

Propozycje rozszerzenia JoSKIPI

Michał Marcińczuk

28.09.2010 r.

1. Dopasowanie sekwencji tokenów i indeksowanie tablicą

```
match( variable, <lista_warunków>)
```

Operator dopasowuje sekwencję tokenów spełniających listę warunków. Z dopasowanych elementów tworzony jest indeks dopasowań, który zapisany jest pod zmienną `variable`. Operator **match** działa podobnie do operatora **accept**, tj. po każdym spełnionym warunku zwiększany jest indeks `$_`, ale z tą różnicą, że dodatkowo tworzony jest indeks dopasowań. Jeżeli sekwencja jest dopasowana operator zwraca wartość `true`, a w przeciwnym wypadku `false`.

Np.

```
match($m,  
  inter(base[$_], { 'spółka', 'firma' } ),  
  regex(orth[$_], '\p{Lu}\p{Ll}*'),  
  inter(orth[$_], {S.A.})  
)
```

Wykonane na zdaniu:

```
Spółka Kompi S.A. podpisała umowę z firmą Wiwi S.A.
```

dopasuje się w dwóch miejscach, indeksując:

```
$m[0] = "Spółka"  
$m[1] = "Kompi"  
$m[2] = "S.A."
```

i

```
$m[0] = "firma"  
$m[1] = "Wiwi"  
$m[2] = "S.A."
```

Jeżeli jednym z warunków jest operator `and` lub `or`, w którym występuje kilkakrotne odwołanie do zmiennej `$_`, to zmienna ta jest inkrementowana tylko raz, np.

```
match($m,  
  and(  
    inter(base[$_], { 'spółka', 'firma' } ), // $_=0  
    regex(orth[$_], '\p{Lu}\p{Ll}*') // $_=0  
  ),  
  regex(orth[$_], '\p{Lu}\p{Ll}*'), // $_=+1  
  or(  
    inter(orth[$_], {S.A.}), // $_=+2  
    inter(orth[$_], {Sp.}) // $_=+2  
  )  
)
```

Dodatkowe uwagi dotyczące zachowania operatora **mark** dla pustych dopasowań znajdują się w pkt. 8 i 9.

2. Dodawanie anotacji

```
mark( from, to, annotation_name)
```

Dodaje anotację `annotation_name` rozciągającą się od indeksu `from` do `to` włącznie.

Np.

```
match($m,  
    intern(base[$_], { 'spółka', 'firma' } ),  
    regex(orth[$_], '\p{Lu}\p{Ll}*'),  
    intern(orth[$_], {S.A.})  
),  
mark($match[1], $match[2], 'COMPANY_NAM')
```

wykonane na zdaniu:

```
Spółka Kompi S.A. podpisała umowę z firmą Wiwi S.A.
```

zwróci:

```
Spółka <COMPANY_NAM>Kompi S.A.<COMPANY_NAM> podpisała umowę z firmą  
<COMPANY_NAM>Wiwi S.A.<COMPANY_NAM>
```

3. Dopasowanie anotacji

```
is( position, annotation_name)
```

Dopasowuje ciąg tokenów zaczynający się od pozycji `position` będący w całości oznaczony jako anotacja `annotation_name`.

Np.

```
match($m,  
    is($_, "PERSON")  
)
```

Dla tekstu:

```
<PERSON>Jan Nowak<PERSON> spotkał się z <PERSON>Markiem Kowalskim<PERSON>.
```

operator dopasuje się w dwa razy:

```
$m[0] = "Jan Nowak"
```

i

```
$m[0] = "Markiem Kowalskim"
```

Możliwość reanotacji:

Np.

```
match($m,  
  is($_, "PERSON"),  
  is($_, "LAST_NAM")  
) ,  
mark($m[0], $m[1], "PERSON")
```

Dla tekstu:

<PERSON>Pan Jan</PERSON> <LAST_NAM>Włoszyński</LAST_NAM> został wybrany na stanowisko premiera.

zwróci:

<PERSON>Pan Jan <LAST_NAM>Włoszyński</LAST_NAM></PERSON> został wybrany na stanowisko premiera.

4.Usuwanie anotacji

```
unmark( index, annotation_name)
```

Usuwa anotację o nazwie `annotation_name` dopasowaną do indeksu `index`.

Np.

```
match($m,  
  text($_, "ul. "),  
  is($_, "PERSON_LAST_NAM")  
) ,  
unmark($m[1], "PERSON_LAST_NAM")
```

Dla tekstu:

Budynek znajduje się przy ul. <PERSON_LAST_NAM>Mickiewicza</PERSON_LAST_NAM>.

zwróci:

Budynek znajduje się przy ul. Mickiewicza.

