Verbal Phraseology
...To a Parser

Agnieszka Patejuk

INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES
ul. Jana Kazimierza 5, 01-248 Warszawa

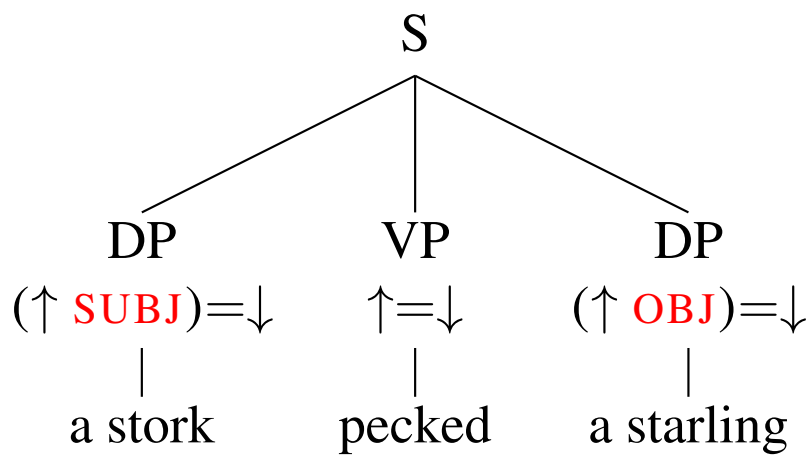
2nd PARSEME Training School
27 June – 1 July 2016, La Rochelle

Plan

- intro:
  - intro to LFG
  - about POLFIE and Walenty
  - bird's eye view of conversion
  - GF assignment
  - formalising constraints
- conversion of:
  - basic arguments (non-lexicalised)
  - lexicalised arguments

LFG formalism in a (tiny) nutshell

- *Lexical Functional Grammar* (LFG; Bresnan 1982, 2000, Dalrymple 2001),
- **generative**, but **not transformational** (declarative),
- **constraint-based**, highly **lexicalised**,
- **formalised**, **implementable** (in XLE),
- uses **parallel** levels of **representation**:



$$\begin{bmatrix} \text{PRED} & \text{`PECK}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\ \text{SUBJ } \boxed{1} & \begin{bmatrix} \text{PRED} & \text{`STORK'} \end{bmatrix} \\ \text{OBJ } \boxed{2} & \begin{bmatrix} \text{PRED} & \text{`STARLING'} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

- offers analyses of many **typologically diverse** languages (English, Warlpiri, Russian, Urdu...), **actively developed**.

Levels of representation

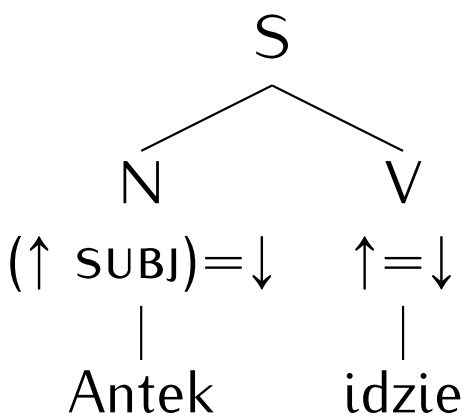
Two basic levels of representation:

- **c-structure**:
  - constituent structure – a **tree**,
  - based on **syntactic categories**,
  - **surface** structure,
  - **language dependent**;
- **f-structure**:
  - functional structure – an **attribute-value matrix**,
  - based on **grammatical functions**,
  - **deep** structure,
  - **more universal**,
  - close to **semantics** (but this is not semantics).

Other: s(emantic)-structure, a(rgument)-structure, i(nformation)-structure...

Sample structures

Antek idzie.
Antek.NOM walks



$$\begin{bmatrix} \text{PRED} & \text{`WALK}\langle \boxed{1} \rangle\text{'} \\ \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`ANTEK'} \\ \text{CASE} & \text{NOM} \\ \text{GEND} & \text{M1} \\ \text{NUM} & \text{SG} \end{bmatrix} \\ \text{TENSE} & \text{PRES} \end{bmatrix}$$

Grammatical functions

- **primitive notion** in LFG (not derived),
- **syntactic** notion,
- **may be correlated with semantics** to some degree,
- describe the relation between **the head and the dependent**,
- **independent of the position** in the tree,
- **universally available**,
- but not every language uses the entire set.

Basic GF set

- governed – **arguments**:
  - **subject**: SUBJ (*Antek walks*),
  - **complements**:
    - direct: OBJ (*Eryk likes Antek*),
    - indirect: OBJ_θ (*Antek gave Eryk a record*),
    - prepositional: OBL (*Eryk waits for Antek*),
    - clausal: COMP (*Antek says that he walks*),
    - infinitival: XCOMP (*Antek wants to walk*),
    - predicative: XCOMP-PRED (*Antek is nice*);
- ungoverned – **modifiers**:
  - plain: ADJUNCT (Antek walks *quickly*),
  - controlled: XADJUNCT (Antek walks *jumping/drunk*),
  - possessive: POSS (*Antek's sister walks*).

F-structures: more examples

Antek dał Erykowi płytę.
Antek gave Eryk record

$$\begin{bmatrix}
\text{PRED} & \text{`GIVE}\langle[1],[2],[3]\rangle\text{'} \\
\text{SUBJ} & [1]\begin{bmatrix}\text{PRED} & \text{`ANTEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBJ} & [2]\begin{bmatrix}\text{PRED} & \text{`RECORD'} \\ \text{CASE} & \text{ACC}\end{bmatrix} \\
\text{OBJ}_\theta & [3]\begin{bmatrix}\text{PRED} & \text{`ERYK'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

Antek mówi, że idzie.
Antek says that walks

$$\begin{bmatrix}
\text{PRED} & \text{`SAY}\langle[1],[2]\rangle\text{'} \\
\text{SUBJ} & [1]\begin{bmatrix}\text{PRED} & \text{`ANTEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{COMP} & [2]\begin{bmatrix}
\text{PRED} & \text{`WALK}\langle[3]\rangle\text{'} \\
\text{SUBJ} & [3]\begin{bmatrix}\text{PRED} & \text{`PRO'}\end{bmatrix} \\
\text{TENSE} & \text{PRES} \\
\text{COMP-FORM} & \text{THAT}
\end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structures: more examples

Antek dał Erykowi płytę.
Antek gave Eryk record

$$\begin{bmatrix}
\text{PRED} & \text{`GIVE}\langle[1],[2],[3]\rangle\text{'} \\
\text{SUBJ} & [1]\begin{bmatrix}\text{PRED} & \text{`ANTEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBJ} & [2]\begin{bmatrix}\text{PRED} & \text{`RECORD'} \\ \text{CASE} & \text{ACC}\end{bmatrix} \\
\text{OBJ}_\theta & [3]\begin{bmatrix}\text{PRED} & \text{`ERYK'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

Antek mówi, że idzie.
Antek says that walks

$$\begin{bmatrix}
\text{PRED} & \text{`SAY}\langle[1],[2]\rangle\text{'} \\
\text{SUBJ} & [1]\begin{bmatrix}\text{PRED} & \text{`ANTEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{COMP} & [2]\begin{bmatrix}
\text{PRED} & \text{`WALK}\langle[3]\rangle\text{'} \\
\text{SUBJ} & [3]\begin{bmatrix}\text{PRED} & \text{`PRO'}\end{bmatrix} \\
\text{TENSE} & \text{PRES} \\
\text{COMP-FORM} & \text{THAT}
\end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structures: subject vs object control

Antek kazał Erykowi iść.
Antek ordered Eryk walk

$$\begin{bmatrix}
\text{PRED} & \text{`ORDER}\langle\boxed{1},\boxed{2},\boxed{3}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{`ANTEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBJ}_\theta & \boxed{2}\begin{bmatrix}\text{PRED} & \text{`ERYK'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\
\text{XCOMP} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{`WALK}\langle\boxed{2}\rangle\text{'} \\ \text{SUBJ} & \boxed{2}\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

Antek obiecał Erykowi iść.
Antek promised Eryk walk

$$\begin{bmatrix}
\text{PRED} & \text{`PROMISE}\langle\boxed{1},\boxed{2},\boxed{3}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{`ANTEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBJ}_\theta & \boxed{2}\begin{bmatrix}\text{PRED} & \text{`ERYK'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\
\text{XCOMP} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{`WALK}\langle\boxed{1}\rangle\text{'} \\ \text{SUBJ} & \boxed{1}\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

Lexical entries

```
Antek   N   *   (^ PRED)='Antek'
                (^ NUM)= sg
                (^ CASE)= nom
                (^ GEND)= m1

idzie   V   *   (^ PRED)='walk<(^ SUBJ)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom

kazał   V   *   (^ PRED)='order<(^ SUBJ)(^ OBJ-TH)(^ XCOMP)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom
                (^ SUBJ GEND)=c m1
                (^ OBJ-TH)=(^ XCOMP SUBJ)

form    V   *   PRED
                constraints
```

Lexical entries

```
Antek   N   *   (^ PRED)='Antek'
                (^ NUM)= sg
                (^ CASE)= nom
                (^ GEND)= m1

idzie   V   *   (^ PRED)='walk<(^ SUBJ)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom

kazał   V   *   (^ PRED)='order<(^ SUBJ)(^ OBJ-TH)(^ XCOMP)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom
                (^ SUBJ GEND)=c m1
                (^ OBJ-TH)=(^ XCOMP SUBJ)

form    V   *   PRED
                constraints
```

Lexical entries

```
Antek   N   *   (^ PRED)='Antek'
                (^ NUM)= sg
                (^ CASE)= nom
                (^ GEND)= m1

idzie   V   *   (^ PRED)='walk<(^ SUBJ)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom

kazał   V   *   (^ PRED)='order<(^ SUBJ)(^ OBJ-TH)(^ XCOMP)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom
                (^ SUBJ GEND)=c m1
                (^ OBJ-TH)=(^ XCOMP SUBJ)

form    V   *   PRED
                constraints
```

Lexical entries

```
Antek   N   *   (^ PRED)='Antek'
                (^ NUM)= sg
                (^ CASE)= nom
                (^ GEND)= m1

idzie   V   *   (^ PRED)='walk<(^ SUBJ)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom

kazał   V   *   (^ PRED)='order<(^ SUBJ)(^ OBJ-TH)(^ XCOMP)>'
                (^ SUBJ NUM)=c sg
                (^ SUBJ CASE)=c nom
                (^ SUBJ GEND)=c m1
                (^ OBJ-TH)=(^ XCOMP SUBJ)

form    V   *   PRED
                constraints
```

About POLFIE

POLFIE grammar (Patejuk and Przepiórkowski 2012):

- **Lexical-Functional Grammar** approach (LFG),
- implemented **in XLE** (platform dedicated to LFG),
- created by maximising the use of existing Polish resources:
  - **previous implemented grammars**,
  - **morphosyntactic** information from **Morfeusz**,
  - **valency** information from **Walenty**,
- **used** for the construction of **structure bank**,
- uses **OT marks** for disambiguation,
- available on an open source license: **GPL3**.

About Walenty

General information:

- 84379 **schemata** for 15195 **lemmata**,
- **human- and machine-readable**,
- **framework-independent**, uses its own formalism,
- **can be converted** to various formalisms,
- schemata illustrated with **attested examples**,
- **open source**, available from: `http://walenty.ipipan.waw.pl/`,
- **formats**: plain text (syntax only), XML.

Linguistic features (Przepiórkowski *et al.* 2014):

- 2 levels: **morphosyntax**, **semantics**,
- **structural case**, **passivisation**, **control** relations,
- explicit account of **coordination** (unlike category coordination),
- some arguments defined by **semantics** rather than category (e.g. manner, location, duration, path),
- **lexicalised** arguments: rich component.

