Krótki opis języka JOSKIPI

Adam Radziszewski

2 września 2008

Streszczenie

Język JOSKIPI służy do zapisu wyrażeń funkcyjnych wartościowanych na wybranym zdaniu z korpusu. Umożliwia też zapis reguł ujednoznaczniających oraz wzorców dla tagera. Niniejszy dokument stanowi nieformalny przegląd operatorów JOSKIPI. Przedmiotem opisu jest język, nie zaś jego implementacja ani sposób użycia biblioteki.

1 Pochodzenie i zastosowania

Język JOSKIPI został stworzony na potrzeby tagera TaKIPI [Pia06, Pia07]. Operatory języka odwołują się bezpośrednio do systemu znaczników morfosyntaktycznych korpusu IPI PAN [Prz04]. Tager wraz z odsyłaczami do artykułów jest dostępny pod poniższym adresem:

`http://www.plwordnet.pwr.wroc.pl/g419/tagger/`

Obecnie JOSKIPI ma następujące zastosowania:

- wyszukiwanie tokenów spełaniających podane ograniczenia,
- opis reguł ujednoznaczniających,
- opis wzorców dla tagera.

Wyszukiwanie tokenów może zarówno wspomagać korzystanie z korpusu (choć brak narzędzia z graficznym interfejsem użytkownika), jak i pisanie programów przetwarzających język polski (ujednoznacznianie, wydobywanie relacji składniowych i semantycznych).

2 Wyrażenia

W języku JOSKIPI można zapisać:

- wyrażenia funkcyjne i ograniczenia,
- wyrażenia ustawiające wartości zmiennych,
- reguły ujednoznaczniające,
- definicje wzorców dla tagera.

2.1 Pozycje

Działanie wyrażeń JOSKIPI ograniczone jest do granic zdania rozumianego jako ciąg tokenów. Zdanie przetwarzane jest token po tokenie. Dostęp do tokenów ze zdania realizowany jest przez *pozycje*. Przetwarzany obecnie token wskazywany jest przez *bieżącą pozycję*.

W języku JOSKIPI pozycję określa się w jeden z poniższych sposobów:

- 0 oznacza bieżącą; w przypadku ujednoznaczniania określa aktualnie ujednoznaczniany token, w przypadku wyszukiwaniu tokenów jest to token rozważany — być może spełniający opisany warunek,
- n, gdzie $n \in \mathbf{Z}$, oznacza pozycję względem bieżącej,
- `$Zmienna` oznacza pozycję, na którą wskazuje zmienna,
- `$+`n`Zmienna`, `$-`n`Zmienna` oznacza pozycję względem zmiennej,
- `begin`, `end` to początek i koniec zdania,
- `current` oznacza pozycję bieżącą (nie wiem czy jest sens tego używać, ale przyjmie to parser),
- `nowhere` oznacza pozycję nieustawioną (jak wyżej).

2.2 Wyrażenia funkcyjne

Wyrażenia funkcyjne zwracają *zbiory wartości*. W zależności od wyrażenia, mogą być to wartości z dziedziny atrybutów (pochodzące wprost z systemu znaczników morfosyntaktycznych, np. `acc` — biernik, wartość przypadka), wartości z dziedziny napisów (formy napotkane wyrazów oraz ich formy hasłowe). Z punktu widzenia języka mogą to być również wartości logiczne; w tym opisie jednak wyrażenia funkcyjne zwracające wartości logiczne potraktowane zostały osobno — są to ograniczenia (testy, predykaty).

Wyrażeniami funkcyjnymi są operatory proste, `catflt` (operator filtrujący), `agrflt` (filtr uzgodnienia) oraz operatory oparte o prawdopodobieństwo (ich implementacja nie została przetestowana; `dispos` i `discat`).

W implementacji JOSKIPI można znaleźć jeszcze operator `fixsimp`, który wg opisu miał obcinać n znaków z początku lub końca napisu; nie wiem jednak czy sama implementacja jest skończona i nie jest on obsługiwany przez parser.

2.2.1 Operatory proste

Operatory proste mają postać `oper[poz]`, gdzie `poz` to definicja pozycji.

- Operatory zwracające napisy: `orth` (zwraca formę napotkaną) oraz `base` (zwraca zbiór możliwych lematów — form hasłowych tokenu). Napisy zwracane są w postaci niezmienionej — mogą zawierać wielkie i małe litery oraz różne inne znaki; lematy zwyczajowo nie zawierają wielkich liter, lecz jest to kwestia nie tyle JOSKIPI, co analizatora morfologicznego lub korpusu.
- Operator zwracający zbiór klas fleksyjnych tokenu: `flex`.
- Operatory zwracające zbiory wartości danego atrybutu z zestawu znaczników: `nmb` (liczba), `cas` (przypadek), `gnd` (rodzaj), `per` (osoba), `deg` (stopień), `asp` (aspekt), `acm` (akomodacyjność), `acn` (akcentowość), `ppr` (poprzyimkowość), `agg` (aglutynacja), `vcl` (wokaliczność), `tnt` (typ literału liczbowego, dotyczy tylko rozszerzonego tagsetu)

Przykładowo, `flex[1]` zwraca zbiór możliwych klas fleksyjnych tokenu następującego po bieżącym.

