A Computational Grammar for Georgian

Paul Meurer

AKSIS, University of Bergen, Norway

Seventh International Tbilisi Symposium
on Language, Logic and Computation
Tbilisi, October 1 – 5, 2007

Outline

1. Introduction: The Georgian grammar project
2. Lexical-Functional Grammar
3. Morphology and Morphosyntax
4. Some aspects of Georgian syntax
5. Tools for LFG grammar development

Outline

1 Introduction: The Georgian grammar project

2 Lexical-Functional Grammar

3 Morphology and Morphosyntax

4 Some aspects of Georgian syntax

5 Tools for LFG grammar development

A computational grammar for Georgian

The Georgian grammar project

- is aimed at developing a full-scale computational grammar for Georgian

A computational grammar for Georgian

The Georgian grammar project

- is aimed at developing a full-scale computational grammar for Georgian
- uses LFG (Lexical Functional Grammar) as a syntactic formalism (output of a parse consists of c- and f-structures, no semantic representation yet)

A computational grammar for Georgian

The Georgian grammar project

- is aimed at developing a full-scale computational grammar for Georgian
- uses LFG (Lexical Functional Grammar) as a syntactic formalism (output of a parse consists of c- and f-structures, no semantic representation yet)
- uses XLE (Xerox Linguistic Environment) as a parsing engine

A computational grammar for Georgian

The Georgian grammar project

- is aimed at developing a full-scale computational grammar for Georgian
- uses LFG (Lexical Functional Grammar) as a syntactic formalism (output of a parse consists of c- and f-structures, no semantic representation yet)
- uses XLE (Xerox Linguistic Environment) as a parsing engine
- uses visualization, treebanking and corpus tools developed at Aksis/University of Bergen

A computational grammar for Georgian

The Georgian grammar project

- is aimed at developing a full-scale computational grammar for Georgian
- uses LFG (Lexical Functional Grammar) as a syntactic formalism (output of a parse consists of c- and f-structures, no semantic representation yet)
- uses XLE (Xerox Linguistic Environment) as a parsing engine
- uses visualization, treebanking and corpus tools developed at Aksis/University of Bergen
- is part of the international Parallel Grammar (ParGram) project, which coordinates the development of LFG grammars in a parallel manner using XLE

A computational grammar for Georgian

Time frames and status quo:

- started around 1995 with morphology, 2005 with syntax

A computational grammar for Georgian

Time frames and status quo:

- started around 1995 with morphology, 2005 with syntax
- most of the morphology is covered (missing: proper names; compounds)

A computational grammar for Georgian

Time frames and status quo:

- started around 1995 with morphology, 2005 with syntax
- most of the morphology is covered (missing: proper names; compounds)
- most basic syntactic constructions are covered (incomplete: subcategorization frames, constructions involving infinite forms, appositions, much more)

A computational grammar for Georgian

Time frames and status quo:

- started around 1995 with morphology, 2005 with syntax
- most of the morphology is covered (missing: proper names; compounds)
- most basic syntactic constructions are covered (incomplete: subcategorization frames, constructions involving infinite forms, appositions, much more)
- funding: none

Outline

1 Introduction: The Georgian grammar project

2 Lexical-Functional Grammar

3 Morphology and Morphosyntax

4 Some aspects of Georgian syntax

5 Tools for LFG grammar development

Lexical-Functional Grammar (LFG)

- a generative linguistic framework

Lexical-Functional Grammar (LFG)

- a generative linguistic framework
- initiated by Joan Bresnan and Ronald Kaplan in the 1970s to overcome conceptual and explanatory shortcomings of Chomsky's transformational grammar

Lexical-Functional Grammar (LFG)

- a generative linguistic framework
- initiated by Joan Bresnan and Ronald Kaplan in the 1970s to overcome conceptual and explanatory shortcomings of Chomsky's transformational grammar
- constraint-based; no transformations

Lexical-Functional Grammar (LFG)

- a generative linguistic framework
- initiated by Joan Bresnan and Ronald Kaplan in the 1970s to overcome conceptual and explanatory shortcomings of Chomsky's transformational grammar
- constraint-based; no transformations
- rigid formalism, well-suited for implementation and efficient parsing, as well as for theoretical work

C- and F-structure

- parallel description of linguistic entities by:

C- and F-structure

- parallel description of linguistic entities by:
  - C(onstituent)-structure: phrase structure tree (often modeled according to Xbar-syntax principles)

C- and F-structure

- parallel description of linguistic entities by:
  - C(onstituent)-structure: phrase structure tree (often modeled according to Xbar-syntax principles)
  - F(unctional)-structure: attribute-value matrix which recursively correlates the semantic argument structure of predicates with grammatical functions

C- and F-structure

- parallel description of linguistic entities by:
  - C(onstituent)-structure: phrase structure tree (often modeled according to Xbar-syntax principles)
  - F(unctional)-structure: attribute-value matrix which recursively correlates the semantic argument structure of predicates with grammatical functions

C- and f-structure are related by a projection relation

C- and F-structure

- parallel description of linguistic entities by:
  - C(onstituent)-structure: phrase structure tree (often modeled according to Xbar-syntax principles)
  - F(unctional)-structure: attribute-value matrix which recursively correlates the semantic argument structure of predicates with grammatical functions

  C- and f-structure are related by a projection relation
- Structural relations are formally described by phrase structure rules annotated with functional equations

C- and F-structure

- parallel description of linguistic entities by:
  - C(onstituent)-structure: phrase structure tree (often modeled according to Xbar-syntax principles)
  - F(unctional)-structure: attribute-value matrix which recursively correlates the semantic argument structure of predicates with grammatical functions

  C- and f-structure are related by a projection relation
- Structural relations are formally described by phrase structure rules annotated with functional equations
- Lexical items (lemmas and morphological features) are annotated with a lexical category and functional equations (including argument structures for verbs)

C- and F-structure: Example

bavšvebi tamašoben.
children they-play.

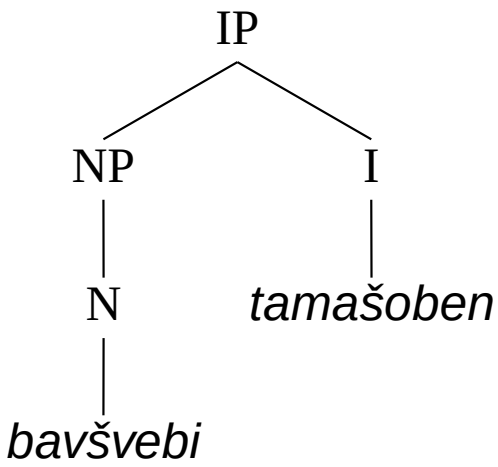
'The children are playing.'

C- and F-structure: Example

bavšvebi tamašoben.
children they-play.

'The children are playing.'

c-structure

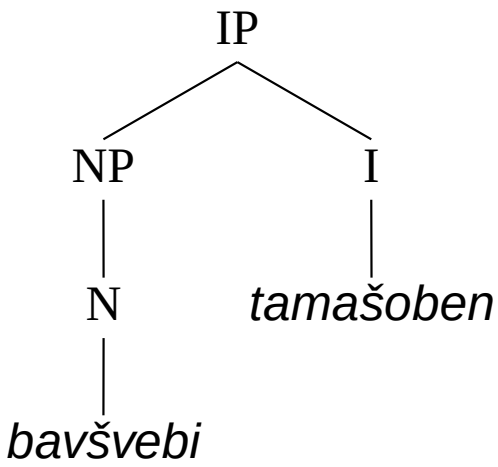


C- and F-structure: Example

bavšvebi tamašoben.
children they-play.

'The children are playing.'

c-structure



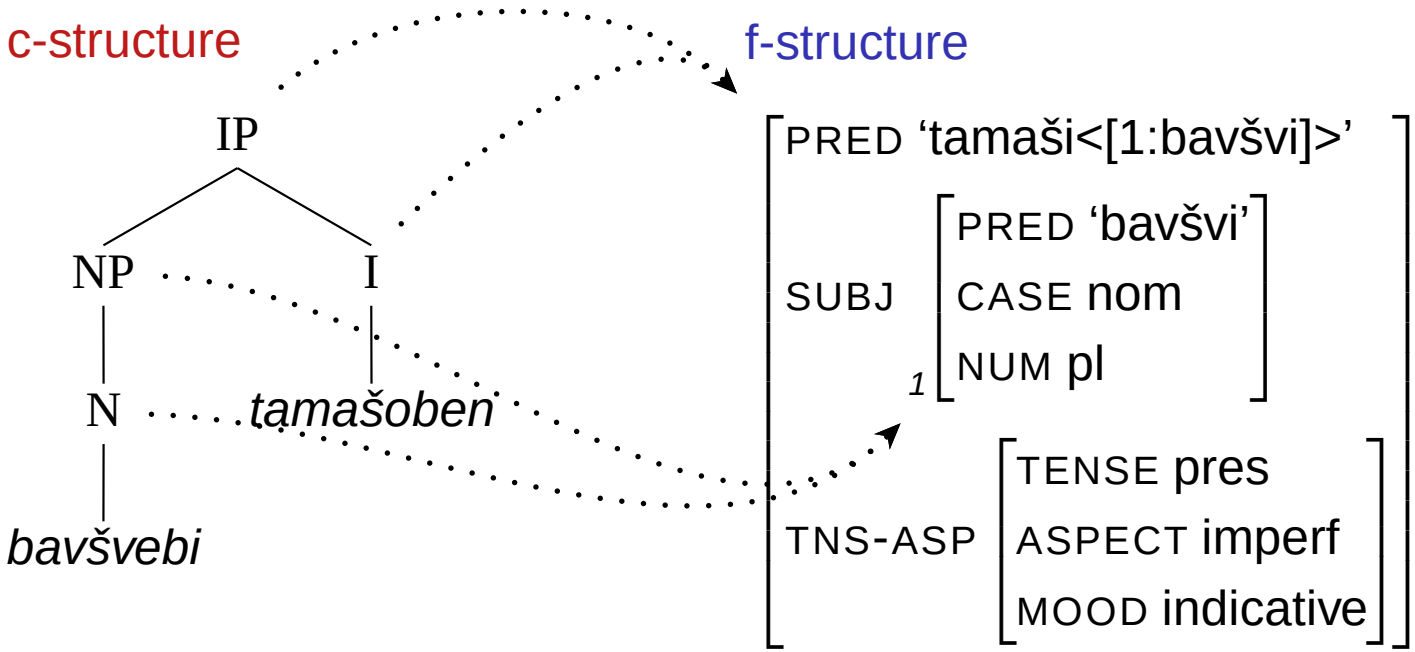
f-structure

$$\begin{bmatrix} \text{PRED } \text{'tamaši<[1:bavšvi]>'} \\ \text{SUBJ } {}_{1}\begin{bmatrix} \text{PRED } \text{'bavšvi'} \\ \text{CASE nom} \\ \text{NUM pl} \end{bmatrix} \\ \text{TNS-ASP } \begin{bmatrix} \text{TENSE pres} \\ \text{ASPECT imperf} \\ \text{MOOD indicative} \end{bmatrix} \end{bmatrix}$$

C- and F-structure: Example

bavšvebi tamašoben.
children they-play.

'The children are playing.'



Parsing with XLE

Parsing steps:

Parsing with XLE

Parsing steps:

- Tokenization

Parsing with XLE

Parsing steps:

- Tokenization
- Morphological analysis

Parsing with XLE

Parsing steps:

- Tokenization
- Morphological analysis
- Lexical insertion

Parsing with XLE

Parsing steps:

- Tokenization
- Morphological analysis
- Lexical insertion
  - lexemes and morphological features are entries in the LFG lexicon

Parsing with XLE

Parsing steps:

- Tokenization
- Morphological analysis
- Lexical insertion
  - lexemes and morphological features are entries in the LFG lexicon
  - they are annotated with (sub)lexical categories and functional equations

Parsing with XLE

Parsing steps:

- Tokenization
- Morphological analysis
- Lexical insertion
  - lexemes and morphological features are entries in the LFG lexicon
  - they are annotated with (sub)lexical categories and functional equations
  - these are used to initialize the parse chart

Parsing with XLE

Parsing steps:

- Tokenization
- Morphological analysis
- Lexical insertion
  - lexemes and morphological features are entries in the LFG lexicon
  - they are annotated with (sub)lexical categories and functional equations
  - these are used to initialize the parse chart
- Chart parsing with phrase structure rules

Parsing with XLE

Parsing steps:

- Tokenization
- Morphological analysis
- Lexical insertion
  - lexemes and morphological features are entries in the LFG lexicon
  - they are annotated with (sub)lexical categories and functional equations
  - these are used to initialize the parse chart
- Chart parsing with phrase structure rules
- Solving of functional equations

Outline

1 Introduction: The Georgian grammar project

2 Lexical-Functional Grammar

3 Morphology and Morphosyntax

4 Some aspects of Georgian syntax

5 Tools for LFG grammar development

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification
  - Disjunctive unification with a lexicon of existing forms to discard nonexisting verb analyses

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification
  - Disjunctive unification with a lexicon of existing forms to discard nonexisting verb analyses
  - Implemented in Common Lisp, based on Parc Xerox's old fsa module

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification
  - Disjunctive unification with a lexicon of existing forms to discard nonexisting verb analyses
  - Implemented in Common Lisp, based on Parc Xerox's old fsa module
- New implementation based on *fst* (Xerox finite state tool)

