Constructions at Play!

Katrien Beuls, Jens Nevens and Paul Van Eecke

Artificial Intelligence Laboratory, VUB, Belgium

Tutorial at the 10th International Conference on Construction Grammar, Paris, 20-07-2018

VRIJE UNIVERSITEIT BRUSSEL

Sneak Preview

https://vimeo.com/80252375

Today's schedule

09:00 – 10:00	Introduction to FCG
10:00 – 11:30	Live-coding session
11:30 – 12:00	*Coffee break*
12:00 – 13:00	Exercises
13:00 – 14:00	*Lunch break*
14:00 – 16:00	The use of CxG in AI applications
17:00 – 19:00	*Farewell cheese and wine*

FLUID CONSTRUCTION GRAMMAR

Main goals

1. Verify models of construction grammar in terms of consistency and preciseness
2. Empirically test construction grammar models on corpora
3. Establish a standard for exchanging and integrating these models
4. Exploit construction grammar insights for language technology

Fluid Construction Grammar

is NOT:

- a grammar
- a linguistic theory
- a machine learning algorithm

Fluid Construction Grammar

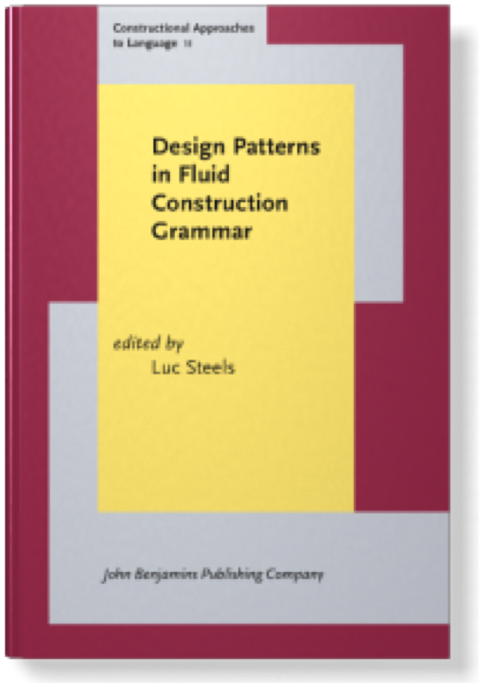
= a computational platform that provides the necessary building blocks for implementing construction grammars

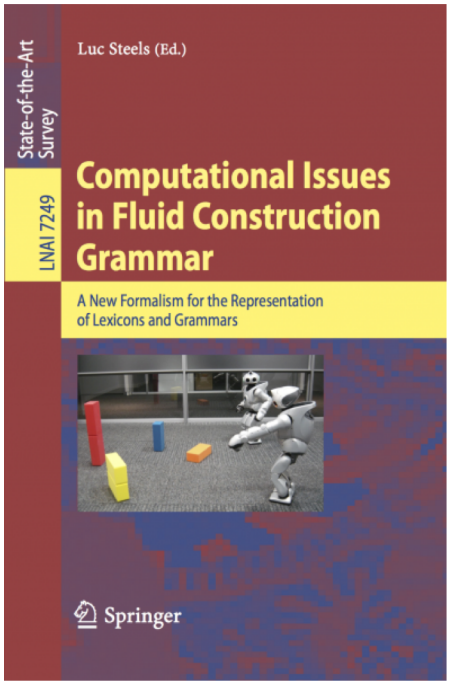
- constructicons supporting the lexicon-grammar continuum
- constructions as meaning/form mappings
- bi-directional processing
- usage-based learning of constructions

very few assumptions

A long history...

- Around since 2000 (Luc Steels)
- Mature, stable version in 2011





A new FCG

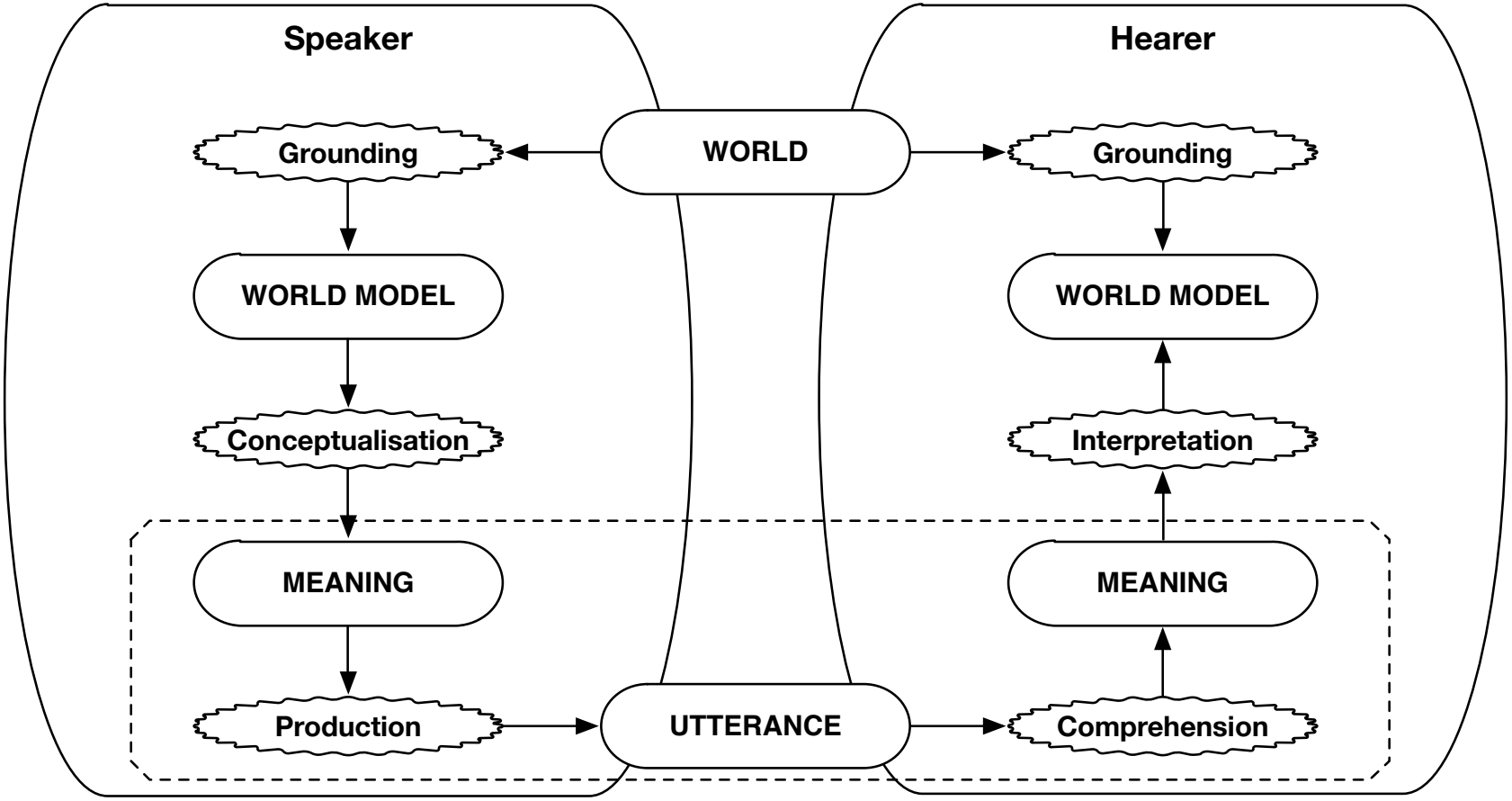
- Introduced in
  - Steels, Luc (2017). Basics of Fluid Construction Grammar. *Constructions and Frames*, 9(2).
  - Van Eecke, Paul and Katrien Beuls (2018) “Syntax and semantics of FCG”. *ELA Github Wiki*

- Main features
  - Representations and visualizations that are closer to Construction Grammar intuitions
  - Easier to learn and use

What is FCG used for?

