

## 1.1 Wprowadzenie

```

graph TD
    s[s] --- syngrNG[syngr:NG]
    s --- synw1[synw]
    s --- synw2[synw]
    s --- syngrPrepNG[syngr:PrepNG]
    s --- namedOrgName[named:orgName:]
    s --- synw3[synw]

    syngrNG -- head:semh --> synw4[synw]
    syngrNG -- head:synh --> synw4
    synw4 --> Radni[Radni]
    Radni --> sub1[sub]

    synw1 --> wypowiedzieli[wypowiedzieli]
    wypowiedzieli --> sub2[sub]

    synw2 --> sie[się]
    sie --> sub3[sub]

    synw2 --> takze[także]
    takze --> sub4[sub]

    syngrPrepNG -- head:synh --> synw5[synw]
    syngrPrepNG -- head:semh --> synw6[synw]
    syngrPrepNG -- nonhead --> synw7[synw]
    syngrPrepNG -- nonhead --> synw8[synw]
    syngrPrepNG -- nonhead --> synw9[synw]

    synw5 --> w[w]
    w --> sub5[sub]

    synw5 --> sprawie[sprawie]
    sprawie --> sub6[sub]

    synw6 --> koniecznosci[konieczności]
    koniecznosci --> sub7[sub]

    synw7 --> zmian[zmian]
    zmian --> sub8[sub]

    synw8 --> kursow[kursów]
    kursow --> sub9[sub]

    synw9 --> MZK[MZK]
    MZK --> sub10[sub]

    synw3 --> period[.]
    period --> sub11[sub]
  
```

Krawędzie są w istocie skierowane (od ojca do dziecka); na rysunku wszystkie krawędzie prowadzą z góry na dół.

- **wartości proste:** napis, liczba całkowita, boolean  
(np. wartością **orth** dla segmentu może być napis *Sejm*, a wartością **case** dla jego interpretacji — **nom**)<sup>1</sup>
- inne struktury atrybutowe
- **listy** oraz nieco ogólniejsze **listy wieloznaczne** (patrz p. 2.3)  
(np. wartością atrybutu **sons** dla grupy składniowej jest lista jej węzłów-dzieci)
- wartość **null**, oznaczająca **wynik nieokreślony**  
(np. wartość atrybutu **case** dla interpretacji segmentu przysłówkowego)

To samo stosuje się do wartości innych atrybutów skończenie-wartościowych (**number**, **gender**, **type** itp.).

- atrybuty węzłów i krawędzi:
  - **type** typ węzła/krawędzi (o wartościach zależnych od korpusu; patrz dodatki)
- atrybuty węzłów (odzwierciedlające strukturę grafu składni):
  - **sons** lista węzłów-dzieci (w porządku krawędziowym; patrz p. 3.3)
  - **span** lista segmentów-potomków (w porządku segmentowym; patrz p. 3.3)
- atrybuty list:
  - **size** liczba elementów listy (jako wartość liczbowa)
  - **join** dla list napisów: konkatenacja wszystkich elementów listy (np. `{poszedł, em}.join` zwraca wynik *poszedłem*)

## 1.3 Rodzaje krawędzi i ich znaczenie

Model danych rozróżnia dwa rodzaje krawędzi:

- >: krawędzie **zwyczajne**, reprezentujące:
  - dominację składniową w drzewach składnikowych rozbioru zdania (NKJP, Składnica, c-struktury LFG);
  - zależności atrybutowe między f-strukturami LFG; (nazwę atrybutu modelujemy jako wartość **type**; zależność nienazwaną oznaczającą “zawieranie” modelujemy poprzez **type=\_HAS**)
- ->: krawędzie **drugorzędne**, reprezentujące:
  - rzutowanie pomiędzy c-strukturami i f-strukturami LFG; (modelujemy jako krawędź drugorzędną z **type=phi**)
  - koreferencje.

Sposób osadzenia obu rodzajów krawędzi w języku jest analogiczny; rozróżnienie ma na celu m.in. zapewnienie sensownej interpretacji sztucznych atrybutów takich, jak **sons** albo **span**.

## 2 Zapytania o struktury atrybutowe

### 2.1 Specyfikacje napisów; zapytania proste

Najprostszym zapytaniem o segmenty (**zapytaniem prostym**) jest dowolna poprawna **specyfikacja napisu**.

#### PRZYKŁADY

- **chciałbym** trzy segmenty: `[chciał][by][m]`  
W ogólności — ciąg segmentów *nierozdzielonych spacjami*, tworzących łącznie podane słowo.
- **"(ws|u)trzyma.\*"** słowo (segment lub ciąg) pasujące do wyrażenia regularnego  
Wyrażenia regularne wymagają użycia cudzysłowów. Składnia wyrażeń regularnych — jak w Pythonie 3.
- **komisji/i** wystąpienie słowa *komisji* bez uwzględniania wielkości liter  
Flaga `/i` powoduje zignorowanie wielkości liter. Pełne omówienie flag znajduje się w p. 2.3

Podobnie jak w starym Poliqarpie, można takie zapytania łączyć w ciągi — wynikiem jest ciąg segmentów:

#### PRZYKŁAD

- Nie chciałbym                      ciąg słów: *[Nie] [chciał][by][m]*

## 2.2 Zapytania o wartości atrybutów

Ogólna specyfikacja struktury atrybutowej ma postać

[ *warunek* ]

gdzie *warunek* zbudowany jest ze specyfikacji atrybutów, mających najczęściej postać

*atrybut*=wartość    lub    *atrybut*!=wartość

Warunki można łączyć przy pomocy operatorów logicznych:

&&    ||    !    =>

#### PRZYKŁADY

- []                      opisuje dowolny węzeł (brak warunków oznacza **true**)
- [**type**=syng<sub>r</sub>:AdjG]  
  [syng<sub>r</sub>:AdjG]  
  [AdjG]                znajduje grupy przymiotnikowe w korpusie  
                            równoważne: atrybut **type** jest domyślny i można pominąć jego nazwę  
                            równoważne z [**type**="(.\*)?AdjG(:.\*)?"] (hierarchia typów)
- [**seg** && **orth**=Sejm]                znajduje wystąpienia segmentu *Sejm*

Warunek **type=seg** jest istotny, ponieważ grupy i słowa składniowe również mają atrybut **orth**.