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification
  - Disjunctive unification with a lexicon of existing forms to discard nonexisting verb analyses
  - Implemented in Common Lisp, based on Parc Xerox's old fsa module
- New implementation based on *fst* (Xerox finite state tool)
  - automatically derived from old implementation

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification
  - Disjunctive unification with a lexicon of existing forms to discard nonexisting verb analyses
  - Implemented in Common Lisp, based on Parc Xerox's old fsa module
- New implementation based on *fst* (Xerox finite state tool)
  - automatically derived from old implementation
  - flag diacritics mimic feature structure unification; compiled out at the end $\Rightarrow$ pure finite state

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification
  - Disjunctive unification with a lexicon of existing forms to discard nonexisting verb analyses
  - Implemented in Common Lisp, based on Parc Xerox's old fsa module
- New implementation based on *fst* (Xerox finite state tool)
  - automatically derived from old implementation
  - flag diacritics mimic feature structure unification; compiled out at the end $\Rightarrow$ pure finite state
  - lexicon compiled into the transducer

Morphology

Parsing model

- First approach: Finite state transducer augmented with feature structure unification
  - Disjunctive unification with a lexicon of existing forms to discard nonexisting verb analyses
  - Implemented in Common Lisp, based on Parc Xerox's old fsa module
- New implementation based on *fst* (Xerox finite state tool)
  - automatically derived from old implementation
  - flag diacritics mimic feature structure unification; compiled out at the end $\Rightarrow$ pure finite state
  - lexicon compiled into the transducer
  - interfaces well with XLE

Morphology

The lexicon

- Derived from Kita Tschenkélis ‘Georgisch-Deutsches Wörterbuch’ (52 000 entries, 3 823 verb entries) and other sources (74 000 nouns and adjectives altogether)

Morphology

The lexicon

- Derived from Kita Tschenkélis ‘Georgisch-Deutsches Wörterbuch’ (52 000 entries, 3 823 verb entries) and other sources (74 000 nouns and adjectives altogether)

Morphology

Typical analyses:

‘wine’

ġvino → ġvino+N+Nom+Sg

Morphology

Typical analyses:

‘wine’
ġvino → ġvino+N+Nom+Sg

‘for the girls, too’
gogo-eb-isa-tvis-ac → gogo+N+Anim+Full+Gen+Pl+Tvis+C

Morphology

Typical analyses:

‘wine’
ġvino → ġvino+N+Nom+Sg

‘for the girls, too’
gogo-eb-isa-tvis-ac → gogo+N+Anim+Full+Gen+Pl+Tvis+C

‘in childhood’
bavšvob-isa-s → bavšvoba+N+DGen+DSg+Dat+Sg

Morphology

Typical analyses:

‘wine’
ġvino → ġvino+N+Nom+Sg

‘for the girls, too’
gogo-eb-isa-tvis-ac → gogo+N+Anim+Full+Gen+Pl+Tvis+C

‘in childhood’
bavšvob-isa-s → bavšvoba+N+DGen+DSg+Dat+Sg

‘I apparently painted it’/‘he will paint it for me’
da-mi-xaṭ-av-s →

{ da-xaṭva-3569-5+V+Trans+Perf+Subj1Sg+Obj3
| da-xaṭva-3569-18+V+Trans+Perf+Subj1Sg+Obj3
| da-xaṭva-3569-18+V+Trans+Fut+Subj3Sg+Obj1Sg } s

From lexeme to grammatical function

Each verb lexeme in the LFG lexicon is associated with one or more subcategorization frames (argument structures) and a mapping of each of the arguments to a grammatical function (one of SUBJ(ect), OBJ(ect), OBJben(eficiary), OBL(ique), etc.).

From lexeme to grammatical function

Each verb lexeme in the LFG lexicon is associated with one or more subcategorization frames (argument structures) and a mapping of each of the arguments to a grammatical function (one of SUBJ(ect), OBJ(ect), OBJben(eficiary), OBL(ique), etc.).

- *ga-v-u-ḳet-eb* 'I will do it for him/her':

  ga-ḳeteba<agent, benefic, theme>

From lexeme to grammatical function

Each verb lexeme in the LFG lexicon is associated with one or more subcategorization frames (argument structures) and a mapping of each of the arguments to a grammatical function (one of SUBJ(ect), OBJ(ect), OBJben(eficiary), OBL(ique), etc.).

- *ga-v-u-ḳet-eb* 'I will do it for him/her':

ga-ḳeteba<agent, benefic, theme>
↓ ↓ ↓
ga-ḳeteba<SUBJ, OBJben, OBJ>

From lexeme to grammatical function

Each verb lexeme in the LFG lexicon is associated with one or more subcategorization frames (argument structures) and a mapping of each of the arguments to a grammatical function (one of SUBJ(ect), OBJ(ect), OBJben(eficiary), OBL(ique), etc.).

- *ga-v-u-ḳet-eb* 'I will do it for him/her':

ga-ḳeteba<agent, benefic, theme>
↓ ↓ ↓
ga-ḳeteba<SUBJ, OBJben, OBJ>

The verb classification in Tschenkéli's *Georgisch–deutsches Wörterbuch* could be used directly to automatically derive a preliminary version of the Georgian LFG verb lexicon

From lexeme to grammatical function

Each verb lexeme in the LFG lexicon is associated with one or more subcategorization frames (argument structures) and a mapping of each of the arguments to a grammatical function (one of SUBJ(ect), OBJ(ect), OBJben(eficiary), OBL(ique), etc.).

- *ga-v-u-ḳet-eb* 'I will do it for him/her':

  ga-ḳeteba<agent, benefic, theme>

  ↓ ↓ ↓

  ga-ḳeteba<SUBJ, OBJben, OBJ>

The verb classification in Tschenkéli's *Georgisch–deutsches Wörterbuch* could be used directly to automatically derive a preliminary version of the Georgian LFG verb lexicon

- Example: Tschenkéli's class T^3 maps to the argument structure

  P<SUBJ, OBJben, OBJ>

From lexeme to grammatical function

In many cases the correct frames are not (easily) deducible from Tschenkéli´s classification:

From lexeme to grammatical function

In many cases the correct frames are not (easily) deducible from Tschenkéli´s classification:

- verbs taking oblique or genitive arguments:

  ča-tvla<SUBJ, OBJ, OBL_{adv}> ‘consider sb. to be sth.’
  še-šineba<SUBJ, OBJ_{gen}> ‘be afraid of sb./sth.’

From lexeme to grammatical function

In many cases the correct frames are not (easily) deducible from Tschenkéli´s classification:

- verbs taking oblique or genitive arguments:

  ča-tvla<SUBJ, OBJ, OBL_{adv}> ‘consider sb. to be sth.’
  še-šineba<SUBJ, OBJ_{gen}> ‘be afraid of sb./sth.’

- Class III verbs: can be transitive and intransitive

From lexeme to grammatical function

In many cases the correct frames are not (easily) deducible from Tschenkéli´s classification:

- verbs taking oblique or genitive arguments:

  ča-tvla<SUBJ, OBJ, OBL_{adv}> ‘consider sb. to be sth.’
  še-šineba<SUBJ, OBJ_{gen}> ‘be afraid of sb./sth.’

- Class III verbs: can be transitive and intransitive
- verbs taking clausal arguments

From lexeme to grammatical function

In many cases the correct frames are not (easily) deducible from Tschenkéli´s classification:

- verbs taking oblique or genitive arguments:

  ča-tvla<SUBJ, OBJ, OBL_{adv}> ‘consider sb. to be sth.’
  še-šineba<SUBJ, OBJ_{gen}> ‘be afraid of sb./sth.’

- Class III verbs: can be transitive and intransitive
- verbs taking clausal arguments
- morphological passives: can be passives or unaccusatives

  ga-ḳet-deba (mtavrobis mier): ‘it will be done (by the government)’
  ga-ḳeteba<OBL-AG, SUBJ>

  *da-brun-deba (*dedis mier)*: ‘(s)he will return’
  da-bruneba<SUBJ>

Case and affix alignment

ṣtudenṭ-ma	ċeril-i	mo-m-ċer-a.
student.ERG	letter.NOM	he-wrote-it-to_me.

The student wrote me a letter.

mi-ċera< SUBJ, OBJben, OBJ >

Case and affix alignment

student.ERG	letter.NOM	he-wrote-it-to_me.

The student wrote me a letter.

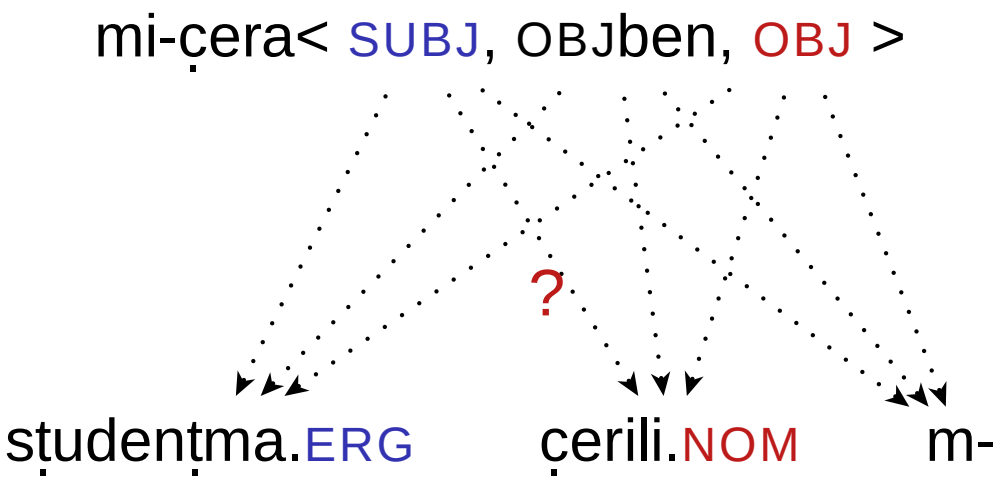
mi-cera< SUBJ, OBJben, OBJ >

studentma.ERG cerili.NOM m-

Case and affix alignment

ṣtudenṭ-ma ċeril-i mo-m-ċer-a.
student.ERG letter.NOM he-wrote-it-to_me.

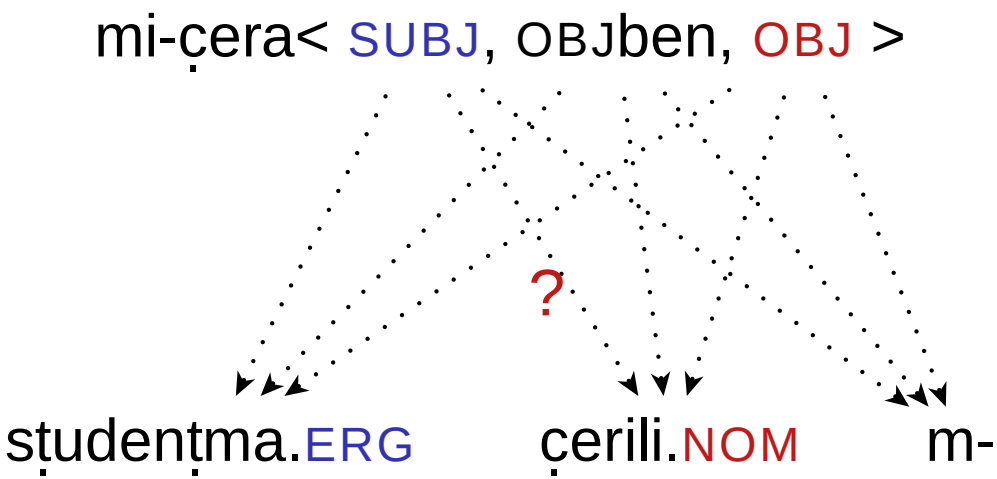
The student wrote me a letter.



Case and affix alignment

ṣtudenṭ-ma ċeril-i mo-m-ċer-a.
student.ERG letter.NOM he-wrote-it-to_me.

The student wrote me a letter.



Georgian is head- and dependent-marking: verbal affixes and nominal case code grammatical functions.

Case and affix alignment: Facts

Case alignment patterns

	SUBJ	OBJ	OBJben
A	NOM	DAT	DAT
B	ERG	NOM	DAT
C	DAT	NOM	*-tvis*

Case and affix alignment: Facts

Case alignment patterns

	SUBJ	OBJ	OBJben
A	NOM	DAT	DAT
B	ERG	NOM	DAT
C	DAT	NOM	*-tvis*

Alignment depending on verb class and tense group

	I *trans.*	II *unacc.*	III *unerg.*	IV *indir.*
present	A	A	A	C
aorist	B	A	B	C
perfect	C	A	C	C

Case and affix alignment: Facts

Case alignment patterns

	SUBJ	OBJ	OBJben
A	NOM	DAT	DAT
B	ERG	NOM	DAT
C	DAT	NOM	*-tvis*

Alignment depending on verb class and tense group

	I *trans.*	II *unacc.*	III *unerg.*	IV *indir.*
present	A	A	A	C
aorist	B	A	B	C
perfect	C	A	C	C

Person/number affix alignment patterns

	SUBJ	OBJ	OBJben
A = B	*v-* (FSUBJ)	*m-* (FOBJ)	*h-* (FOBJ)
C	*h-* (FOBJ)	*v-* (FSUBJ)	-

Case and affix alignment: Implementation

Affixes: Alignment coded in the morphology

Example: 1st person plural (morphological) subject marker

gv-	→	+FObj1Pl	→	+Subj1Pl (perfect)
			→	+Obj1Pl (otherwise)

ga-gv-i-ḳet-eb-i-a 'we apparently did it'

→ ga-ḳeteba-904-3+V+Perf+FSubj3+FObj1Pl+Trans

→ ga-ḳeteba-904-3+V+Perf+Subj1Pl+Obj3+Trans

Functional equations are attached to morphology features:

+Subj1Pl:	(↑ SUBJ PERS) = 1
	(↑ SUBJ NUM) = pl.
+Obj1Pl:	(↑ _MORPH-SYNT _AGR _OBJ PERS) = 1
	(↑ _MORPH-SYNT _AGR _OBJ NUM) = pl.

