Gramatyka TAG dla języka polskiego

Katarzyna Krasnowska

IPI PAN

25 lutego 2013

Plan prezentacji

Tree Adjoining Grammar – formalna definicja

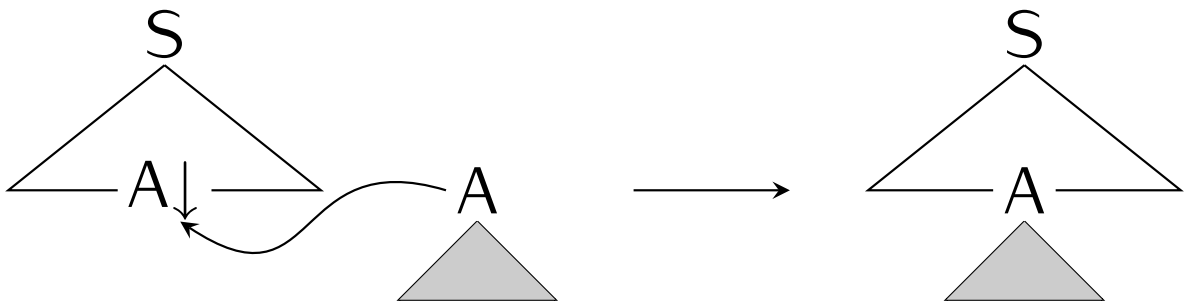
Gramatyka TAG (Joshi i Schabes, 1997) to 5-krotka $\langle \Sigma, NT, I, A, S \rangle$:

- Σ – skończony zbiór terminali
- NT – skończony zbiór nieterminali
- I – skończony zbiór drzew *początkowych* (ang. *initial trees*)
- A – skończony zbiór drzew *pomocniczych* (ang. *auxiliary trees*); $I \cap A = \emptyset$
- $S \in NT$ – symbol początkowy

Tree Adjoining Grammar

- Gramatyka słabo kontekstowa
- Parsowalna wielomianowo ($\mathcal{O}(n^6)$)
- Słabo równoważna m.in. formalizmowi CCG (Combinatory Categorial Grammar)

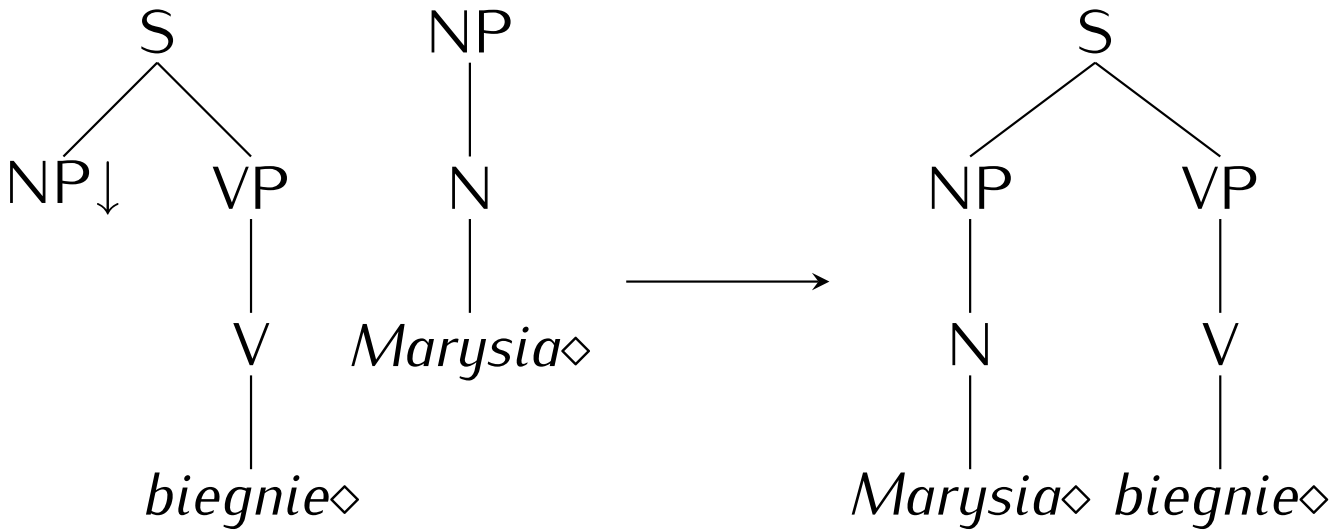
Operacje na drzewach: podstawienie (*substitution*)



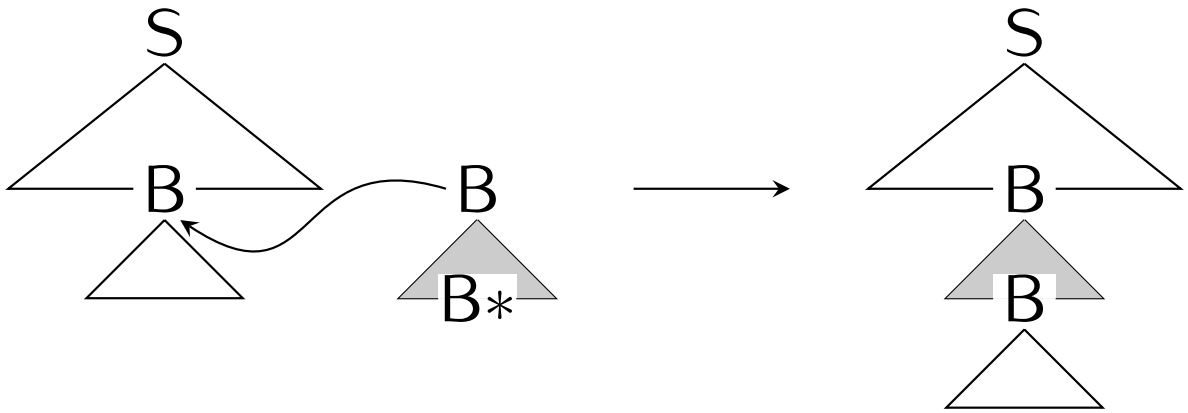
- A↓ – miejsce podstawienia (*substitution node*/*site*) (nieterminalny liść)

Operacje na drzewach: podstawienie (*substitution*)

- Przykład:



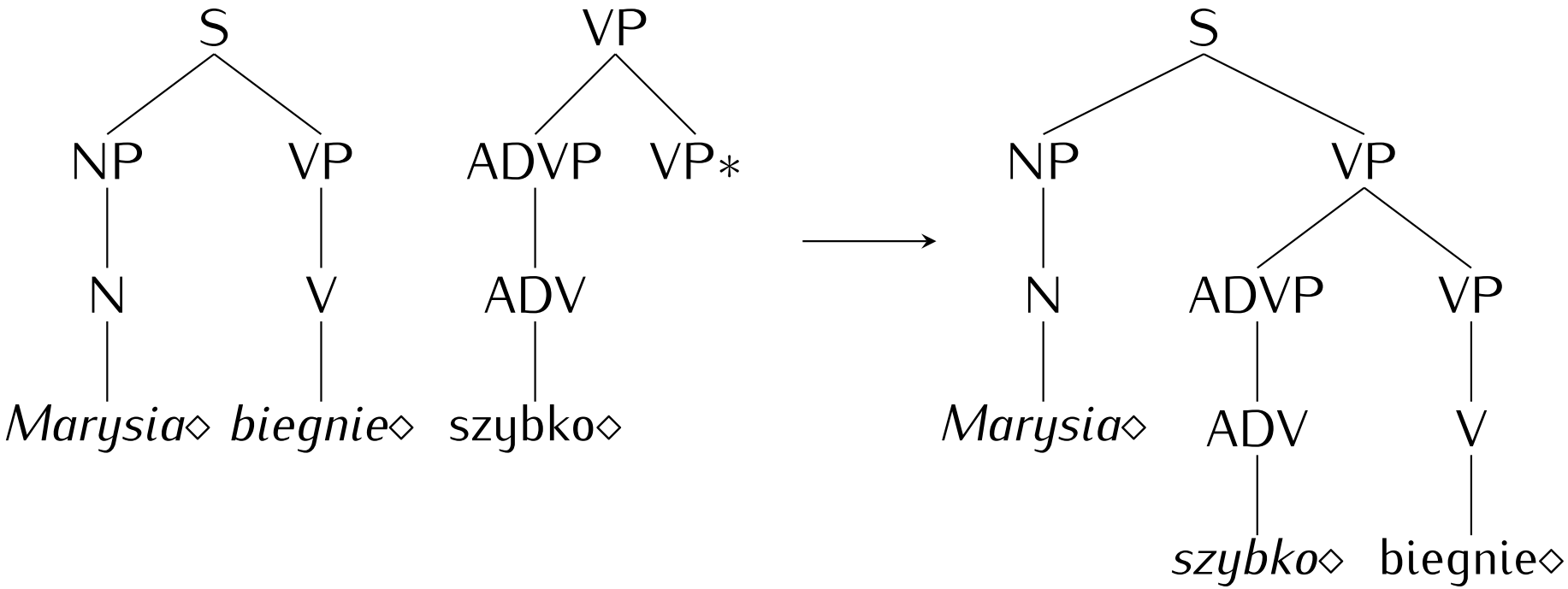
Operacje na drzewach: dołączenie (*adjunction*)



- B∗ – tzw. *foot node* (nieterminalny liść o etykiecie identycznej z korzeniem drzewa)

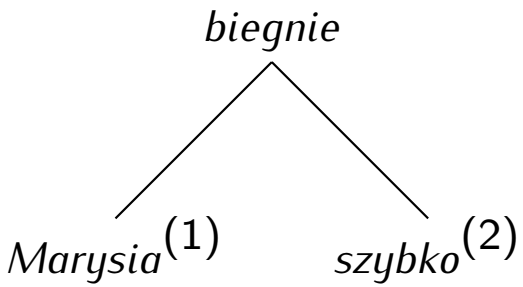
Operacje na drzewach: dołączenie (*adjunction*)

- Przykład:



Drzewo wyprowadzenia

- Dla każdego drzewa elementarnego zaznaczony jest adres Gorna węzła, w którym dokonano podstawienia/dołączenia
- Drzewo wyprowadzenia dla zdania „*Marysia szybko biegnie*”:



LTAG

Lexicalised Tree Adjoining Grammar

- Każde drzewo elementarne posiada co najmniej jeden liść – terminal (element leksykalny, ang. *anchor*)
- Dopuszczalne są dodatkowe leksemy w liściach (*co-anchors*)
- element leksykalny oznaczany jest symbolem $\diamond$.