- Implementing construction grammar models
  - E.g. *Force dynamic meanings of argument structure constructions*
- Evolutionary linguistics experiments
  - E.g. *Emergence of the NP*
- Language technology applications
  - E.g. *Visual question answering*

Semiotic cycle



FCG Constructicon

- Models one language user
- Lexicon-grammar continuum
- Different modes of organization
  - no internal structure
  - families of constructions (subsets)
  - network of constructions
- Holds configurations for processing the grammar

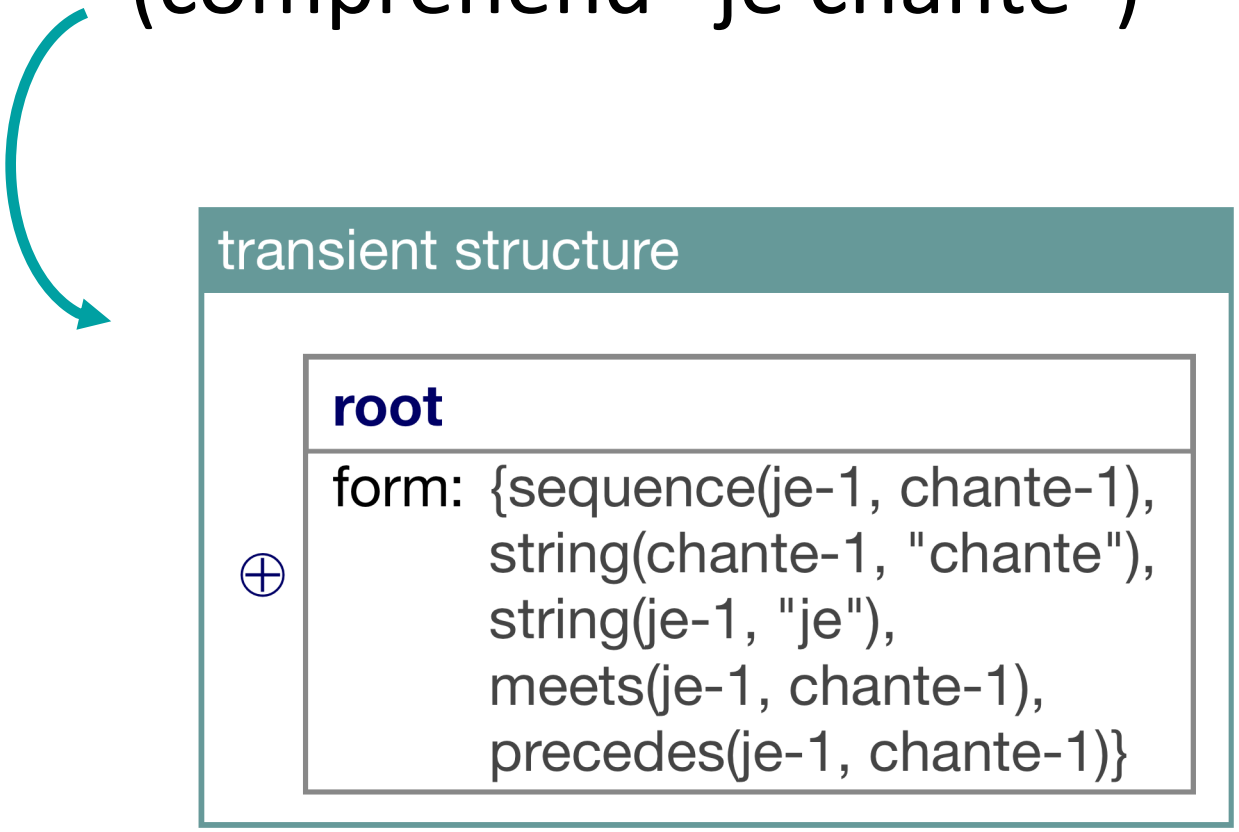
Bi-directional processing

- comprehend
  - utterance
  - constructicon (*cxn-inventory*)

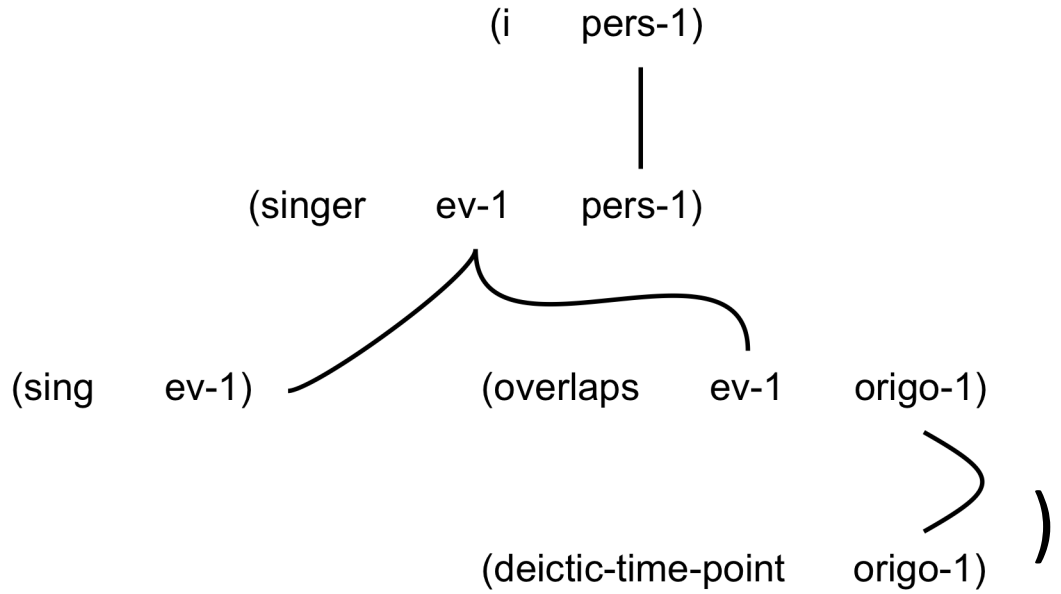
  >> meaning representation
- formulate
  - meaning representation
  - constructicon (*cxn-inventory*)

  >> utterance

(comprehend “je chante”)