Case and affix alignment: Implementation

Case: Alignment coded in the syntax

Equations attached to verb lexicon entry

Example: transitive/unergative subject

{ (↑ SUBJ PRED) = 'pro'
| @(*ifelse* (↑ _TENSEGROUP) $=_c$ pres
[(↑ SUBJ CASE) = nom]
[@(*ifelse* (↑ _TENSEGROUP) $=_c$ aor
[(↑ SUBJ CASE) = erg]
[(↑ SUBJ CASE) = dat])]) }.

Outline

1 Introduction: The Georgian grammar project

2 Lexical-Functional Grammar

3 Morphology and Morphosyntax

4 Some aspects of Georgian syntax

5 Tools for LFG grammar development

Free word order

‘Free word order’ at the phrase level

Free word order

‘Free word order’ at the phrase level

- Subject and complements cannot be distinguished configurationally: no VP

Free word order

‘Free word order’ at the phrase level

- Subject and complements cannot be distinguished configurationally: no VP
- Finite verb and other constituents can occur in almost arbitrary order; or: an arbitrary permutation of the toplevel constituents of a grammatical sentence results in a grammatical sentence with the same propositional truth value

Free word order

'Free word order' at the phrase level

- Subject and complements cannot be distinguished configurationally: no VP
- Finite verb and other constituents can occur in almost arbitrary order; or: an arbitrary permutation of the toplevel constituents of a grammatical sentence results in a grammatical sentence with the same propositional truth value
- This is to be expected: since grammatical functions are coded morphologically, there is no need to repeat the coding configurationally

Free word order

'Free word order' at the phrase level

- Subject and complements cannot be distinguished configurationally: no VP
- Finite verb and other constituents can occur in almost arbitrary order; or: an arbitrary permutation of the toplevel constituents of a grammatical sentence results in a grammatical sentence with the same propositional truth value
- This is to be expected: since grammatical functions are coded morphologically, there is no need to repeat the coding configurationally

$\Rightarrow$ First approximation:

$$S \rightarrow V, XP^*$$

Information structure and Discurse functions

Position is significant for Information structure: it is used to code the discurse functions FOCUS and TOPIC.

Information structure and Discurse functions

Position is significant for Information structure: it is used to code the discurse functions FOCUS and TOPIC.

- FOCUS: immediately in front of inflected verb or in last position

Information structure and Discurse functions

Position is significant for Information structure: it is used to code the discurse functions FOCUS and TOPIC.

- FOCUS: immediately in front of inflected verb or in last position
- TOPIC: initial position(s) to the left of FOCUS and verb

Information structure and Discurse functions

Position is significant for Information structure: it is used to code the discurse functions FOCUS and TOPIC.

- FOCUS: immediately in front of inflected verb or in last position
- TOPIC: initial position(s) to the left of FOCUS and verb

$\Rightarrow$ Revision of phrase structure rules (compliant with Xbar theory, Bresnan 2001):

Information structure and Discurse functions

Position is significant for Information structure: it is used to code the discurse functions FOCUS and TOPIC.

- FOCUS: immediately in front of inflected verb or in last position
- TOPIC: initial position(s) to the left of FOCUS and verb

$\Rightarrow$ Revision of phrase structure rules (compliant with Xbar theory, Bresnan 2001):

$I \rightarrow$ finite verb

$I' \rightarrow I\ (S)$

$S \rightarrow XP+$

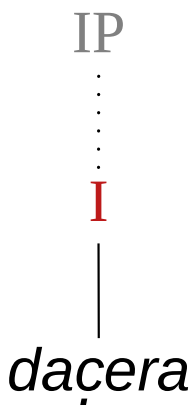
$IP \rightarrow (XP)\ I'$

$IP \rightarrow XP\ IP$

Phrase structure rules

I is the category of the finite (inflected) verb:

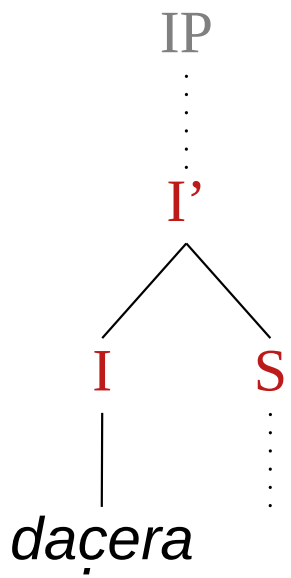
I → finite verb



Phrase structure rules

The nonprojective category S is the Complement of I:

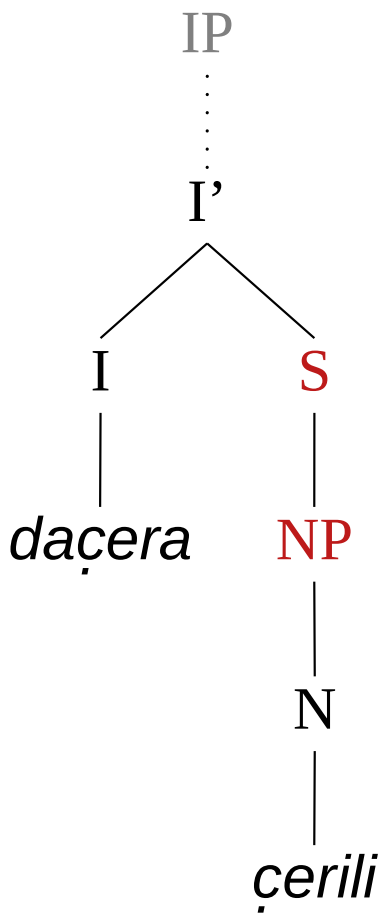
I' → I (S)



Phrase structure rules

S contains all material to the right of the verb:

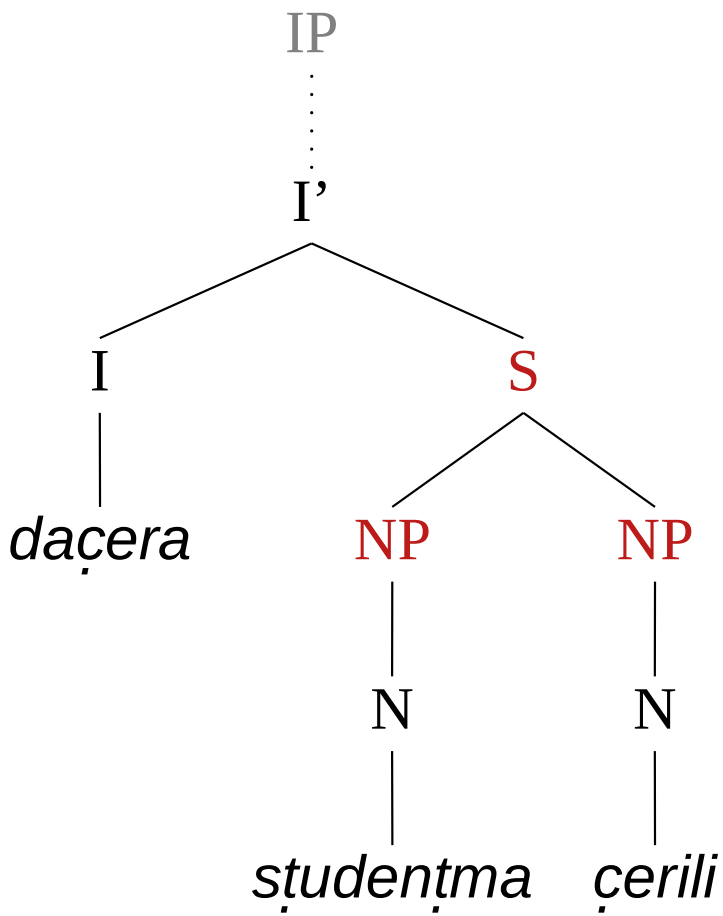
$S \rightarrow XP+$



Phrase structure rules

S contains all material to the right of the verb:

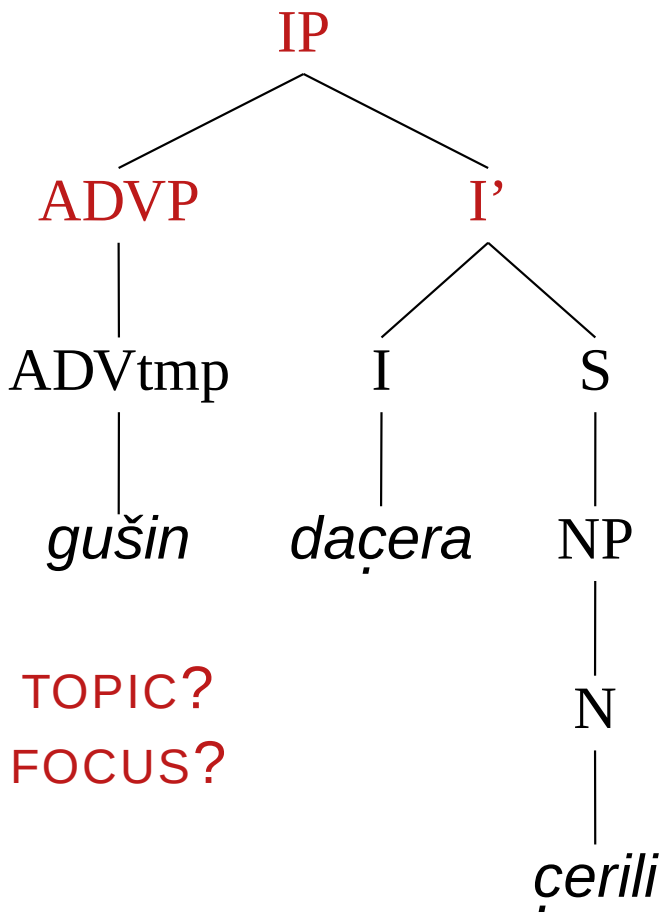
$S \rightarrow XP+$



Phrase structure rules

The Specifier of I is often TOPIC or FOCUS position:

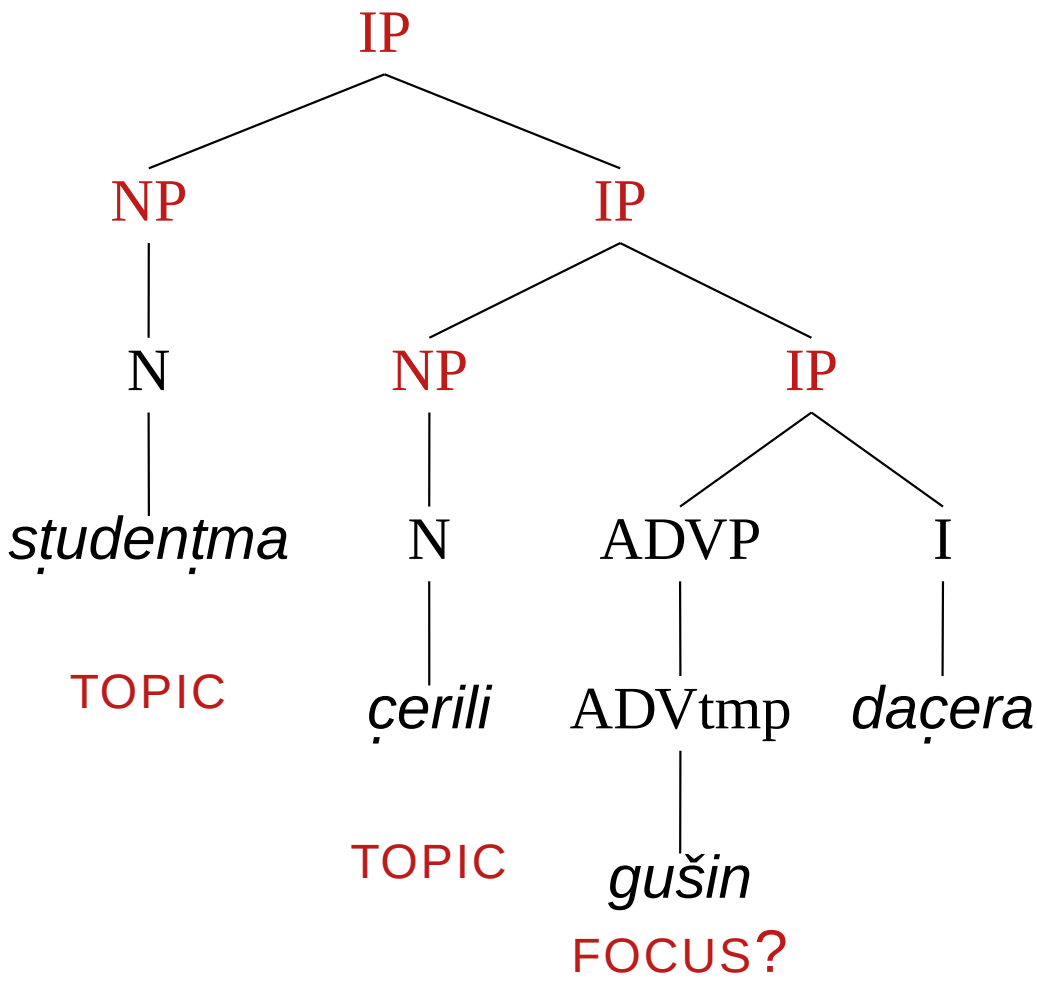
IP → (XP) I'