About Walenty

General information:

- 84379 **schemata** for 15195 **lemmata**,
- **human- and machine-readable**,
- **framework-independent**, uses its own formalism,
- **can be converted** to various formalisms,
- schemata illustrated with **attested examples**,
- **open source**, available from: `http://walenty.ipipan.waw.pl/`,
- **formats**: plain text (syntax only), XML.

Linguistic features (Przepiórkowski *et al.* 2014):

- 2 levels: **morphosyntax**, **semantics**,
- **structural case**, **passivisation**, **control** relations,
- explicit account of **coordination** (unlike category coordination),
- some arguments defined by **semantics** rather than category (e.g. manner, location, duration, path),
- **lexicalised** arguments: rich component.

Conversion: bird's eye view

- choose the grammatical function for each dependent
- construct the PRED attribute:
  - lemma
  - arguments: semantic vs non-semantic
  - markers (e.g. reflexive) not included
- impose relevant constraints:
  - for each dependent
  - for each realisation
- further:
  - passive: create active/passive schemata
  - reduction: decide which dependents are obligatory

Conversion: bird's eye view

- choose the grammatical function for each dependent
- construct the PRED attribute:
  - lemma
  - arguments: semantic vs non-semantic
  - markers (e.g. reflexive) not included
- impose relevant constraints:
  - for each dependent
  - for each realisation
- further:
  - passive: create active/passive schemata
  - reduction: decide which dependents are obligatory

Conversion: bird's eye view

- choose the grammatical function for each dependent
- construct the PRED attribute:
  - lemma
  - arguments: semantic vs non-semantic
  - markers (e.g. reflexive) not included
- impose relevant constraints:
  - for each dependent
  - for each realisation
- further:
  - passive: create active/passive schemata
  - reduction: decide which dependents are obligatory

Conversion: bird's eye view

- choose the grammatical function for each dependent
- construct the PRED attribute:
  - lemma
  - arguments: semantic vs non-semantic
  - markers (e.g. reflexive) not included
- impose relevant constraints:
  - for each dependent
  - for each realisation
- further:
  - passive: create active/passive schemata
  - reduction: decide which dependents are obligatory

Conversion: bird's eye view

- choose the grammatical function for each dependent
- construct the PRED attribute:
  - lemma
  - arguments: semantic vs non-semantic
  - markers (e.g. reflexive) not included
- impose relevant constraints:
  - for each dependent
  - for each realisation
- further:
  - passive: create active/passive schemata
  - reduction: decide which dependents are obligatory

Conversion: bird's eye view

- choose the grammatical function for each dependent
- construct the PRED attribute:
  - lemma
  - arguments: semantic vs non-semantic
  - markers (e.g. reflexive) not included
- impose relevant constraints:
  - for each dependent
  - for each realisation
- further:
  - passive: create active/passive schemata
  - reduction: decide which dependents are obligatory

Choosing the grammatical function

Only 2 GFs marked in Walenty (for a reason):

- subject
- passivisable object

GF assignment depends on:

- the entire set of realisations
- control relations
- other GFs assigned (GF uniqueness)

Choosing the grammatical function

Only 2 GFs marked in Walenty (for a reason):

- subject
- passivisable object

GF assignment depends on:

- the entire set of realisations
- control relations
- other GFs assigned (GF uniqueness)

Only one realisation

- [np ∨ ncp ∨ adjp] ∧
  - controllee → XCOMP-PRED
  - case == dat → OBJ-TH
  - [case == str ∨ case == part] → OBL-STR
  - case == gen → OBL-GEN
  - case == inst → OBL-INST
- [prepnp ∨ prepncp ∨ prepadjp ∨ comprepnp] →
  - controllee → XCOMP-PRED
  - OBL (numerical index is appended when there is more than one argument of this type: OBL2, OBL3, etc.)
- cp → COMP
- infp → XCOMP
- advp → OBL-ADV
- xp(sem) → OBL-SEM (e.g. xp(abl) → OBL-ABL)
- refl → marker (co-head, not a GF)

Only one realisation

- [np ∨ ncp ∨ adjp] ∧
  - controllee → XCOMP-PRED
  - case == dat → OBJ-TH
  - [case == str ∨ case == part] → OBL-STR
  - case == gen → OBL-GEN
  - case == inst → OBL-INST
- [prepnp ∨ prepncp ∨ prepadjp ∨ comprepnp] →
  - controllee → XCOMP-PRED
  - OBL (numerical index is appended when there is more than one argument of this type: OBL2, OBL3, etc.)
- cp → COMP
- infp → XCOMP
- advp → OBL-ADV
- xp(sem) → OBL-SEM (e.g. xp(abl) → OBL-ABL)
- refl → marker (co-head, not a GF)

Only one realisation

- [np ∨ ncp ∨ adjp] ∧
  - controllee → XCOMP-PRED
  - case == dat → OBJ-TH
  - [case == str ∨ case == part] → OBL-STR
  - case == gen → OBL-GEN
  - case == inst → OBL-INST
- [prepnp ∨ prepncp ∨ prepadjp ∨ comprepnp] →
  - controllee → XCOMP-PRED
  - OBL (numerical index is appended when there is more than one argument of this type: OBL2, OBL3, etc.)
- cp → COMP
- infp → XCOMP
- advp → OBL-ADV
- xp(sem) → OBL-SEM (e.g. xp(abl) → OBL-ABL)
- refl → marker (co-head, not a GF)

Only one realisation

- [np ∨ ncp ∨ adjp] ∧
  - controllee → XCOMP-PRED
  - case == dat → OBJ-TH
  - [case == str ∨ case == part] → OBL-STR
  - case == gen → OBL-GEN
  - case == inst → OBL-INST
- [prepnp ∨ prepncp ∨ prepadjp ∨ comprepnp] →
  - controllee → XCOMP-PRED
  - OBL (numerical index is appended when there is more than one argument of this type: OBL2, OBL3, etc.)
- cp → COMP
- infp → XCOMP
- advp → OBL-ADV
- xp(sem) → OBL-SEM (e.g. xp(abl) → OBL-ABL)
- refl → marker (co-head, not a GF)

Only one realisation

- [np ∨ ncp ∨ adjp] ∧
  - controllee → XCOMP-PRED
  - case == dat → OBJ-TH
  - [case == str ∨ case == part] → OBL-STR
  - case == gen → OBL-GEN
  - case == inst → OBL-INST
- [prepnp ∨ prepncp ∨ prepadjp ∨ comprepnp] →
  - controllee → XCOMP-PRED
  - OBL (numerical index is appended when there is more than one argument of this type: OBL2, OBL3, etc.)
- cp → COMP
- infp → XCOMP
- advp → OBL-ADV
- xp(sem) → OBL-SEM (e.g. xp(abl) → OBL-ABL)
- refl → marker (co-head, not a GF)

Only one realisation

- [np ∨ ncp ∨ adjp] ∧
  - controllee → XCOMP-PRED
  - case == dat → OBJ-TH
  - [case == str ∨ case == part] → OBL-STR
  - case == gen → OBL-GEN
  - case == inst → OBL-INST
- [prepnp ∨ prepncp ∨ prepadjp ∨ comprepnp] →
  - controllee → XCOMP-PRED
  - OBL (numerical index is appended when there is more than one argument of this type: OBL2, OBL3, etc.)
- cp → COMP
- infp → XCOMP
- advp → OBL-ADV
- xp(sem) → OBL-SEM (e.g. xp(abl) → OBL-ABL)
- refl → marker (co-head, not a GF)

Only one realisation

- [np ∨ ncp ∨ adjp] ∧
  - controllee → XCOMP-PRED
  - case == dat → OBJ-TH
  - [case == str ∨ case == part] → OBL-STR
  - case == gen → OBL-GEN
  - case == inst → OBL-INST
- [prepnp ∨ prepncp ∨ prepadjp ∨ comprepnp] →
  - controllee → XCOMP-PRED
  - OBL (numerical index is appended when there is more than one argument of this type: OBL2, OBL3, etc.)
- cp → COMP
- infp → XCOMP
- advp → OBL-ADV
- xp(sem) → OBL-SEM (e.g. xp(abl) → OBL-ABL)
- refl → marker (co-head, not a GF)

Examples: ADRESOWAĆ ‘address’

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  ‘Jan addressed a/the letter to Maria.’

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

  GF assignment:
  - `subj` → SUBJ (given)
  - `obj` → OBJ (given)
  - `prepnp` → OBL (no control)

Examples: ADRESOWAĆ 'address'

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

  GF assignment:
  - `subj` → SUBJ (given)
  - `obj` → OBJ (given)
  - `prepnp` → OBL (no control)

Examples: ADRESOWAĆ ‘address’

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  ‘Jan addressed a/the letter to Maria.’

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

  GF assignment:
  - `subj` → SUBJ (given)
  - `obj` → OBJ (given)
  - `prepnp` → OBL (no control)

Examples: ADRESOWAĆ ‘address’

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  ‘Jan addressed a/the letter to Maria.’

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

  GF assignment:
  - `subj` → SUBJ (given)
  - `obj` → OBJ (given)
  - `prepnp` → OBL (no control)

Examples: ADRESOWAĆ 'address'

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

  GF assignment:
  - `subj` → SUBJ (given)
  - `obj` → OBJ (given)
  - `prepnp` → OBL (no control)

Examples: ADRESOWAĆ 'address'

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

  GF assignment:
  - `subj` → SUBJ (given)
  - `obj` → OBJ (given)
  - `prepnp` → OBL (no control)

Examples: ADRESOWAĆ 'address'

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

  GF assignment:
  - `subj` → SUBJ (given)
  - `obj` → OBJ (given)
  - `prepnp` → OBL (no control)

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

GF assignment:

- `subj` → SUBJ (given)
- `np(dat)` → OBJ-TH
- `infp` → XCOMP (controlled by OBJ-TH)

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

GF assignment:

- `subj` → SUBJ (given)
- `np(dat)` → OBJ-TH
- `infp` → XCOMP (controlled by OBJ-TH)

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

GF assignment:

- `subj` → SUBJ (given)
- `np(dat)` → OBJ-TH
- `infp` → XCOMP (controlled by OBJ-TH)

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

GF assignment:

- `subj` → SUBJ (given)
- `np(dat)` → OBJ-TH
- `infp` → XCOMP (controlled by OBJ-TH)

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

GF assignment:

- `subj` → SUBJ (given)
- `np(dat)` → OBJ-TH
- `infp` → XCOMP (controlled by OBJ-TH)

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

GF assignment:

- `subj` → SUBJ (given)
- `xp(mod)` → OBL-MOD
- `xp(locat)` → OBL-LOCAT

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

GF assignment:

- `subj` → SUBJ (given)
- `xp(mod)` → OBL-MOD
- `xp(locat)` → OBL-LOCAT

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

GF assignment:

- `subj` → SUBJ (given)
- `xp(mod)` → OBL-MOD
- `xp(locat)` → OBL-LOCAT

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

GF assignment:

- `subj` → SUBJ (given)
- `xp(mod)` → OBL-MOD
- `xp(locat)` → OBL-LOCAT

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

GF assignment:

- `subj` → SUBJ (given)
- `xp(mod)` → OBL-MOD
- `xp(locat)` → OBL-LOCAT

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

GF assignment:

- `subj` → SUBJ (given)
- `xp(mod)` → OBL-MOD
- `xp(locat)` → OBL-LOCAT

More than one realisation

- for each realisation, choose the GF as in the previous slide
- make a list of candidate GFs
- choose the highest ranked GF:

#	GF
4	OBL-<SEM>, OBL-ADV
3	OBL
2	OBL-GEN, OBL-INST, OBL-STR, OBJ-TH
1	COMP, XCOMP

Result: COMP and XCOMP treated as elsewhere GFs

Examples: BAĆ SIĘ 'fear'

- Boisz się bezrobocia i że zabraknie Ci środków na utrzymanie?
  fear.2.SG RM unemployment.GEN and that lack you means for subsistence
  'Are you afraid of unemployment and that you'll have no means of subsistence?'
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2),
  - `cp(że)` → COMP (1)

Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na utrzymanie?
  fear.2.SG RM unemployment.GEN and that lack you means for subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2),
  - `cp(że)` → COMP (1)

Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2),
  - `cp(że)` → COMP (1)

Examples: BAĆ SIĘ 'fear'

- Boisz się bezrobocia i że zabraknie Ci środków na utrzymanie?
  fear.2.SG RM unemployment.GEN and that lack you means for subsistence
  'Are you afraid of unemployment and that you'll have no means of subsistence?'
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2),
  - `cp(że)` → COMP (1)

Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2),
  - `cp(że)` → COMP (1)

Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2),
  - `cp(że)` → COMP (1)

Examples: BAĆ SIĘ 'fear'

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  'Are you afraid of unemployment and that you'll have no means of subsistence?'
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2),
  - `cp(że)` → COMP (1)

Examples: BAĆ SIĘ 'fear'

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  'Are you afraid of unemployment and that you'll have no means of subsistence?'
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Some features:

- inherent reflexive marker is part of lemma (unlike reflexive pronouns),
- syntactic positions explicitly defined via the coordination test,
- `subj` → SUBJ (given),
- two set elements, ranking used:
  - `np(gen)` → OBL-GEN (2) → WINS,
  - `cp(że)` → COMP (1)

Constraint types

- **defining**:
  - assign a value to an attribute
  - `(PATH ATTR)= val`
  - `(^ NUM)= sg`
- **checking**:
  - check the value of an attribute
  - `(PATH ATTR)=c val`
  - `(^ NUM)=c sg`
- **existential**:
  - check that the attribute is present (no matter the value)
  - `(PATH ATTR)`
  - `(^ NUM)`

More notation

- conjunction:
  - `A B`
  - `(^ NUM)=c sg (^ CASE)=c nom`
- disjunction:
  - `{ A | B }`
  - `{(^ NUM)=c sg | (^ CASE)=c nom}`
- negation:
  - `~A`
  - `~(^ NUM)=c sg`
  - `(^ NUM)~= sg`

Two ways of formalising constraints

How to formalise *"Case of GF is accusative or genitive"*?

- GF is non-coordinate: $\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}$ $\boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}$
- GF is coordinate:
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}\right\}$
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}\right\}$
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}\right\}$

- `{(^ GF CASE)=c acc | (^ GF CASE)=c gen}`
- `(^ GF PRED: {(<- CASE)=c acc | (<- CASE)=c gen})`

Two ways of formalising constraints

How to formalise *"Case of GF is accusative or genitive"*?

- GF is non-coordinate: $\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}$ $\boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}$
- GF is coordinate:
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}\right\}$
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}\right\}$
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}\right\}$

- `{(^ GF CASE)=c acc | (^ GF CASE)=c gen}`
- `(^ GF PRED: {(<- CASE)=c acc | (<- CASE)=c gen})`

Two ways of formalising constraints

How to formalise *"Case of GF is accusative or genitive"*?

- GF is non-coordinate: $\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}$ $\boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}$
- GF is coordinate:
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}\right\}$
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}\right\}$
  - $\boxed{0}\left\{\boxed{1}\begin{bmatrix}\text{CASE} & \text{ACC}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CASE} & \text{GEN}\end{bmatrix}\right\}$

- `{(^ GF CASE)=c acc | (^ GF CASE)=c gen}`
- `(^ GF PRED: {(<- CASE)=c acc | (<- CASE)=c gen})`

Examples: ADRESOWAĆ 'address' – prepositional complement

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- preposition is non-semantic: `PFORM`, no `PRED`
- GF: `prepnp` → OBL
- constraints for `prepnp`:
  - preposition form is DO: `(^ OBL PFORM)=c do`
  - case required by the preposition is GEN: `(^ OBL CASE)=c gen`

Examples: ADRESOWAĆ 'address' – prepositional complement

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- preposition is non-semantic: `PFORM`, no `PRED`
- GF: `prepnp` → OBL
- constraints for `prepnp`:
  - preposition form is DO: `(^ OBL PFORM)=c do`
  - case required by the preposition is GEN: `(^ OBL CASE)=c gen`

Examples: ADRESOWAĆ 'address' – prepositional complement

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- preposition is non-semantic: `PFORM`, no `PRED`
- GF: `prepnp` → OBL
- constraints for `prepnp`:
  - preposition form is DO: `(^ OBL PFORM)=c do`
  - case required by the preposition is GEN: `(^ OBL CASE)=c gen`

Examples: ADRESOWAĆ 'address' – prepositional complement

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- preposition is non-semantic: `PFORM`, no `PRED`
- GF: `prepnp` → OBL
- constraints for `prepnp`:
  - preposition form is DO: `(^ OBL PFORM)=c do`
  - case required by the preposition is GEN: `(^ OBL CASE)=c gen`

Examples: ADRESOWAĆ 'address' – prepositional complement

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- preposition is non-semantic: `PFORM`, no `PRED`
- GF: `prepnp` → OBL
- constraints for `prepnp`:
  - preposition form is DO: `(^ OBL PFORM)=c do`
  - case required by the preposition is GEN: `(^ OBL CASE)=c gen`

Examples: ADRESOWAĆ 'address' – prepositional complement

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- preposition is non-semantic: `PFORM`, no `PRED`
- GF: `prepnp` → OBL
- constraints for `prepnp`:
  - preposition form is DO: `(^ OBL PFORM)=c do`
  - case required by the preposition is GEN: `(^ OBL CASE)=c gen`

Examples: ADRESOWAĆ 'address' – structural subject (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: SUBJ (given)
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when gerund head: `{(^ CAT)=c ger (^ SUBJ CASE)=c gen`
  - otherwise NOM: `| (^ CAT)~= ger (^ SUBJ CASE)=c nom}`

Examples: ADRESOWAĆ 'address' – structural subject (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: SUBJ (given)
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when gerund head: `{(^ CAT)=c ger (^ SUBJ CASE)=c gen`
  - otherwise NOM: `| (^ CAT)~= ger (^ SUBJ CASE)=c nom}`

Examples: ADRESOWAĆ 'address' – structural subject (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: SUBJ (given)
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when gerund head: `{(^ CAT)=c ger (^ SUBJ CASE)=c gen`
  - otherwise NOM: `| (^ CAT)~= ger (^ SUBJ CASE)=c nom}`

Examples: ADRESOWAĆ 'address' – structural subject (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: SUBJ (given)
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when gerund head: `{(^ CAT)=c ger (^ SUBJ CASE)=c gen`
  - otherwise NOM: `| (^ CAT)~= ger (^ SUBJ CASE)=c nom}`

Examples: ADRESOWAĆ 'address' – structural subject (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: SUBJ (given)
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when gerund head: `{(^ CAT)=c ger (^ SUBJ CASE)=c gen`
  - otherwise NOM: `| (^ CAT)~= ger (^ SUBJ CASE)=c nom}`

Examples: ADRESOWAĆ 'address' – structural object (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: OBJ (given) → passivisation possible
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when negation present: `{(^ NEG)=c + (^ OBJ CASE)=c gen`
  - otherwise ACC: `| ~(^ NEG) (^ OBJ CASE)=c acc}`

Examples: ADRESOWAĆ 'address' – structural object (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: OBJ (given) → passivisation possible
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when negation present: `{(^ NEG)=c + (^ OBJ CASE)=c gen`
  - otherwise ACC: `| ~(^ NEG) (^ OBJ CASE)=c acc}`

Examples: ADRESOWAĆ 'address' – structural object (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: OBJ (given) → passivisation possible
- constraint for `np`: case – structural, depends on syntactic context:
  - `gen` when negation present: `{(^ NEG)=c + (^ OBJ CASE)=c gen`
  - otherwise ACC: `| ~(^ NEG) (^ OBJ CASE)=c acc}`

Examples: ADRESOWAĆ 'address' – structural object (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: OBJ (given) → passivisation possible
- constraint for `np`: case – structural, depends on syntactic context:
  - gen when negation present: `{(^ NEG)=c + (^ OBJ CASE)=c gen`
  - otherwise ACC: `| ~(^ NEG) (^ OBJ CASE)=c acc}`

Examples: ADRESOWAĆ 'address' – structural object (simplified)

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- `adresować: _: imperf:`
  `subj{np(str)} + obj{np(str)} + {prepnp(do,gen)}`

- non-coordinate position
- GF: OBJ (given) → passivisation possible
- constraint for `np`: case – structural, depends on syntactic context:
  - `gen` when negation present: `{(^ NEG)=c + (^ OBJ CASE)=c gen`
  - otherwise ACC: `| ~(^ NEG) (^ OBJ CASE)=c acc}`

Examples: ADRESOWAĆ 'address' – passivisation

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- List był adresowany do Marii (przez Jana).
  letter.NOM was addressed to Maria.GEN by Jan.ACC
  'A/The letter was addressed to Maria (by Jan).'

The GF assignment changes under passive voice:

- active object becomes the passive subject:
  `(^ OBJ) --> (^ SUBJ)`
- active subject:
  - becomes the passive oblique: `(^ SUBJ) --> (^ OBL-AG)`
  - or is dropped: `(^ SUBJ) --> NULL`

Examples: ADRESOWAĆ 'address' – passivisation

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'

- List był adresowany do Marii (przez Jana).
  letter.NOM was addressed to Maria.GEN by Jan.ACC
  'A/The letter was addressed to Maria (by Jan).'

The GF assignment changes under passive voice:

- active object becomes the passive subject:
  `(^ OBJ) --> (^ SUBJ)`
- active subject:
  - becomes the passive oblique: `(^ SUBJ) --> (^ OBL-AG)`
  - or is dropped: `(^ SUBJ) --> NULL`

Examples: ADRESOWAĆ 'address' – passivisation

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'
- List był adresowany do Marii (przez Jana).
  letter.NOM was addressed to Maria.GEN by Jan.ACC
  'A/The letter was addressed to Maria (by Jan).'

The GF assignment changes under passive voice:

- active object becomes the passive subject:
  `(^ OBJ) --> (^ SUBJ)`
- active subject:
  - becomes the passive oblique: `(^ SUBJ) --> (^ OBL-AG)`
  - or is dropped: `(^ SUBJ) --> NULL`

Examples: ADRESOWAĆ 'address' – passivisation

- Jan adresował list do Marii.
  Jan.NOM addressed letter.ACC to Maria.GEN
  'Jan addressed a/the letter to Maria.'
- List był adresowany do Marii (przez Jana).
  letter.NOM was addressed to Maria.GEN by Jan.ACC
  'A/The letter was addressed to Maria (by Jan).'