2.2.2 Operator filtrujący

Operator ma następującą składnię: `catflt(poz,jakie,filtr)`, gdzie `poz` to definicja pozycji, natomiast `jakie` i `filtr` to zbiory atrybutów lub konkretnych wartości. Zbiór `jakie` wybiera tagi, które należy w ogóle brać pod uwagę; brane są wszystkie te:

- których klasa fleksyjna jest podana wprost w tym zbiorze,
- które mają określoną wartość któregokolwiek z atrybutów podanych w tym zbiorze,
- które mają wartość któregokolwiek z atrybutów należącą do tego zbioru.

Natomiast `filtr` to maska określająca, które atrybuty lub ich wartości nas interesują (wartości tagów zostaną rzutowane na tęże maskę).

Przykładowo, `catflt(0,{subst,adj},{gnd,gen})` zwróci zbiór wartości rodzaju i przypadka (przypadka tylko jeśli przypadkiem będzie `gen`) jedynie z tych leksemów tokenu na pozycji bieżącej (stąd 0), które należą do klasy fleksyjnej `subst` lub `adj`.

2.2.3 Filtr uzgodnienia

Operator ma następującą składnię: `agrflt(poz1,poz2,atr_uzg,bity,filtr)`, gdzie `poz1` i `poz2` to pozycja początkowa i końcowa, `atr_uzg` to atrybuty (lub konkretne wartości), na których ma zajść uzgodnienie, `filtr` to zbiory atrybutów lub konkretnych wartości, które określają co ma zostać zwrócone (maska, podobnie jak w `catflt`).

Operator sprawdza, czy zachodzi słabe uzgodnienie (patrz pkt. 2.3.3). Jeśli nie, zwraca zbiór pusty. Jeśli zachodzi, wykreślane są wszystkie tagi niespełniające uzgodnienia, po czym zwracana jest wartość atrybutów tagów pozostałych przefiltrowanych przez `filtr`. **Uwaga:** w obecnej implementacji `filtr` musi być podzbiorem `atr_uzg`, w przeciwnym razie operator zwróci zbiór pusty.

Przykładowo, w frazie *swoją*`/acc.inst` *moc*`/acc.nom` *pobudzającą*`/acc.inst` operator ten pozwoliłby na wybranie jedynego przypadka spełniającego uzgodnienie — biernika. Operator taki mógłby mieć postać `agrflt(-1,1,{cas,nmb,gnd},3,{cas})` (zakładając, że jesteśmy na tokenie *moc*).

2.2.4 Operatory oparte o prawdopodobieństwo

Operatory te zostały kiedyś wprowadzone w ramach eksperymentu, ich implementacja nie została przetestowana. Ich użycie może mieć sens w przypadku tagera (tager przypisuje prawdopodobieństwa; w przypadku czytania korpusu lub użycia analizatora informacji tej brak lub jest szczątkowa). Operatory wybierają pierwszy tag o najwyższym prawdopodobieństwie. Operator `dispos` zwraca klasę fleksyjną takiego taga, natomiast `discat` zwraca wartości jego (pozostałych) atrybutów. Nie ma możliwości filtrowania tego zbioru.

2.3 Ograniczenia

Ograniczenia należy traktować jako predykaty, czyli funkcje zwracające wartości logiczne. W zależności od kontekstu ich użycia, mogą służyć realizacji różnych celów: wyszukiwania z ograniczeniami, uwarunkowania wywołania innego operatora, czy po prostu pobrania wartości logicznej.

2.3.1 Operatory logiczne

Dostępne są trzy operatory: `and`, `or`, `not` o swej zwykłej semantyce. Operatory są przedrostkowe i mogą zawierać dowolną liczbę argumentów. Wyrażenie `not(a1, a2, ..., an)` jest równoważne wyrażeniu `not(or(a1, a2, ..., an))`.

2.3.2 Kwantifikatory

Kwantyfikatory działają na zakresie pozycji. Są dwa: `only` (odpowiednik kwantyfikatora uniwersalnego) oraz `atleast` (kwantyfikator szczegółowy z licznością). Operatory te mają następującą składnię:

```
only(poz1,poz2,$P,ogr)
```

```
atleast(poz1,poz2,$P,ogr,n)
```

Wyrażenie `only` jest spełnione, jeśli dla każdej wartości zmiennej `$P` z zakresu `poz1`–`poz2` ograniczenie `ogr` jest spełnione. Wyrażenie `atleast` wymaga, by ograniczenie było spełnione co najmniej `n` razy.