5.Sprawdzenie, czy anotacja jest częścią innej anotacji

```
part( index, annotation_name)
```

Np. zaznacz wszystkie nazwiska, które są częścią nazwy firmy:

```
match($m,  
  and(  
    is($_, "PERSON_LAST_NAM")  
    ,part($_, "COMPANY")  
  )  
)
```

6. Dopasowanie tekstu bez uwzględniania podziału na tokeny

```
text( position, tekst)
```

Operator dopasowuje sekwencję tokenów, dla których złączenie orth-ów równe jest zadanej wartości `tekst`. Operator będzie potrzebny, kiedy nie ma pewności, jak dany tekst może zostać podzielony przez tokenizator.

Np.

```
text($_, 'Sp. z o.o.')
```

zostanie dopasowany do "[Sp.] [z] [o.o.]" jak i "[Sp][.] [z] [o.][o.]", gdzie [.] oznaczają podział na tokeny.

Przykład połączenia z operatorem `match`:

```
match($m,  
  intern(base[$_], { 'spółka', 'firma' } ),  
  regex(orth[$_], '\p{Lu}\p{Ll}*'),  
  tekst($_, 'Sp. z o.o.')  
)
```

dla zdania:

```
Spółka XYZ Sp. z o.o.
```

wynikiem dopasowania będzie:

```
$m[0] = "Spółka"  
$m[1] = "XYZ"  
$m[2] = "Sp. z o.o."
```

7. Zwielokrotnione dopasowanie

```
repeat( position, condition)
```

Dopasowuje sekwencję tokenów począwszy od pozycji `position` spełniających określone warunku. Operator dopasowuje najdłuższą możliwą sekwencję.

Np.

```
match($m,  
  intern(base[$_], { 'pan', 'pani' } ),  
  repeat($_, regex(orth[$_], '\p{Lu}\p{Ll}*'))  
)
```

dla zdania:

```
Pan Maciej Nowak pracuje w Intercom.
```

dopasuje:

```
$m[0] = "Pan"  
$m[1] = "Maciej Nowak"
```

Warianty:

`repeat(position, condition)` - domyślnie dopasuje najdłuższą sekwencję

`repeat(position, condition, number_of_repeats)` - podana liczba powtórzeń

`repeat(position, condition, min_repeats, max_repeats)` - liczba powtórzeń określona przedziałem

Możliwość zagnieżdżenia:

Np.

```
match($m,
  is($_, 'PERSON') ,
  repeat($_, // ", Upper*"
    text($_, ','),
    repeat($_,
      regex(orth[$_], '\p{Lu}\p{Ll}*' )
    )
  )
)
```

Przykład działania dla tekstu:

"W spotkaniu uczestniczył <PERSON>Pan Jarek Nowak</PERSON>, Marek Kowalski, Iwona Wójcik i Inga Wieczorek."

dopasuje:

```
$m[0] = "Pan Jarek Nowak"
$m[1] = ", Marek Kowalski, Iwona Wójcik"
```

8. Iteracja po powtórzeniach

```
for( group, var, action )
```

Dla ostatniego przykładu z pkt. 6, drugi element składa się z powtórzonych dopasowań, tj. , 'Marek Kowalski' i ', Iwona Wójcik' na pierwszym poziomie zagnieżdżenia. Powinna istnieć możliwość iteracji po tych elementach na potrzeby zastosowania operatora mark.

Np.

```
match($m,
  is($_, 'PERSON') ,
  repeat($_,
    text($_, ','),
    repeat($_,
      regex(orth[$_], '\p{Lu}\p{Ll}*' )
    )
  )
),
for($m[1], $v, mark($v[1], 'PERSON'))
```

Dla zdania:

```
"W spotkaniu uczestniczył <PERSON>Pan Jarek Nowak</PERSON>, Marek Kowalski, Iwona Wójcik i Inga Wieczorek."
```

zwróci wynik:

```
"W spotkaniu uczestniczył <PERSON>Pan Jarek Nowak</PERSON>, <PERSON>Marek Kowalski</PERSON>, <PERSON>Iwona Wójcik</PERSON> i Inga Wieczorek."
```

9. Opcjonalne dopasowanie

```
optional( condition )
```

Uproszczony zapis dla operatora:

```
repeat( condition, 0, 1)
```

Np.

```
match( $m,  
  inter( base[$_], "firma"),  
  regex( orth[$_], "\p{Lu}\p{Ll}*" ),  
  optional( text($_, "S.A.") )  
),  
mark( $m[1], $m[2], "COMPANY_NAM")
```

dla tekstu:

```
Firma Intext S.A. i firma Kontra podpisały umowę.
```

zwróci wynik:

```
Firma <COMPANY_NAM>Intext S.A.<COMPANY_NAM>. i firma <COMPANY_NAM>Kontra <COMPANY_NAM> podpisały umowę.
```

Pomimo, że w drugim dopasowaniu warunek opcjonalny nie został dopasowany, to odwołanie do indeksu `$m[2]` było poprawne. W tym przypadku operator `mark` pomija wszystkie puste indeksy z początku i końca zakresu.