- [**doc** && **taxonomy\_type**="#typ\_qmow"]                znajduje dokumenty quasi-mówione

Napis zawierający znak inny niż litera, cyfra, podkreślenie i dwukropek (bądź napis z samych cyfr) musi być zawarty w cudzysłowach.

Warunek **type=doc** nie jest semantycznie istotny, ale w obecnej implementacji przyspieszy wyszukiwanie.

Jeśli wartość atrybutu jest strukturą (lub udostępnia atrybuty sztuczne), można się do nich odwołać poprzez kropkę:

#### PRZYKŁADY

- [**sons.size**=5]                      węzeł z dokładnie 5 dziećmi (w sensie krawędziowym; patrz p. 3.3)
- [**sons.orth.join**=oto]              węzeł z jednym dzieckiem *oto* lub dwoma *o*, *to*

## 2.3 Zapytania o atrybuty morfosyntaktyczne

### 2.3.1 Wprowadzenie

Węzły anotowane morfosyntaktycznie (w NKJP: segmenty i słowa składniowe) udostępniają atrybut **msd**, zawierający listę wieloznaczną struktur nazywanych *interpretacjami morfosyntaktycznymi*.

**Interpretacja morfosyntaktyczna** jest po prostu strukturą atrybutową, przechowującą informacje morfosyntaktyczne w rozmaitych atrybutach (w przypadku korpusu NKJP są to m.in. **base**, **pos**, **case** itp.; szczegóły podano w dodatku A).

### 2.3.2 Listy wieloznaczne

**Lista wieloznaczna** jest listą, wzbogaconą o podział swoich elementów na **wybrane** oraz **niewybrane**. W kontekście listy interpretacji węzła, „wybraność” odnosi się do pozytywnego wyniku częściowej dezambiguacji (a więc jest rozumiana w takim sensie, w jakim tradycyjne Poliqarpowe operatory  $=$ ,  $==$  różnią się od  $\sim$ ,  $\sim\sim$ ). Każda lista wieloznaczna  $L$  ma następujące atrybuty:

- **sel** zwraca (zwykłą) listę wybranych elementów należących do  $L$
- **all** zwraca (zwykłą) listę wszystkich elementów należących do  $L$

#### PRZYKŁAD

- `[seg && msd.all.size = 3]` segment posiadający dokładnie 3 interpretacje

**Zalecany** sposobem specyfikowania list wieloznacznych — o ile to możliwe — jest użycie **operatorów kwantyfikujących**:

- $L \sim X$  na liście wieloznacznej  $L$  *istnieje* element  $X$
- $L \sim\sim X$  na liście wieloznacznej  $L$  *istnieją* elementy i *wszystkie* one są równe  $X$
- $L = X$  na liście wieloznacznej  $L$  *istnieje* *wybrany* element  $X$
- $L == X$  na liście wieloznacznej  $L$  *istnieją* *wybrane* elementy i *wszystkie* one są równe  $X$

$X$  może być dowolną poprawną specyfikacją wartości danego typu (np. wyrażeniem regularnym).

Wszystkie powyższe operatory dopuszczają negację:  $L \not\Box X$  oznacza, że nie zachodzi  $L \Box X$ .

Przykłady użycia tych operatorów będą podane w punkcie 2.3.4.

### 2.3.3 Atrybuty pochodne

Oprócz atrybutu **msd**, węzły anotowane morfosyntaktycznie (segmenty i słowa składniowe) udostępniają:

- **atrybuty pochodne** — mapowane z atrybutów interpretacji dla wygody użytkownika.  
W przypadku korpusu NKJP, są to:
  - **base** lista wieloznaczna wartości atrybutu **base** dla kolejnych interpretacji zawartych w **msd**
  - **pos** lista wieloznaczna wartości atrybutu **pos** dla kolejnych interpretacji zawartych w **msd**
  - ...
- **atrybuty pochodne złożone** — mapowane z list (ściślej: krotek) atrybutów interpretacji. Na przykład:
  - **{pos, case}** lista wieloznaczna *par* wartości atrybutów **pos** i **case**
  - ...

### 2.3.4 Przykłady

Proste przykłady z wykorzystaniem atrybutów pochodnych i operatorów kwantyfikujących:

- (a) [`seg && base=przyjąć`] segment, w którym istnieje wybrana interpretacja z formą bazową *przyjąć*
- (b) [`number~~sg`] węzeł, który posiada interpretacje i wszystkie one są w liczbie pojedynczej
- Dzięki warunkowi „posiada interpretacje” wynikami tego zapytania *nie* będą np. węzły typu `doc`.
- (c) [`pos~adj && case~nom`] węzeł z pewną interpretacją przymiotnikową oraz pewną mianownikową

Wykorzystanie złożonych atrybutów pochodnych:

- (d) [`{pos, case}~{adj, nom}`] węzeł z pewną interpretacją przymiotnikową i *równocześnie* mianownikową

Delikatnie mocniejszy warunek niż w (c); niewyraźalny w starym Poliqarpie!

Wykorzystanie nietrywialnych specyfikacji napisów:

- (e) [`base~"u.*ć" && gender="m."`] węzeł z pewną interpretacją z formą bazową postaci *u...ć* oraz pewną interpretacją w jednym z rodzajów męskich
- (f) [`base==trzymać/x`] węzeł, który posiada wybrane interpretacje i wszystkie one mają formę bazową *zawierającą* słowo *jechać*

Flaga `/x` oznacza specyfikację napisu *zawierającego* podany w zapytaniu.

Pełne omówienie flag znajduje się w p. 5.