(formulate



transient structure

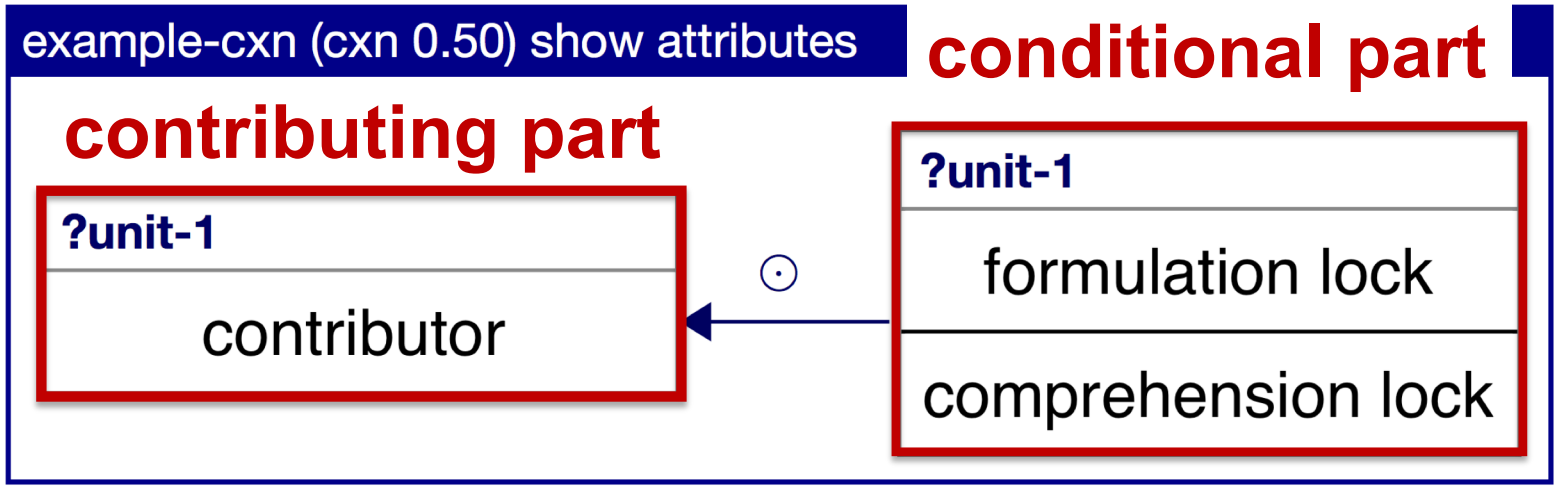
root

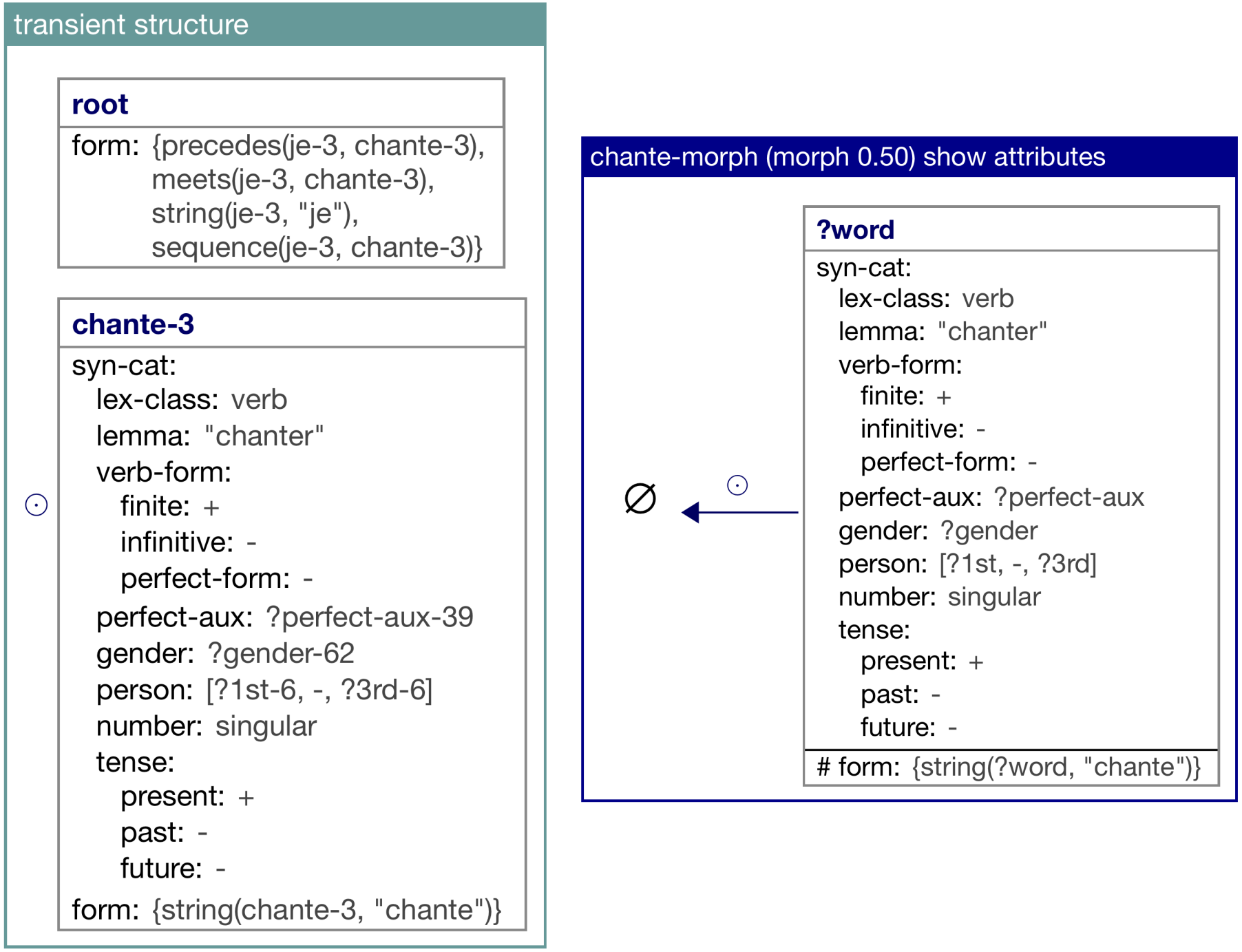
meaning: {i(pers-1),
sing(ev-1),
singer(ev-1, pers-1),
deictic-time-point(origo-1),
overlaps(ev-1, origo-1)}