Phrase structure rules

Material adjoint to IP is TOPIC:

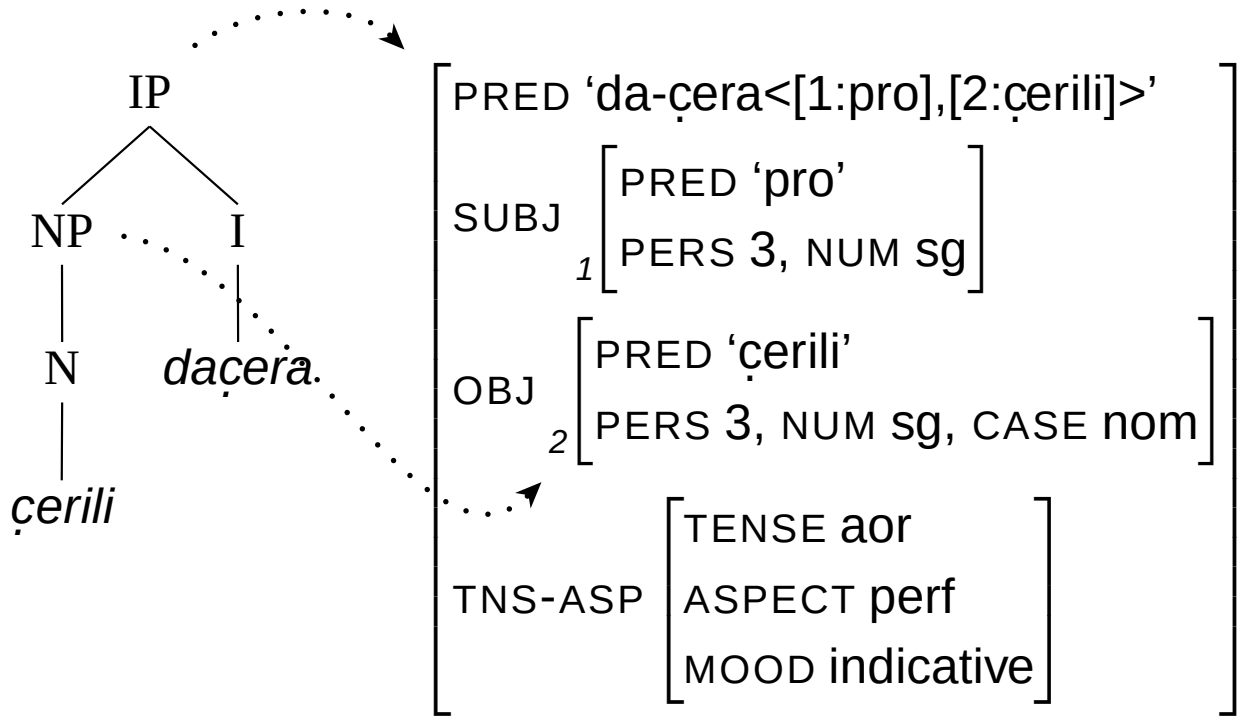
$IP \rightarrow XP\ IP$



Pro-drop and case syntax: Example

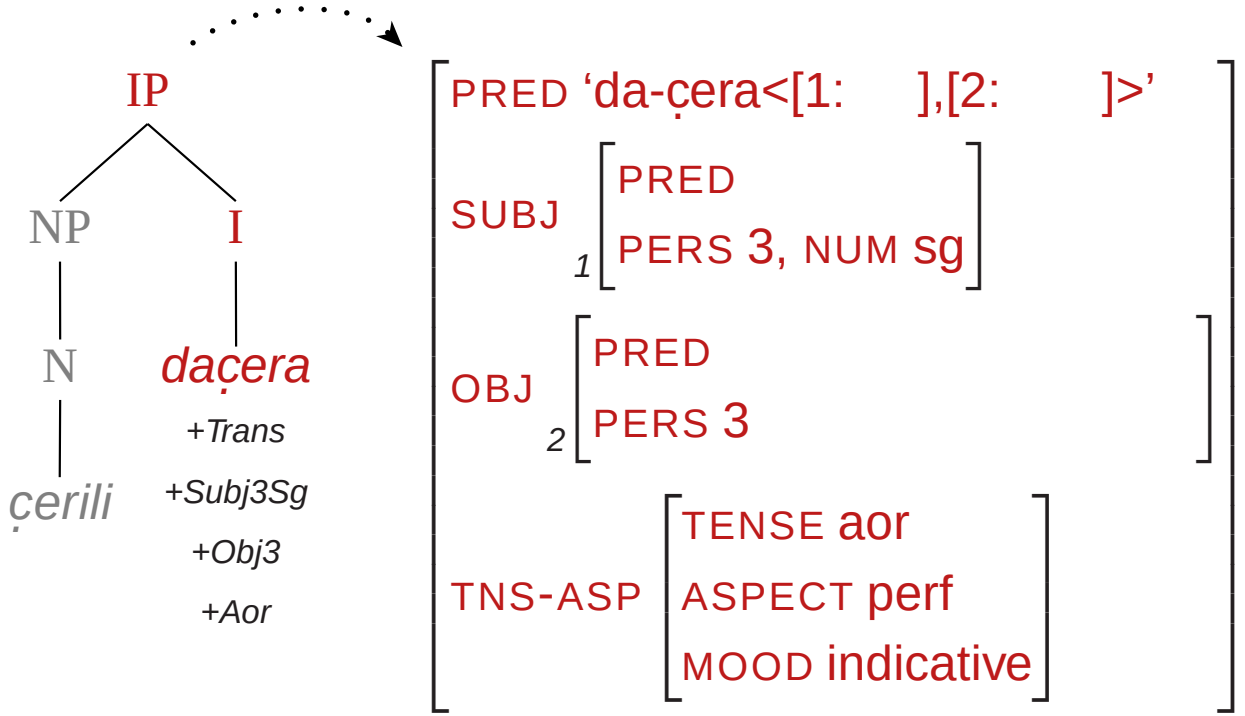
ცerili daცera.
letter.NOM he-wrote-it.AOR

'He wrote a letter.'



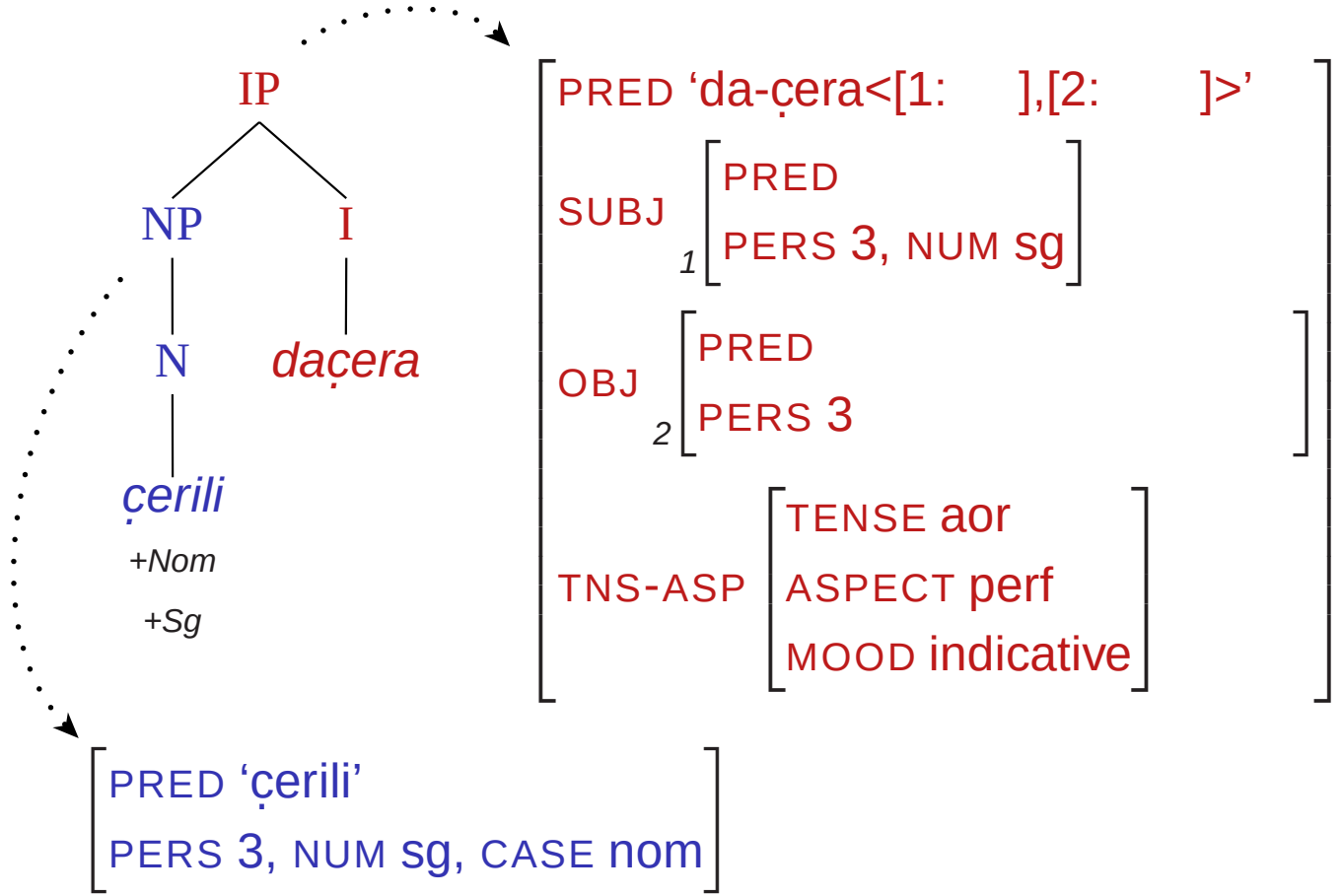
Pro-drop and case syntax: Example

‘He wrote a letter.’



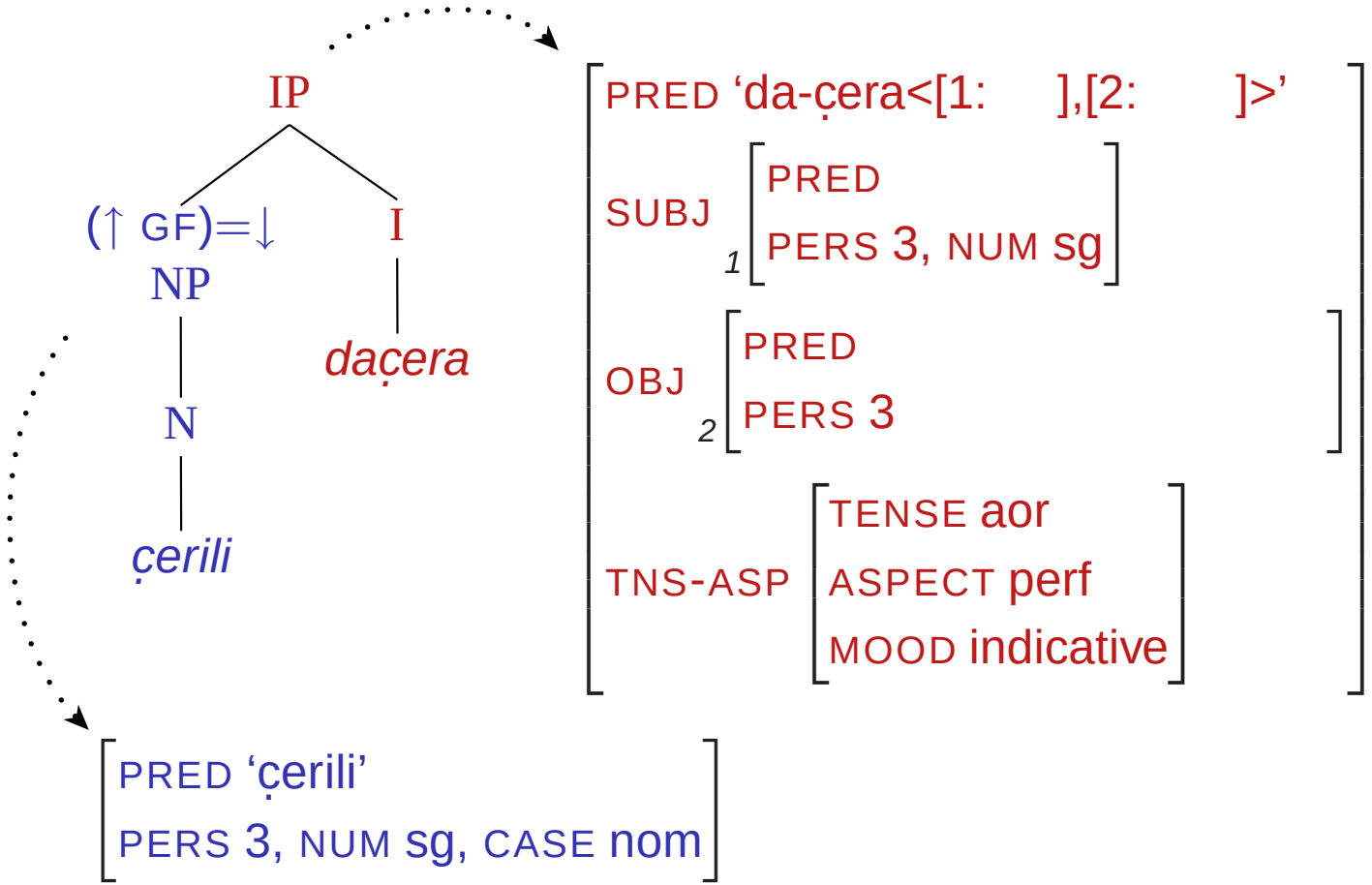
Pro-drop and case syntax: Example

‘He wrote a letter.’