The GF assignment changes under passive voice:

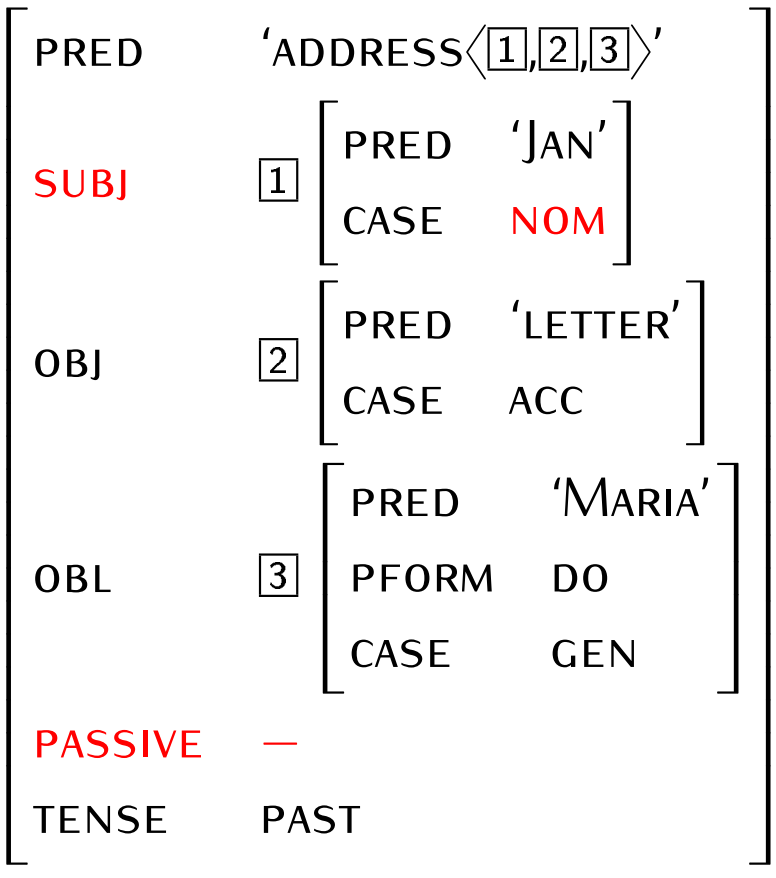
- active object becomes the passive subject:
  `(^ OBJ) --> (^ SUBJ)`
- active subject:
  - becomes the passive oblique: `(^ SUBJ) --> (^ OBL-AG)`
  - or is dropped: `(^ SUBJ) --> NULL`

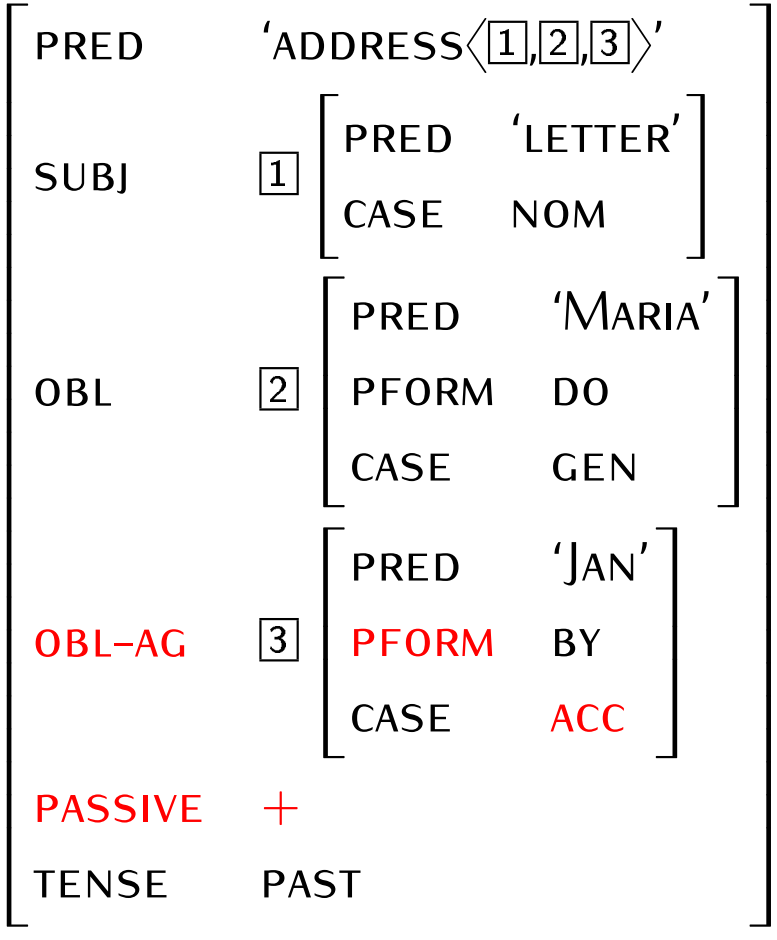
F-structures for ADRESOWAĆ: active and passive

$$\begin{bmatrix}
\text{PRED} & \text{'ADDRESS}\langle\boxed{1},\boxed{2},\boxed{3}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{'JAN'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBJ} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{'LETTER'} \\ \text{CASE} & \text{ACC}\end{bmatrix} \\
\text{OBL} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{'MARIA'} \\ \text{PFORM} & \text{DO} \\ \text{CASE} & \text{GEN}\end{bmatrix} \\
\text{PASSIVE} & - \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

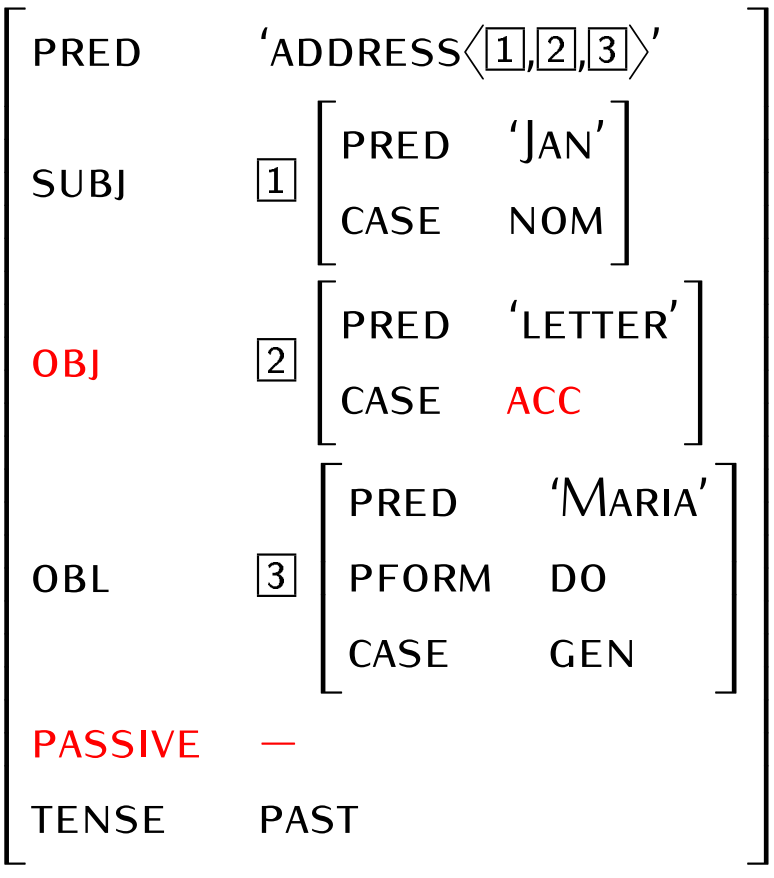
$$\begin{bmatrix}
\text{PRED} & \text{'ADDRESS}\langle\boxed{1},\boxed{2},\boxed{3}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{'LETTER'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBL} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{'MARIA'} \\ \text{PFORM} & \text{DO} \\ \text{CASE} & \text{GEN}\end{bmatrix} \\
\text{OBL-AG} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{'JAN'} \\ \text{PFORM} & \text{BY} \\ \text{CASE} & \text{ACC}\end{bmatrix} \\
\text{PASSIVE} & + \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

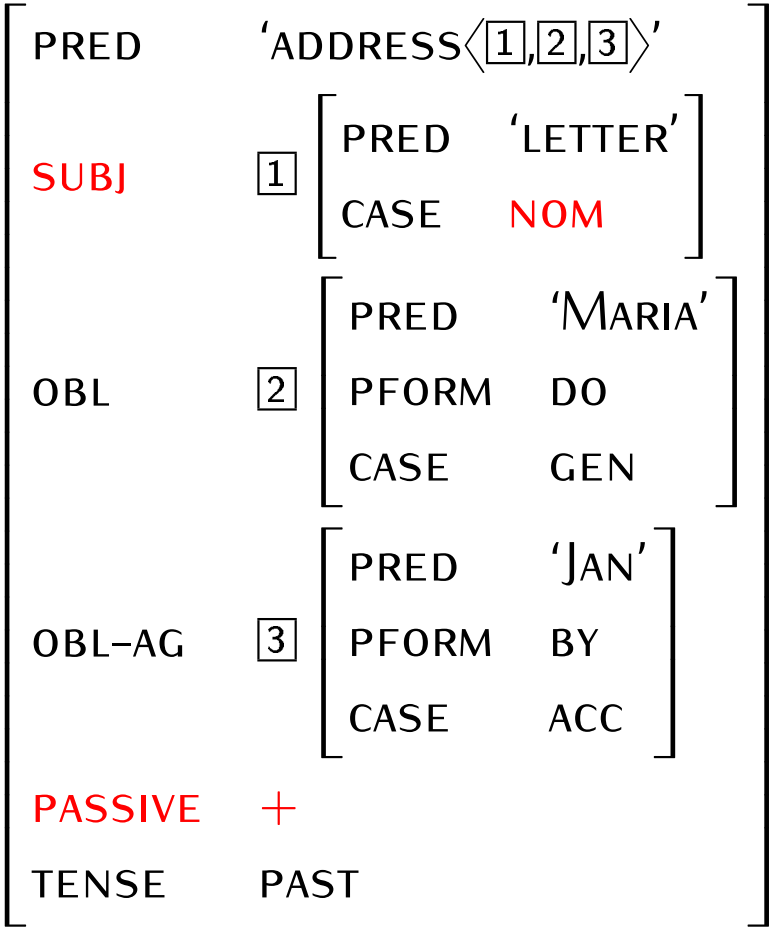
F-structures for ADRESOWAĆ: active and passive





F-structures for ADRESOWAĆ: active and passive





Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na utrzymanie?
  fear.2.SG RM unemployment.GEN and that lack you means for subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Constraints:

- subj → SUBJ (given),
- coordinate position (by definition):
  - GF: OBL-GEN,
  - np(gen): (^ GF CASE)=c gen,
  - cp(że): (^ GF COMP-FORM)=c że,
  - off-path constraint:
    (^ GF PRED: {(<- CASE)=c gen | (<- COMP-FORM)=c że})

Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Constraints:

- `subj` → SUBJ (given),
- coordinate position (by definition):
  - GF: OBL-GEN,
  - `np(gen): (^ GF CASE)=c gen`,
  - `cp(że): (^ GF COMP-FORM)=c że`,
  - off-path constraint:
    `(^ GF PRED: {(<- CASE)=c gen | (<- COMP-FORM)=c że})`

Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Constraints:

- `subj` → SUBJ (given),
- coordinate position (by definition):
  - GF: OBL-GEN,
  - `np(gen): (^ GF CASE)=c gen`,
  - `cp(że): (^ GF COMP-FORM)=c że`,
  - off-path constraint:
    `(^ GF PRED: {(<- CASE)=c gen | (<- COMP-FORM)=c że})`

Examples: BAĆ SIĘ ‘fear’

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  ‘Are you afraid of unemployment and that you’ll have no means of subsistence?’
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Constraints:

- `subj` → SUBJ (given),
- coordinate position (by definition):
  - GF: OBL-GEN,
  - `np(gen)`: `(^ GF CASE)=c gen`,
  - `cp(że)`: `(^ GF COMP-FORM)=c że`,
  - off-path constraint:
    `(^ GF PRED: {(<- CASE)=c gen | (<- COMP-FORM)=c że})`

Examples: BAĆ SIĘ 'fear'

- Boisz się bezrobocia i że zabraknie Ci środków na utrzymanie?
  fear.2.SG RM unemployment.GEN and that lack you means for subsistence
  'Are you afraid of unemployment and that you'll have no means of subsistence?'
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Constraints:

- `subj` → SUBJ (given),
- coordinate position (by definition):
  - GF: OBL-GEN,
  - `np(gen)`: `(^ GF CASE)=c gen`,
  - `cp(że)`: `(^ GF COMP-FORM)=c że`,
  - off-path constraint:
    `(^ GF PRED: {(<- CASE)=c gen | (<- COMP-FORM)=c że})`

Examples: BAĆ SIĘ 'fear'

- Boisz się bezrobocia i że zabraknie Ci środków na
  fear.2.SG RM unemployment.GEN and that lack you means for
  utrzymanie?
  subsistence
  'Are you afraid of unemployment and that you'll have no means of subsistence?'
- `bać się: _: imperf:`
  `subj{np(str)} + {np(gen); cp(że)}`