**Uwaga.** Operatory kwantyfikujące i atrybuty pochodne jako takie nie rozszerzają siły wyrazu języka, jednak ich użycie jest zalecane ze względu na wydajność:

- (g) [`msd.all contains [pos=adj && case=nom]`] równoważna, ale mniej wydajna wersja zapytania (d)

Różnica wydajności wynika stąd, że operatory kwantyfikujące i atrybuty pochodne — poprzez zawężenie ekspresywności i bardziej bezpośrednie intencji zapytania — pozwalają na łatwą optymalizację zapytania. Być może różnica ta zostanie zniwelowana w przyszłych wersjach.

## 3 Zapytania wielowęzłowe

### 3.1 Wyrażenia regularne poziome

Podobnie jak w starym Poliqarpie, zapytanie może mieć postać wyrażenia regularnego zastosowanego do specyfikacji segmentów.

Dostępne są następujące konstrukcje wyrażeń regularnych:

- konkatenacja
- alternatywa: `|`
- kwantyfikatory podstawowe: `?`, `*`, `+`
- kwantyfikatory zliczające: `{m, n}`, `{m, }`, `{, n}`, `{n}`

## PRZYKŁADY

- Czy? to jest      znajduje ciągi *to jest* oraz *Czy to jest*
- (to|jest){2,}      znajduje ciągi *jest to*, *to jest to*, *to to* itp.
- [pos~adj]{3,5}      znajduje ciągi od trzech do pięciu potencjalnych przymiotników
- się []{2} lubi      znajduje frazę *się lubi* rozdzieloną przez dokładnie dwa segmenty, np. *się bardzo nie lubi*

Podobnie jak dowolne podzapytanie, wyrażenie regularne może zostać zmodyfikowane flagą:

## PRZYKŁAD

- (prezydium sejmu)/i      znajduje ciąg *Prezydium Sejmu*

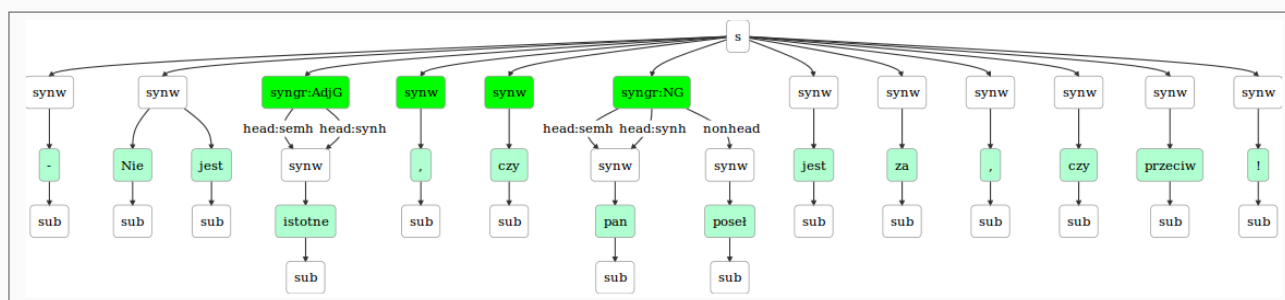
Flaga /i powoduje zignorowanie wielkości liter. Pełne omówienie flag znajduje się w p. 5.

Nawiasy są niezbędne — w zapytaniu *prezydium sejmu/i* flaga dotyczyłaby jedynie słowa *sejmu*.

Domyślnie wyrażenia regularne odnoszą się jedynie do segmentów. Jednak po włączeniu flagi /rha za dopasowanie zostanie uznany dowolny ciąg węzłów, których *zasięgi* (zbiory segmentów potomnych) kolejno przylegają do siebie:

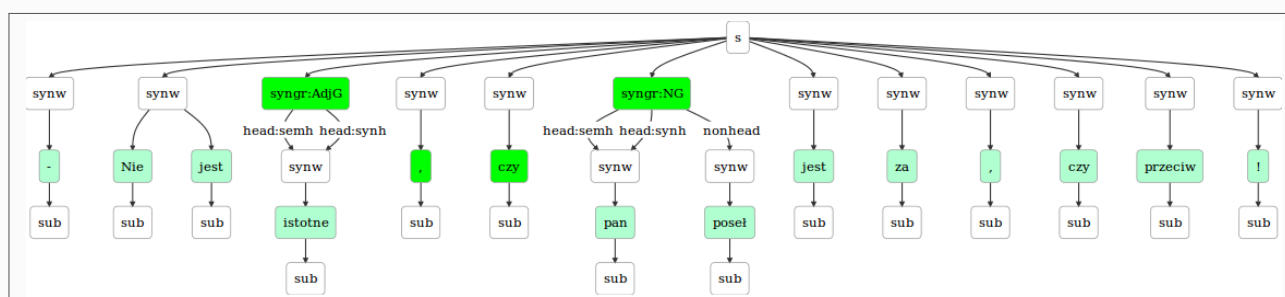
## PRZYKŁADY POZIOMYCH WYRAŻEŃ REGULARNYCH POWYŻEJ SEGMENTÓW (FLAGA /rha)

- (a) ([syngr:AdjG] [synw]{2} [syngr:NG])/rha      znajduje m.in. zdanie:



- *Nie jest istotne, czy pan poseł jest za, czy przeciw!*

- (b) ([syngr:AdjG] []{2} [syng:NG])/rha      wybierze w tym samym zdaniu inne węzły:



- *Nie jest istotne, czy pan poseł jest za, czy przeciw!*

W rzeczywistości powyższe zapytanie ma w przedstawionym zdaniu aż 9 wyników (w tym wynik z (a)): każdy z segmentów *czy* oraz *,* mógłby zostać zastąpiony przez swojego ojca (słowo składniowe) lub swoje jedyne dziecko (podsegment).

Aby uniknąć przytłaczania użytkownika wieloma tego typu wynikami, system domyślnie wybiera po jednym wyniku z każdego zdania wg pewnych heurystyk. (Stan ten można zmieniać przy pomocy odpowiednich flag; patrz p. 5). Do tych heurystyk zalicza się w szczególności dążenie do wybrania jak największej liczby segmentów.