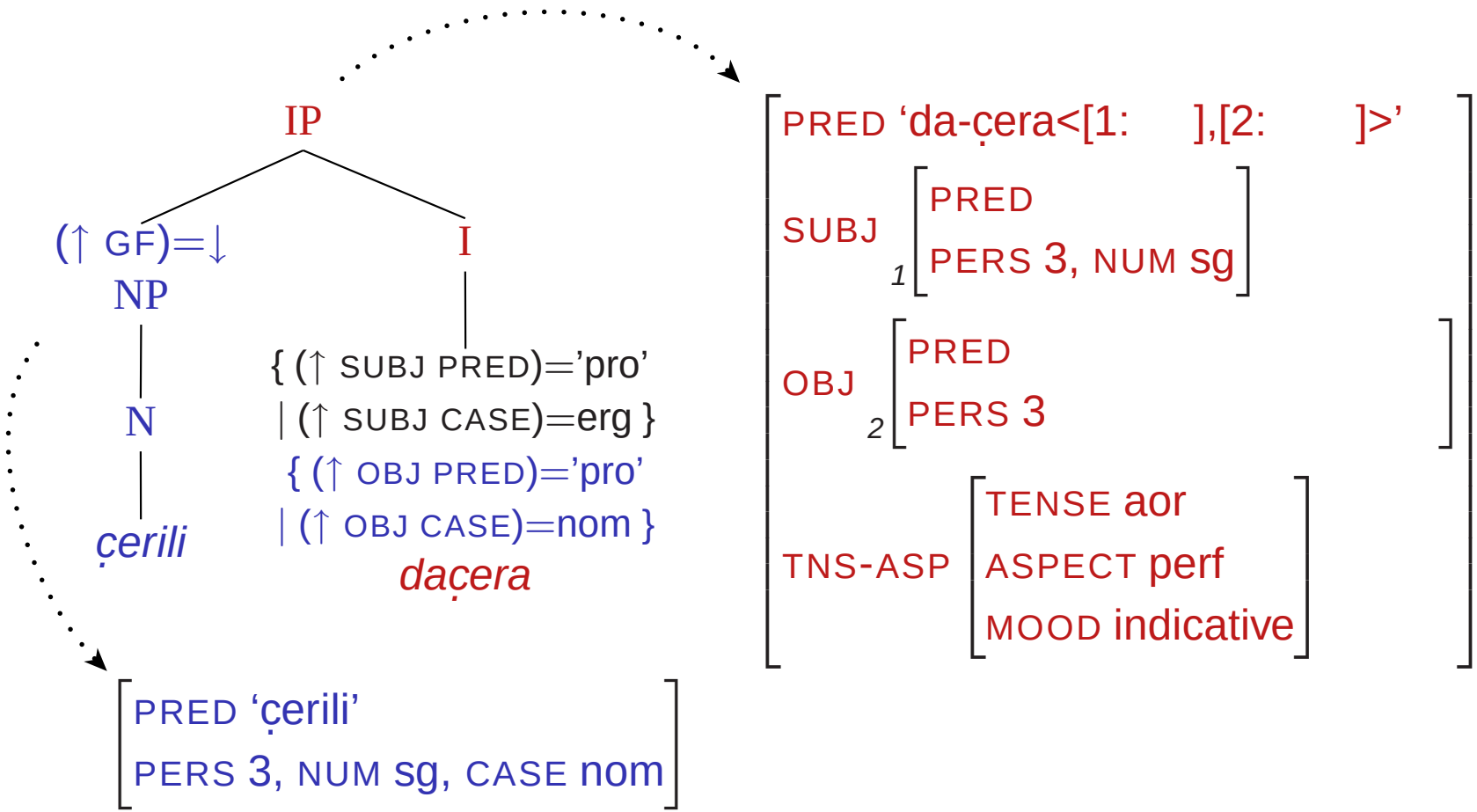
Pro-drop and case syntax: Example

'He wrote a letter.'



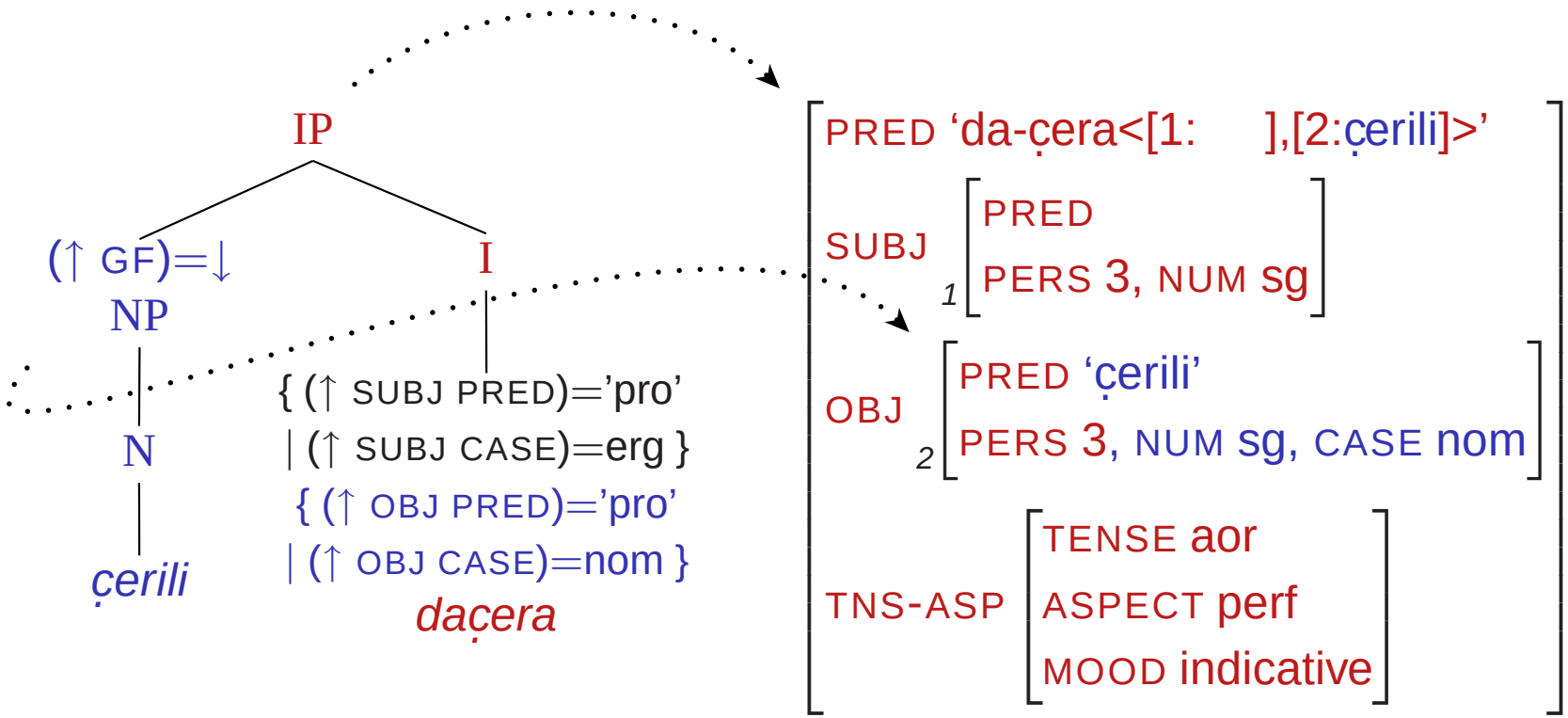
Pro-drop and case syntax: Example

‘He wrote a letter.’



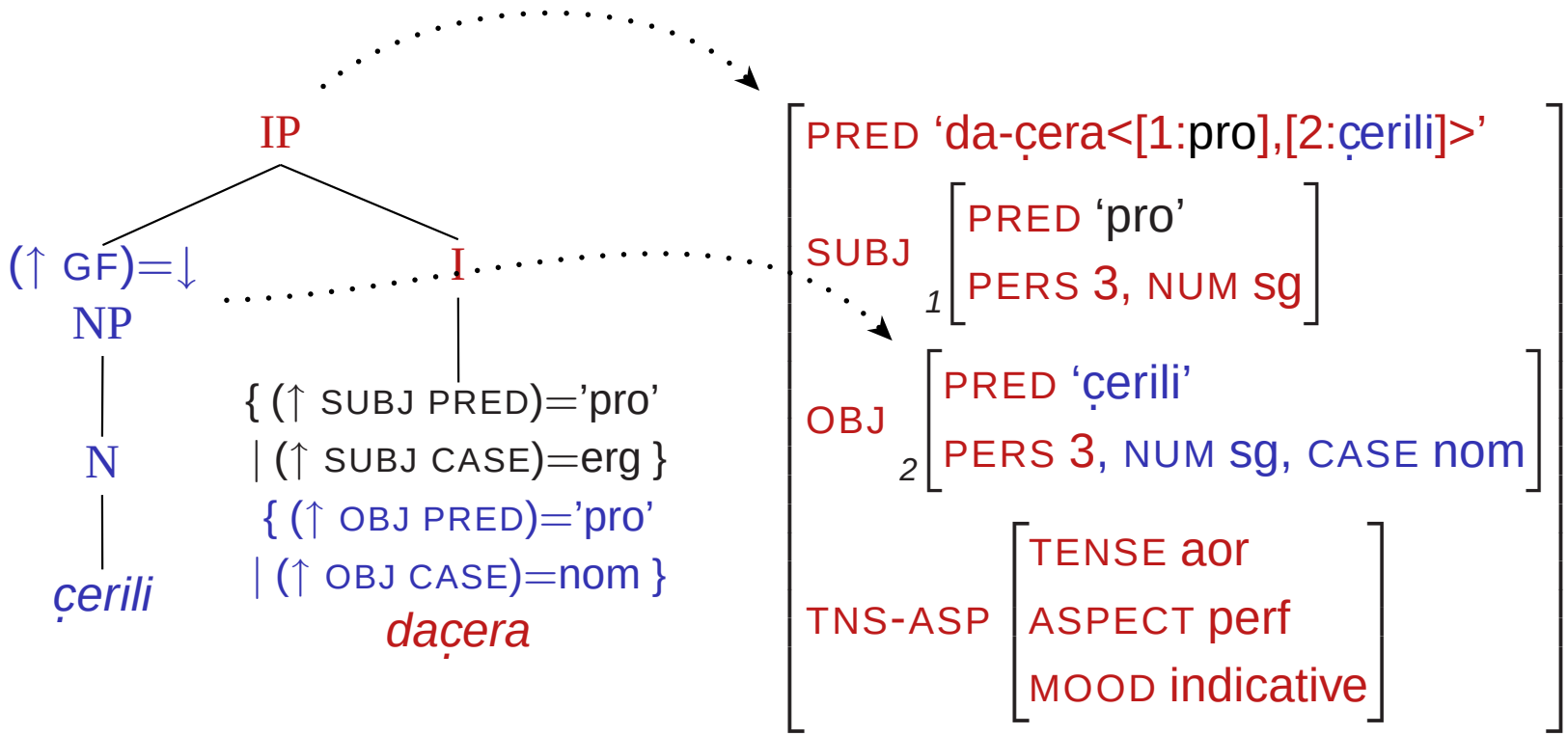
Pro-drop and case syntax: Example

‘He wrote a letter.’