Constraints:

- `subj` → SUBJ (given),
- coordinate position (by definition):
  - GF: OBL-GEN,
  - `np(gen): (^ GF CASE)=c gen`,
  - `cp(że): (^ GF COMP-FORM)=c że`,
  - off-path constraint:
    `(^ GF PRED: {(<- CASE)=c gen | (<- COMP-FORM)=c że})`

F-structure for BAĆ SIĘ: unlike coordination

$$\begin{bmatrix}
\text{PRED} & \text{`FEAR}\langle\boxed{1},\boxed{2}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{`PRO'}\end{bmatrix} \\
\text{OBL}_{gen} & \boxed{2}\begin{bmatrix}
\left\{\begin{bmatrix}\text{PRED} & \text{`UNEMPLOYMENT'} \\ \text{CASE} & \text{GEN}\end{bmatrix}, \begin{bmatrix}\text{PRED} & \text{`LACK}\langle\boxed{3},\boxed{4}\rangle\text{'} \\ \text{OBJ}_{\theta} & \boxed{4}\begin{bmatrix}\text{PRED} & \text{`YOU'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\ \text{OBL}_{gen} & \boxed{4}\begin{bmatrix}\text{PRED} & \text{`MEANS'} \\ \text{CASE} & \text{GEN}\end{bmatrix} \\ \text{COMP-FORM} & \dot{\text{Z}}\text{E}\end{bmatrix}\right\} \\
\text{COORD-FORM} & \text{I}
\end{bmatrix}
\end{bmatrix}$$

F-structure for BAĆ SIĘ: unlike coordination

$$\begin{bmatrix}
\text{PRED} & \text{`FEAR}\langle\boxed{1},\boxed{2}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{`PRO'}\end{bmatrix} \\
\text{OBL}_{gen} & \boxed{2}\left\{\begin{bmatrix}\text{PRED} & \text{`UNEMPLOYMENT'} \\ \text{CASE} & \text{GEN}\end{bmatrix}, \begin{bmatrix}\text{PRED} & \text{`LACK}\langle\boxed{3},\boxed{4}\rangle\text{'} \\ \text{OBJ}_{\theta} & \boxed{4}\begin{bmatrix}\text{PRED} & \text{`YOU'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\ \text{OBL}_{gen} & \boxed{4}\begin{bmatrix}\text{PRED} & \text{`MEANS'} \\ \text{CASE} & \text{GEN}\end{bmatrix} \\ \text{COMP-FORM} & \dot{\text{Z}}\text{E}\end{bmatrix}\right\} \\
 & \text{COORD-FORM} \quad \text{I}
\end{bmatrix}$$

F-structure for BAĆ SIĘ: unlike coordination

$$\begin{bmatrix}
\text{PRED} & \text{`FEAR}\langle\boxed{1},\boxed{2}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{`PRO'}\end{bmatrix} \\
\text{OBL}_{gen} & \boxed{2}\begin{bmatrix}
\left\{\begin{bmatrix}\text{PRED} & \text{`UNEMPLOYMENT'} \\ \text{CASE} & \text{GEN}\end{bmatrix}, \begin{bmatrix}\text{PRED} & \text{`LACK}\langle\boxed{3},\boxed{4}\rangle\text{'} \\ \text{OBJ}_\theta & \boxed{4}\begin{bmatrix}\text{PRED} & \text{`YOU'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\ \text{OBL}_{gen} & \boxed{4}\begin{bmatrix}\text{PRED} & \text{`MEANS'} \\ \text{CASE} & \text{GEN}\end{bmatrix} \\ \text{COMP-FORM} & \dot{\text{Z}}\text{E}\end{bmatrix}\right\} \\
\text{COORD-FORM} & \text{I}
\end{bmatrix}
\end{bmatrix}$$

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

Constraints:

- `subj` → SUBJ (given)
- `np(dat)` →:
  - OBJ-TH, non-coordinate position
  - `(^ GF CASE)=c dat`
- `infp` →:
  - XCOMP, non-coordinate position
  - controlled by OBJ-TH: `(^ OBJ-TH)=(^ XCOMP SUBJ)`

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

Constraints:

- `subj` → SUBJ (given)
- `np(dat)` →:
  - OBJ-TH, non-coordinate position
  - `(^ GF CASE)=c dat`
- `infp` →:
  - XCOMP, non-coordinate position
  - controlled by OBJ-TH: `(^ OBJ-TH)=(^ XCOMP SUBJ)`

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

Constraints:

- `subj` → SUBJ (given)
- `np(dat)` →:
  - OBJ-TH, non-coordinate position
  - `(^ GF CASE)=c dat`
- `infp` →:
  - XCOMP, non-coordinate position
  - controlled by OBJ-TH: `(^ OBJ-TH)=(^ XCOMP SUBJ)`

Examples: KAZAĆ ‘order’

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  ‘Jan ordered Maria to sing.’
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

Constraints:

- `subj` → SUBJ (given)
- `np(dat)` →:
  - OBJ-TH, non-coordinate position
  - `(^ GF CASE)=c dat`
- `infp` →:
  - XCOMP, non-coordinate position
  - controlled by OBJ-TH: `(^ OBJ-TH)=(^ XCOMP SUBJ)`

Examples: KAZAĆ ‘order’

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  ‘Jan ordered Maria to sing.’
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

Constraints:

- `subj` → SUBJ (given)
- `np(dat)` →:
  - OBJ-TH, non-coordinate position
  - `(^ GF CASE)=c dat`
- `infp` →:
  - XCOMP, non-coordinate position
  - controlled by OBJ-TH: `(^ OBJ-TH)=(^ XCOMP SUBJ)`

Examples: KAZAĆ 'order'

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  'Jan ordered Maria to sing.'
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

Constraints:

- `subj` → SUBJ (given)
- `np(dat)` →:
  - OBJ-TH, non-coordinate position
  - `(^ GF CASE)=c dat`
- `infp` →:
  - XCOMP, non-coordinate position
  - controlled by OBJ-TH: `(^ OBJ-TH)=(^ XCOMP SUBJ)`

Examples: KAZAĆ ‘order’

- Jan kazał Marii śpiewać.
  Jan.NOM ordered Maria.DAT sing.INF
  ‘Jan ordered Maria to sing.’
- `kazać: _: perf: subj{np(str)} + controller{np(dat)} + controllee{infp(_)}`

Constraints:

- `subj` → SUBJ (given)
- `np(dat)` →:
  - OBJ-TH, non-coordinate position
  - `(^ GF CASE)=c dat`
- `infp` →:
  - XCOMP, non-coordinate position
  - controlled by OBJ-TH: `(^ OBJ-TH)=(^ XCOMP SUBJ)`

F-structure for KAZAĆ: object control

$$\begin{bmatrix}
\text{PRED} & \text{`ORDER}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`JAN'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBJ}_\theta & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`MARYSIA'} \\ \text{CASE} & \text{DAT} \end{bmatrix} \\
\text{XCOMP} & \boxed{3} \begin{bmatrix} \text{PRED} & \text{`SING}\langle \boxed{2} \rangle\text{'} \\ \text{SUBJ} & \boxed{2} \end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

Constraints:

- `subj` → SUBJ (given)
- `xp(mod)` →:
  - OBL-MOD, coordinate position
  - off-path constraint with all realisations
- `xp(locat)` →:
  - OBL-LOCAT, coordinate position
  - off-path constraint with all realisations

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

Constraints:

- `subj` → SUBJ (given)
- `xp(mod)` →:
  - OBL-MOD, coordinate position
  - off-path constraint with all realisations
- `xp(locat)` →:
  - OBL-LOCAT, coordinate position
  - off-path constraint with all realisations

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

Constraints:

- `subj` → SUBJ (given)
- `xp(mod)` →:
  - OBL-MOD, coordinate position
  - off-path constraint with all realisations
- `xp(locat)` →:
  - OBL-LOCAT, coordinate position
  - off-path constraint with all realisations

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

Constraints:

- `subj` → SUBJ (given)
- `xp(mod)` →:
  - OBL-MOD, coordinate position
  - off-path constraint with all realisations
- `xp(locat)` →:
  - OBL-LOCAT, coordinate position
  - off-path constraint with all realisations

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

Constraints:

- `subj` → SUBJ (given)
- `xp(mod)` →:
  - OBL-MOD, coordinate position
  - off-path constraint with all realisations
- `xp(locat)` →:
  - OBL-LOCAT, coordinate position
  - off-path constraint with all realisations

Examples: FUNKCJONOWAĆ ‘function’

- Jan dobrze funkcjonuje w nowej roli.
  Jan.NOM well functions in new.LOC role.LOC
  ‘Jan functions well in his new role.’
- `funkcjonować: _: imperf:`
  `subj{np(str)} + {xp(mod)} + {xp(locat)}`

Constraints:

- `subj` → SUBJ (given)
- `xp(mod)` →:
  - OBL-MOD, coordinate position
  - off-path constraint with all realisations
- `xp(locat)` →:
  - OBL-LOCAT, coordinate position
  - off-path constraint with all realisations

Converting semantically defined xp(sem) phrases

List (abbreviated) of realisations of xp(locat):

```
xp(locat)-->
    advp(locat)
    prepnp(między,inst)
    prepnp(nad,inst)
    prepnp(pod,inst)
    prepnp(ponad,inst)
    prepnp(przy,loc)
    prepnp(w,loc)
```

```
xp(locat)-->
    advp(locat)
    prepnp(between,inst)
    prepnp(above,inst)
    prepnp(under,inst)
    prepnp(over,inst)
    prepnp(near,loc)
    prepnp(in,loc)
```

Off-path constraint where each disjunct corresponds to one realisation:

```
(^ GF PRED: { ...  | (-> FN)=c między (<- OBJ CASE)=c inst |
...  | (-> FN)=c w (<- OBJ CASE)=c loc | ...})
```

Converting semantically defined xp(sem) phrases

List (abbreviated) of realisations of xp(locat):

```
xp(locat)-->
    advp(locat)
    prepnp(między,inst)
    prepnp(nad,inst)
    prepnp(pod,inst)
    prepnp(ponad,inst)
    prepnp(przy,loc)
    prepnp(w,loc)
```

```
xp(locat)-->
    advp(locat)
    prepnp(between,inst)
    prepnp(above,inst)
    prepnp(under,inst)
    prepnp(over,inst)
    prepnp(near,loc)
    prepnp(in,loc)
```

Off-path constraint where each disjunct corresponds to one realisation:

```
(^ GF PRED: { ...   | (-> FN)=c między (<- OBJ CASE)=c inst |
...   | (-> FN)=c w (<- OBJ CASE)=c loc | ...})
```

Converting semantically defined xp(sem) phrases

List (abbreviated) of realisations of xp(locat):

```
xp(locat)-->
    advp(locat)
    prepnp(między,inst)
    prepnp(nad,inst)
    prepnp(pod,inst)
    prepnp(ponad,inst)
    prepnp(przy,loc)
    prepnp(w,loc)
```

```
xp(locat)-->
    advp(locat)
    prepnp(between,inst)
    prepnp(above,inst)
    prepnp(under,inst)
    prepnp(over,inst)
    prepnp(near,loc)
    prepnp(in,loc)
```

Off-path constraint where each disjunct corresponds to one realisation:

```
(^ GF PRED: { ...  | (-> FN)=c między (<- OBJ CASE)=c inst |
...  | (-> FN)=c w (<- OBJ CASE)=c loc | ...})
```