## 3.2 Dominacja i wyrażenia regularne pionowe

### 3.2.1 Podstawowe zapytania

**Dominację bezpośrednią** (rodzic-dziecko) między węzłami oznaczamy symbolem  $>$  (lub  $->$ ).

- `[synw && pos=Noun] > [pos!~subst]`  
słowo składniowe z wybraną interpretacją rzeczownikową posiadające dziecko, dla którego nie istnieje żadna interpretacja ściśle rzeczownikowa, np. *wywłaszczanie*

**Dominację pośrednią** (łańcuch krawędzi) uzyskujemy poprzez dodanie odpowiedniego kwantyfikatora wyrażen regularnych:

`? * + {m,n} {m,} {,n} {n}`

Przy tym zliczaniu podlega łączna ilość krawędzi łańcucha; w szczególności liczba 0 oznacza, że początek i koniec łańcucha są tym samym węzłem.

- `[s] >{,3} [accentability=nakc]`  
węzeł z wybraną interpretacją nieakcentowaną, będący co najwyżej prawnikiem korzenia zdania
- `[syng] >? [person~~pri]`  
słowo składniowe albo samo będące niewątpliwie w pierwszej osobie, albo mające takie dziecko  
Jeśli zarówno słowo, jak i jego dziecko mają tę własność, dziecko zostanie uwzględnione w wyniku zapytania — w związku z heurystyką preferującą długie dopasowania dla wyrażen regularnych.

*Bezpośrednia*<sup>2</sup> dominacja może zawierać po prawej stronie rozbudowane podzapytanie:

- `[synw] > (nie powinny)`  
frazę *nie powinny* złożoną z dzieci pewnego słowa składniowego

### 3.2.2 Dodatkowe modyfikatory

Modyfikator `[ ]` umożliwia specyfikację **atrybutów** krawędzi (o ile takowe atrybuty zostały określone):

- `[ ] >[!synh] [ ]` dominacja, dla której atrybut `synh` ma wartość `false`
- `[fs] >[ADJUNCT] [fs]` LFG: druga ze struktur jest wartością atrybutu `ADJUNCT` dla pierwszej (modelowane jako dominacja o odpowiedniej wartości atrybutu `type`)
- `[fs] >ADJUNCT [fs]` równoważna składnia dla modyfikatorów postaci `[spec_typu]`

Przy pomocy modyfikatora `{: :}` można wyspecyfikować **węzły pośrednie** dominacji pośredniej:

<sup>2</sup>Planujemy taką możliwość również dla ogólnej dominacji pośredniej. Obecnie funkcjonalność ta jest częściowo realizowana przez operator `within` (patrz p. 4.2).

- `[s] >{: [sons.size = 2] :}{3} [seg]`  
segment, którego ojciec i dziadek mają po dwoje dzieci, a pradziadek jest korzeniem zdania
- `[s] >{: [type!=syng:NG] :}{+} [seg]`  
ścieżka od korzenia do segmentu niezawierająca żadnej grupy nominalnej

Modyfikator `[1]` (odp. `[-1]`) wymaga przechodzenia wyłącznie do pierwszych (odp. ostatnich) dzieci:

- `[syng:NG && span.size > 5] >[-1]+ [seg]`  
skrajny prawy potomek grupy nominalnej mającej łącznie ponad 5 potomków
- Uwaga: Zapytanie `[syng:NG && span.size > 5].span[-1]` nie jest równoważne powyższemu, ponieważ `span[-1]` zwraca skrajnego prawego potomka wg porządku segmentowego, podczas gdy `>[-1]+` operuje wg porządku krawędziowego (p. punkt 3.3).

Kolejność modyfikatorów jest dowolna — następujące operatory są równoważne:

`>[1] [synh] {: [syng] :}{+}`    `>{: [syng] :}{+} [1] [synh]`    `>[synh]+[1] {: [syng] :}{+}`

### 3.2.3 Wyrażenia regularne względem typów krawędzi pionowych

Specjalna składnia — zasadniczo na użytek modelu LFG — umożliwia budowanie **ogólnych** wyrażeń regularnych w pionie, o ile jedyne specyfikacje pośrednie dotyczą typów odwiedzanych krawędzi:

- `[fs] >(OBL []{2,} PRED) [semform]`  
f-struktura, z której przejście atrybutami `OBL`, ... ( $\geq 2$  po drodze), `PRED` daje formę semantyczną
- `[fs] >(OBL (_HAS | ADJUNCT){2,} PRED) [semform]`  
j.w., ale wszystkie atrybuty po drodze to `_HAS` lub `ADJUNCT`

### 3.2.4 Dodatki składniowe

Dodatki na użytek LFG:

- `[IP] >> [fs]`    rzutowanie c-struktury typu `nt:IP` na f-strukturę; “>>”  $\equiv$  “->phi”
- `[IP] >><< [seg]`    dwie c-struktury typów `nt:IP` oraz `seg` rzutują na tę samą f-strukturę

## 3.3 Wyrażenia regularne poziome względem porządku krawędziowego

Model danych dopuszcza sytuację, w której graf składni nie jest drzewem. Oznacza to, że w ogólności musimy rozważać dwa sposoby porządkowania linearnego węzłów (por. przykład na str. 9):

- **porządek segmentowy** — liniowe uporządkowanie segmentów wg kolejności wystąpienia w tekście źródłowym; wyznacza również uporządkowanie (w ogólności — *częściowe*) pozostałych węzłów, zgodnie z przykładami z p. 3.1
- **porządek krawędziowy** — liniowe uporządkowanie krawędzi wychodzących z danego węzła do