Plan prezentacji

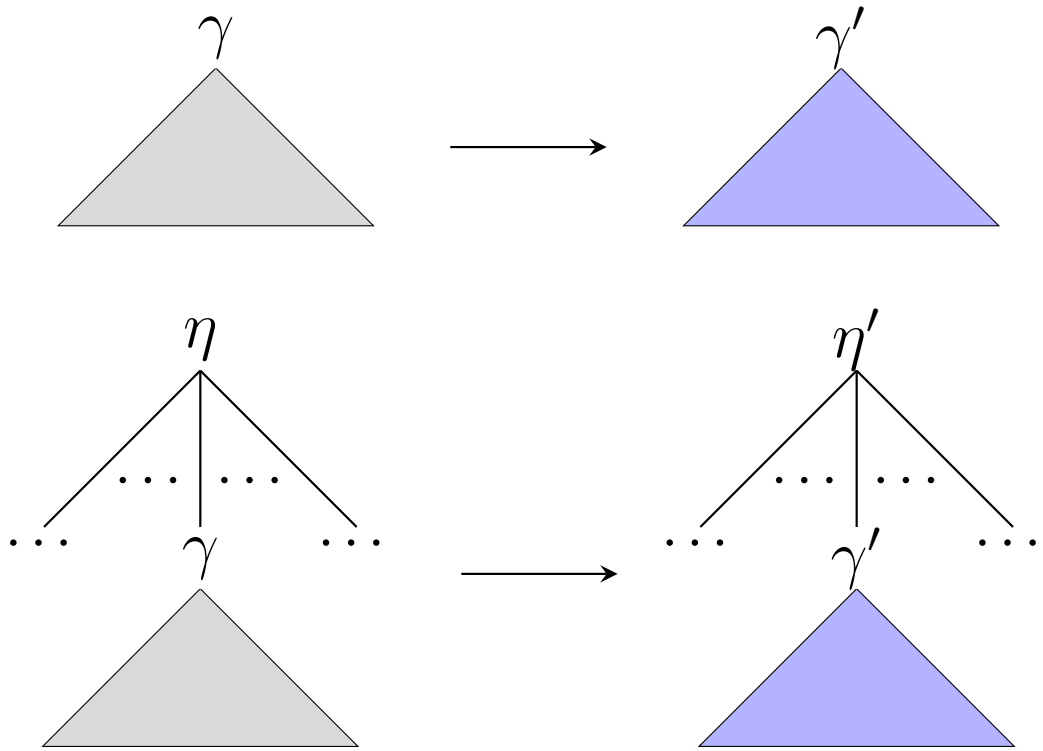
Procedura ekstrakcji

(Chen i Vijay-Shanker, 2000):
Działanie procedury w węźle η (wynik – drzewo elementarne α):

- Stwórz η' – kopię η – korzeń drzewa α
- Dla każdego dziecka η nie będącego elementem głównym, zdecyduj, czy jest ono argumentem
- Dla każdego γ – dziecka η, jeśli jest ono...

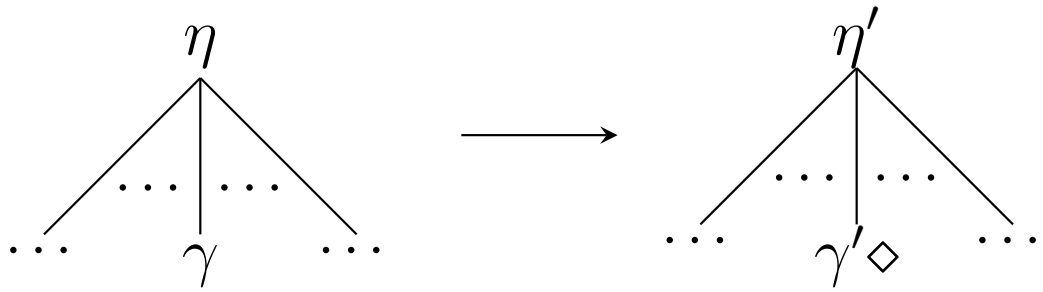
Procedura ekstrakcji

...nieterminalnym elementem głównym – uruchom procedurę rekurencyjnie dla γ i dołącz jej wynik jako dziecko η.



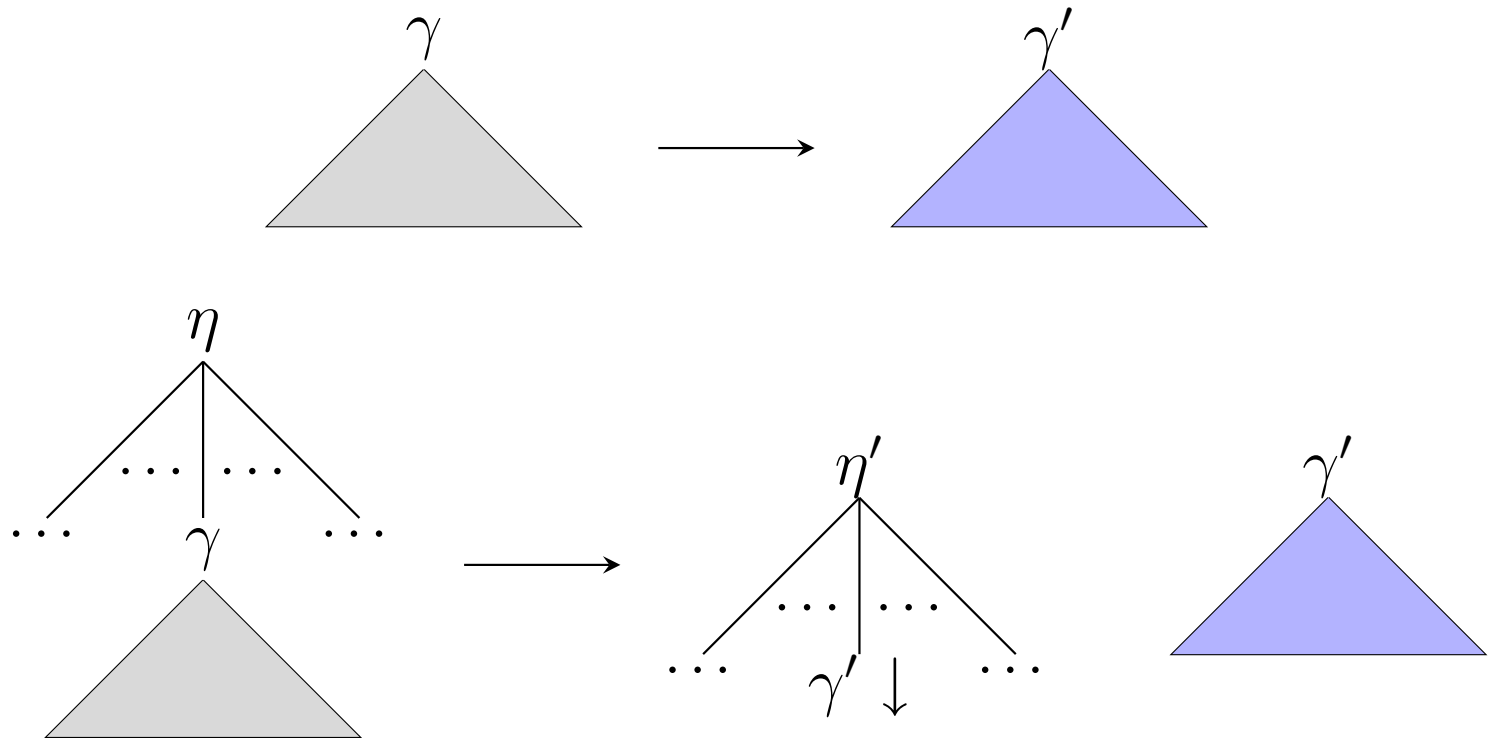
Procedura ekstrakcji

...terminalnym elementem głównym – dołącz kopię γ jako dziecko η' i oznacz jako element leksykalny.



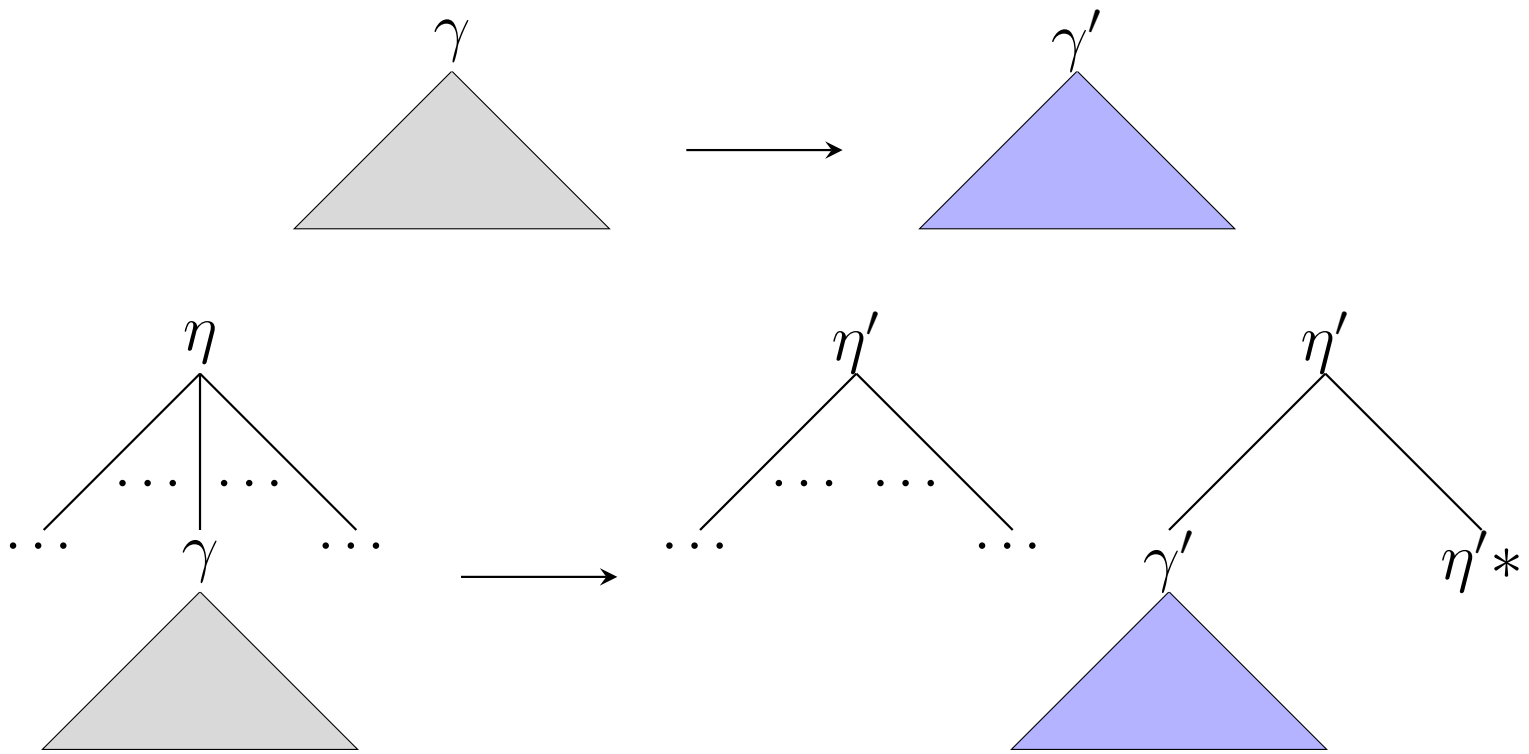
Procedura ekstrakcji

...argumentem – dołącz kopię γ jako dziecko η'; uruchom procedurę dla γ, tworząc nowe drzewo początkowe.