F-structure for FUNKCJONOWAĆ: semantic arguments

$$\begin{bmatrix}
\text{PRED} & \text{`FUNCTION}\langle \boxed{1},\boxed{2},\boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{`JAN'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBL-MOD} & \boxed{2}\begin{bmatrix} \text{PRED} & \text{`WELL'} \end{bmatrix} \\
\text{OBL-LOCAT} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{`IN}\langle \boxed{4} \rangle\text{'} \\ \text{OBJ} & \boxed{4}\begin{bmatrix} \text{PRED} & \text{`ROLE'} \\ \text{CASE} & \text{LOC} \\ \text{ADJUNCT} & \left\{\begin{bmatrix} \text{PRED} & \text{`NEW'} \end{bmatrix}\right\} \end{bmatrix} \end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structure for FUNKCJONOWAĆ: semantic arguments

$$\begin{bmatrix}
\text{PRED} & \text{`FUNCTION}\langle \boxed{1},\boxed{2},\boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{`JAN'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBL-MOD} & \boxed{2}\begin{bmatrix} \text{PRED} & \text{`WELL'} \end{bmatrix} \\
\text{OBL-LOCAT} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{`IN}\langle \boxed{4} \rangle\text{'} \\ \text{OBJ} & \boxed{4}\begin{bmatrix} \text{PRED} & \text{`ROLE'} \\ \text{CASE} & \text{LOC} \\ \text{ADJUNCT} & \left\{\begin{bmatrix} \text{PRED} & \text{`NEW'} \end{bmatrix}\right\} \end{bmatrix} \end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structure for FUNKCJONOWAĆ: semantic arguments

$$\begin{bmatrix}
\text{PRED} & \text{`FUNCTION}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix} \text{PRED} & \text{`JAN'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBL-MOD} & \boxed{2}\begin{bmatrix} \text{PRED} & \text{`WELL'} \end{bmatrix} \\
\text{OBL-LOCAT} & \boxed{3}\begin{bmatrix} \text{PRED} & \text{`IN}\langle \boxed{4} \rangle\text{'} \\ \text{OBJ} & \boxed{4}\begin{bmatrix} \text{PRED} & \text{`ROLE'} \\ \text{CASE} & \text{LOC} \\ \text{ADJUNCT} & \left\{\begin{bmatrix} \text{PRED} & \text{`NEW'} \end{bmatrix}\right\} \end{bmatrix} \end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

General schema of lexicalised phrases

`lex(basecat(params),<lexparams>,lemma,mod)`

- metacategory,
- base category with standard constraints,
- lexicalised constraints over base category (variable number),
- lemma,
- modification type.

Examples:

- `{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}`
- `{lex(np(str),sg,'strona',ratr1({possp}))}`
- `subj{lex(np(str),sg,'krew',ratr)} + {lex(prepnp(w,loc),pl,'żyła',ratr)}`
- `subj{lex(np(str),sg,'bóg',atr({adjp(agr)}))} + obj{np(str)} + {lex(xp(adl[prepnp(do,gen)]),sg,'siebie',natr)}`

General schema of lexicalised phrases

`lex(basecat(params),<lexparams>,lemma,mod)`

- metacategory,
- base category with standard constraints,
- lexicalised constraints over base category (variable number),
- lemma,
- modification type.

Examples:

- `{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}`
- `{lex(np(str),sg,'strona',ratr1({possp}))}`
- `subj{lex(np(str),sg,'krew',ratr)} + {lex(prepnp(w,loc),pl,'żyła',ratr)}`
- `subj{lex(np(str),sg,'bóg',atr({adjp(agr)}))} + obj{np(str)} + {lex(xp(adl[prepnp(do,gen)]),sg,'siebie',natr)}`

General schema of lexicalised phrases

`lex(basecat(params),<lexparams>,lemma,mod)`

- metacategory,
- base category with standard constraints,
- lexicalised constraints over base category (variable number),
- lemma,
- modification type.

Examples:

- `{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}`
- `{lex(np(str),sg,'strona',ratr1({possp}))}`
- `subj{lex(np(str),sg,'krew',ratr)} + {lex(prepnp(w,loc),pl,'żyła',ratr)}`
- `subj{lex(np(str),sg,'bóg',atr({adjp(agr)}))} + obj{np(str)} + {lex(xp(adl[prepnp(do,gen)]),sg,'siebie',natr)}`

General schema of lexicalised phrases

`lex(basecat(params),<lexparams>,lemma,mod)`

- metacategory,
- base category with standard constraints,
- lexicalised constraints over base category (variable number),
- lemma,
- modification type.

Examples:

- `{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}`
- `{lex(np(str),sg,'strona',ratr1({possp}))}`
- `subj{lex(np(str),sg,'krew',ratr)} + {lex(prepnp(w,loc),pl,'żyła',ratr)}`
- `subj{lex(np(str),sg,'bóg',atr({adjp(agr)}))} + obj{np(str)} + {lex(xp(adl[prepnp(do,gen)]),sg,'siebie',natr)}`

General schema of lexicalised phrases

`lex(basecat(params),<lexparams>,lemma,mod)`

- metacategory,
- base category with standard constraints,
- lexicalised constraints over base category (variable number),
- lemma,
- modification type.

Examples:

- `{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}`
- `{lex(np(str),sg,'strona',ratr1({possp}))}`
- `subj{lex(np(str),sg,'krew',ratr)} +`
  `{lex(prepnp(w,loc),pl,'żyła',ratr)}`
- `subj{lex(np(str),sg,'bóg',atr({adjp(agr)}))} +`
  `obj{np(str)} +`
  `{lex(xp(adl[prepnp(do,gen)]),sg,'siebie',natr)}`

General schema of lexicalised phrases

`lex(basecat(params),<lexparams>,lemma,mod)`

- metacategory,
- base category with standard constraints,
- lexicalised constraints over base category (variable number),
- lemma,
- modification type.

Examples:

- `{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}`
- `{lex(np(str),sg,'strona',ratr1({possp}))}`
- `subj{lex(np(str),sg,'krew',ratr)} + {lex(prepnp(w,loc),pl,'żyła',ratr)}`
- `subj{lex(np(str),sg,'bóg',atr({adjp(agr)}))} + obj{np(str)} + {lex(xp(adl[prepnp(do,gen)]),sg,'siebie',natr)}`

Conversion of lexicalised dependents: bird's eye view

- choose the grammatical function for each dependent
- impose relevant constraints:
  - as in base category,
  - lexicalised:
    - lemma,
    - extra morphosyntactic constraints,
    - modification.

Conversion of lexicalised dependents: bird's eye view

- choose the grammatical function for each dependent
- impose relevant constraints:
  - as in base category,
  - lexicalised:
    - lemma,
    - extra morphosyntactic constraints,
    - modification.

Conversion of lexicalised dependents: bird's eye view

- choose the grammatical function for each dependent
- impose relevant constraints:
  - as in base category,
  - lexicalised:
    - lemma,
    - extra morphosyntactic constraints,
    - modification.

Conversion of lexicalised dependents: bird's eye view

- choose the grammatical function for each dependent
- impose relevant constraints:
  - as in base category,
  - lexicalised:
    - lemma,
    - extra morphosyntactic constraints,
    - modification.

Conversion of lexicalised dependents: bird's eye view

- choose the grammatical function for each dependent
- impose relevant constraints:
  - as in base category,
  - lexicalised:
    - lemma,
    - extra morphosyntactic constraints,
    - modification.

Conversion of lexicalised dependents: bird's eye view

- choose the grammatical function for each dependent
- impose relevant constraints:
  - as in base category,
  - lexicalised:
    - lemma,
    - extra morphosyntactic constraints,
    - modification.

Conversion of lexicalised dependents: bird's eye view

- choose the grammatical function for each dependent
- impose relevant constraints:
  - as in base category,
  - lexicalised:
    - lemma,
    - extra morphosyntactic constraints,
    - modification.

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.: `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.:
  `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.: `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.: `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.: `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.:
  `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.:
  `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.:
  `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.:
  `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Base categories and constraints

- `lex(np(case),number,lemma,mod)`
- `lex(adjp(case),number,gender,degree,lemma,mod)`
- `lex(prepnp(pform,case),number,lemma,mod)`
- `lex(xp(sem[cat(params)],lexparams,lemma,mod)`; e.g.:
  `lex(xp(adl[prepnp(pform,case)]),number,lemma,mod)`

Modification types

- `natr`: no further modification
- `atr(...)`: modification allowed (optional)
- `atr1(...)`: only one modifier allowed
- `ratr(...)`: modification required (obligatory)
- `ratr1(...)`: only one modifier required

No further modification: natr

Janek wziął na wstrzymanie.
Janek.NOM took on stoppage.ACC
'Janek decided to wait / not to take action.'

wziąć: subj{np(str)} +
{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c wstrzymanie
- ~(^ OBL GF), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

No further modification: natr

Janek wziął na wstrzymanie.
Janek.NOM took on stoppage.ACC
'Janek decided to wait / not to take action.'

wziąć: subj{np(str)} +
{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c wstrzymanie
- ~(^ OBL GF), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

No further modification: natr

Janek wziął na wstrzymanie.
Janek.NOM took on stoppage.ACC
'Janek decided to wait / not to take action.'

```
wziąć: subj{np(str)} +
{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c wstrzymanie
- ~(^ OBL GF), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

No further modification: natr

Janek wziął na wstrzymanie.
Janek.NOM took on stoppage.ACC
'Janek decided to wait / not to take action.'

wziąć: subj{np(str)} +
{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c wstrzymanie
- ~(^ OBL GF), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

No further modification: natr

Janek wziął na wstrzymanie.
Janek.NOM took on stoppage.ACC
'Janek decided to wait / not to take action.'

```
wziąć: subj{np(str)} +
{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c wstrzymanie
- ~(^ OBL GF), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

No further modification: natr

Janek wziął na wstrzymanie.
Janek.NOM took on stoppage.ACC
'Janek decided to wait / not to take action.'

```
wziąć: subj{np(str)} +
{lex(prepnp(na,acc),sg,'wstrzymanie',natr)}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c wstrzymanie
- ~(^ OBL GF), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

F-structure for WZIĄĆ NA WSTRZYMANIE

$$\begin{bmatrix} \text{PRED} & \text{`TAKE}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\ \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`JANEK'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\ \text{OBL} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`STOPPAGE'} \\ \text{PFORM} & \text{NA} \\ \text{CASE} & \text{ACC} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

F-structure for WZIĄĆ NA WSTRZYMANIE

$$\begin{bmatrix} \text{PRED} & \text{`TAKE}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\ \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`JANEK'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\ \text{OBL} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`STOPPAGE'} \\ \text{PFORM} & \text{NA} \\ \text{CASE} & \text{ACC} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

Modification allowed: atr

Jan bije Marii (gromkie) brawo.
Jan.NOM strikes Maria.DAT loud.ACC applause.ACC
'Jan (loudly) applauds Maria.'

bić: subj{np(str)} + {np(dat)} +
{lex(np(str),_,'brawo',atr)}

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- no number constraint
- (^ OBL-STR PRED FN)=c brawo
- {(^ OBL-STR ADJUNCT)}

Modification allowed: atr

Jan bije Marii (gromkie) brawo.
Jan.NOM strikes Maria.DAT loud.ACC applause.ACC
'Jan (loudly) applauds Maria.'

```
bić: subj{np(str)} + {np(dat)} +
{lex(np(str),_,'brawo',atr)}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- no number constraint
- (^ OBL-STR PRED FN)=c brawo
- {(^ OBL-STR ADJUNCT)}

Modification allowed: atr

Jan bije Marii (gromkie) brawo.
Jan.NOM strikes Maria.DAT loud.ACC applause.ACC
'Jan (loudly) applauds Maria.'

```
bić: subj{np(str)} + {np(dat)} +
{lex(np(str),_,'brawo',atr)}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- no number constraint
- (^ OBL-STR PRED FN)=c brawo
- {(^ OBL-STR ADJUNCT)}

Modification allowed: atr

Jan bije Marii (gromkie) brawo.
Jan.NOM strikes Maria.DAT loud.ACC applause.ACC
'Jan (loudly) applauds Maria.'