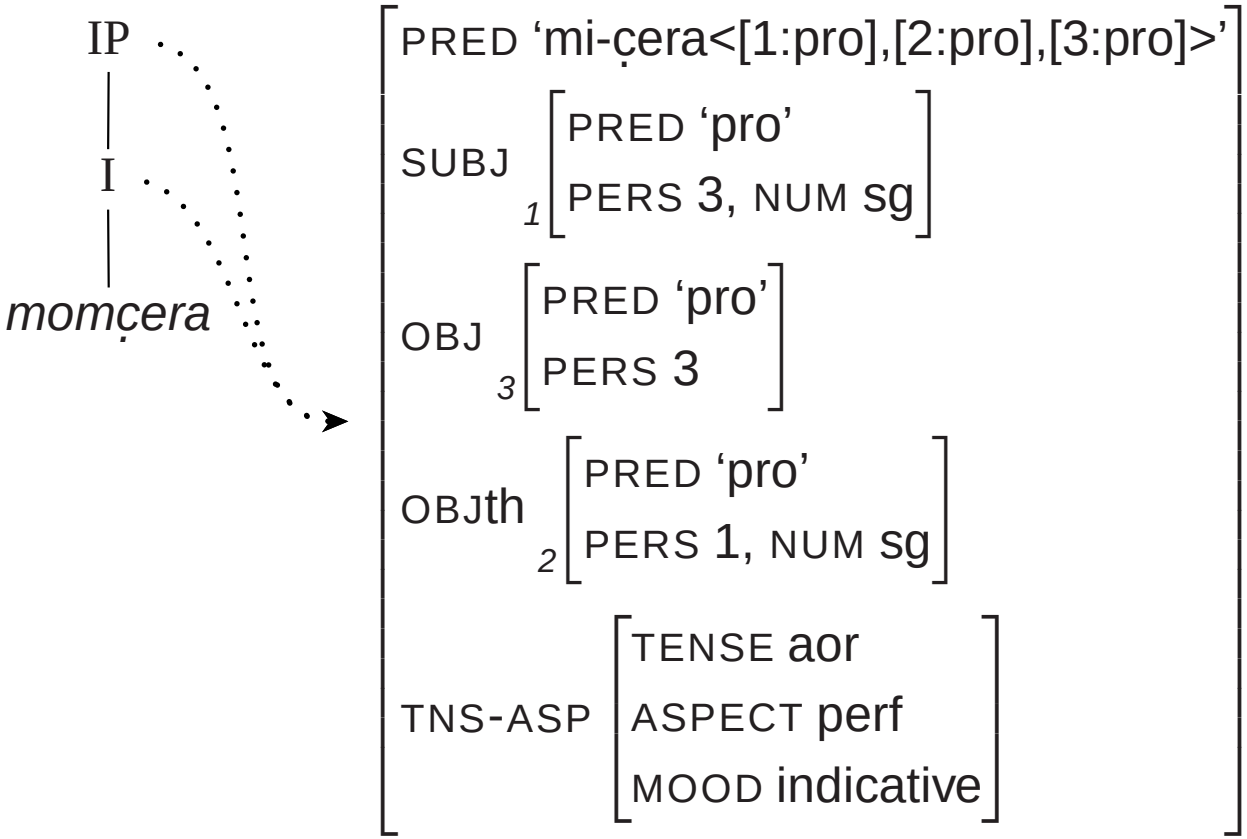
Pro-drop and case syntax: Example

‘He wrote a letter.’



Pro-drop: Example

‘She wrote it to me.’



Predicate coordination

Predicate coordination: two verbs coordinate at V level (as opposed to sentence coordination with ellipsis), and share their arguments.

Predicate coordination

Predicate coordination: two verbs coordinate at V level (as opposed to sentence coordination with ellipsis), and share their arguments.

1. Crosslinguistically common restriction: Both verbs should assign the same semantic roles and grammatical functions to their shared arguments.

Predicate coordination

Predicate coordination: two verbs coordinate at V level (as opposed to sentence coordination with ellipsis), and share their arguments.

1. Crosslinguistically common restriction: Both verbs should assign the same semantic roles and grammatical functions to their shared arguments.
2. Frequent additional restriction: The case of a common argument should be licensed by both verbs, or, when there is case syncretism, be compatible with both verbs' requirements.

Predicate coordination

Predicate coordination: two verbs coordinate at V level (as opposed to sentence coordination with ellipsis), and share their arguments.

1. Crosslinguistically common restriction: Both verbs should assign the same semantic roles and grammatical functions to their shared arguments.
2. Frequent additional restriction: The case of a common argument should be licensed by both verbs, or, when there is case syncretism, be compatible with both verbs' requirements.

In Georgian: less restrictive conditions:

Predicate coordination

Predicate coordination: two verbs coordinate at V level (as opposed to sentence coordination with ellipsis), and share their arguments.

1. Crosslinguistically common restriction: Both verbs should assign the same semantic roles and grammatical functions to their shared arguments.
2. Frequent additional restriction: The case of a common argument should be licensed by both verbs, or, when there is case syncretism, be compatible with both verbs' requirements.

In Georgian: less restrictive conditions:

1. The case of an argument needs only to be licensed by the verb nearest to it.

Predicate coordination: Examples

mas	[uqvars	da	apasebs]	tavis	meuġle-s.
he.DAT	loves	and	esteems	his-own	wife.DAT

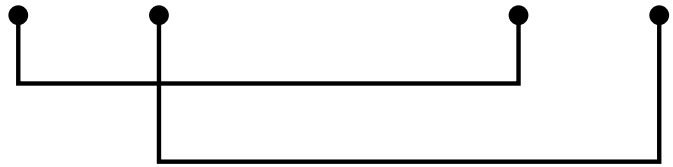
‘He loves and esteems his own wife.’

Predicate coordination: Examples

mas [uqvars da apasebs] tavis meuġle-s.
he.DAT loves and esteems his-own wife.DAT

'He loves and esteems his own wife.'

	uqvars (IV) *'he loves her'*	apasebs (I) *'he esteems her'*
thematic roles	< exp, theme >	< exp, theme >
functions	SUBJ, OBJ	SUBJ, OBJ
case marking	DAT, NOM	NOM, DAT



Predicate coordination: Examples

*me [miqvars da mapasebs] čemi meuġle.
I I-love-her and she-esteems-me my wife.NOM

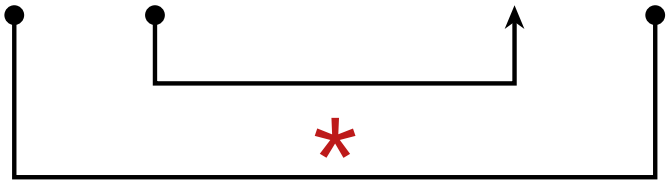
‘I [love, and am esteemed by,] my own wife.’

Predicate coordination: Examples

*me [miqvars da mapasebs] čemi meuġle.
I I-love-her and she-esteems-me my wife.NOM

'I [love, and am esteemed by,] my own wife.'

	uqvars (IV) *'he loves her'*	apasebs (I) *'he esteems her'*
thematic roles	< exp, theme >	< exp, theme >
functions	SUBJ, OBJ	SUBJ, OBJ
case marking	DAT, NOM	NOM, DAT



Predicate coordination: Examples

??misi moġvaċeoba mosċons da pasdeba xalxis mier.
his public-activity it-likes-it and it-is-esteemed the-people by

‘His public activity is liked and esteemed by the people.’

Predicate coordination: Examples

??misi moġvaċeoba mosċons da pasdeba xalxis mier.
his public-activity it-likes-it and it-is-esteemed the-people by

'His public activity is liked and esteemed by the people.'

	mosċons (IV) *'he likes it'*	pasdeba (II) *'it is esteemed'*
case marking	DAT, NOM	mier, NOM
functions	SUBJ, OBJ	OBL, SUBJ
thematic roles	< exp, theme >	< exp, theme >

Predicate coordination: Implementation

Simple case: both verbs license the same cases

Predicate coordination: Implementation

Simple case: both verbs license the same cases

giam iqida da ṭaiḳitxa es ṭigni.
Gia bought and read this book.

Predicate coordination: Implementation

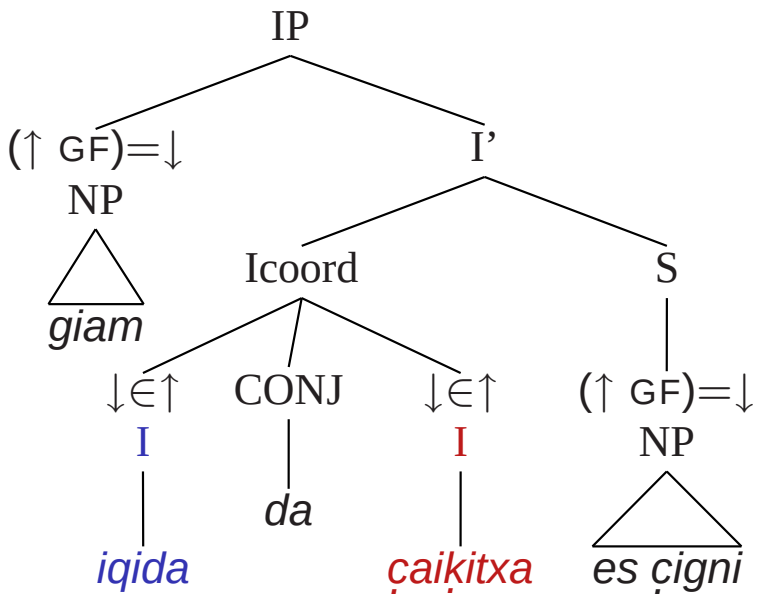
Simple case: both verbs license the same cases

giam iqida da ṭaiḳitxa es ṭigni.
Gia bought and read this book.

Predicate coordination: Implementation

Simple case: both verbs license the same cases

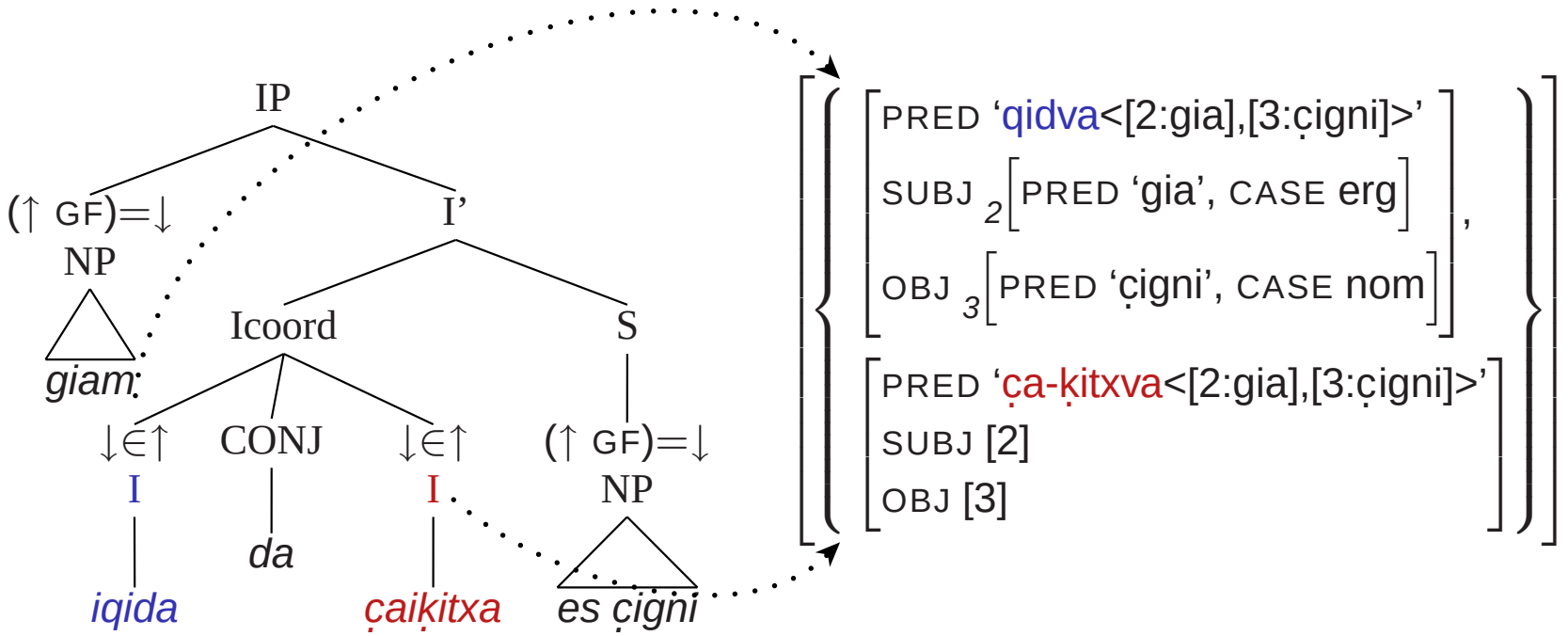
giam iqida da ṭaiḳitxa es ṭigni.
Gia bought and read this book.



Predicate coordination: Implementation

Simple case: both verbs license the same cases

giam iqida da ca-ḳitxa es c-igni.
Gia bought and read this book.