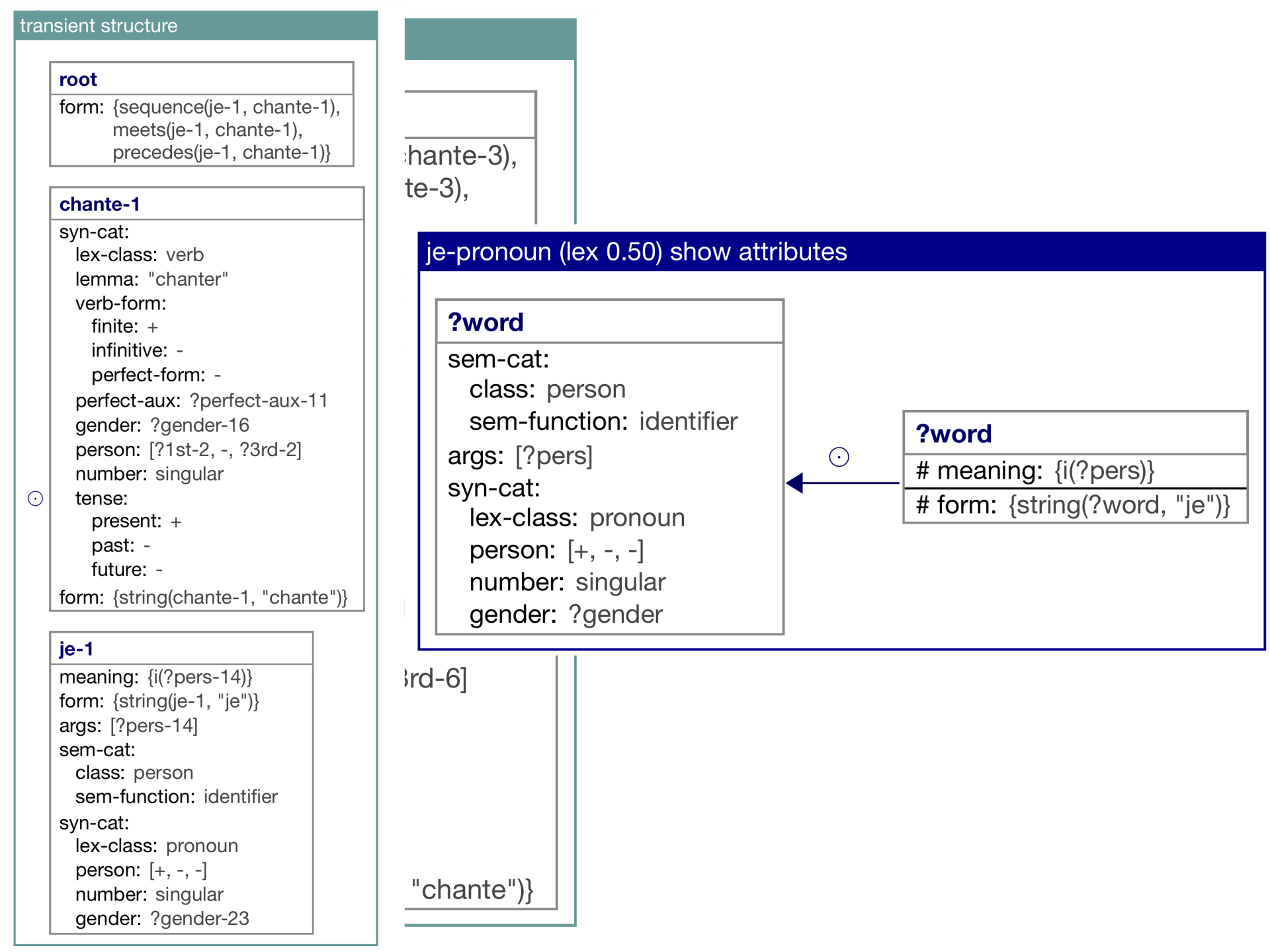
FCG construction

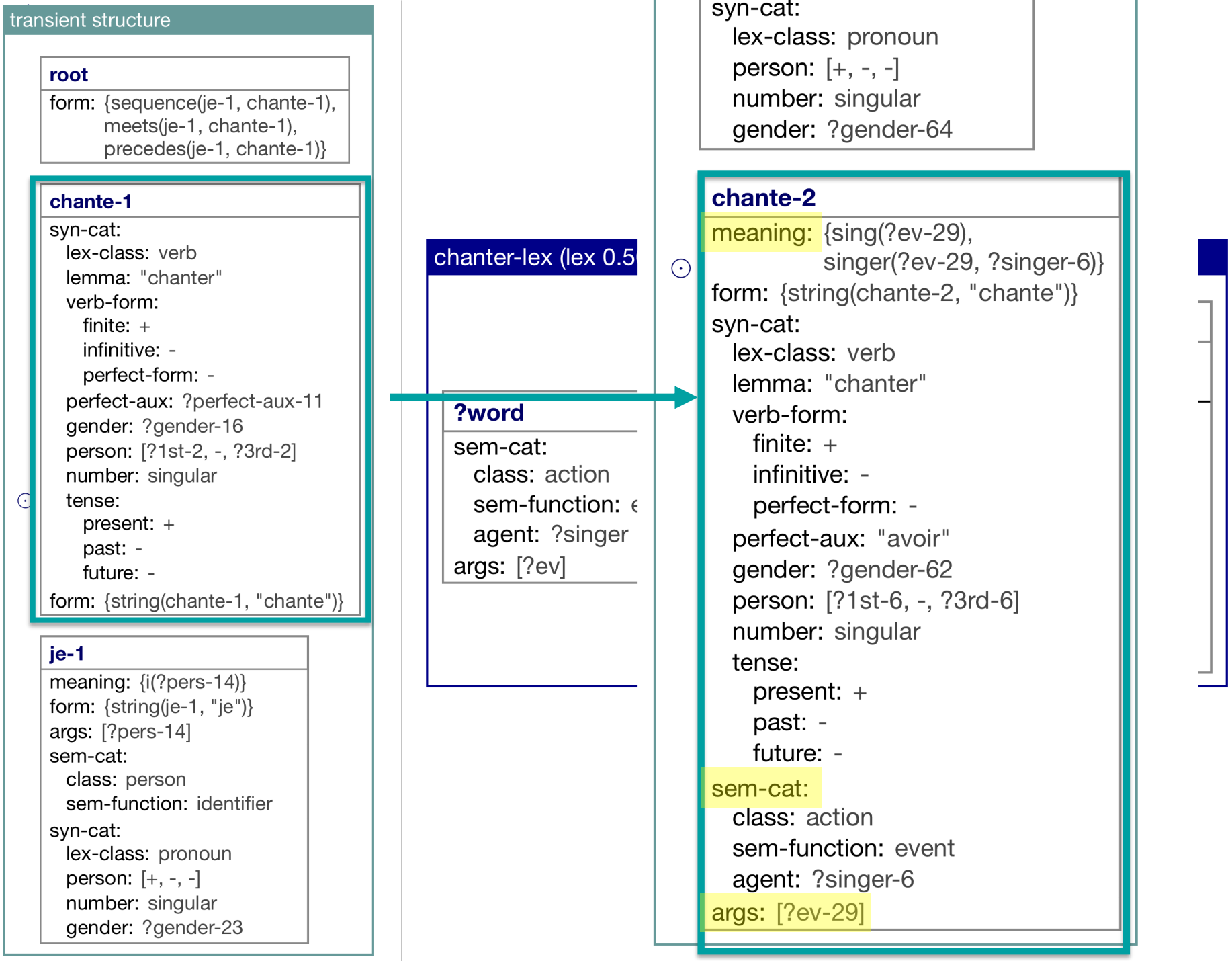
- A coupling between any form and whatever meaning it has
- A construction can consult any aspect of the transient structure to decide how to do so
- Abstract schema that can be used to expand any aspect of a transient structure

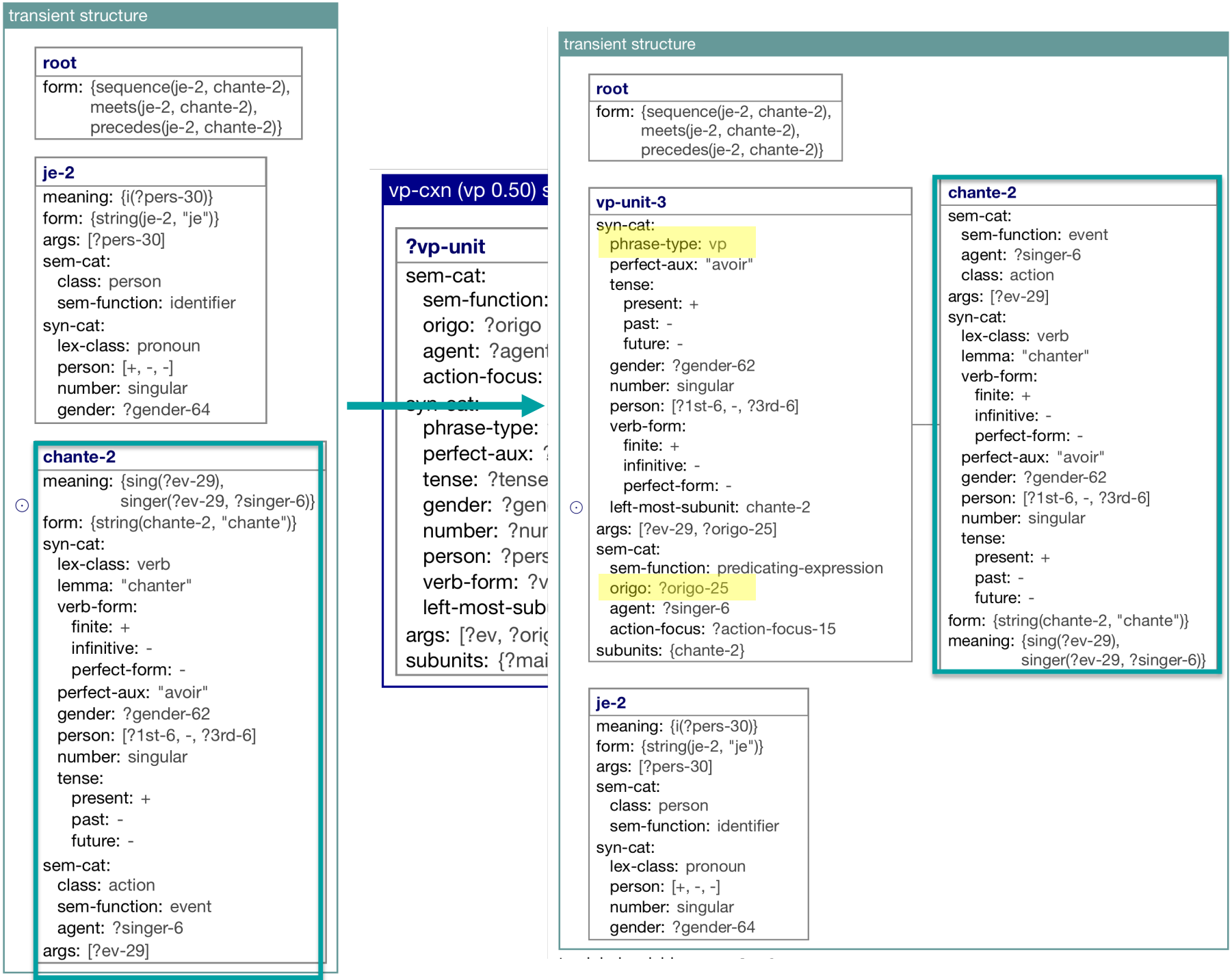
A construction = a data structure with two parts