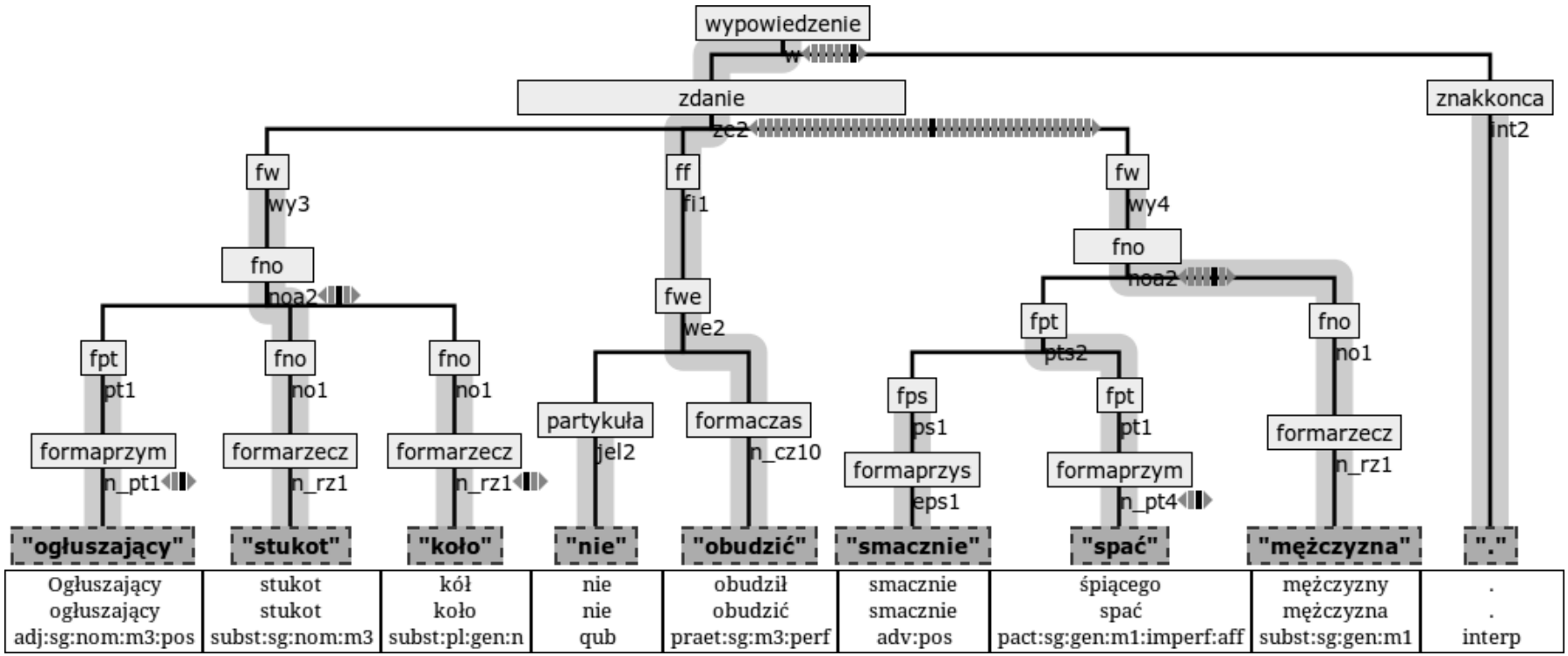


Procedura ekstrakcji

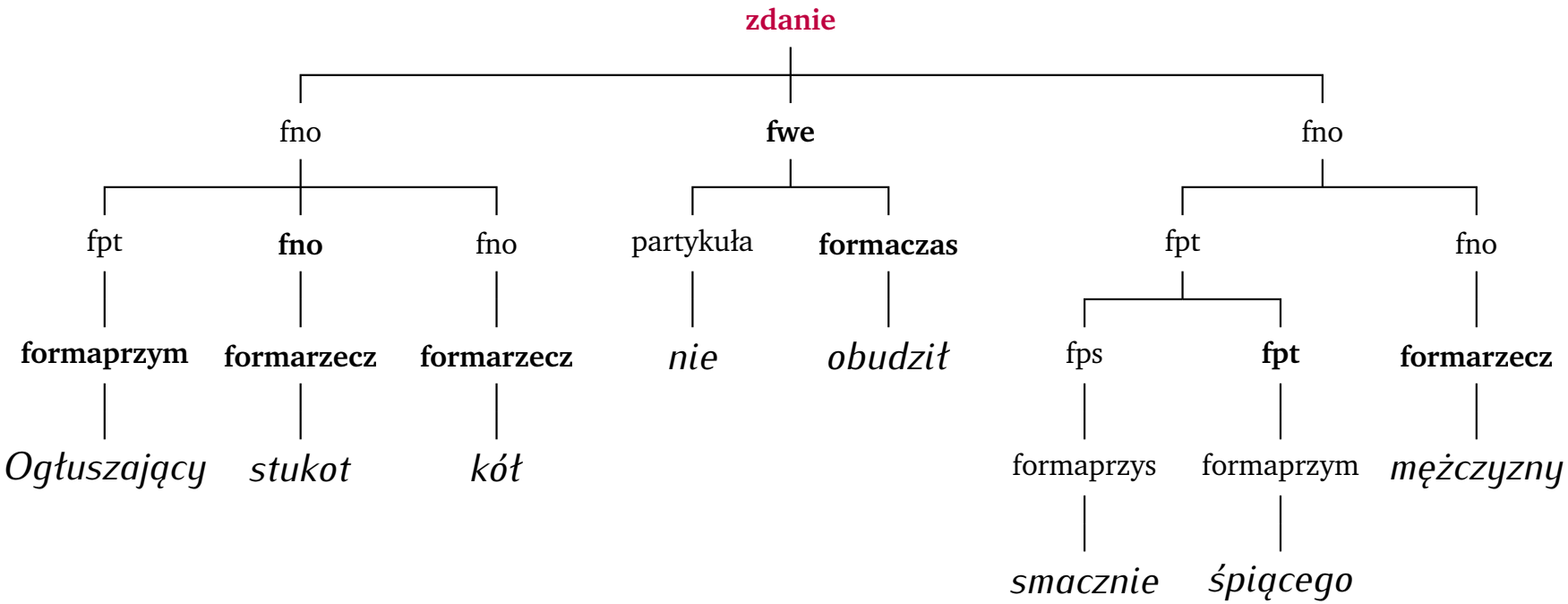
...nie jest argumentem – uruchom procedurę dla γ, tworząc nowe drzewo początkowe, i przekształć je w drzewo pomocnicze.