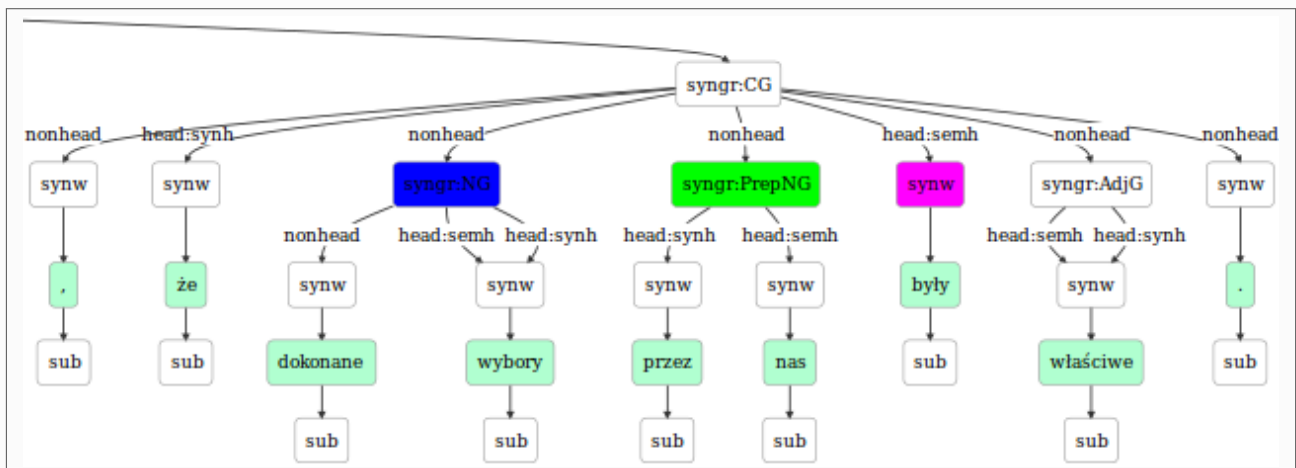
jego dzieci; wartością atrybutu **sons** jest lista dzieci uporządkowana właśnie według tego porządku. Jest on rozważany wyłącznie w kontekście danego węzła-ojca.

Dostęp do porządku krawędziowego dzieci danego węzła jest możliwy poprzez atrybut **sons** tego węzła, w połączeniu z konstrukcją **wewnątrzlistowych wyrażeń regularnych**:

- $\{ | \text{wyr\_reg} | \}$  specyfikuje listę o zawartości pasującej do wyrażenia regularnego *wyr\_reg*
- $L = \{ | \text{wyr\_reg} | \}$  wykonuje wyrażenie *wyr\_reg* na ciągu elementów listy *L*
- $N.\text{sons} = \{ | \text{wyr\_reg} | \}$  wykonuje wyrażenie *wyr\_reg* na liście dzieci węzła *N* (wg kolejności elementów na liście, czyli wg porządku krawędziowego)

#### PRZYKŁAD NIEZGODNOŚCI PORZĄDKÓW

- W poniższym zdaniu<sup>3</sup> węzeł niebieski bezpośrednio poprzedza zielony w porządku krawędziowym:



[Jestem głęboko przekonany, że czas pokaże], że *dokonane* *przez nas* *wybory* *były* właściwe.

- Tymczasem nie jest to prawdą w porządku segmentowym, ponieważ według tego porządku ostatni segment w zasięgu węzła niebieskiego (*wybory*) nie jest bezpośrednim poprzednikiem pierwszego segmentu w zasięgu węzła zielonego (*przez*).
- Z drugiej strony, węzeł niebieski bezpośrednio poprzedza węzeł różowy w porządku segmentowym, ale nie w porządku krawędziowym.

#### PRZYKŁADY ZAPYTAŃ O PORZĄDEK KRAWĘDZIOWY

- (a)  $[\text{syng:NG}].\text{sons} = \{ | [\text{pos!} \sim \text{Noun}] \{3, \} | \}$   
grupa nominalna, która przynajmniej troje dzieci i wszystkie jej dzieci (słowa składniowe) *nie są* niewątpliwymi rzeczownikami, np. *art. 1 ust. 2 pkt 3*
- (b)  $[\text{synw}].\text{sons} = \{ | [] * [\text{orth}=\text{nie}] [\text{pos}!="\text{inf}|\text{fin}|\text{praet}|\text{bedzie}"] [] * | \}$   
słowo składniowe, wśród którego dzieci występuje segment *nie*, poprzedzający (w porządku krawędziowym) segment posiadający wybraną interpretację nieczasownikową, np. *nie wiadomo*

**Uwaga:** Korzystając z porządku krawędziowego należy mieć na uwadze, że w aktualnej implementacji jest ono **utrudnione** z dwóch przyczyn, które zamierzamy wkrótce usunąć:

- W przypadku, gdy głowa składniowa grupy składniowej pokrywa się z semantyczną, model danych odzwierciedla to w postaci dwóch krawędzi (typu **head:synh** oraz **head:semh**). Wskutek tego głowa składniowo-semantyczna występuje *podwójnie* na liście zwracanej przez atrybut **sons** grupy.
- Narzędzia pozwalające na ograniczenie się do wybranej warstwy anotacji są dostępne jedynie częściowo. Wskutek tego węzły pochodzące z różnych warstw (np. węzły składniowe oraz węzły jednostek nazewniczych) podlegają wspólnemu uszeregowaniu w ramach porządku krawędziowego korzenia zdania (który jest ich wspólnym ojcem). Skutkuje to dużą liczbą pozornych zaburzeń drzewiastości w modelu danych.

<sup>3</sup>Rysunek uzyskano wykorzystując zapytanie:

$([].\text{sons} = \{ | [] * \$A \$C := [\text{type}!="\text{named}.*"] [] * | \} \ \&\& \ \$A != \$C \ \&\& \ !(\$A \$C) \ \&\& \ \$A \$D := \$E := [\text{type}=\text{synw}] ) / \text{rha/PS}$

W chwili obecnej utrudnienia te można omijać, odpowiednio rozbudowując zapytania, a zwłaszcza dodając w kluczowych miejscach warunki ograniczające warstwę anotacji (np. `[type!="named:.*"]`). Przykładem może być zapytanie użyte do uzyskania z korpusu przykładowego zdania użytego powyżej (patrz przypis do przykładu).