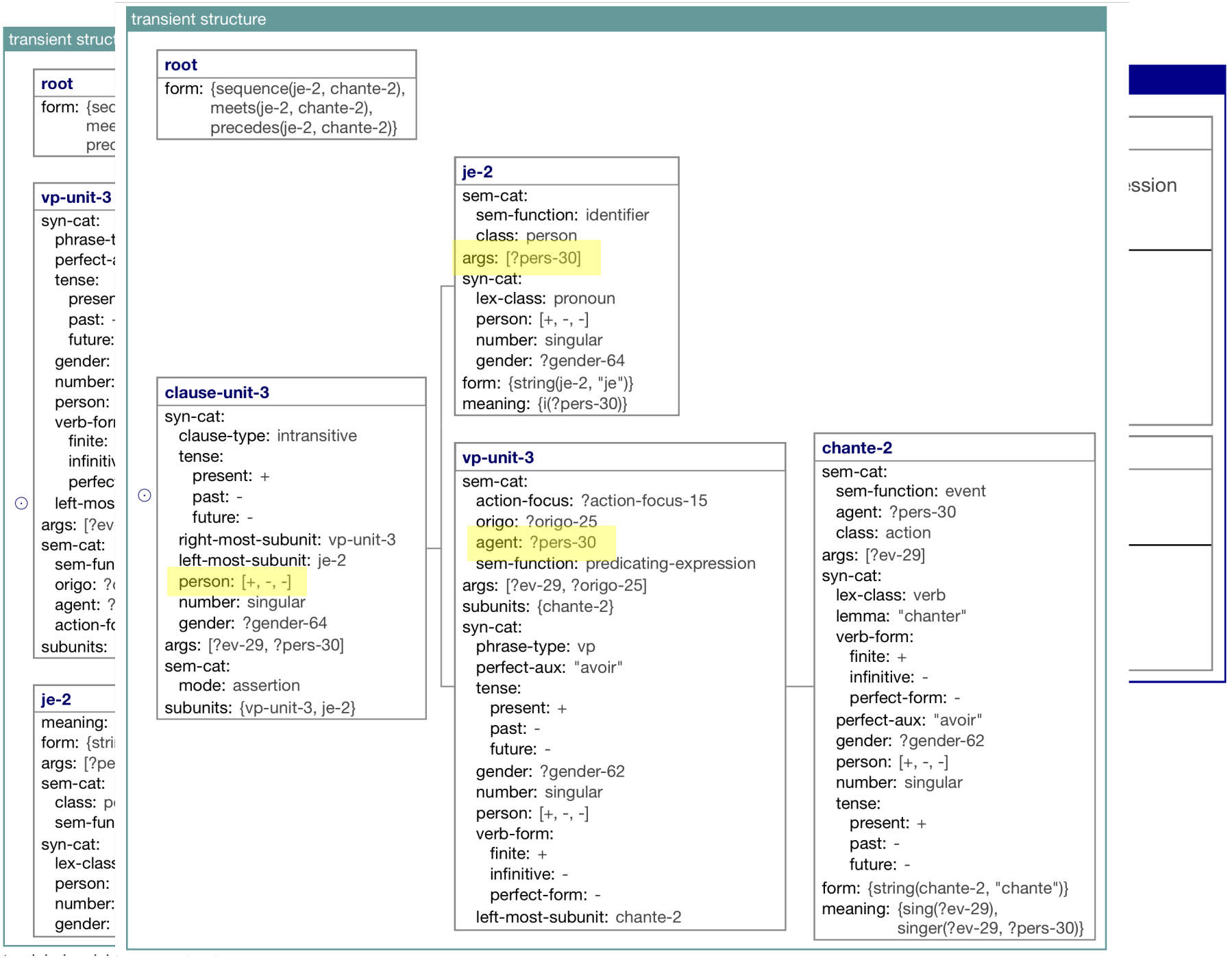


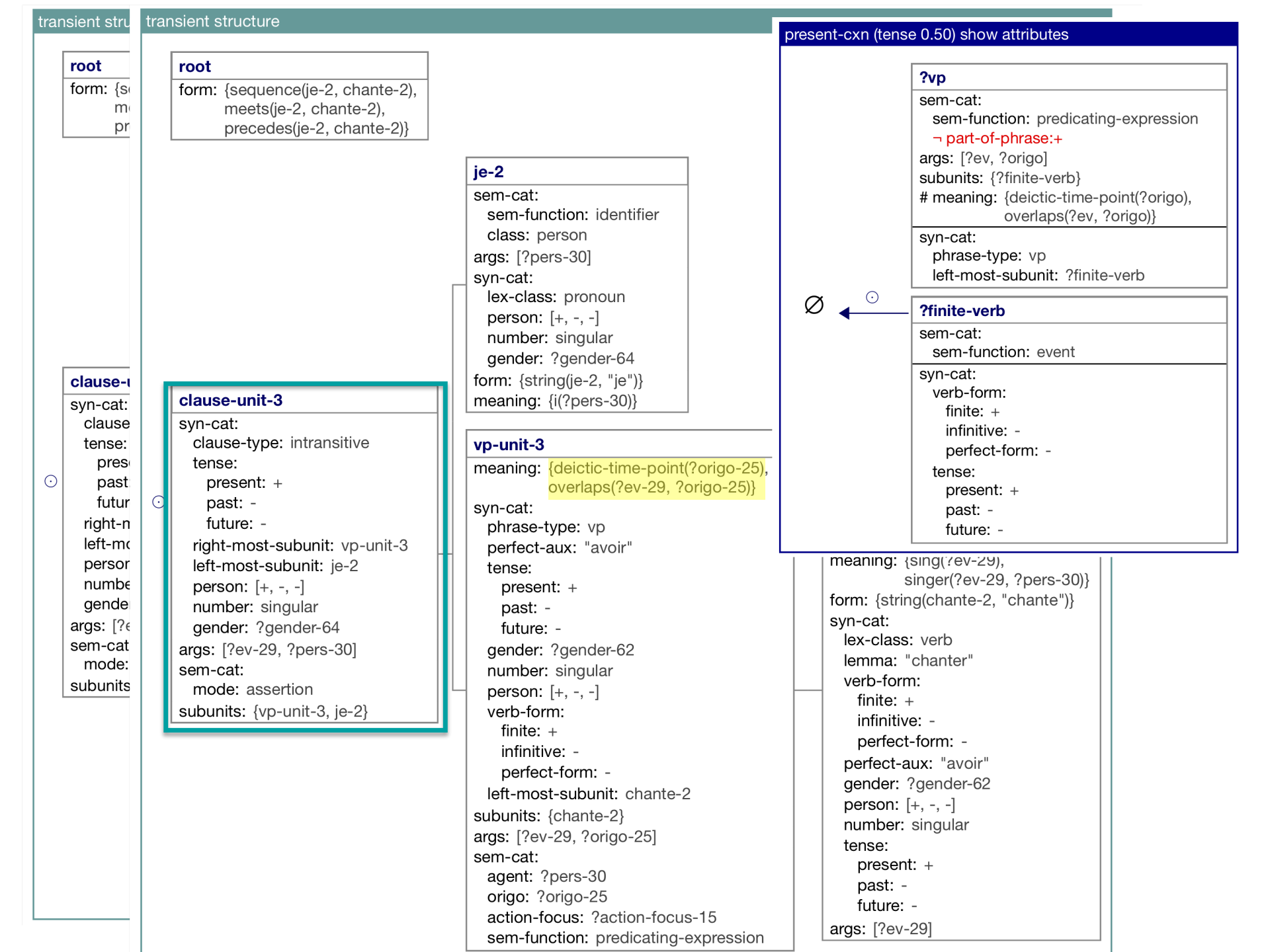

transient structure
root
form: {precedes(je-3, chante-3),
meets(je-3, chante-3),
string(je-3, "je"),
sequence(je-3, chante-3)}
chante-3
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: ?perfect-aux-39
gender: ?gender-62
person: [?1st-6, -, ?3rd-6]
number: singular
tense:
present: +
past: -
future: -
form: {string(chante-3, "chante")}
chante-morph (morph 0.50) show attributes
?word
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: ?perfect-aux
gender: ?gender
person: [?1st, -, ?3rd]
number: singular
tense:
present: +
past: -
future: -
form: {string(?word, "chante")}