```
bić: subj{np(str)} + {np(dat)} +
{lex(np(str),_,'brawo',atr)}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- no number constraint
- (^ OBL-STR PRED FN)=c brawo
- {(^ OBL-STR ADJUNCT)}

Modification allowed: atr

Jan bije Marii (gromkie) brawo.
Jan.NOM strikes Maria.DAT loud.ACC applause.ACC
'Jan (loudly) applauds Maria.'

```
bić: subj{np(str)} + {np(dat)} +
{lex(np(str),_,'brawo',atr)}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- no number constraint
- (^ OBL-STR PRED FN)=c brawo
- {(^ OBL-STR ADJUNCT)}

Modification allowed: atr

Jan bije Marii (gromkie) brawo.
Jan.NOM strikes Maria.DAT loud.ACC applause.ACC
'Jan (loudly) applauds Maria.'

```
bić: subj{np(str)} + {np(dat)} +
{lex(np(str),_,'brawo',atr)}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- no number constraint
- (^ OBL-STR PRED FN)=c brawo
- {(^ OBL-STR ADJUNCT)}

F-structure for BIĆ BRAWO

$$\begin{bmatrix} \text{PRED} & \text{`STRIKE}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\ \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`JAN'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\ \text{OBL-STR} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`APPLAUSE'} \\ \text{CASE} & \text{ACC} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`LOUD'} \end{bmatrix} \right\} \end{bmatrix} \\ \text{OBJ}_\theta & \boxed{3} \begin{bmatrix} \text{PRED} & \text{`MARIA'} \\ \text{CASE} & \text{DAT} \end{bmatrix} \\ \text{TENSE} & \text{PRES} \end{bmatrix}$$

F-structure for BIĆ BRAWO

$$\begin{bmatrix}
\text{PRED} & \text{`STRIKE}\langle\boxed{1},\boxed{2},\boxed{3}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{`JAN'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBL-STR} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{`APPLAUSE'} \\ \text{CASE} & \text{ACC} \\ \text{ADJUNCT} & \left\{\begin{bmatrix}\text{PRED} & \text{`LOUD'}\end{bmatrix}\right\}\end{bmatrix} \\
\text{OBJ}_\theta & \boxed{3}\begin{bmatrix}\text{PRED} & \text{`MARIA'} \\ \text{CASE} & \text{DAT}\end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structure for BIĆ BRAWO

$$\begin{bmatrix}
\text{PRED} & \text{'STRIKE}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{'JAN'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBL-STR} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{'APPLAUSE'} \\ \text{CASE} & \text{ACC} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{'LOUD'} \end{bmatrix} \right\} \end{bmatrix} \\
\text{OBJ}_\theta & \boxed{3} \begin{bmatrix} \text{PRED} & \text{'MARIA'} \\ \text{CASE} & \text{DAT} \end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required: ratr

*(Gorąca) krew płynie w *(jego) żyłach.
hot.NOM blood.NOM flows in his.LOC veins.LOC
'Hot blood runs in his veins.'

płynąć:

```
subj{lex(np(str),sg,'krew',ratr({adjp(agr)}+{possp}))} +
{lex(prepnp(w,loc),pl,'żyła',ratr({adjp(agr)}+{possp}))}
```

- (^ SUBJ CASE)=c nom
- (^ SUBJ NUM)=c sg
- (^ SUBJ PRED FN)=c krew
- {(^ SUBJ ADJUNCT CAT)=c adj | (^ SUBJ POSS)}
- ~(^ SUBJ GF−ADJUNCT−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

F-structure for PŁYNĄĆ W ŻYŁACH

$$\begin{bmatrix}
\text{PRED} & \text{`FLOW}\langle \boxed{1}, \boxed{2} \rangle\text{'} & \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`BLOOD'} \\ \text{CASE} & \text{NOM} \\ \text{NUM} & \text{SG} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`HOT'} \end{bmatrix} \right\} \end{bmatrix} & \\
\text{OBL} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`VEIN'} \\ \text{PFORM} & \text{W} \\ \text{CASE} & \text{LOC} \\ \text{NUM} & \text{PL} \\ \text{POSS} & \begin{bmatrix} \text{PRED} & \text{`HE'} \end{bmatrix} \end{bmatrix} & \\
\text{TENSE} & \text{PRES} &
\end{bmatrix}$$

F-structure for PŁYNĄĆ W ŻYŁACH

$$\begin{bmatrix}
\text{PRED} & \text{`FLOW}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix}
\text{PRED} & \text{`BLOOD'} \\
\text{CASE} & \text{NOM} \\
\text{NUM} & \text{SG} \\
\text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`HOT'} \end{bmatrix} \right\}
\end{bmatrix} \\
\text{OBL} & \boxed{2} \begin{bmatrix}
\text{PRED} & \text{`VEIN'} \\
\text{PFORM} & \text{W} \\
\text{CASE} & \text{LOC} \\
\text{NUM} & \text{PL} \\
\text{POSS} & \begin{bmatrix} \text{PRED} & \text{`HE'} \end{bmatrix}
\end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structure for PŁYNĄĆ W ŻYŁACH

$$\begin{bmatrix}
\text{PRED} & \text{`FLOW}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix}
\text{PRED} & \text{`BLOOD'} \\
\text{CASE} & \text{NOM} \\
\text{NUM} & \text{SG} \\
\text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`HOT'} \end{bmatrix} \right\}
\end{bmatrix} \\
\text{OBL} & \boxed{2} \begin{bmatrix}
\text{PRED} & \text{`VEIN'} \\
\text{PFORM} & \text{W} \\
\text{CASE} & \text{LOC} \\
\text{NUM} & \text{PL} \\
\text{POSS} & \begin{bmatrix} \text{PRED} & \text{`HE'} \end{bmatrix}
\end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structure for PŁYNĄĆ W ŻYŁACH

$$\begin{bmatrix}
\text{PRED} & \text{`FLOW}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix}
\text{PRED} & \text{`BLOOD'} \\
\text{CASE} & \text{NOM} \\
\text{NUM} & \text{SG} \\
\text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`HOT'} \end{bmatrix} \right\}
\end{bmatrix} \\
\text{OBL} & \boxed{2} \begin{bmatrix}
\text{PRED} & \text{`VEIN'} \\
\text{PFORM} & \text{W} \\
\text{CASE} & \text{LOC} \\
\text{NUM} & \text{PL} \\
\text{POSS} & \begin{bmatrix} \text{PRED} & \text{`HE'} \end{bmatrix}
\end{bmatrix} \\
\text{TENSE} & \text{PRES}
\end{bmatrix}$$

F-structure for PŁYNĄĆ W ŻYŁACH

$$\begin{bmatrix}
\text{PRED} & \text{`FLOW}\langle \boxed{1}, \boxed{2} \rangle\text{'} & \\
\text{SUBJ} & \boxed{1} & \begin{bmatrix}
\text{PRED} & \text{`BLOOD'} \\
\text{CASE} & \text{NOM} \\
\text{NUM} & \text{SG} \\
\text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`HOT'} \end{bmatrix} \right\}
\end{bmatrix} \\
\text{OBL} & \boxed{2} & \begin{bmatrix}
\text{PRED} & \text{`VEIN'} \\
\text{PFORM} & \text{W} \\
\text{CASE} & \text{LOC} \\
\text{NUM} & \text{PL} \\
\text{POSS} & \begin{bmatrix} \text{PRED} & \text{`HE'} \end{bmatrix}
\end{bmatrix} \\
\text{TENSE} & \text{PRES} &
\end{bmatrix}$$

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

Modification required exactly once: ratr1

Janek wziął stronę *(Marysi).
Janek.NOM took side.ACC Marysia.GEN
'Janek took Marysia's side.'

```
wziąć: subj{np(str)} +
{lex(np(str),sg,'strona',ratr1({possp}))}
```

- {~(^ NEG) (^ OBL-STR CASE)=c acc
  | (^ NEG)=c + (^ OBL-STR CASE)=c gen}
- (^ OBL-STR NUM)=c sg
- (^ OBL-STR PRED FN)=c strona
- (^ OBL-STR POSS) ~(^ OBL-STR GF−POSS), where GF = {SUBJ|OBJ|OBL|...|ADJUNCT}

F-structure for WZIĄĆ STRONĘ

$$\begin{bmatrix} \text{PRED} & \text{`TAKE}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\ \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`JANEK'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\ \text{OBL-STR} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`SIDE'} \\ \text{CASE} & \text{ACC} \\ \text{POSS} & \begin{bmatrix} \text{PRED} & \text{`MARYSIA'} \end{bmatrix} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

F-structure for WZIĄĆ STRONĘ

$$\begin{bmatrix}
\text{PRED} & \text{`TAKE}\langle\boxed{1},\boxed{2}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{`JANEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBL-STR} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{`SIDE'} \\ \text{CASE} & \text{ACC} \\ \text{POSS} & \begin{bmatrix}\text{PRED} & \text{`MARYSIA'}\end{bmatrix}\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

F-structure for WZIĄĆ STRONĘ

$$\begin{bmatrix} \text{PRED} & \text{`TAKE}\langle \boxed{1}, \boxed{2} \rangle\text{'} \\ \text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`JANEK'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\ \text{OBL-STR} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`SIDE'} \\ \text{CASE} & \text{ACC} \\ \text{POSS} & \begin{bmatrix} \text{PRED} & \text{`MARYSIA'} \end{bmatrix} \end{bmatrix} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

Problem with ratr1 in conversion

Janek zbił łobuza na *(kwaśne) jabłko.
Janek.NOM beat rascal.ACC to sour apple
'Janek beat the rascal badly.'

```
zbić: subj{np(str)} + obj{np(str)} +
{lex(prepnp(na,acc),sg,'jabłko',
ratr1({lex(adjp(agr),agr,agr,pos,'kwaśny',natr)}))}
```

- (^ OBL PFORM)=c na (^ OBL CASE)=c acc
- (^ OBL NUM)=c sg
- (^ OBL PRED FN)=c jabłko
- (^ OBL ADJUNCT $)=%DEP
  (%DEP PRED FN)=c kwaśny (%DEP CAT)=c adj
  ~[(PATH ADJUNCT $) <h %DEP] "nothing before"
  ~[%DEP <h (PATH ADJUNCT $)] "nothing after"

F-structure for ZBIĆ NA KWAŚNE JABŁKO

$$\begin{bmatrix}
\text{PRED} & \text{`BEAT}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`JANEK'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBJ} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`RASCAL'} \\ \text{CASE} & \text{ACC} \end{bmatrix} \\
\text{OBL} & \boxed{3} \begin{bmatrix} \text{PRED} & \text{`APPLE'} \\ \text{PFORM} & \text{NA} \\ \text{CASE} & \text{ACC} \\ \text{NUM} & \text{SG} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`SOUR'} \end{bmatrix} \right\} \end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

F-structure for ZBIĆ NA KWAŚNE JABŁKO

$$\begin{bmatrix}
\text{PRED} & \text{'BEAT}\langle\boxed{1},\boxed{2},\boxed{3}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{'JANEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBJ} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{'RASCAL'} \\ \text{CASE} & \text{ACC}\end{bmatrix} \\
\text{OBL} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{'APPLE'} \\ \text{PFORM} & \text{NA} \\ \text{CASE} & \text{ACC} \\ \text{NUM} & \text{SG} \\ \text{ADJUNCT} & \left\{\begin{bmatrix}\text{PRED} & \text{'SOUR'}\end{bmatrix}\right\}\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