## 4 Zapytania ogólne

### 4.1 Budowa zapytania

Najogólniej, zapytanie jest rozbudowanym **wyrażeniem logicznym**. Może ono mieć postać:

- jednego z rodzajów omówionych w punktach 2 i 3
  - wówczas zwraca `true`, jeśli znaleziono dopasowanie, i `false` w przeciwnym wypadku;
- kombinacji logicznej powyższych zapytań
  - wówczas jego wartość jest obliczana zgodnie z regułami działań logicznych.

#### PRZYKŁADY

- `Sejm && [base=komisja]` w zdaniu istnieje słowo *Sejm* oraz forma słowa *komisja*
- `Sejm && ([base=poseł] => [base="pani?"])`  
w zdaniu istnieje słowo *Sejm*, a przy tym pamiętano o grzeczności  
(jeśli istnieje forma słowa *poseł*, to istnieje też forma słowa *pan* lub *pani*)
- `Sejm && [base=poseł] && ![base="pani?"]`  
w zdaniu istnieje słowo *Sejm*, a przy tym nie pamiętano o grzeczności (p. wyżej)

### 4.2 Zasięg zapytania

Domyślnym **zasięgiem** zapytania jest **zdanie**. Oznacza to, że wszelkie warunki egzystencjalne bez jawnego zasięgu interpretowane będą w kontekście bieżącego zdania — jak w przykładach z p. 4.1.

Na zmianę zasięgu zapytania (lub podzapytania!) pozwala słowo kluczowe `within`. Można go użyć na dwa sposoby:

- *zapytanie* `within typ`                      szukaj w zasięgu węzłów o typie *typ*  
(kompatybilne ze starym Poliqarpem)
- *zapytanie* `within spec_węzła`                      szukaj w zasięgu węzłów spełniających *spec\_węzła*

W przypadku węzła typu `doc`, daje to nowy sposób dostępu do metadanych. Jednak w ogólności *spec\_węzła* może określać węzeł dowolnego typu.

#### PRZYKŁADY

- `(Sejm && [base=komisja]) within p`  
znajduje akapity zawierające słowo *Sejm* oraz formę słowa *komisja*
- `(Prezydium Sejmu) within [doc && taxonomy_type="#typ_qmow"]`  
znajduje frazę *Prezydium Sejmu* w dokumentach quasi-mówionych
- `(([base=komisja] && i) within syng:NG) && [base=porządek]`  
znajduje zdanie zawierające formę słowa *porządek* oraz wieloczłonową nazwę komisji (a ściślej, grupę nominalną (`syng:NG`), która z kolei zawiera w sobie formę słowa *komisja* oraz spójnik *i*)

**Uwaga:** Ze względu na typowy schemat użycia, planujemy zmianę priorytetu wiązania operatora `within` względem jego lewego argumentu na **bardzo słaby**.

## 4.3 Zmienne

Zmienne mogą służyć dwom celom:

- wielokrotnemu odwoływaniu się w zapytaniu do pewnego obiektu;
- wyróżnieniu wybranych obiektów w wyniku zapytania.

Dostępne są następujące operacje na zmiennych:

- `$X:=wart`      **zapis** wartości zmiennej
- `$X+=wart`      **dopisanie** wartości do zmiennej (typu listowego)
- `$X`      **odczyt** wartości zmiennej

Nazwa zmiennej musi rozpoczynać się od znaku `$`. Dalszy ciąg musi rozpoczynać się od litery i składać ze znaków liter, cyfr oraz podkreślenia.

Zmiennych nie trzeba inicjalizować — zmienne niezainicjowane zostaną niejawnie obłożone kwantyfikatorem egzystencjalnym w bieżącym zasięgu.