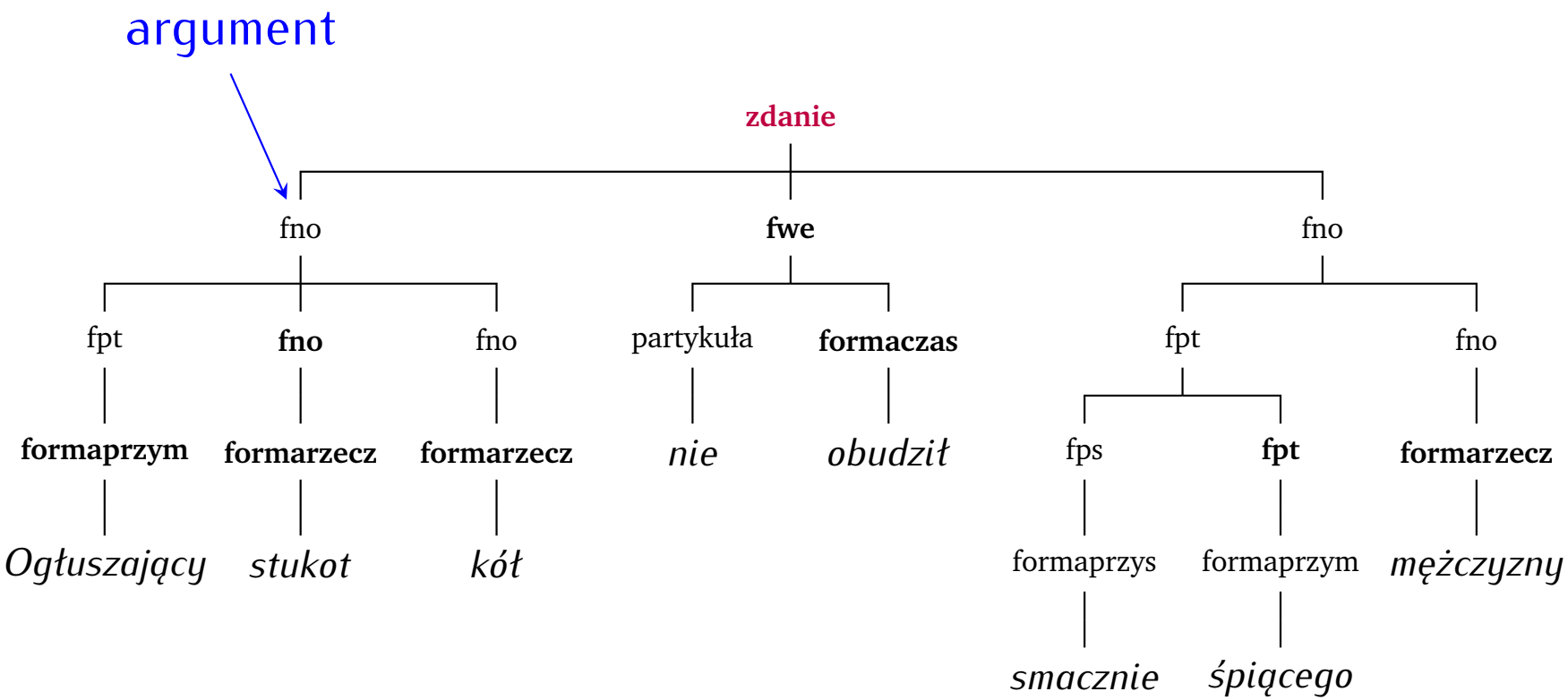
Przykład – drzewo ze *Składnicy*



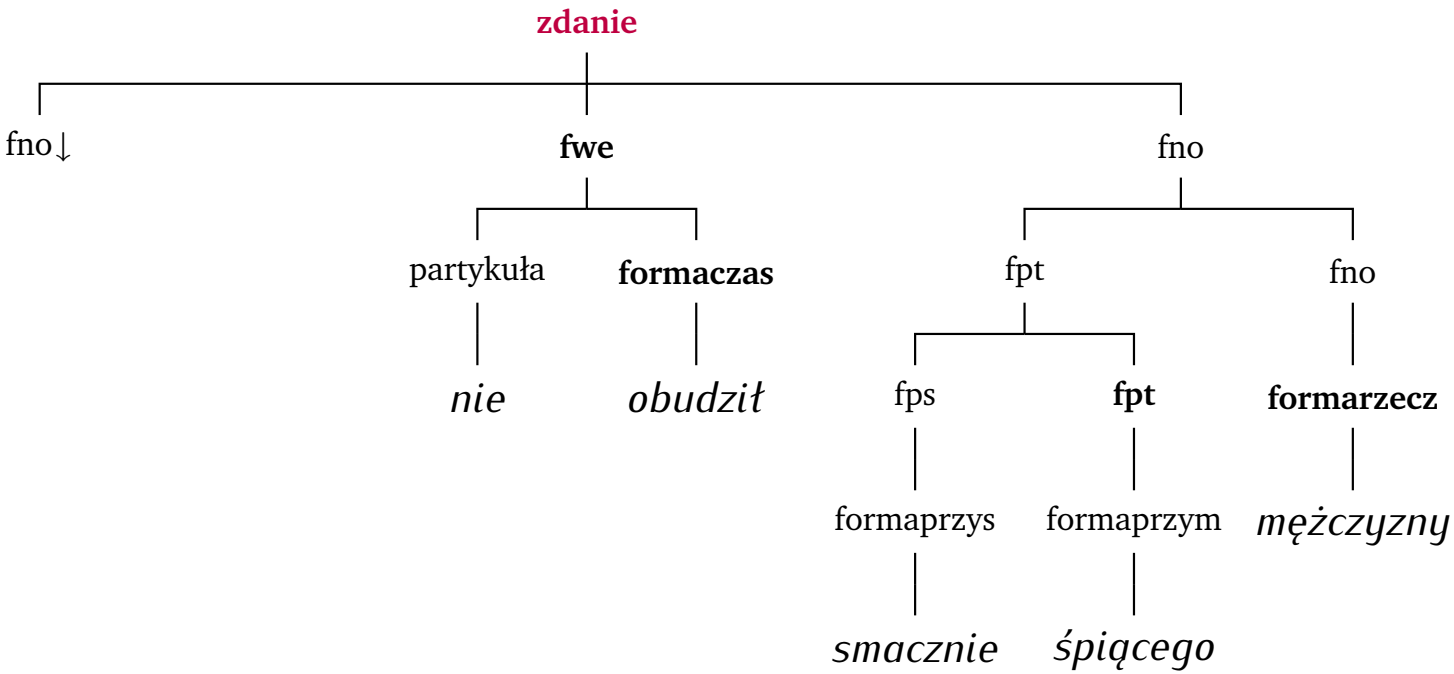
Przykład


zdanie
fno
fwe
fno
fpt
fno
fno
partykuła
formaczas
fpt
fno
formaprzym
formarzecz
formarzecz
nie
obudził
fps
fpt
formarzecz
Ogłuszający
stukot
kół
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

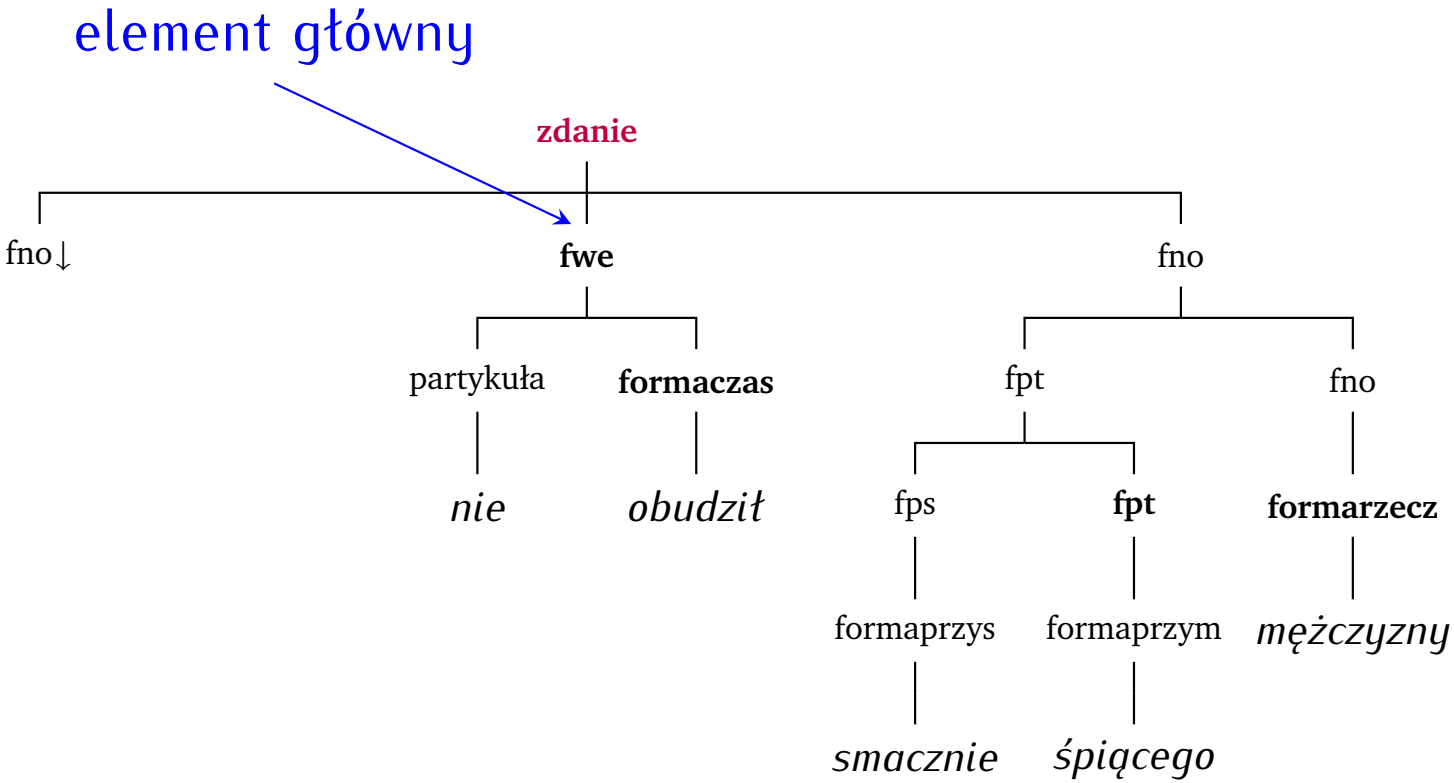
Przykład


argument
zdanie
fno
fwe
fno
fpt
fno
fno
partykuła
formaczas
fpt
fno
formaprzym
formarzecz
formarzecz
nie
obudził
fps
fpt
formarzecz
Ogłuszający
stukot
kół
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

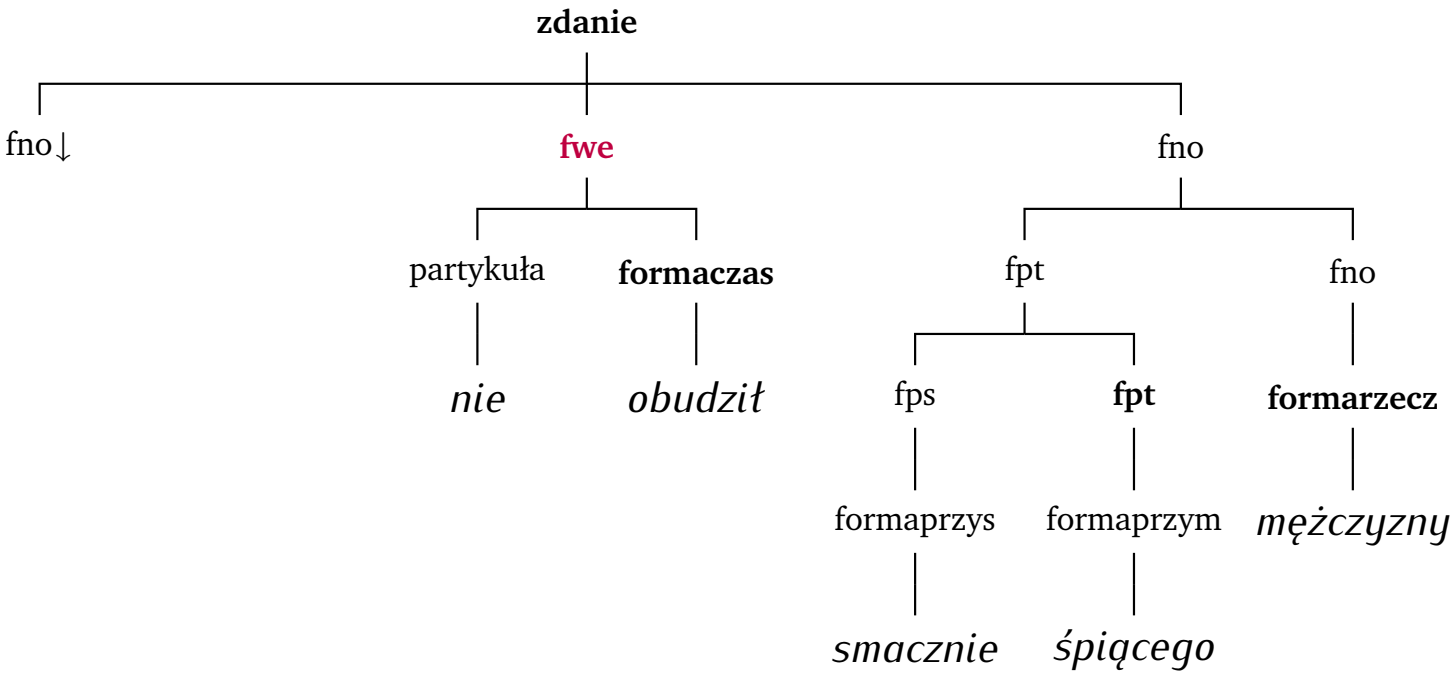
Przykład


zdanie
fno↓
fwe
fno
partykuła
formaczas
fpt
fno
nie
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