F-structure for ZBIĆ NA KWAŚNE JABŁKO

$$\begin{bmatrix}
\text{PRED} & \text{'BEAT}\langle\boxed{1},\boxed{2},\boxed{3}\rangle\text{'} \\
\text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{'JANEK'} \\ \text{CASE} & \text{NOM}\end{bmatrix} \\
\text{OBJ} & \boxed{2}\begin{bmatrix}\text{PRED} & \text{'RASCAL'} \\ \text{CASE} & \text{ACC}\end{bmatrix} \\
\text{OBL} & \boxed{3}\begin{bmatrix}\text{PRED} & \text{'APPLE'} \\ \text{PFORM} & \text{NA} \\ \text{CASE} & \text{ACC} \\ \text{NUM} & \text{SG} \\ \text{ADJUNCT} & \left\{\begin{bmatrix}\text{PRED} & \text{'SOUR'}\end{bmatrix}\right\}\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

An example of coordination and embedding

My również witamy Cię serdecznie i z (szeroko)
we.NOM also welcome you.ACC cordially and with widely
*(otwartymi) ramionami.
open.INST arms.INST
'We are also welcoming you cordially and with (widely) open arms.'

```
witać: subj{np(str)} + obj{np(str)} +
{xp(mod);lex(prepnp(z,inst),pl,XOR('ramię','ręka'),
ratr1({lex(adjp(agr),agr,agr,pos,'otwarty',
atr1({lex(advp(mod),pos,'szeroko',natr)}))}))}
```

- coordinate position: plain `xp(mod)` and lexicalised `prepnp`
- XOR (either... or...) lemma specification
- lexicalised `prepnp` must be modified by a lexicalised `adjp`
- lexicalised `adjp` may be modified by a lexicalised `advp`

An example of coordination and embedding

My również witamy Cię serdecznie i z (szeroko)
we.NOM also welcome you.ACC cordially and with widely
*(otwartymi) ramionami.
open.INST arms.INST
'We are also welcoming you cordially and with (widely) open arms.'

```
witać: subj{np(str)} + obj{np(str)} +
{xp(mod);lex(prepnp(z,inst),pl,XOR('ramię','ręka'),
ratr1({lex(adjp(agr),agr,agr,pos,'otwarty',
atr1({lex(advp(mod),pos,'szeroko',natr)}))}))}
```

- coordinate position: plain `xp(mod)` and lexicalised `prepnp`
- XOR (either... or...) lemma specification
- lexicalised `prepnp` must be modified by a lexicalised `adjp`
- lexicalised `adjp` may be modified by a lexicalised `advp`

An example of coordination and embedding

My również witamy Cię serdecznie i z (szeroko)
we.NOM also welcome you.ACC cordially and with widely
*(otwartymi) ramionami.
open.INST arms.INST
'We are also welcoming you cordially and with (widely) open arms.'

```
witać: subj{np(str)} + obj{np(str)} +
{xp(mod);lex(prepnp(z,inst),pl,XOR('ramię','ręka'),
ratr1({lex(adjp(agr),agr,agr,pos,'otwarty',
atr1({lex(advp(mod),pos,'szeroko',natr)}))})})}
```

- coordinate position: plain `xp(mod)` and lexicalised `prepnp`
- XOR (either... or...) lemma specification
- lexicalised `prepnp` must be modified by a lexicalised `adjp`
- lexicalised `adjp` may be modified by a lexicalised `advp`

An example of coordination and embedding

My również witamy Cię serdecznie i z (szeroko)
we.NOM also welcome you.ACC cordially and with widely
*(otwartymi) ramionami.
open.INST arms.INST
'We are also welcoming you cordially and with (widely) open arms.'

```
witać: subj{np(str)} + obj{np(str)} +
{xp(mod);lex(prepnp(z,inst),pl,XOR('ramię','ręka'),
ratr1({lex(adjp(agr),agr,agr,pos,'otwarty',
atr1({lex(advp(mod),pos,'szeroko',natr)}))}))}
```

- coordinate position: plain `xp(mod)` and lexicalised `prepnp`
- XOR (either... or...) lemma specification
- lexicalised `prepnp` must be modified by a lexicalised `adjp`
- lexicalised `adjp` may be modified by a lexicalised `advp`

An example of coordination and embedding

My również witamy Cię serdecznie i z (szeroko)
we.NOM also welcome you.ACC cordially and with widely
*(otwartymi) ramionami.
open.INST arms.INST
'We are also welcoming you cordially and with (widely) open arms.'

```
witać: subj{np(str)} + obj{np(str)} +
{xp(mod);lex(prepnp(z,inst),pl,XOR('ramię','ręka'),
ratr1({lex(adjp(agr),agr,agr,pos,'otwarty',
atr1({lex(advp(mod),pos,'szeroko',natr)}))}))}
```

- coordinate position: plain `xp(mod)` and lexicalised `prepnp`
- XOR (either... or...) lemma specification
- lexicalised `prepnp` must be modified by a lexicalised `adjp`
- lexicalised `adjp` may be modified by a lexicalised `advp`

An example of coordination and embedding

My również witamy Cię serdecznie i z (szeroko)
we.NOM also welcome you.ACC cordially and with widely
*(otwartymi) ramionami.
open.INST arms.INST
'We are also welcoming you cordially and with (widely) open arms.'

```
witać: subj{np(str)} + obj{np(str)} +
{xp(mod);lex(prepnp(z,inst),pl,XOR('ramię','ręka'),
ratr1({lex(adjp(agr),agr,agr,pos,'otwarty',
atr1({lex(advp(mod),pos,'szeroko',natr)}))}))}
```

- coordinate position: plain `xp(mod)` and lexicalised `prepnp`
- XOR (either... or...) lemma specification
- lexicalised `prepnp` must be modified by a lexicalised `adjp`
- lexicalised `adjp` may be modified by a lexicalised `advp`

An example of coordination and embedding

My również witamy Cię serdecznie i z (szeroko)
we.NOM also welcome you.ACC cordially and with widely
*(otwartymi) ramionami.
open.INST arms.INST
'We are also welcoming you cordially and with (widely) open arms.'

```
witać: subj{np(str)} + obj{np(str)} +
{xp(mod);lex(prepnp(z,inst),pl,XOR('ramię','ręka'),
ratr1({lex(adjp(agr),agr,agr,pos,'otwarty',
atr1({lex(advp(mod),pos,'szeroko',natr)}))}))}
```

- coordinate position: plain `xp(mod)` and lexicalised `prepnp`
- XOR (either... or...) lemma specification
- lexicalised `prepnp` must be modified by a lexicalised `adjp`
- lexicalised `adjp` may be modified by a lexicalised `advp`

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI


```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI


```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI


```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI


```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI


```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

Formalisation of WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

```
(^ OBL-MOD PRED FN)=c z (^ OBL-MOD OBJ CASE)=c inst
(^ OBL-MOD OBJ NUM)=c pl
(^ OBL-MOD OBJ PRED FN)$c {ramię ręka}

(^ OBL-MOD ADJUNCT $)=%DEP
(%DEP PRED FN)=c otwarty (%DEP CAT)=c adj
~[(^ OBL-MOD ADJUNCT $) <h %DEP]
~[%DEP <h (^ OBL-MOD ADJUNCT $)]

{
(%DEP ADJUNCT $)=%DEPEMB
(%DEPEMB PRED FN)=c szeroko (%DEPEMB CAT)=c adv
~[(%DEP ADJUNCT $) <h %DEPEMB]
~[%DEPEMB <h (%DEP ADJUNCT $)]
~(%DEPEMB GF)
}
```

F-structure for WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

$$\begin{bmatrix}
\text{PRED} & \text{`WELCOME}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`WE'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBJ} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`YOU'} \\ \text{CASE} & \text{ACC} \end{bmatrix} \\
\text{OBL-MOD} & \boxed{3} \begin{bmatrix}
\left\{ \begin{bmatrix} \text{PRED} & \text{`CORDIALLY'} \end{bmatrix}, \right. \\
\left. \begin{bmatrix} \text{PRED} & \text{`WITH}\langle \boxed{4} \rangle\text{'} \\ \text{OBJ} & \boxed{4} \begin{bmatrix} \text{PRED} & \text{`ARM'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`OPEN'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`WIDELY'} \end{bmatrix} \right\} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix} \right\} \\
\text{COORD-FORM} & \text{I}
\end{bmatrix} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}$$

F-structure for WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

$$
\begin{bmatrix}
\text{PRED} & \text{`WELCOME}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`WE'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBJ} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`YOU'} \\ \text{CASE} & \text{ACC} \end{bmatrix} \\
\text{OBL-MOD} & \boxed{3} \left\{ \begin{bmatrix} \text{PRED} & \text{`CORDIALLY'} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{`WITH}\langle \boxed{4} \rangle\text{'} \\ \text{OBJ} & \boxed{4} \begin{bmatrix} \text{PRED} & \text{`ARM'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`OPEN'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`WIDELY'} \end{bmatrix} \right\} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix} \right\} \\
& \text{COORD-FORM} \quad \text{I} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}
$$

F-structure for WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

$$
\begin{bmatrix}
\text{PRED} & \text{`WELCOME}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`WE'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBJ} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`YOU'} \\ \text{CASE} & \text{ACC} \end{bmatrix} \\
\text{OBL-MOD} & \boxed{3} \left\{ \begin{bmatrix} \text{PRED} & \text{`CORDIALLY'} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{`WITH}\langle \boxed{4} \rangle\text{'} \\ \text{OBJ} & \boxed{4} \begin{bmatrix} \text{PRED} & \text{`ARM'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`OPEN'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`WIDELY'} \end{bmatrix} \right\} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix} \right\} \\
 & \text{COORD-FORM} \quad \text{I} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}
$$

F-structure for WITAĆ Z (SZEROKO) OTWARTYMI RAMIONAMI

$$
\begin{bmatrix}
\text{PRED} & \text{`WELCOME}\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle\text{'} \\
\text{SUBJ} & \boxed{1} \begin{bmatrix} \text{PRED} & \text{`WE'} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\
\text{OBJ} & \boxed{2} \begin{bmatrix} \text{PRED} & \text{`YOU'} \\ \text{CASE} & \text{ACC} \end{bmatrix} \\
\text{OBL-MOD} & \boxed{3} \left\{ \begin{bmatrix} \text{PRED} & \text{`CORDIALLY'} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{`WITH}\langle \boxed{4} \rangle\text{'} \\ \text{OBJ} & \boxed{4} \begin{bmatrix} \text{PRED} & \text{`ARM'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`OPEN'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{`WIDELY'} \end{bmatrix} \right\} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix} \right\} \\
 & \text{COORD-FORM} \quad \text{I} \\
\text{TENSE} & \text{PAST}
\end{bmatrix}
$$

Representing MWEs in LFG?

- `LEX` attribute inside relevant dependents
- extra `MWEPRED` attribute
- pointer to Walenty schemata (idea: Agata Savary)
- pointer to XML identifiers of dependents
- pointer to WordNet sense of the MWE
- more?...

Conclusion

We have presented:

- a short introduction to LFG
- POLFIE: an implemented LFG grammar of Polish
- conversion of Walenty to XLE/LFG formalism:
  - GF assignment
  - imposing constraints
  - basic, non-lexicalised arguments
  - lexicalised arguments

POLFIE

`http://zil.ipipan.waw.pl/LFG/`

POLFIE in XLE-Web

`http://iness.mozart.ipipan.waw.pl/iness/xle-web`

Bresnan, J., editor (1982). *The Mental Representation of Grammatical Relations*. MIT Press Series on Cognitive Theory and Mental Representation. The MIT Press, Cambridge, MA.

Bresnan, J. (2000). *Lexical-Functional Syntax*. Blackwell Textbooks in Linguistics. Blackwell.

Dalrymple, M. (2001). *Lexical-Functional Grammar*. Academic Press.

Patejuk, A. and Przepiórkowski, A. (2012). Towards an LFG parser for Polish: An exercise in parasitic grammar development. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, pages 3849–3852, Istanbul, Turkey. ELRA.

Przepiórkowski, A., Hajnicz, E., Patejuk, A., Woliński, M., Skwarski, F., and Świdziński, M. (2014). Walenty: Towards a comprehensive valence dictionary of Polish. In N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pages 2785–2792, Reykjavík, Iceland. ELRA.