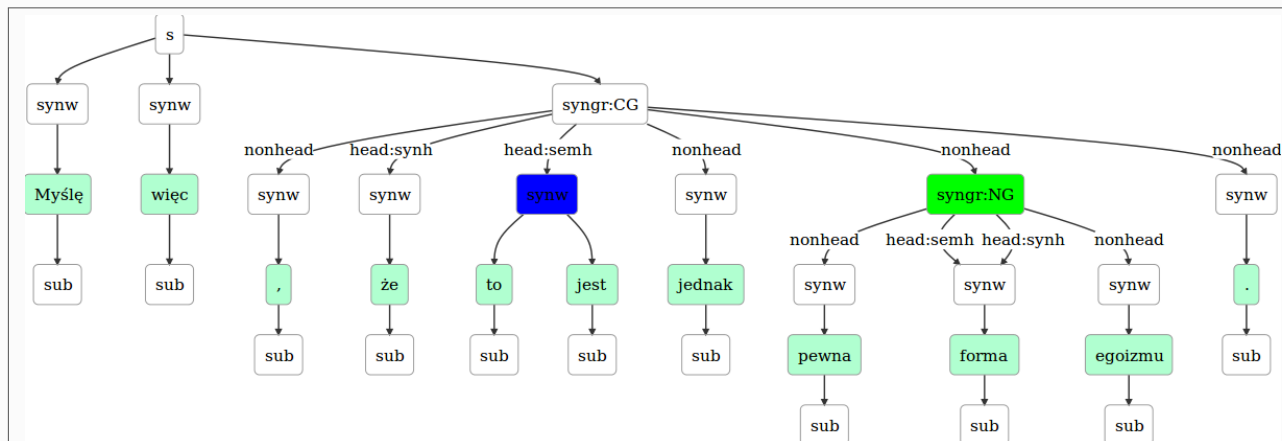
### PRZYKŁADY

- `$X:=[syng:NG] > [base=komisja] && $X > *` i  
znajduje grupę nominalną `$X`, zawierającą formę słowa *komisja* oraz spójnik *i*
- `([base=komisja] $X:=[] * i $Y:=[] *) within syng:NG`  
znajduje np. grupę nominalną *Komisja Rolnictwa i Gospodarki Żywnościowej*;  
przypisuje zmiennej `$X` segment *Rolnictwa*, zaś zmiennej `$Y` segmenty *Gospodarki Żywnościowej*
- `[base=$B:= ".*trzymać"]`  
znajduje węzeł posiadający *jakąś* wybraną formę bazową typu *otrzymać* lub *utrzymać*;  
zapisuje tę formę do `$B`
- `[$B:=base=".*trzymać"]`  
z pozoru prostszy sposób na to samo, ale w tej wersji do `$B` zostanie zapisana pełna lista wieloznaczna będąca wartością `base`!
- `[type=$T] > [seg]`  
“podejrzenie” wartości atrybutu (tu: `type`) w znalezionym węźle (tu: dominującym segment)
- `[$T:=type] > [seg]`  
znów, z pozoru prościej, ale nie zadziała: wyrażenie `$T:=type` jest napisem, a specyfikacja struktury atrybutowej powinna zawierać wartość logiczną

---

PRZYKŁAD (NIESTETY WOLNO DZIAŁAJĄCY)

- (`[[] .sons = { [ [] * $A [] + $B [] * | } && $A $B && $B.type!="named:.*")/rha` znajduje węzeł `$B`, który następuje po `$A` bezpośrednio w porządku segmentowym, ale niebezpośrednio w porządku krawędziowym ich wspólnego ojca, a ponadto nie jest jednostką nazewniczą; wynikiem zapytania są na przykład węzły zaznaczone w poniższym zdaniu:



Myślę więc, że *to* jednak *jest*  *pewna forma egoizmu.*

## 5 Flagi

## 5.1 Sposób użycia

**Flaga** jest modyfikatorem całego zapytania lub jego fragmentu, wpływająca na wybrany aspekt semantyki języka.

### PRZYKŁAD

- do ([base=komisja] [pos~subst]\*)/i ignoruje wielkość liter w całym podzapytaniu

Nazwy flag rozpoczynają się od znaku /. Priorytet ich wiązania jest dość silny:

- pan poseł/i ignoruje wielkość liter jedynie dla słowa *poseł*

Flagi tworzą **grupy**, w ramach których wzajemnie się wykluczają. Przykładowo, w jednej grupie znajdują się flagi `/i` (ignoruj wielkość liter) oraz `/I` (uwzględnij wielkość liter), z których zawsze dokładnie jedna jest aktywna. Jeśli do danego fragmentu zapytania zastosowano wielokrotnie flagi z jednej grupy, uwzględniany jest wybór dokonany najgłębiej w sensie składni zapytania.

### PRZYKŁAD

- (pan poseł/i)/I/i jest równoważne z (pan/I) (poseł/i)

Flagi dzielą się na **krótkie** (o jednoliterowych nazwach) oraz **długie** (pozostałe). Dla zachowania kompatybilności ze starym Poliqarpem, zezwalamy na łączenie flag krótkich:

### PRZYKŁAD

- sejm/ix jest równoważne z sejm/i/x

## 5.2 Przegląd flag

Każda ramka odpowiada jednej grupie flag. Symbolem + oznaczamy flagi domyślnie stosowane do zapytania.

Flagi dotyczące **znakowych wyrażeń regularnych**:

- uwzględnianie wielkości liter:

- /i	ignoruj wielkość liter
+ /I	uwzględnij wielkość liter

- (nie)kompletność specyfikacji słów:

- most/x	znajduje <i>most, pomost, mostu, pomostu</i>
- most/xl	znajduje <i>most, pomost</i>
- most/xr	znajduje <i>most, mostu</i>
+ most/X	znajduje <i>most</i>

Flagi kontrolujące **rozpatrywane warstwy anotacji**:

- specyfikacje węzłów (w tym zapytania proste):

- /ls	dopasowują się jedynie do <i>segmentów</i>
+ /la	dopasowują się do dowolnych <i>węzłów</i>

- poziome wyrażenia regularne (z wyjątkiem wewnątrzlistowych):

+ /rhs	dopasowują się jedynie do ciągów <i>segmentów</i>
- /rha	dopasowują się do ciągów dowolnych <i>węzłów</i>

Flagi kontrolujące **liczbę i postać wyników**:

- niejawna zmienna całosciowa:

+ /ps	utwórz zmienną $\$_Q$ przechowującą cały wynik zapytania
- /ps( <i>nazwa</i> )	utwórz zmienną $\$_{nazwa}$ przechowującą cały wynik zapytania
- /PS	nie twórz zmiennej o tej roli

- liczba możliwych wyników w zdaniu:

- /pm	zezwól na wiele wyników w jednym zdaniu
+ /PM	wyświetlaj $\leq 1$ wynik dla każdego zdania

**Uwaga:** Flaga /PM ma pierwszeństwo przed wszelkimi „wylewnymi” flagami opisanymi poniżej.

- filtrowanie wyników wg zasięgu zmiennych:

- `/pmv` zezwól na wiele wyników nieróżniących się zasięgami wartości zmiennych
- + `/PMV` wyświetlaj  $\leq 1$  wynik dla danego zestawu zasięgów wartości zmiennych

- heurystyka segmentowa:

- + `/ppv` wyróżnij wyniki zawierające więcej segmentów
- `/PPV` nie wyróżniaj takich wyników

- heurystyka maksymalnego dopasowania:

- + `/pps` wyróżnij wyniki o maksymalnym łącznym zasięgu
- `/PPS` nie wyróżniaj takich wyników

- dopuszczenie wyników niewyróżnionych:

- `/pnp` wyświetl wszystkie wyniki
- + `/PNP` wyświetl jedynie wyniki wyróżnione przez `/ppv` i/lub `/pps`

Flagi kontrolujące **limity wydajnościowe**:

- limit czasu wykonania w jednym zdaniu (ogólniej — jednostce zasięgu):