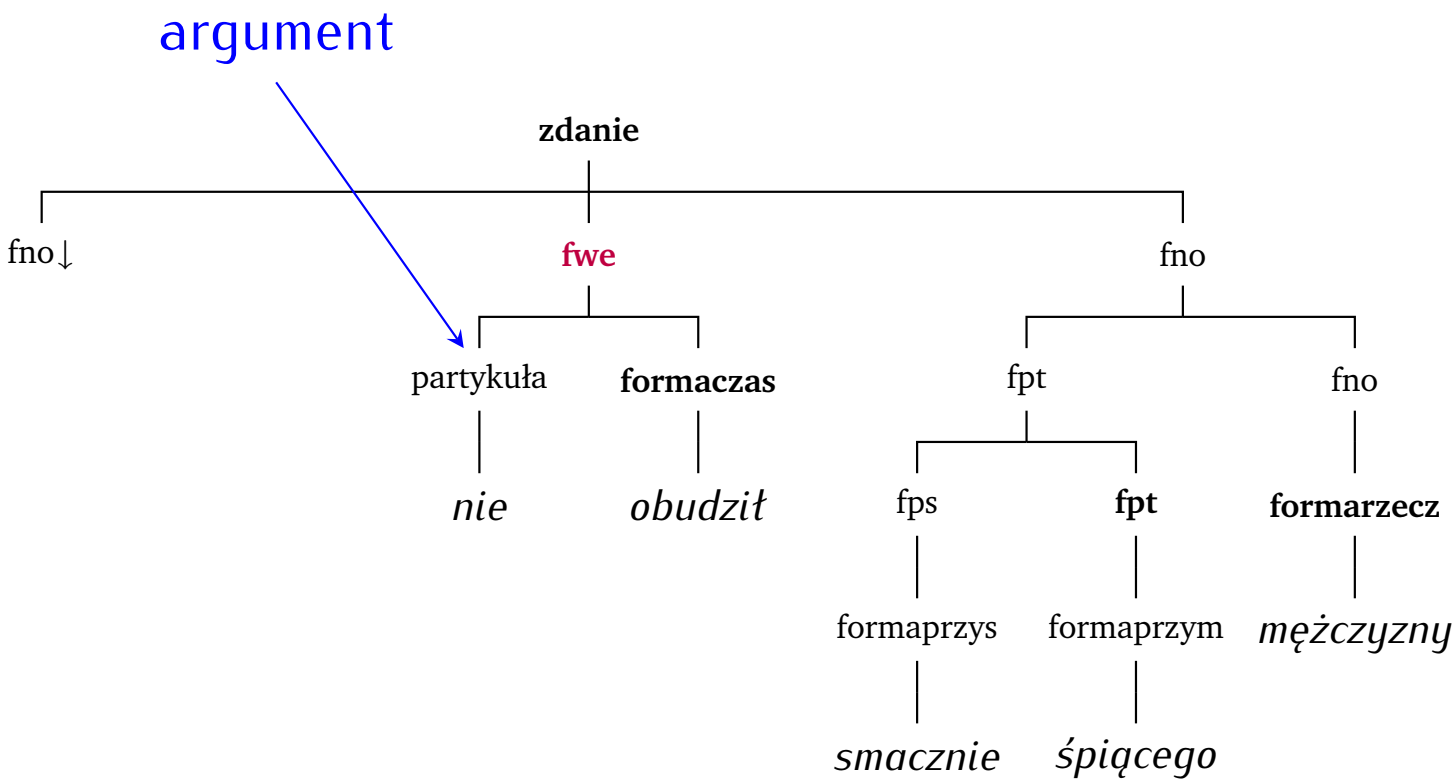
Przykład


element główny
zdanie
fno↓
fwe
fno
partykuła
formaczas
fpt
fno
nie
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

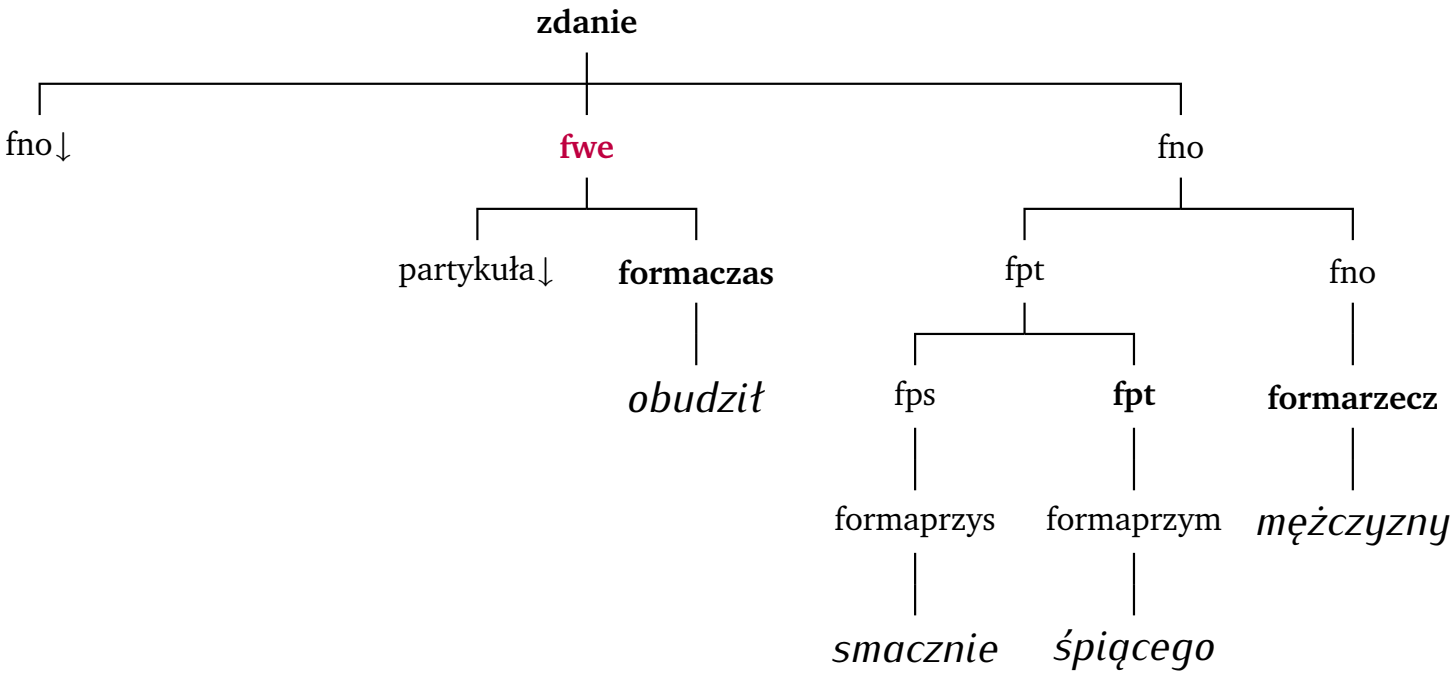
Przykład


zdanie
fno↓
fwe
fno
partykuła
formaczas
fpt
fno
nie
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

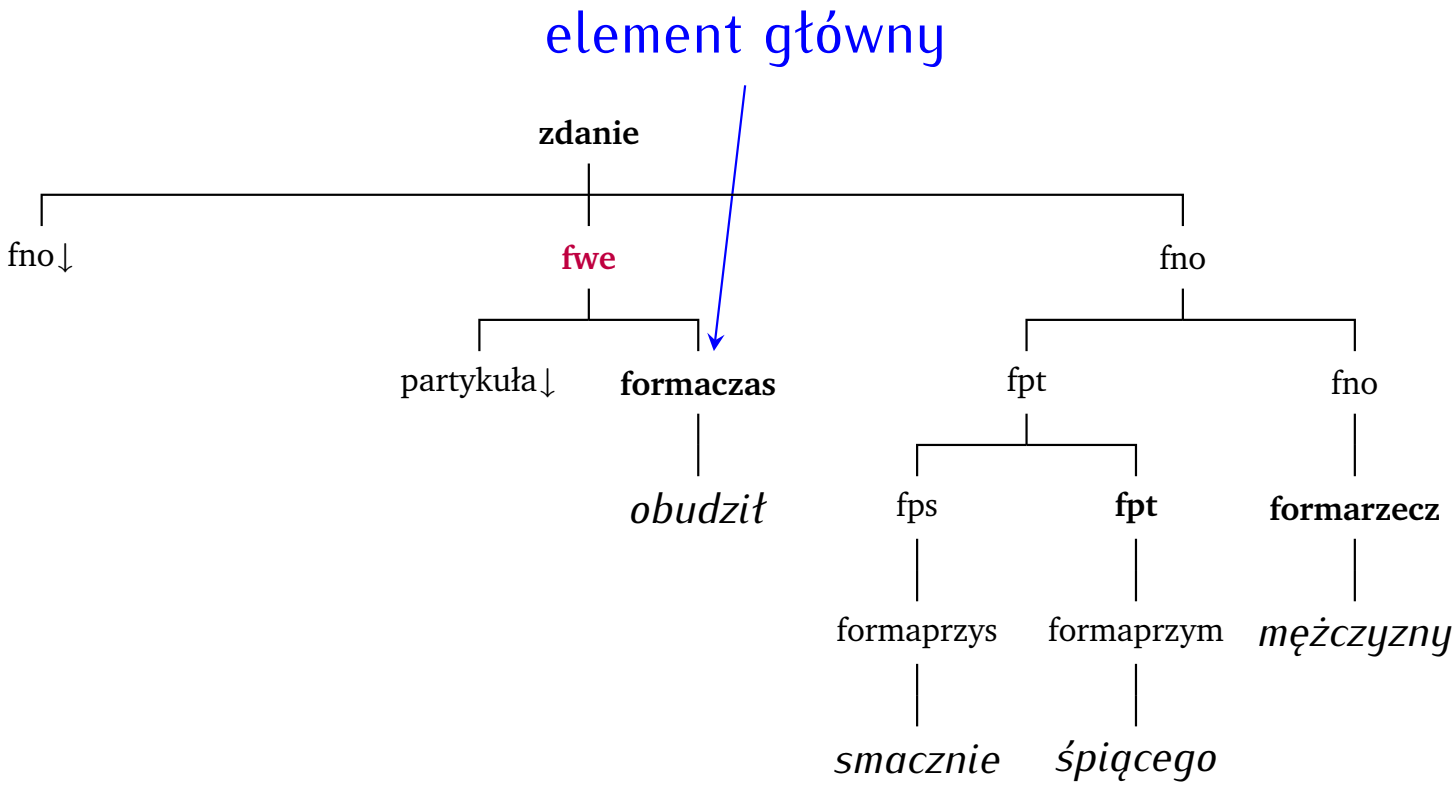
Przykład


argument
zdanie
fno↓
fwe
fno
partykuła
formaczas
fpt
fno
nie
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