transient structure
root
form: {sequence(je-1, chante-1),
meets(je-1, chante-1),
precedes(je-1, chante-1)}
chante-1
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: ?perfect-aux-11
gender: ?gender-16
person: [?1st-2, -, ?3rd-2]
number: singular
tense:
present: +
past: -
future: -
form: {string(chante-1, "chante")}
je-1
meaning: {i(?pers-14)}
form: {string(je-1, "je")}
args: [?pers-14]
sem-cat:
class: person
sem-function: identifier
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender-23
hante-3),
te-3),
je-pronoun (lex 0.50) show attributes
?word
sem-cat:
class: person
sem-function: identifier
args: [?pers]
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender
?word
meaning: {i(?pers)}
form: {string(?word, "je")}
rd-6]
"chante")}

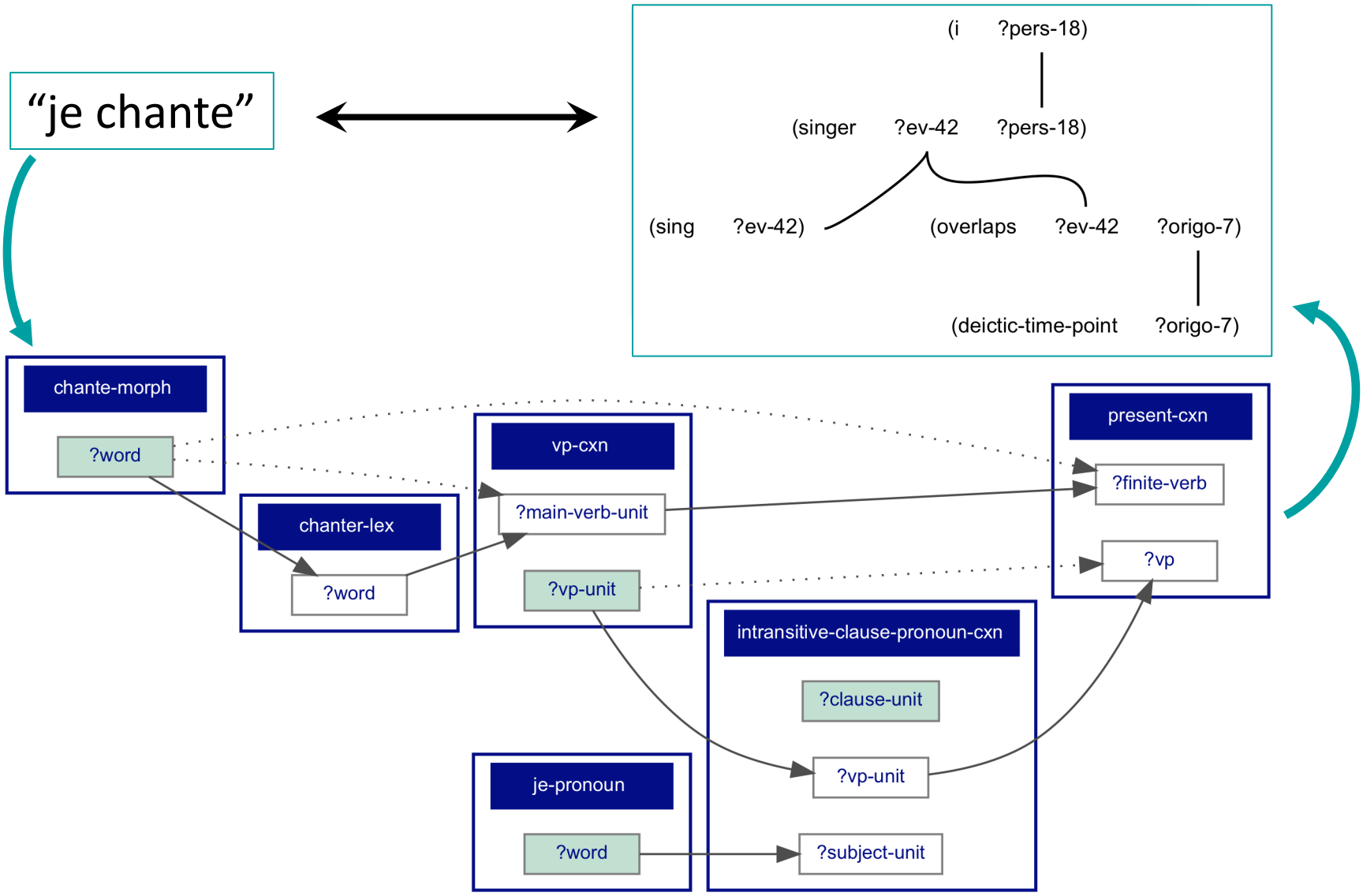

transient structure
root
form: {sequence(je-1, chante-1),
meets(je-1, chante-1),
precedes(je-1, chante-1)}
chante-1
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: ?perfect-aux-11
gender: ?gender-16
person: [?1st-2, -, ?3rd-2]
number: singular
tense:
present: +
past: -
future: -
form: {string(chante-1, "chante")}
je-1
meaning: {i(?pers-14)}
form: {string(je-1, "je")}
args: [?pers-14]
sem-cat:
class: person
sem-function: identifier
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender-23
chanter-lex (lex 0.5
?word
sem-cat:
class: action
sem-function: e
agent: ?singer
args: [?ev]
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender-64
chante-2
meaning: {sing(?ev-29),
singer(?ev-29, ?singer-6)}
form: {string(chante-2, "chante")}
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: "avoir"
gender: ?gender-62
person: [?1st-6, -, ?3rd-6]
number: singular
tense:
present: +
past: -
future: -
sem-cat:
class: action
sem-function: event
agent: ?singer-6
args: [?ev-29]


transient structure
root
form: {sequence(je-2, chante-2),
meets(je-2, chante-2),
precedes(je-2, chante-2)}
je-2
meaning: {i(?pers-30)}
form: {string(je-2, "je")}
args: [?pers-30]
sem-cat:
class: person
sem-function: identifier
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender-64
chante-2
meaning: {sing(?ev-29),
singer(?ev-29, ?singer-6)}
form: {string(chante-2, "chante")}
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: "avoir"
gender: ?gender-62
person: [?1st-6, -, ?3rd-6]
number: singular
tense:
present: +
past: -
future: -
sem-cat:
class: action
sem-function: event
agent: ?singer-6
args: [?ev-29]
vp-cxn (vp 0.50)
?vp-unit
sem-cat:
sem-function:
origo: ?origo
agent: ?agent
action-focus:
phrase-type:
perfect-aux:
tense: ?tense
gender: ?gen
number: ?nur
person: ?pers
verb-form: ?v
left-most-sub
args: [?ev, ?orig
subunits: {?mai
transient structure
root
form: {sequence(je-2, chante-2),
meets(je-2, chante-2),
precedes(je-2, chante-2)}
vp-unit-3
syn-cat:
phrase-type: vp
perfect-aux: "avoir"
tense:
present: +
past: -
future: -
gender: ?gender-62
number: singular
person: [?1st-6, -, ?3rd-6]
verb-form:
finite: +
infinitive: -
perfect-form: -
left-most-subunit: chante-2
args: [?ev-29, ?origo-25]
sem-cat:
sem-function: predicating-expression
origo: ?origo-25
agent: ?singer-6
action-focus: ?action-focus-15
subunits: {chante-2}
chante-2
sem-cat:
sem-function: event
agent: ?singer-6
class: action
args: [?ev-29]
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: "avoir"
gender: ?gender-62
person: [?1st-6, -, ?3rd-6]
number: singular
tense:
present: +
past: -
future: -
form: {string(chante-2, "chante")}
meaning: {sing(?ev-29),
singer(?ev-29, ?singer-6)}
je-2
meaning: {i(?pers-30)}
form: {string(je-2, "je")}
args: [?pers-30]
sem-cat:
class: person
sem-function: identifier
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender-64