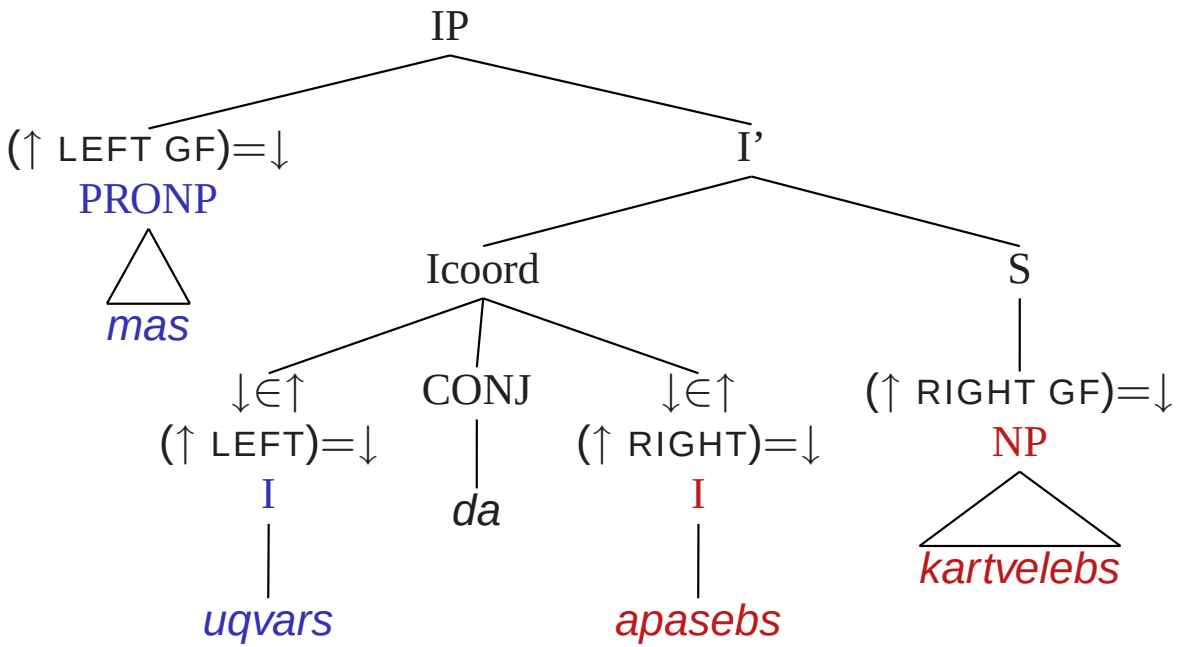


Predicate coordination: Implementation

General case: the verbs license different cases

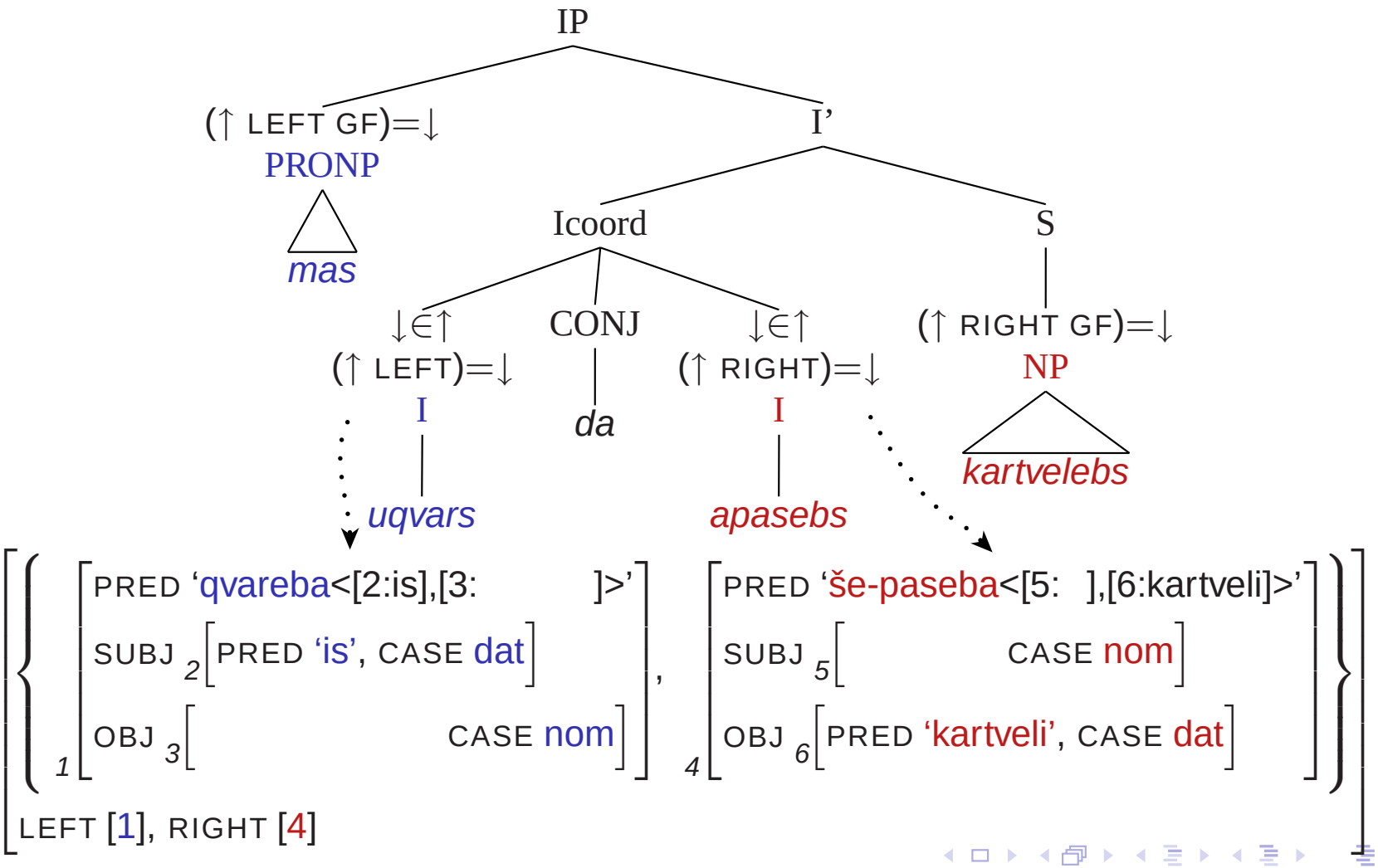
Predicate coordination: Implementation

General case: the verbs license different cases

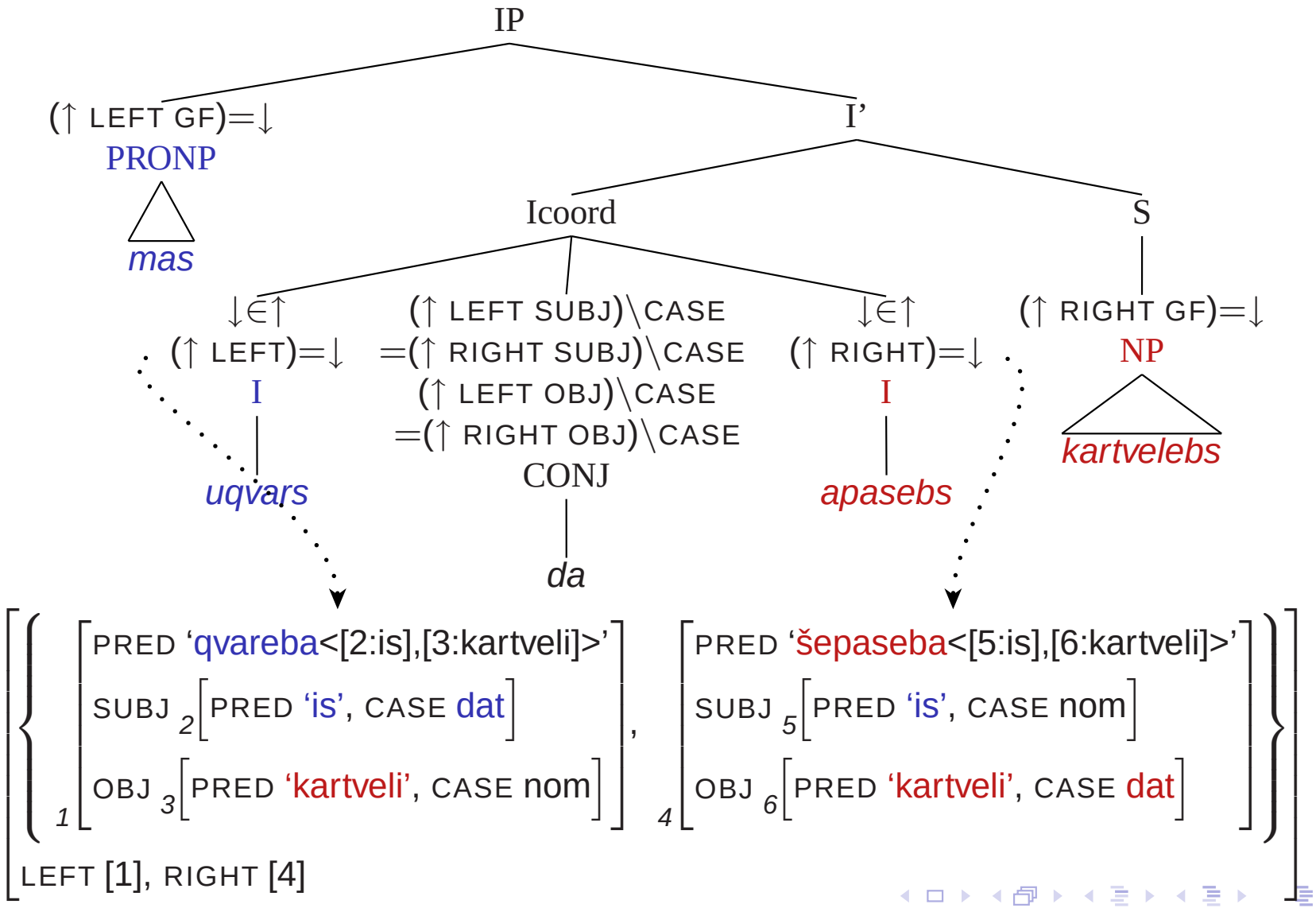


Predicate coordination: Implementation

General case: the verbs license different cases



Predicate coordination: Implementation



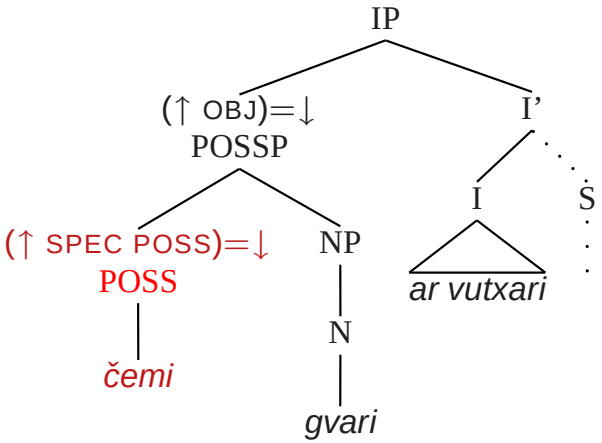
Discontinuous constituents

čem-i	gvar-i	ar	v-u-txar-i
[[my.NOM].POSS	last-name.NOM].NP	not	I.to-him.told.it.

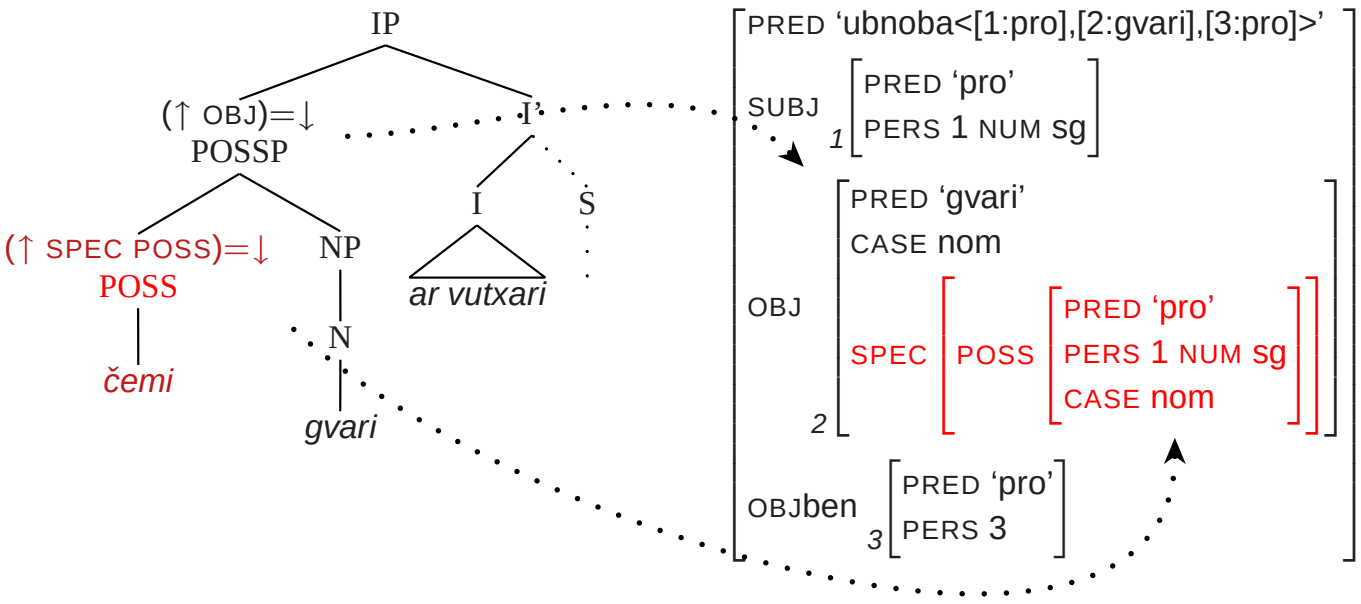
‘I did not tell him my last name.’

gvar-i	ar	v-u-txar-i	čem-i
[last-name.NOM].NP	not	I.to-him.told.it	[my.NOM].POSS.