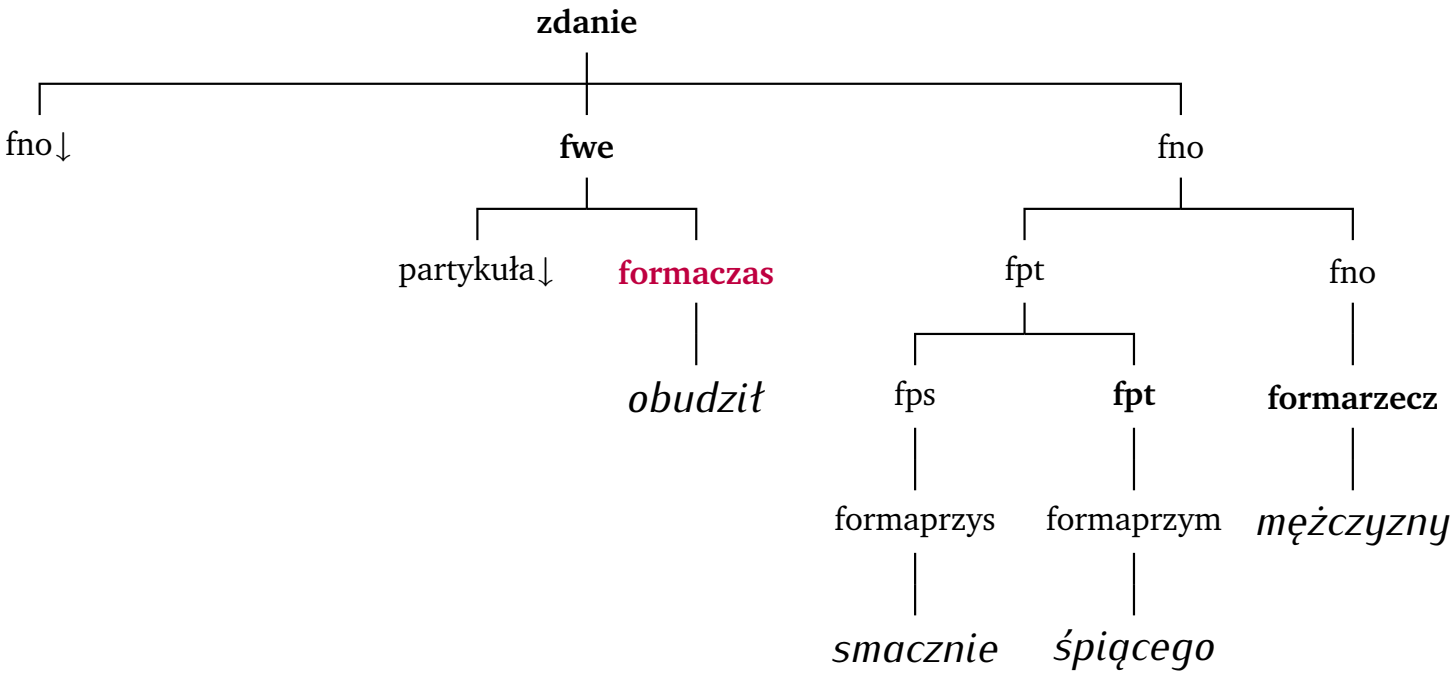
Przykład


zdanie
fno↓
fwe
fno
partykuła↓
formaczas
fpt
fno
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

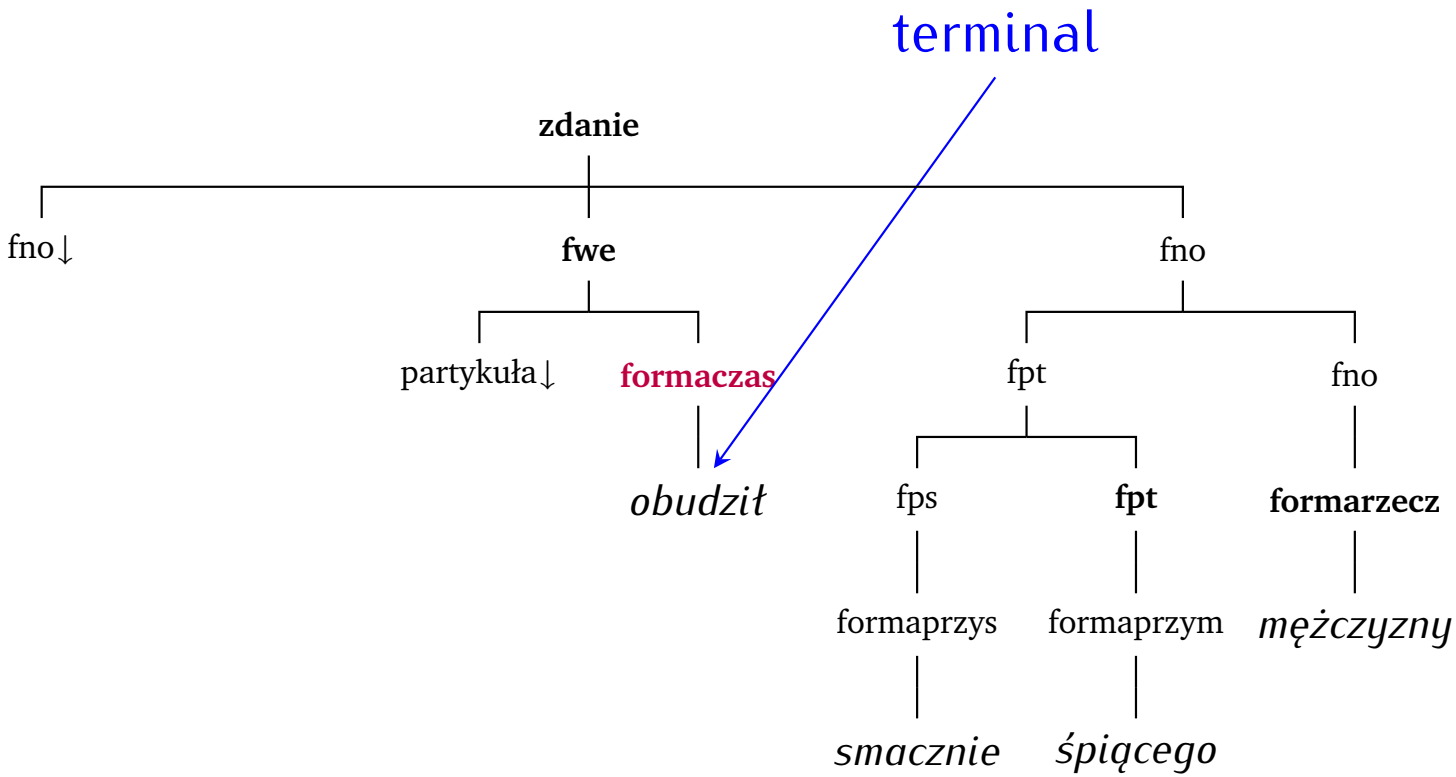
Przykład


element główny
zdanie
fno↓
fwe
fno
partykuła↓
formaczas
fpt
fno
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

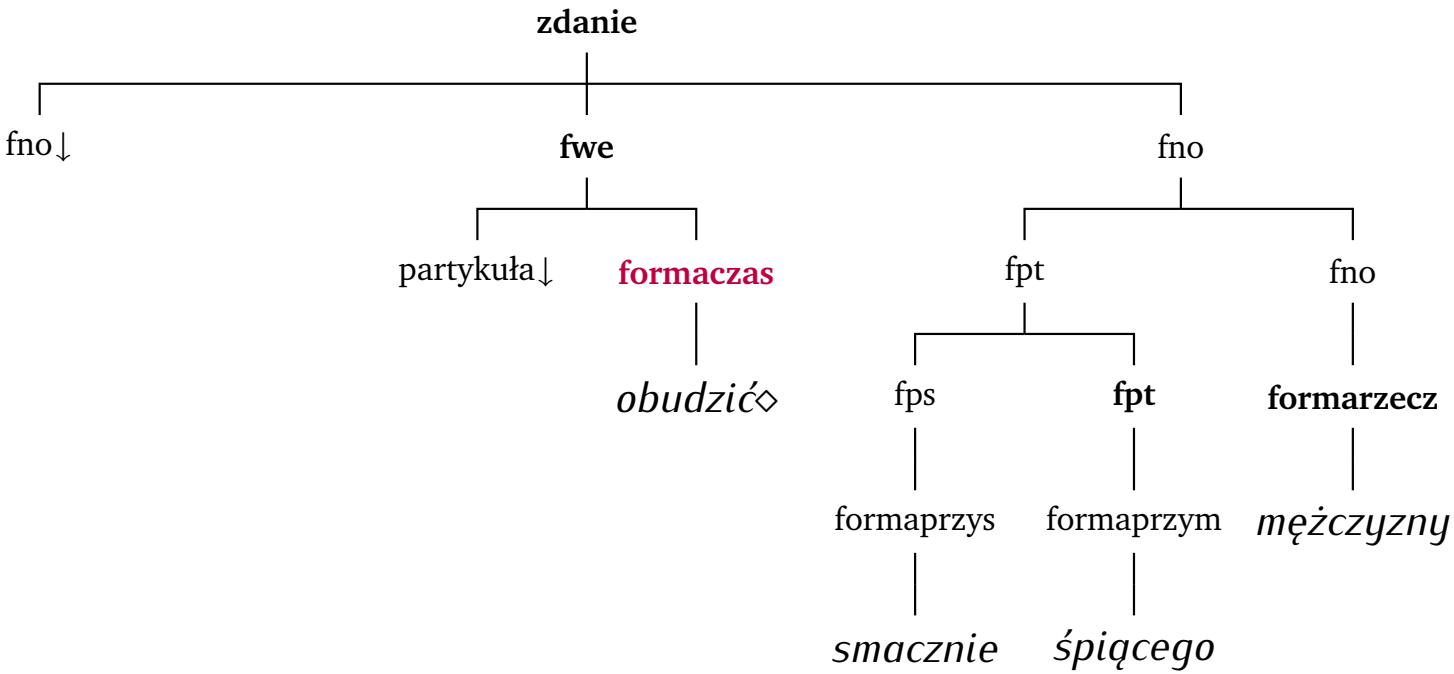
Przykład


zdanie
fno↓
fwe
fno
partykuła↓
formaczas
fpt
fno
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