10. Indeksowanie `match` z użyciem operatora `or` i różnej długości sekwencji

Rozpatrzmy następującą sytuację:

```

match( $m,
  or(
    accept(
      inter( base[$_], "przewodniczący"),
      inter( base[$_], "zarząd")
    ),
    accept(
      inter( base[$_], "prezes")
    )
  ),
  inter( base[$_], "pan"),
  repeat( regex( $_, "\p{Lu}\p{Ll}*" ) )
)

```

W powyższej regule, wewnątrz operatora **or** znajdują się dwie alternatywne sekwencje o różnych długościach. W tym przypadku, oczekujemy, że jeżeli zostanie spełniony pierwszy warunek **accept**, to indeks `$_` zostanie zwiększony o 2, w przypadku drugiego warunku o 1. Zatem, pomimo wcześniejszego założenia, że indeks nie jest zwiększany wielokrotnie wewnątrz operatora **or** tutaj jest to możliwe - dzięki operatorowi **accept**.

Dla tekstu:

Przewodniczący zarządu Pan Marek Nowak i prezes Pan Waldek Kruszyńska doszli do porozumienia w kwestii sprzedaży gruntu.

powstaną następujące dopasowania:

```

$m[0] = "Przewodniczący zarządu"
$m[1] = "Pan"
$m[2] = "Marek Nowak"

```

i

```

$m[0] = "prezes"
$m[1] = "Pan"
$m[2] = "Waldek Kruszyńska"

```

11. Odpytanie Słowności

Możliwość sprawdzenia, czy dane słowo jest synonimem i/lub hiponimem, np.

```

synonym($_, "umowa#n#1") // synonim
kind_of($_, "umowa#n#1") // synonim lub hiponim

```

12. Utworzenie relacji

Dodanie relacji między dowolnymi anotacjami (nawet znajdującymi się w różnych zdaniach), np.:

```

link($a[0], $b[0], "MEMBER_OF")

```

13. Usunięcie relacji

Usunięcie relacji między anotacjami, np.:

```
unlink($a[0], $b[0], "MEMBER_OF")
```

14. Łączenie operatora `llook`, `rlook` z `match`

Mechanizm do znajdowania sekwencji tokenów względem innej sekwencji tokenów, np:

```
match($m,  
    inter(base[$_], "podpisać"),  
    inter(base[$_], "umowa")  
),  
llook(  
    match($a,  
        is($_, "COMPANY")  
    )  
),  
rlook(  
    match($b,  
        is("COMPANY")  
    )  
)  
$e = mark($m[0], $m[1], "EVENT_AGREEMENT_SIGNUP"),  
link($e, $a[0], "AGREEMENT_SIDE"),  
link($e, $b[0], "AGREEMENT_SIDE")
```

15. Operator `exists`

Sprawdza, czy istnieje w badanym kontekście sekwencja o podanych warunkach.

TODO

16. Kontekst zdania i dokumentu

Możliwe powinno być określenie kontekstu działania operatora (zdanie, dokument, okno), w szczególności dla operatorów `rlook`, `llook`, `exists`.

TODO

Przykładowe reguły

Zaznacz wszystko co zaczyna się z dużych liter:

```
rule(  
  match(  
    regex("\p{Lu}{Ll}*" )  
  ),  
  match("UPPERCASE")  
)
```

Oznaczenie nazw miejscowości ze słowników:

```
rule(  
  match(  
    and(  
      inter(nmb, {"sg"}),  
      inter(flex, {"subst"}),  
      inter(base, {"miasto", "miasteczko", "miejscowość"})  
    ),  
    and(  
      is("CITY_GAZ_BASED"),  
      inter(cas, {"nom", cas[0]})  
    )  
  ),  
  mark(1, "GAZ_CITY")  
)
```

Oznaczenie nazw miejscowości po słowach kluczowych:

```
rule(  
  match(  
    and(  
      inter(nmb, {"sg"}),  
      inter(flex, {"subst"}),  
      inter(base, {"miasto", "miasteczko", "miejscowość"})  
    ),  
    repeat( is("UPPERCASE") )  
  ),  
  mark(1, "KEY_CITY")  
)
```

Oznaczenie nazw regionów na podstawie słownika:

```
rule(  
  match(  
    and(  
      inter(nmb, {"sg"}),  
      inter(flex, {"subst"}),  
      inter(base, {"region"})  
    ),  
    and(  
      is("REGION_GAZ_BASED"),  
      inter(cas, {"nom", cas[0]})  
    )  
  ),  
  mark(1, "GAZ_REGION")  
)
```

Analogicznie oznaczenie REGION dla słów „gmina”, „powiat”, „województwo”.

Oznaczenie dla nazw zaczynających się od słowa Morze, Ocean, Jezioro z dużych liter.

```
rule(  
  match(  
    and(  
      is("UPPERCASE"),  
      inter(flex, {"subst"}),  
      is("INT_PROOF_GEOG_NAME")  
    ),  
    inter(flex, {"adj"})  
  ),  
  mark(1, "GEOG_NAME")  
)
```

Oznaczenie skrótów ulic, placów itp.

```
rule(  
  match(  
    inter(base, {"ul", "pl", "al"}),  
    optional( text(".") )  
  ),  
  match("URABN_ABBREV")  
)
```