Propozycje rozszerzenia JoSKIPI

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 indekst `$_` , ale z tą różnicą, że dodatkowo tworzony jest indeks dopasowań. Jeżeli sekwencja jest dopasowana operator zwracana 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] = firmą
$m[1] = Wiwi
$m[2] = S.A.
```

Jeżeli jednym z warunków jest operator `and` lub `or`, w którym występuje kilkukrotne 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 dopasować 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. 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."
```

6. 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 zagnieżdżania:

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"
```

7. 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."*

8. 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.

9. 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 Kruszynka 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 Kruszynka"
```