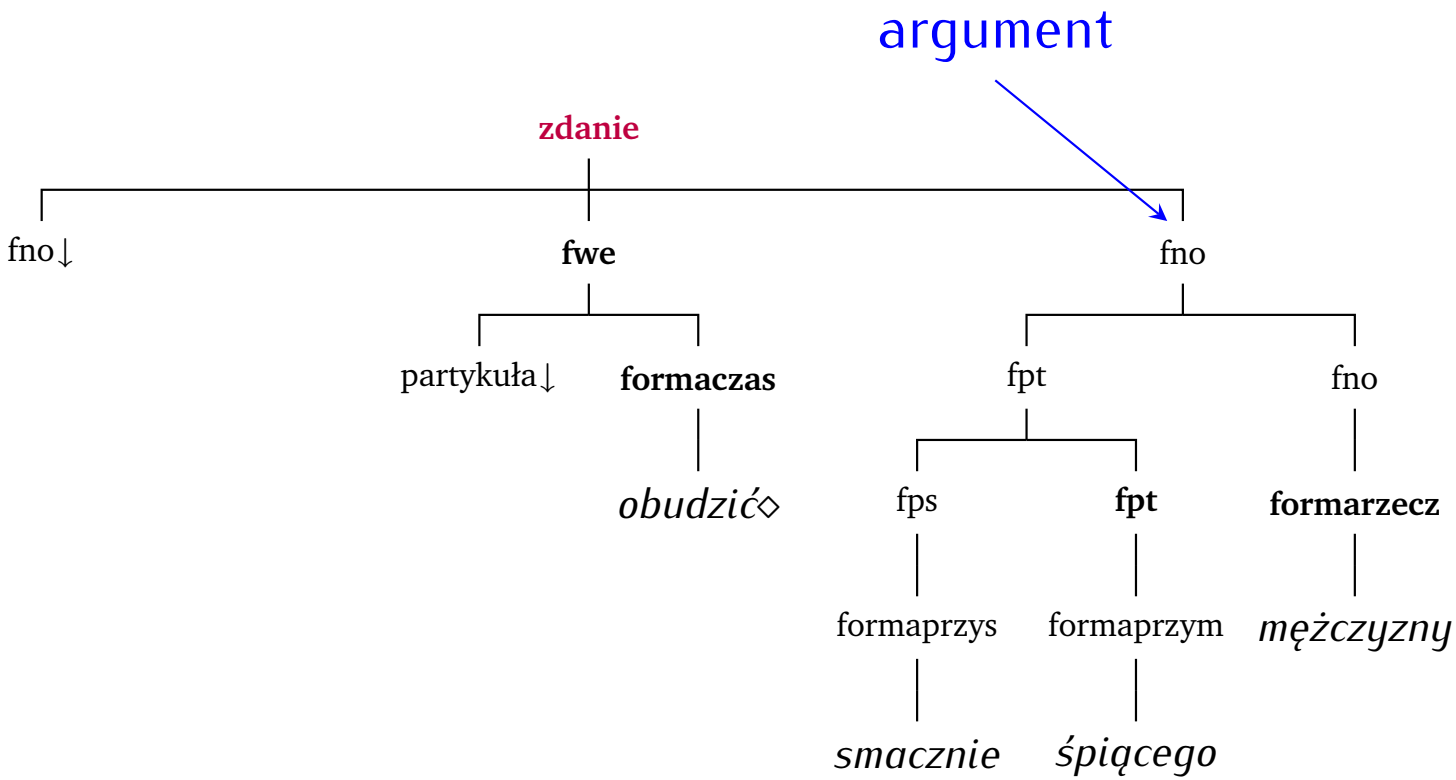
Przykład


terminal
zdanie
fno↓
fwe
fno
partykuła↓
formaczas
fpt
fno
obudził
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

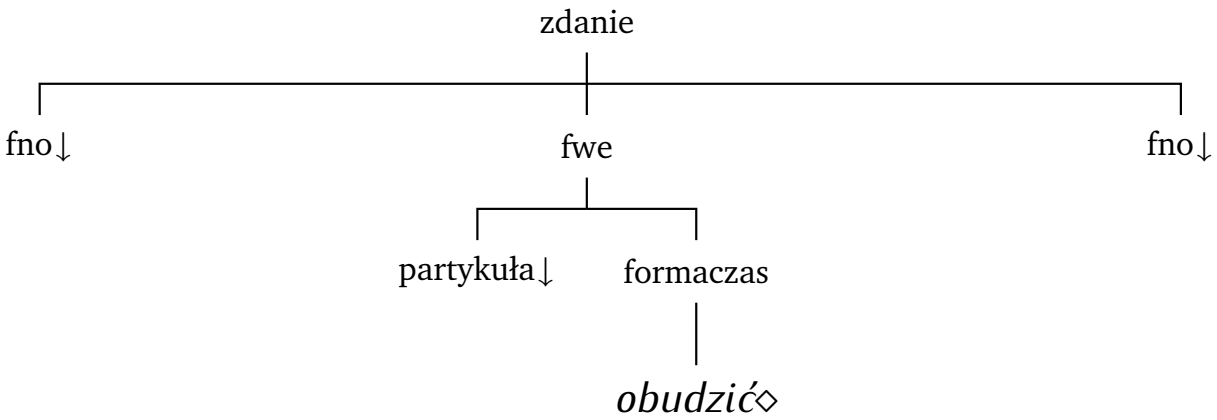
Przykład


zdanie
fno↓
fwe
fno
partykuła↓
formaczas
fpt
fno
obudzić◇
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

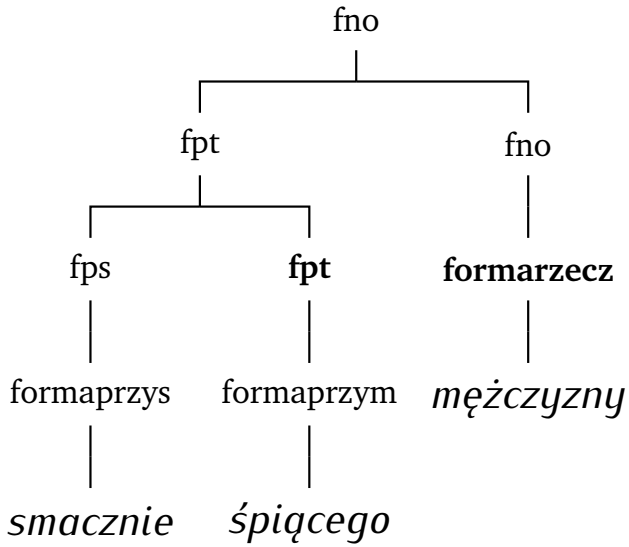
Przykład

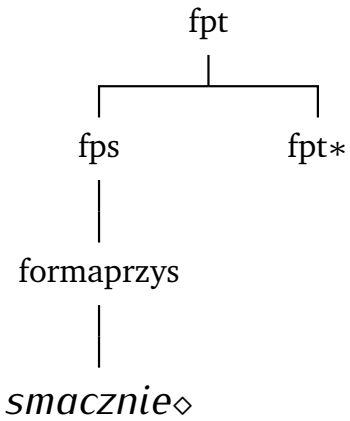

argument
zdanie
fno↓
fwe
fno
partykuła↓
formaczas
fpt
fno
obudzić◇
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego

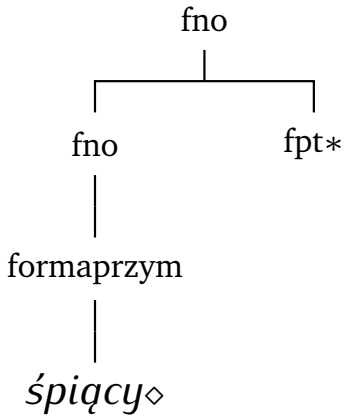
Przykład

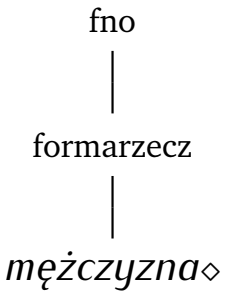


Przykład


fno
fpt
fno
fps
fpt
formarzecz
formaprzys
formaprzym
mężczyzny
smacznie
śpiącego


fpt
fps
fpt∗
formaprzys
smacznie◇


fno
fno
fpt∗
formaprzym
śpiący◇


fno
formarzecz
mężczyzna◇

Plan prezentacji

TuLiPA

Tübingen Linguistic Parsing Architecture (Kallmeyer *et al.* (2008); `https://sourcesup.cru.fr/tulipa/`):

- Parser m.in. dla gramatyk TAG
- Korzysta z 3-warstwowego opisu
- **Gramatyka** składa się z tzw. rodzin drzew
  - drzewa elementarne bez elementów leksykalnych
- **Leksykon** zawiera listę możliwych dopasowań leksemu do rodziny drzew
  - każde takie dopasowanie odpowiada zleksykalizowanemu drzewu elementarnemu
- **Morfoskładnia** dla słów z leksykonu

Fragment gramatyki

```
class ZDANIE_22c
declare ?N1 ?N2 ?N3 ?N4 ?N5 ?N6 ?V1 ?V2 ?V3 ?V4 ?V5
{ <syn> {
  node ?N1
  [cat = ZDANIE, liczba = ?V1, rodzaj = ?V2, osoba = ?V3, czas = ?V4] {
    node ?N2
    [cat = FF, liczba = ?V1, rodzaj = ?V2, osoba = ?V3, czas = ?V4] {
      node ?N3
      [cat = FWE, liczba = ?V1, rodzaj = ?V2, osoba = ?V3, czas = ?V4] {
        node ?N4
        [cat = FORMACZAS, liczba = ?V1, rodzaj = ?V2, osoba = ?V3, czas = ?V4] {
          node ?N5 (mark = anchor)
          [cat = verb, liczba = ?V1, rodzaj = ?V2, osoba = ?V3, czas = ?V4]
        }
      }
    }
    node ?N6 (mark = subst, name = substNode1)
    [cat = FPT, liczba = ?V1, przypadek = ?V5, rodzaj = ?V2]
  }
}
}
```

Fragment leksykonu

```
*ENTRY: pozostać
*CAT: verb
*SEM:
*ACC: 1
*FAM: ZDANIE_22c
*FILTERS: []
*EX: {}
*EQUATIONS:
    substNode1 -> przypadek = mian
*COANCHORS:

*ENTRY: być
*CAT: verb
*SEM:
*ACC: 1
*FAM: ZDANIE_22c
*FILTERS: []
*EX: {}
*EQUATIONS:
    substNode1 -> przypadek = mian
*COANCHORS:
```

TuLiPA-pl

Nieco zmodyfikowana wersja parsera TuLiPA:

- Wymaga tylko dwóch pierwszych warstw gramatyki
- Plik z morfoskładnią jest opcjonalny
- W przypadku jego braku TuLiPA-pl korzysta z Morfeusza.

