiana faktu.

4. W drugiej części można również stworzyć nowe zmienne i/lub przypisać im wartości przy użyciu funkcji `bind`. Funkcji tej nie można użyć w pierwszej części .

5. Reguła może mieć priorytet, opisany przez własność `salience`.

Dopasowywanie faktów

1. Dany fragment sprawdza typ faktu.

2. Reguła jest sprawdzana i wywoływana dla każdej kombinacji faktów zgodnych z przedstawionymi wzorcami.

3. Sprawdzany jest typ (wzorzec) faktu i dodatkowe warunki podane przez użytkownika.

4. Jeśli chcemy pobrać wskazanie na fakt, używamy operatora <-
   `?f <- (wzorzec …).`

Warunki dotyczące wartości slotów

1. Aby warunek opisujący fakt był prawdziwy, muszą być spełnione wszystkie warunki dotyczące jego slotów.

2. Pobranie wartości slotu odbywa się poprzez napisanie w nawiasach nazwy slotu i zmiennej pod którą podstawiamy wartość: `(slot ?zmienna)`.

3. Warunki można zapisać na trzy sposoby:
   - Po pobraniu wszystkich faktów i ich wartości użycie funkcji `test`.
   - Wyrażenie otoczone nawiasami klamrowymi umieszczone wewnątrz opisu faktu, np. `(person {age < 3})`. W wyrażeniu tym można posługiwać się nazwami slotów i istniejących zmiennych. Nie tworzy ono żadnych nowych obiektów ani zmiennych.
   - Łącznie z pobieraniem wartości, umieszczając wyrażenie w notacji uproszczonej tuż po nazwie zmiennej.

4. Jeśli chcemy upewnić się, że wartość istnieje nie pobierając jej, zamiast `?zmienna` używamy tylko `?`.

Dopasowywanie multislotów

1. Multislot przechowuje listę wartości. Jest ona uporządkowana.

2. Jeśli chcemy pobrać nie pojedynczą wartość a listę, wyrażenie należy poprzedzić znakiem dolara `$`. Dotyczy to również pojedynczego znaku zapytania; tu zamiast `?` piszemy `$?`

3. Aby sprawdzić czy wartość należy do listy należy użyć funkcji `(member$ wartosc lista)`.

Operacje na faktach

1. Aby wykonać operacje na faktach w drugiej części reguły należy znać tożsamość faktu. Pobieramy ją w pierwszej części reguły za pomocą operatora <-.

2. Nowy fakt tworzymy za pomocą funkcji `assert`, której jako parametr podajemy listę składającą się z nazwy ramy i listy par slot-wartość.

3. Fakty usuwane są za pomocą funkcji `retract`, której jako parametr podajemy wskazania na fakt.

4. Aby zmienić wartości w slotach należy użyć funkcji `modify`, której pierwszym argumentem jest wskazanie na fakt, a jako kolejne argumenty występują pary slot-nowa wartość .

Uruchomienie systemu

1. Aby uruchomić system, należy użyć funkcji `run`.

2. System będzie przeglądał reguły i wywoływał te, które mogą być wywołane.

3. Działanie skończy się gdy nie będzie możliwości wywołania żadnej reguły.

4. Przed `run` zwykle wywoływana jest funkcja reset, aby powrócić do stanu początkowego usuwając tymczasowe fakty Dzięki temu każde uruchomienie będzie rozpoczynało od tego samego stanu, co powinno zapewnić powtarzalność obserwacji.

REGUŁY

1. Reguła wyznaczająca wagę zamówienia.

```
(
defrule regula_waga_zamowienia
      ?f <- (zamowienie (towar ?t) (ilosc ?i)(waga_zamowienia ?wz&nil))
      (towar (nazwa ?t) (waga_jednostkowa ?wj))
=>
(bind ?m (* ?i ?wj))
(modify ?f (waga_zamowienia ?m))
)
```

2. Reguła wypisująca wagę każdego zamówienia.

```
(
defrule regula_wypisz_wage
      ?f <- (zamowienie (numer_zamowienia ?n) (towar ?t) (ilosc ?i) (waga_zamowienia ?w))
=>
(printout t "Zamowienie numer " ?n " na towar " ?t " w ilosci " ?i " wazy " ?w crlf)
)
```

3. Reguła wypełniająca ramę Potencjalne.

```
(
defrule regula_wypelnij_potencjalne
      (zamowienie (numer_zamowienia ?n))
=>
(assert (potencjalne (numer_zamowienia ?n)))
(printout t "Realizowane jest zamowienie numer " ?n crlf)
)
```

4. Reguła wyznaczająca firmy spełniające ograniczenia wagowe.

```
(
defrule regula_spelnia_ograniczenia_wagowe
      ?w <- (potencjalne (numer_zamowienia ?nr) (wymagania_wagowe $?slot))
      (zamowienie (numer_zamowienia ?nr) (waga_zamowienia ?wz&~nil))
      (firma (maksymalna_waga ?mw) (nazwa ?n))
      (test ( and ( > ?mw ?wz) (not ( member$ ?n $?slot )) ) )
=>
(modify ?w (wymagania_wagowe $?slot ?n))
(printout t "Wagowe - dodano firme " ?n " dla zamowienia " ?nr crlf)
)
```

5. Reguła wyznaczająca firmy spełniające ograniczenia lokalizacji

```
(
defrule regula_spelnia_ograniczenia_lokalizacji
      ?w <- (potencjalne (numer_zamowienia ?nr) (wymagania_lokalizacji $?slot))
      (zamowienie (numer_zamowienia ?nr) (miasto_zrodlowe ?mz)
                  (miasto_docelowe ?md))
      (firma (oddzialy $?o) (nazwa ?n))
      (test (and (not (member$ ?n $?slot)) (member$ ?mz $?o) (member$ ?md $?o)))
=>
(modify ?w (wymagania_lokalizacji $?slot ?n))
(printout t "Lokalizacja - dodano firme " ?n " dla zamowienia " ?nr crlf)
)
```