2.3.3 Uzgodnienia

Istnieją trzy rodzaje uzgodnienia:

- między dwiema pozycjami — `agrpp(poz1,poz2,atr_uzg,n)`,
- ciągłe między dwiema pozycjami — `agr(poz1,poz2,atr_uzg,n)`,
- słabe ciągłe między dwiema pozycjami `wagr(poz1,poz2,atr_uzg,n)`.

Zbiór `atr_uzg` określa atrybuty na których zachodzi uzgodnienie, ewentualnie ich konkretne wartości. Uzgodnienie `agrpp` zachodzi jeśli dwie wskazane pozycje zawierają co najmniej n określonych w zbiorze atrybutów i mają te same wartości tych atrybutów.

Uzgodnienie ciągłe `agr` wymaga, by cały przedział między wskazanymi pozycjami zawierał tokeny, każdy z których ma przynajmniej jeden tag z wartościami wszystkich z podanych atrybutów oraz wartości tych atrybutów są identyczne.

Uzgodnienie ciągłe słabe łagodzi ten warunek i pozwala na pozycje zawierające tagi niedospecyfikowane pod względem atrybutów, w szczególności mogą być to tagi bez żadnego z atrybutów (np. `qub`). Wymagane jest jednak, by pierwszy i ostatni token rozpatrywanego przedziału uzgadniały się na wartościach w pełni dospecyfikowanych tagów — tj. jeśli pośród tagów pierwszego lub ostatniego tokenu będzie np. `qub`, i tak nie będzie on brany pod uwagę. Zostało to wprowadzony, nie uniknąć uzgodnień między słowem a znakiem interpunkcyjnym. W przypadku pozostałych tokenów obecność tego typu tagu wystarczy, by dany tag spełniał ograniczenie.

Przykładowo, fraza *moc nie swoją*:

-1 *moc* `subst:sg:nom:f`, `subst:sg:acc:f`

0 *nie* `qub`, `ppron3:sg:acc:n`, `ppron3:sg:acc:n:ter`, `ppron3:pl:acc:m2.m3.f.n:ter`,

1 *swoją* `adj:sg:acc:f:pos`, `adj:sg:inst:f:pos`

nie zostanie uzgodniona przez `agr(-1,1,{cas,gnd,num},3)`, ponieważ te interpretacje słowa *nie*, które zawierają wszystkie atrybuty, nie zgadzają się z wartościami słów pozostałych. Zachodzi natomiast `wagr(-1,1,{cas,gnd,num},3)` (słowo *nie* ma również interpretację `qub`, która nie jest niezgodna) oraz `agrpp(-1,1,{cas,gnd,num},3)` (patrzymy tylko na słowo *moc* oraz *swoją*).

Uwaga: w implementacji operatora `agr` n nie jest obsłużone i przekazana wartość nie ma znaczenia.

2.4 Wyrażenia ustawiające wartości zmiennych

Istnieją dwa specjalne operatory, których działanie wywołuje skutki uboczne: ustawienie wartości zmiennej. Poza tym, są one ograniczeniami i można je łączyć z innymi ograniczeniami poprzez operatory logiczne. Operatory te wykonują poszukiwanie pozycji spełniającej podane ograniczenie. Operator zwraca wartość prawdy, jeśli pozycję udało się znaleźć, natomiast zmienna zostaje ustawiona na tę właśnie pozycję.

Składnia wygląda następująco:

```
llook(poz1,poz2,$Zmienna,ogr)

rlook(poz1,poz2,$Zmienna,ogr)
```

Operator `llook` wymaga, by `poz1` $\geqslant$ `poz2`, operator `rlook` — odwrotnie (w przeciwnym wypadku zwróci fałsz).

Przykładowo, poniższe wyrażenie próbuje ustawić wartość zmiennej `$Subst` na najbliższy rzeczownik lub gerundium z prawej strony.

```
rlook(1,end,$Subst,
      in(flex[$Subst],{ger,subst})
)
```

2.5 Reguły ujednoznaczniające

2.6 Definicje wzorców

Definicje wzorców określają atrybuty przypisane klasom niejednoznaczności. Tager dla danej klasy oblicza wartości operatorów wchodzących w skład wzorca i przekazuje je klasyfikatorowi.

Literatura

[Pia06] M. Piasecki. Handmade and Automatic Rules for Polish Tagger. 2006.

[Pia07] Maciej Piasecki. Polish tagger TaKIPI: Rule based construction and optimisation. *Task Quarterly*, 11(1–2):151–167, 2007.

[Prz04] Adam Przepiórkowski. *Korpus IPI PAN. Wersja wstępna*. Instytut Podstaw Informatyki, Polska Akademia Nauk, Warszawa, 2004.

Skorowidz