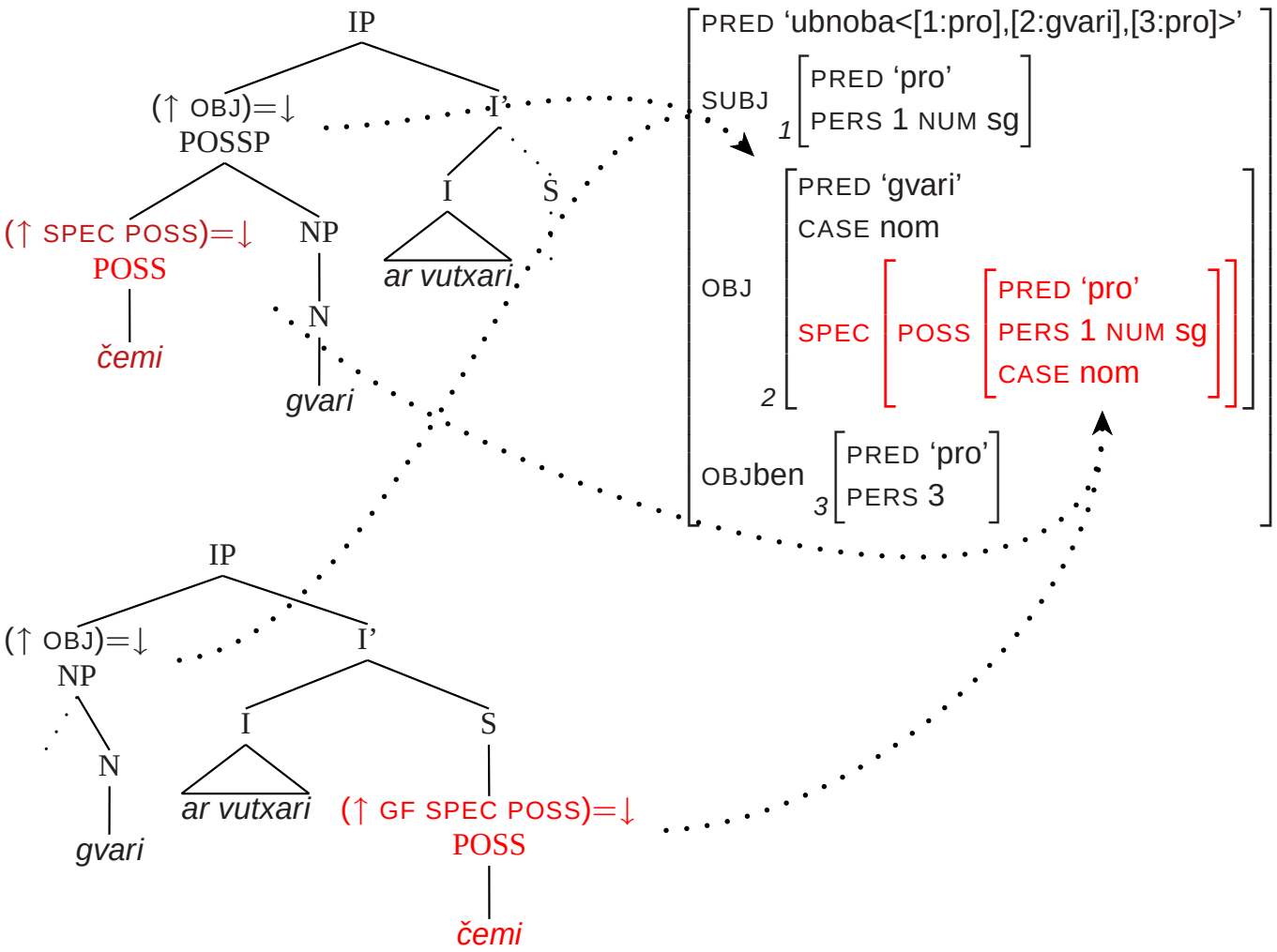
Discontinuous constituents



Discontinuous constituents



Discontinuous constituents



Outline

1 Introduction: The Georgian grammar project

2 Lexical-Functional Grammar

3 Morphology and Morphosyntax

4 Some aspects of Georgian syntax

5 Tools for LFG grammar development

XLE and *fst*

XLE (Xerox Linguistic Environment)

XLE and *fst*

XLE (Xerox Linguistic Environment)

- is a sophisticated development platform for LFG grammars developed by the Palo Alto Research Center with active participation of some of the inventors of LFG.

XLE and *fst*

XLE (Xerox Linguistic Environment)

- is a sophisticated development platform for LFG grammars developed by the Palo Alto Research Center with active participation of some of the inventors of LFG.
- consists of a parser, a generator and a transfer module

XLE and *fst*

XLE (Xerox Linguistic Environment)

- is a sophisticated development platform for LFG grammars developed by the Palo Alto Research Center with active participation of some of the inventors of LFG.
- consists of a parser, a generator and a transfer module
- can be used both from Emacs via a Tcl/Tk interface that provides powerful viewing and debugging facilities, and as a shared library, which opens up for integrating XLE into custom software

XLE and *fst*

XLE (Xerox Linguistic Environment)

- is a sophisticated development platform for LFG grammars developed by the Palo Alto Research Center with active participation of some of the inventors of LFG.
- consists of a parser, a generator and a transfer module
- can be used both from Emacs via a Tcl/Tk interface that provides powerful viewing and debugging facilities, and as a shared library, which opens up for integrating XLE into custom software
- Tokenization and morphological analysis is normally done with the Xerox finite state tool, *fst*

XLE-Web

XLE-Web

XLE-Web

XLE-Web

- easy-to-use pedagogical Web interface to XLE for parsing sentences on the fly

XLE-Web

XLE-Web

- easy-to-use pedagogical Web interface to XLE for parsing sentences on the fly
- in use for several of the ParGram grammars (among others Norwegian, English, German, Welsh and Malagassy)

XLE-Web

XLE-Web

- easy-to-use pedagogical Web interface to XLE for parsing sentences on the fly
- in use for several of the ParGram grammars (among others Norwegian, English, German, Welsh and Malagassy)
- display of c- and f-structures of LFG analyses

XLE-Web

XLE-Web

- easy-to-use pedagogical Web interface to XLE for parsing sentences on the fly
- in use for several of the ParGram grammars (among others Norwegian, English, German, Welsh and Malagassy)
- display of c- and f-structures of LFG analyses
- visualization of the mapping from c- to f-structure

XLE-Web

XLE-Web

- easy-to-use pedagogical Web interface to XLE for parsing sentences on the fly
- in use for several of the ParGram grammars (among others Norwegian, English, German, Welsh and Malagassy)
- display of c- and f-structures of LFG analyses
- visualization of the mapping from c- to f-structure
- display of compact packed representations of c- and f-structures that combine the c- resp. f-structures of all analyses of a given parse into one c- resp. f-structure graph

Grammar development

Tasks when developing a large grammar:
In order to monitor progress, to assess coverage and to compare analyses across different grammar versions:

Grammar development

Tasks when developing a large grammar:
In order to monitor progress, to assess coverage and to compare analyses across different grammar versions:

- run the grammar on a set of sample sentences

Grammar development

Tasks when developing a large grammar:
In order to monitor progress, to assess coverage and to compare analyses across different grammar versions:

- run the grammar on a set of sample sentences
- store the parse results

Grammar development

Tasks when developing a large grammar:
In order to monitor progress, to assess coverage and to compare analyses across different grammar versions:

- run the grammar on a set of sample sentences
- store the parse results
- rerun successive versions of the grammar on the same sentences and compare the results

Grammar development

When the grammar has reached acceptable coverage, one wants to:

Grammar development

When the grammar has reached acceptable coverage, one wants to:

- run the grammar on a larger set of sentences (perhaps chosen from running text)

Grammar development

When the grammar has reached acceptable coverage, one wants to:

- run the grammar on a larger set of sentences (perhaps chosen from running text)
- develop a treebank in the sense of a linguistic resource

Grammar development

When the grammar has reached acceptable coverage, one wants to:

- run the grammar on a larger set of sentences (perhaps chosen from running text)
- develop a treebank in the sense of a linguistic resource

Problems:

Grammar development

When the grammar has reached acceptable coverage, one wants to:

- run the grammar on a larger set of sentences (perhaps chosen from running text)
- develop a treebank in the sense of a linguistic resource

Problems:

- sentences of only moderate complexity often are highly ambiguous

Grammar development

When the grammar has reached acceptable coverage, one wants to:

- run the grammar on a larger set of sentences (perhaps chosen from running text)
- develop a treebank in the sense of a linguistic resource

Problems:

- sentences of only moderate complexity often are highly ambiguous
- the desired or correct reading is only one of the analyses offered by the grammar

Grammar development

When the grammar has reached acceptable coverage, one wants to:

- run the grammar on a larger set of sentences (perhaps chosen from running text)
- develop a treebank in the sense of a linguistic resource

Problems:

- sentences of only moderate complexity often are highly ambiguous
- the desired or correct reading is only one of the analyses offered by the grammar

$\Rightarrow$ Need for manual disambiguation of the parses in an efficient way

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

- developed in the Trepil project (together with Victoria Rosén and Koenraad de Smedt, Bergen)

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

- developed in the Trepil project (together with Victoria Rosén and Koenraad de Smedt, Bergen)
- originally for Norwegian, but language independent

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

- developed in the Trepil project (together with Victoria Rosén and Koenraad de Smedt, Bergen)
- originally for Norwegian, but language independent

Supports a process flow involving

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

- developed in the Trepil project (together with Victoria Rosén and Koenraad de Smedt, Bergen)
- originally for Norwegian, but language independent

Supports a process flow involving

- automatic parsing with XLE

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

- developed in the Trepil project (together with Victoria Rosén and Koenraad de Smedt, Bergen)
- originally for Norwegian, but language independent

Supports a process flow involving

- automatic parsing with XLE
- viewing with XLE-Web

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

- developed in the Trepil project (together with Victoria Rosén and Koenraad de Smedt, Bergen)
- originally for Norwegian, but language independent

Supports a process flow involving

- automatic parsing with XLE
- viewing with XLE-Web
- structural c- and f-structure queries based on the TIGERSearch treebank search tool

LFG Parsebanker

LFG Parsebanker: Web-based toolkit for building and manual disambiguation of an LFG treebank

- developed in the Trepil project (together with Victoria Rosén and Koenraad de Smedt, Bergen)
- originally for Norwegian, but language independent

Supports a process flow involving

- automatic parsing with XLE
- viewing with XLE-Web
- structural c- and f-structure queries based on the TIGERSearch treebank search tool
- efficient manual disambiguation by means of discriminants

Discriminants

Discriminant: ‘Any elementary linguistic property of an analysis that is not shared by all analyses’ (David Carter).

Discriminants

Discriminant: ‘Any elementary linguistic property of an analysis that is not shared by all analyses’ (David Carter).
Our discriminants:

Discriminants

Discriminant: 'Any elementary linguistic property of an analysis that is not shared by all analyses' (David Carter).
Our discriminants:

- specifically designed for LFG grammars

Discriminants

Discriminant: 'Any elementary linguistic property of an analysis that is not shared by all analyses' (David Carter).

Our discriminants:

- specifically designed for LFG grammars
- four major types: lexical, morphological, c-structure and f-structure discriminants

Discriminants

Discriminant: 'Any elementary linguistic property of an analysis that is not shared by all analyses' (David Carter).
Our discriminants:

- specifically designed for LFG grammars
- four major types: lexical, morphological, c-structure and f-structure discriminants
- A lexical discriminant is a word form together with its part of speech

Discriminants

Discriminant: 'Any elementary linguistic property of an analysis that is not shared by all analyses' (David Carter).
Our discriminants:

- specifically designed for LFG grammars
- four major types: lexical, morphological, c-structure and f-structure discriminants
- A lexical discriminant is a word form together with its part of speech
- A morphological discriminant is a base form with the tags it receives from morphological preprocessing

Discriminants

Discriminant: 'Any elementary linguistic property of an analysis that is not shared by all analyses' (David Carter).
Our discriminants:

- specifically designed for LFG grammars
- four major types: lexical, morphological, c-structure and f-structure discriminants
- A lexical discriminant is a word form together with its part of speech
- A morphological discriminant is a base form with the tags it receives from morphological preprocessing
- C-structure discriminants are based on minimal subtrees, a minimal subtree being defined as a mother node and her daughters

Discriminants

Discriminant: 'Any elementary linguistic property of an analysis that is not shared by all analyses' (David Carter).
Our discriminants:

- specifically designed for LFG grammars
- four major types: lexical, morphological, c-structure and f-structure discriminants
- A lexical discriminant is a word form together with its part of speech
- A morphological discriminant is a base form with the tags it receives from morphological preprocessing
- C-structure discriminants are based on minimal subtrees, a minimal subtree being defined as a mother node and her daughters
- F-structure discriminants are based on partial paths through f-structures

A Georgian corpus of non-fictional and fictional texts

An indispensable resource for research in Georgian syntax is a searchable text corpus of decent size.
Available text collections on the Internet:

- the electronic newspaper archive Opentext (> 75 million words)

A Georgian corpus of non-fictional and fictional texts

An indispensable resource for research in Georgian syntax is a searchable text corpus of decent size.
Available text collections on the Internet:

- the electronic newspaper archive Opentext (> 75 million words)
- the text archive of Radio tavisupleba (the Georgian service of Radio Free Europe/Radio Liberty) (around eight million words)

A Georgian corpus of non-fictional and fictional texts

An indispensable resource for research in Georgian syntax is a searchable text corpus of decent size.
Available text collections on the Internet:

- the electronic newspaper archive Opentext (> 75 million words)
- the text archive of Radio tavisupleba (the Georgian service of Radio Free Europe/Radio Liberty) (around eight million words)
- fictional texts (both prose and poetry): the UNESCO project Digital collection of Georgian classical literature (three million words)

A Georgian corpus of non-fictional and fictional texts

An indispensable resource for research in Georgian syntax is a searchable text corpus of decent size.
Available text collections on the Internet:

- the electronic newspaper archive Opentext (> 75 million words)
- the text archive of Radio tavisupleba (the Georgian service of Radio Free Europe/Radio Liberty) (around eight million words)
- fictional texts (both prose and poetry): the UNESCO project Digital collection of Georgian classical literature (three million words)

I have harvested these text collections and imported them into corpus query software based on Corpus Workbench (IMS Stuttgart) which is being developed at Aksis.