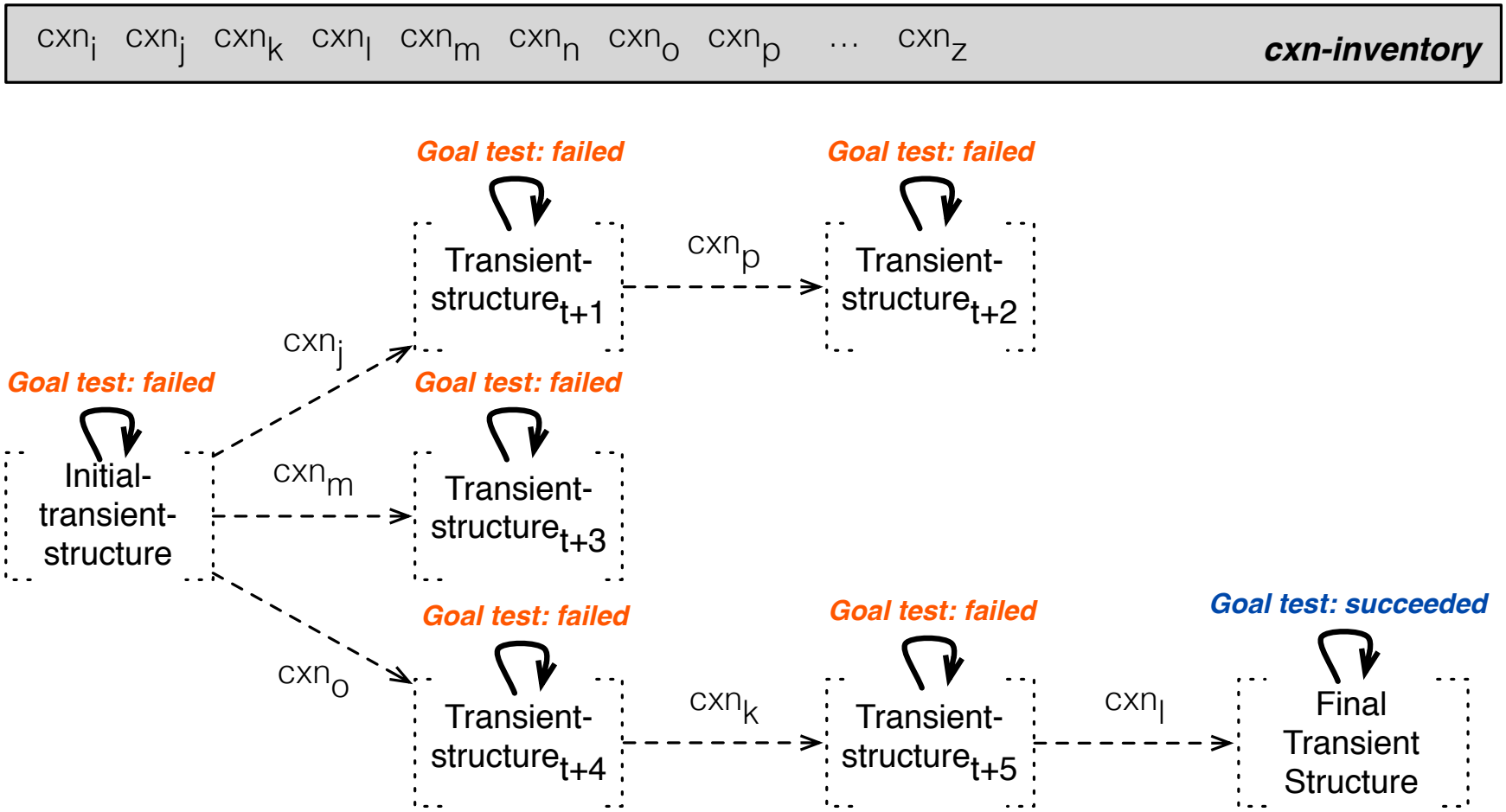

transient structure
root
form: {sequence(je-2, chante-2),
meets(je-2, chante-2),
precedes(je-2, chante-2)}
je-2
sem-cat:
sem-function: identifier
class: person
args: [?pers-30]
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender-64
form: {string(je-2, "je")}
meaning: {i(?pers-30)}
clause-unit-3
syn-cat:
clause-type: intransitive
tense:
present: +
past: -
future: -
right-most-subunit: vp-unit-3
left-most-subunit: je-2
person: [+, -, -]
number: singular
gender: ?gender-64
args: [?ev-29, ?pers-30]
sem-cat:
mode: assertion
subunits: {vp-unit-3, je-2}
vp-unit-3
sem-cat:
action-focus: ?action-focus-15
origo: ?origo-25
agent: ?pers-30
sem-function: predicating-expression
args: [?ev-29, ?origo-25]
subunits: {chante-2}
syn-cat:
phrase-type: vp
perfect-aux: "avoir"
tense:
present: +
past: -
future: -
gender: ?gender-62
number: singular
person: [+, -, -]
verb-form:
finite: +
infinitive: -
perfect-form: -
left-most-subunit: chante-2
chante-2
sem-cat:
sem-function: event
agent: ?pers-30
class: action
args: [?ev-29]
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: "avoir"
gender: ?gender-62
person: [+, -, -]
number: singular
tense:
present: +
past: -
future: -
form: {string(chante-2, "chante")}
meaning: {sing(?ev-29),
singer(?ev-29, ?pers-30)}


transient stru
transient structure
root
form: {s
m
pr
root
form: {sequence(je-2, chante-2),
meets(je-2, chante-2),
precedes(je-2, chante-2)}
present-cxn (tense 0.50) show attributes
?vp
sem-cat:
sem-function: predicating-expression
¬ part-of-phrase:+
args: [?ev, ?origo]
subunits: {?finite-verb}
meaning: {deictic-time-point(?origo),
overlaps(?ev, ?origo)}
syn-cat:
phrase-type: vp
left-most-subunit: ?finite-verb
?finite-verb
sem-cat:
sem-function: event
syn-cat:
verb-form:
finite: +
infinitive: -
perfect-form: -
tense:
present: +
past: -
future: -
je-2
sem-cat:
sem-function: identifier
class: person
args: [?pers-30]
syn-cat:
lex-class: pronoun
person: [+, -, -]
number: singular
gender: ?gender-64
form: {string(je-2, "je")}
meaning: {i(?pers-30)}
clause-u
syn-cat:
clause
tense:
pres
past
futur
right-n
left-mo
persor
numbe
gende
args: [?e
sem-cat
mode:
subunits
clause-unit-3
syn-cat:
clause-type: intransitive
tense:
present: +
past: -
future: -
right-most-subunit: vp-unit-3
left-most-subunit: je-2
person: [+, -, -]
number: singular
gender: ?gender-64
args: [?ev-29, ?pers-30]
sem-cat:
mode: assertion
subunits: {vp-unit-3, je-2}
vp-unit-3
meaning: {deictic-time-point(?origo-25),
overlaps(?ev-29, ?origo-25)}
syn-cat:
phrase-type: vp
perfect-aux: "avoir"
tense:
present: +
past: -
future: -
gender: ?gender-62
number: singular
person: [+, -, -]
verb-form:
finite: +
infinitive: -
perfect-form: -
left-most-subunit: chante-2
subunits: {chante-2}
args: [?ev-29, ?origo-25]
sem-cat:
agent: ?pers-30
origo: ?origo-25
action-focus: ?action-focus-15
sem-function: predicating-expression
meaning: {sing(?ev-29),
singer(?ev-29, ?pers-30)}
form: {string(chante-2, "chante")}
syn-cat:
lex-class: verb
lemma: "chanter"
verb-form:
finite: +
infinitive: -
perfect-form: -
perfect-aux: "avoir"
gender: ?gender-62
person: [+, -, -]
number: singular
tense:
present: +
past: -
future: -
args: [?ev-29]

How can constructions combine freely?