- + `/elt` zatrzymaj wykonanie po 30 sekundach
- `/elt(n)` zatrzymaj wykonanie po *n* sekundach
- `/ELT` brak limitu

- limit liczby *potencjalnych* dopasowań rozważanych jednocześnie w zdaniu (jednostce zasięgu):

- + `/elv` zatrzymaj wykonanie dla  $10^5$  potencjalnych dopasowań
- `/elv(n)` zatrzymaj wykonanie dla  $10^n$  potencjalnych dopasowań
- `/ELV` brak limitu

## A Przegląd atrybutów w korpusie NKJP

### A.1 Atrybut `type`; typy węzłów/krawędzi

Określa typ węzła/krawędzi. Możliwe wartości w korpusie NKJP:

- Typy węzłów składniowych:
  - `seg` segment
  - `synw` słowo składniowe
  - `syngr:typ_grupy` grupa składniowa; po dwukropku rodzaj grupy (np. `syngr:AdjG`)
- Typy jednostek nazewniczych:

- `named: typ_jedn` jednostka nazewnicza;  
po dwukropku rodzaj jednostki (np. `named:persName:surname`);  
w obecnej wersji jednoczłonowe rodzaje zakończone są dwukropkiem (np. `named:orgName:`)
- Typy węzłów segmentacji wielkoskalowej:
  - `s` zdanie
  - `p` akapit
  - `doc` dokument
- Dodatkowy typ węzła (nieobecny w NKJP):
  - `sub` **podsegment**: w ogólności reprezentuje wieloznaczność segmentacji;  
w NKJP — równoważny segmentowi (każdy `seg` ma dokładnie 1 dziecko typu `sub`)
- Typy krawędzi<sup>4</sup>:
  - `head:synh` krawędź prowadząca do głowy składniowej grupy/słowa składniowego
  - `head:semh` krawędź prowadząca do głowy semantycznej grupy/słowa składniowego
  - `nonhead` krawędź prowadząca do nie-głównego dziecka grupy/słowa składniowego
  - `<pusty napis>` żadne z powyższych

## A.2 Proste atrybuty węzłów i krawędzi

Dostępne atrybuty zależą oczywiście od typu danej struktury:

- atrybuty **segmentów**:
  - `orth` postać tekstowa, np. *powiedzieli, by, śmy*
  - `msd` + pochodne informacje morfosyntaktyczne; patrz p. 2.3
- atrybuty **słów składniowych**:
  - `orth` postać tekstowa (ze świadomością spacji), np. *powiedzielibyśmy, wstrzymał się*
  - `msd` + pochodne informacje morfosyntaktyczne; patrz p. 2.3
- atrybuty **grup składniowych**:
  - `orth` j.w., np. *, że Sejm przyjął wniosek o przejście do drugiego czytania.*
- atrybuty **jednostek nazewniczych**:
  - `orth` j.w., np. *Regionalnej Izby Obrachunkowej w Bydgoszczy*
  - `lemma`<sup>5</sup> forma bazowa, np. *Regionalna Izba Obrachunkowa w Bydgoszczy*
  - inne: `certainty`, `comment`, `when`, `derivType`, `derivedFrom`  
(wszystkie typu napisowego)
- atrybuty **akapitów i zdań**:
  - (brak)
- atrybuty **dokumentów** (inaczej mówiąc — **metadane**):

<sup>4</sup>Wg takiego modelu, jeśli głowa składniowa grupy pokrywa się z semantyczną, to prowadzą do niej z tej grupy *dwie* różne krawędzie, typów `head:synh` oraz `head:semh`. Ponieważ rozwiązanie to może utrudniać np. analizę nie-drzewiastości grafów składni, zostanie zapewne wkrótce zmienione.

<sup>5</sup>Nazwa tymczasowa — docelowo planujemy powrót do tradycyjnej nazwy `base`.

- **title** np. *TEI P5 encoded version of sample(s) of Sejm session protocols, 2nd term*
- **docid** np. *NKJP\_1M\_7121900002*
- **source\_author** np. *Kancelaria Sejmu Rzeczypospolitej Polskiej*
- **taxonomy\_type** np. *#typ\_qmow*
- **taxonomy\_channel** np. *#kanal\_prasa\_inne*
- **inne:** **source\_title**, **source\_date\_published**, **source\_idno\_nkjp**, **source\_publisher**, **source\_pubPlace**, **source\_idno\_ISBN**, **source\_note\_text\_origin**  
(wszystkie typu napisowego<sup>6</sup>)
- atrybuty **podsegmentów**:
  - **text** analogiczny do **orth** dla segmentów (a w NKJP równoważny)
  - **newword** czy podsegment rozpoczyna słowo? (**true** lub **false**)
  - **endword** czy podsegment kończy słowo? (**true** lub **false**)

### A.3 Atrybuty morfosyntaktyczne

Węzły anotowane morfosyntaktycznie udostępniają atrybut

- **msd** lista wieloznaczna interpretacji morfosyntaktycznych (patrz p. 2.3)

Atrybuty interpretacji morfosyntaktycznych są zgodne z tagsetem NKJP:  
(wszystkie one mają wartości typu napisowego)

- |                          |                          |                                |
|--------------------------|--------------------------|--------------------------------|
| • <b>accentability</b>   | • <b>cont</b>            | • <b>person</b>                |
| • <b>accommodability</b> | • <b>degree</b>          | • <b>pos</b>                   |
| • <b>agglutination</b>   | • <b>fullstoppedness</b> | • <b>post_prepositionality</b> |
| • <b>aspect</b>          | • <b>gender</b>          | • <b>reflexitivity</b>         |
| • <b>base</b>            | • <b>mood</b>            | • <b>tense</b>                 |
| • <b>brev_pos</b>        | • <b>negation</b>        | • <b>vocalicity</b>            |
| • <b>case</b>            | • <b>number</b>          |                                |

---

<sup>6</sup>Docelowo **source\_date\_published** będzie oczywiście typu liczbowego, z możliwością porównywania z ustalonym rokiem.