pl-TAG

Gramatyka TAG dla języka polskiego (`http://zil.ipipan.waw.pl/plTAG`):

- Uzyskana z 7229 drzew ze *Składnicy*
- 2802 rodziny drzew
  - 1825 początkowych
  - 977 pomocniczych
- 11515 różnych słów w leksykonie
- 23570 drzew elementarnych
- Średnia liczba drzew leksykalizowanych przez słowo: 2,05
- 7953 słów (69%) leksykalizuje tylko jedno drzewo elementarne

pl-TAG – ewaluacja

wynik	zdania	%
rozbiór	2678	37%
brak rozbioru	128	2%
błąd parsera	640	9%
za mało pamięci	3697	51%
za mało czasu	44	1%

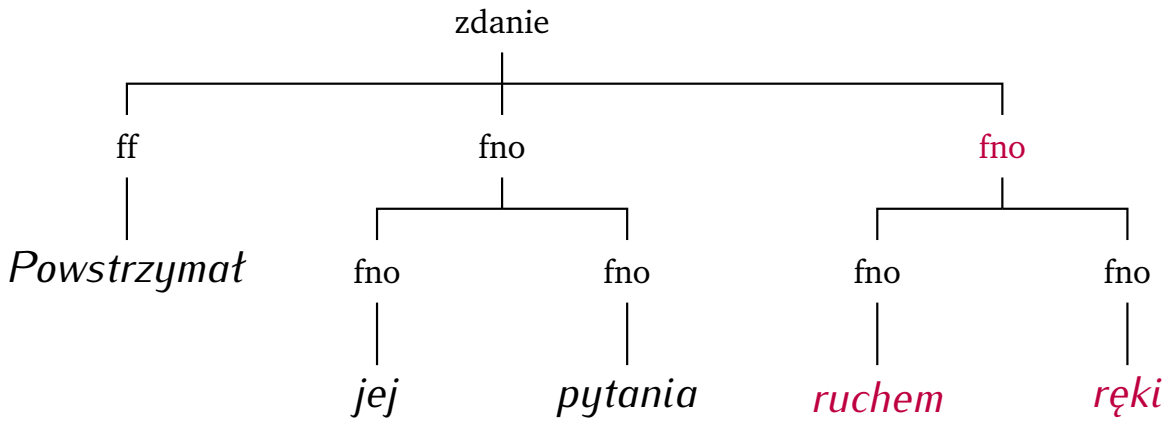
pl-TAG – ewaluacja

Porównanie z rozbiorami ze *Składnicy*:

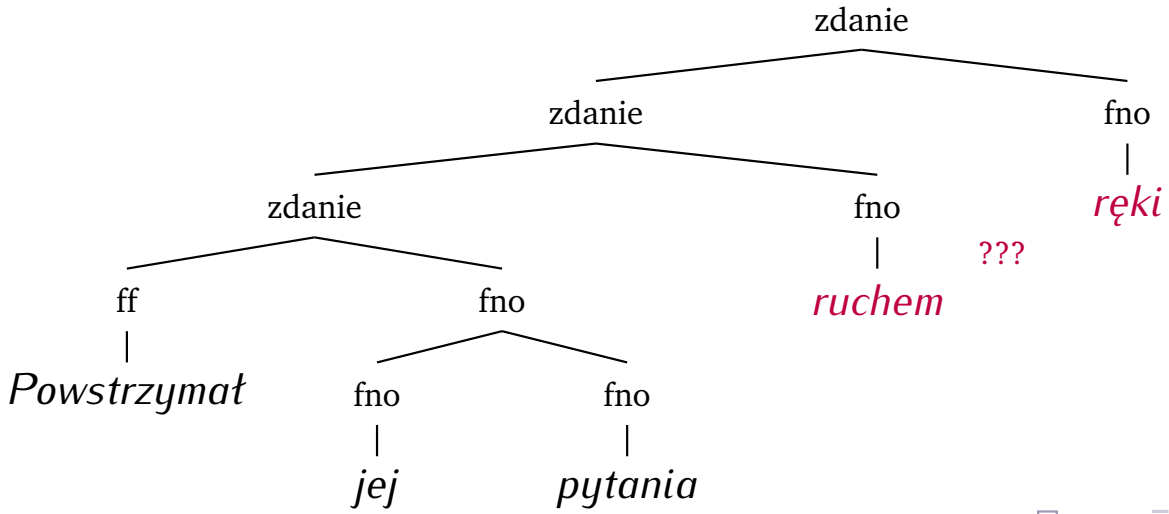
- Wybór najlepiej dopasowanego rozbioru TAG:
  - najwięcej pokrywających się kategorii przypisanych frazom
- Spośród wszystkich fraz w *Składnicy*:
  - 92% identycznie przypisanych kategorii
  - 98,8% dla niepustych rozbiorów TAG

pl-TAG – przykład niedopasowania w rozbiorze

Drzewo ze *Składnicy*:



Rozbiór uzyskany za pomocą gramatyki TAG:



Podsumowanie

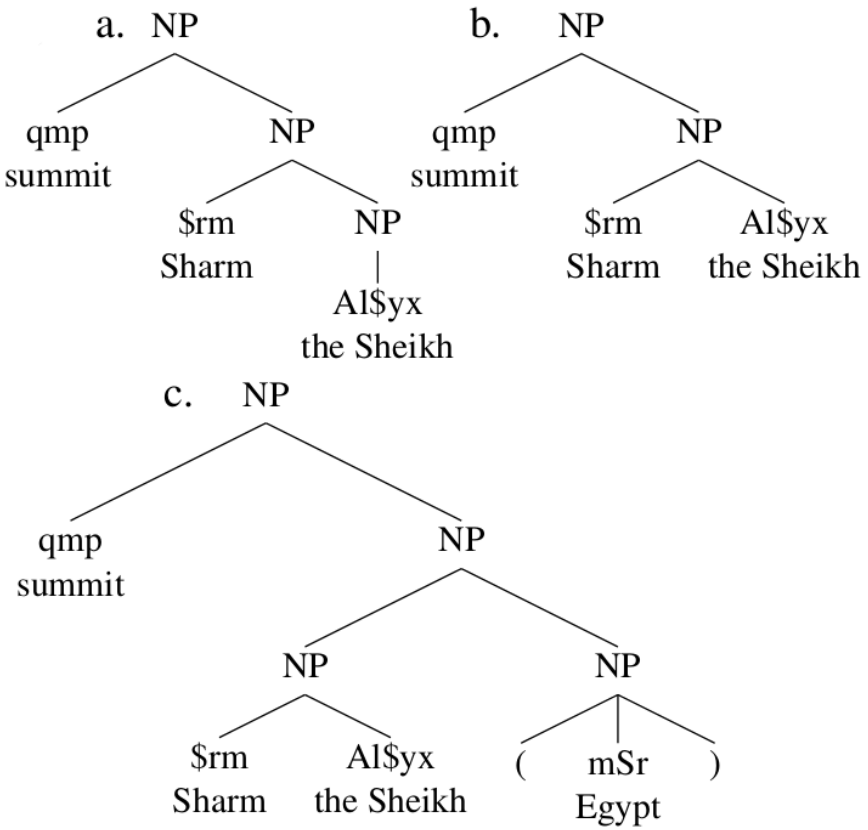
- Prawdopodobnie pierwszy taki eksperyment dla języka polskiego
- Duża (ale nie 100%) zgodność z bankiem drzew, z którego uzyskano gramatykę
- Problemy wydajnościowe
- Trudności np. ze swobodnym szykiem zdań

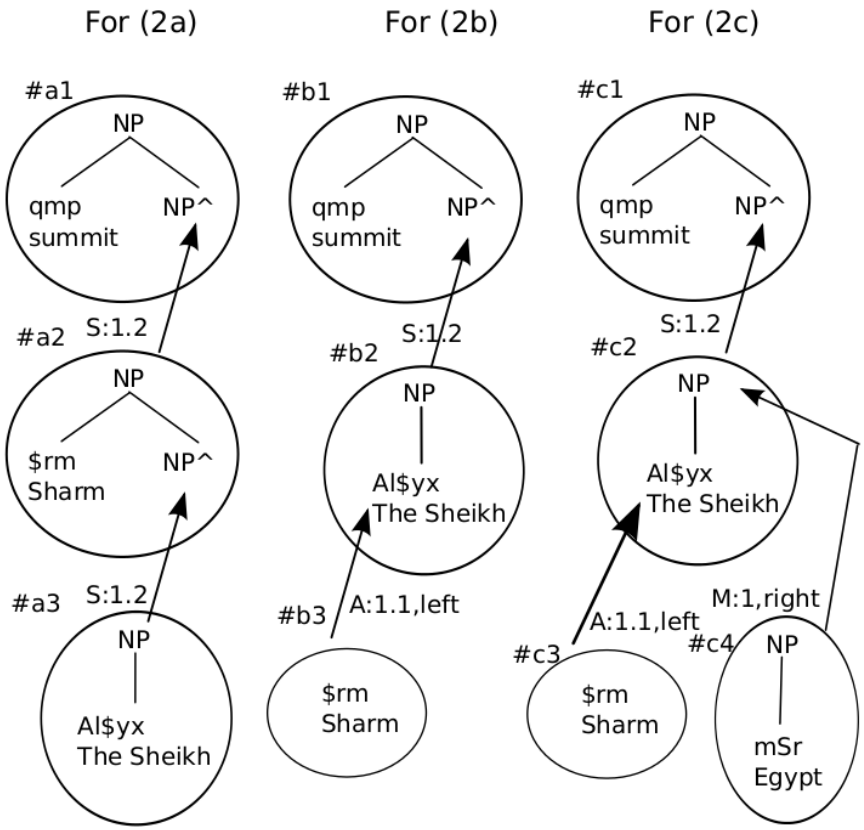
Plan prezentacji

TAG w wykrywaniu błędów

(Kulick *et al.*, 2011):

- Porównanie drzew wyprowadzenia TAG dla identycznych ciągów słów





Bibliografia

Chen, J. i Vijay-Shanker, K. (2000). Automated extraction of Tags from the Penn Treebank. W: *Proceedings of IWPT 2000*.

Joshi, A. i Schabes, Y. (1997). Tree-adjoining grammars. W: *Handbook of Formal Lanaguages and Automata*. Springer-Verlag, Berlin.

Kallmeyer, L., Lichte, T., Maier, W., Parmentier, Y., Dellert, J. i Evang, K. (2008). TuLiPA: Towards a multi-formalism parsing environment for grammar engineering. W: *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, str. 1–8, Manchester, England. Coling 2008 Organizing Committee.

Kulick, S., Bies, A. i Mott, J. (2011). Using derivation trees for treebank error detection. W: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, str. 693–698, Stroudsburg, PA, USA. Association for Computational Linguistics.