6. Reguła wyznaczająca firmy, które są w stanie przewieźć towary zwykłe

```
(
defrule regula_spelnia_ograniczenia_specjalne_1
      ?w <- (potencjalne (numer_zamowienia ?nr) (wymagania_specjalne $?slot))
      (zamowienie (numer_zamowienia ?nr) (towar ?t&~nil))
      (towar_zwykly (nazwa ?t))
      (firma (nazwa ?n))
      (test (not (member$ ?n $?slot)))
=>
(modify ?w (wymagania_specjalne $?slot ?n))
(printout t "Towar zwykly – dodano firme " ?n " dla zamowienia " ?nr crlf)
)
```

7. Reguła wyznaczająca firmy, które są w stanie przewieźć towary kruche

```
(
defrule regula_spelnia_ograniczenia_specjalne_2
      ?w <- (potencjalne (numer_zamowienia ?nr) (wymagania_specjalne $?slot))
      (zamowienie (numer_zamowienia ?nr) (towar ?t&~nil))
      (towar_kruchy (nazwa ?t))
      (firma (nazwa ?n) (specjalne_traktowanie TAK))
      (test (not (member$ ?n $?slot)))
=>
(modify ?w (wymagania_specjalne $?slot ?n))
(printout t "Towar kruchy – dodano firme " ?n " dla zamowienia " ?nr crlf)
)
```

8. Reguła wyznaczająca firmy, które są w stanie przewieźć towary psujące się

```
(
defrule regula_spelnia_ograniczenia_specjalne_3
      ?w <- (potencjalne (numer_zamowienia ?nr) (wymagania_specjalne $?slot))
      (zamowienie (numer_zamowienia ?nr) (towar ?t&~nil))
      (towar_psujacy_sie (nazwa ?t))
      (firma (nazwa ?n) (chlodzenie TAK))
      (test (not (member$ ?n $?slot)))
=>
(modify ?w (wymagania_specjalne $?slot ?n))
(printout t "Towar psujacy sie - dodano firme " ?n " dla zamowienia " ?nr crlf)
)
```

9. Reguła wyznaczająca firmy, które spełniają ograniczenia czasowe

```
(
defrule regula_spelnia_ograniczenia_czasowe
      ?w <- (potencjalne (numer_zamowienia ?nr) (wymagania_czasowe $?slot))
      (zamowienie (numer_zamowienia ?nr) (ekspres ?e))
      (firma (typ_transportu $?tt) (nazwa ?n))
      (test (and (not (member$ ?n $?slot))
            (or (and (= ?e TAK) (member$ "samolot" $?tt))
                (= ?e NIE))))
=>
(modify ?w (wymagania_czasowe $?slot ?n))
(printout t "Czasowe - dodano firme " ?n " dla zamowienia " ?nr crlf)
)
```

10. Reguła wyznaczająca firmy spełniające wszystkie wymagania

```
(
defrule regula_spelnia_wymagania_wszystkie
      ?w <- (potencjalne (wymagania_wagowe $? ?n $?)
                  (wymagania_lokalizacji $?l)
                  (wymagania_specjalne $?s)
                  (wymagania_czasowe $?c)
                  (wymagania_wszystkie $?slot))
(test (and
            (not (member$ ?n $?slot))
            (member$ ?n $?l)
            (member$ ?n $?s)
            (member$ ?n $?c)
            ))
=>
(modify ?w (wymagania_wszystkie $?slot ?n))
(printout t "Firma " ?n " spelnia wszystkie wymagania" crlf)
)
```

Przed wprowadzeniem poniższych reguł, należy zdefiniować dodatkową ramę:

```
(
deftemplate firmy
(declare (ordered TRUE))
)
```

11. Reguła wyznaczająca koszt zamówienia

```
(
defrule regula_wylicz_koszt_zamowienia
      (potencjalne (numer_zamowienia ?nr) (wymagania_wszystkie $? ?n $?))
      (zamowienie (numer_zamowienia ?nr) (waga_zamowienia ?wz) (odleglosc ?o))
      (firma (nazwa ?n) (koszt_za_km ?km) (koszt_zaladunku ?z))
=>
(bind ?koszt_realizacji (+ (* ?z ?wz) (* ?km ?o)))
(assert (firmy ?nr ?n ?koszt_realizacji))
(printout t "Koszt realizacji firmy " ?n " to " ?koszt_realizacji crlf)
)
```

12. Reguła usuwająca asercje z za dużym kosztem realizacji

```
(
defrule regula_koszt_1
      ?p1 <- (firmy ?nr ?n1 ?c1)
      ?p2 <- (firmy ?nr ?n2 ?c2)
      (test (and (neq ?n1 ?n2) (>= ?c1 ?c2)))
=>
(retract ?p1)
)
```

13. Reguła wstawiająca do zamówienia koszt jego realizacji oraz firmy spełniające wszystkie wymagania

```
(
defrule regula_koniec
(declare (salience -1))
      ?p <- (firmy ?nr ?n ?c)
      ?ff <- (zamowienie (numer_zamowienia ?nr) (koszt_realizacji ?&nil) (firma $?slot))
=>
(retract ?p)
(modify ?ff (koszt_realizacji ?c) (firma $?slot ?n))
)
```

ABY ZMIENIĆ ZAWARTOŚĆ FAKTU

```
(reset)

(facts)

(modify nr_faktu (slot nowa_wartosc))

(facts)
```