Language as a problem solving process

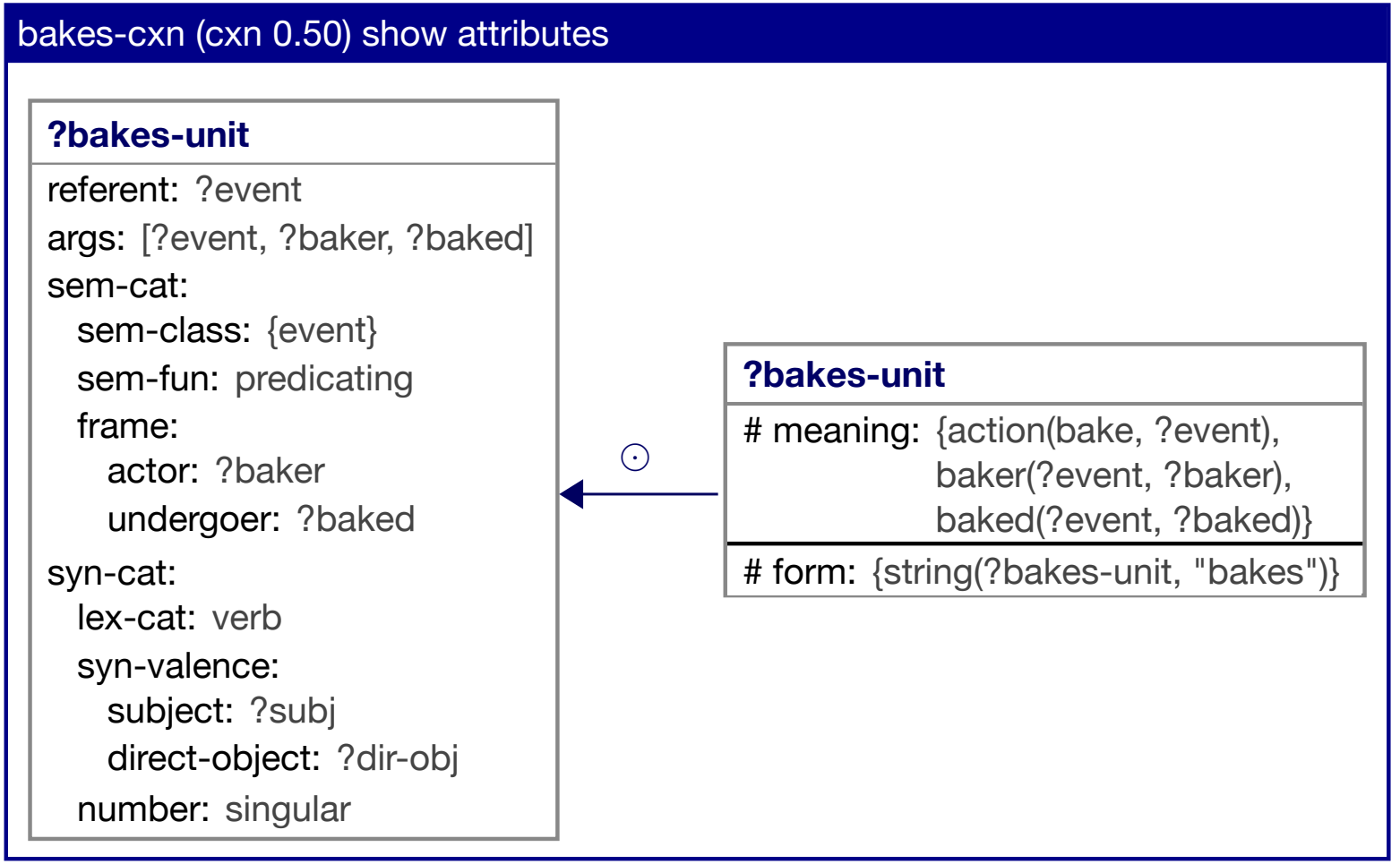


DEMONSTRATION

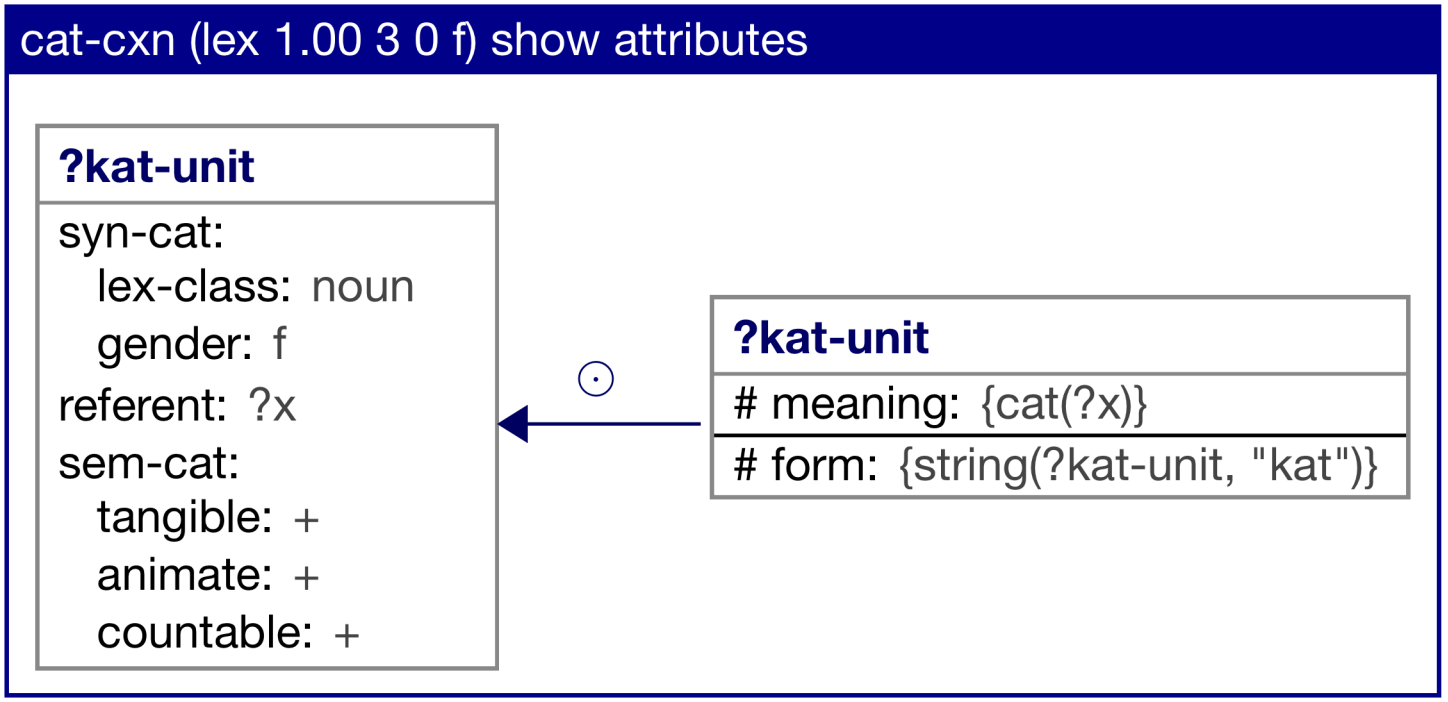
Few assumptions

- No predefined features
  - different languages and different analyses use different features
- No grammaticality judgements
  - communicative success
  - always returns a solution, even if it is partial

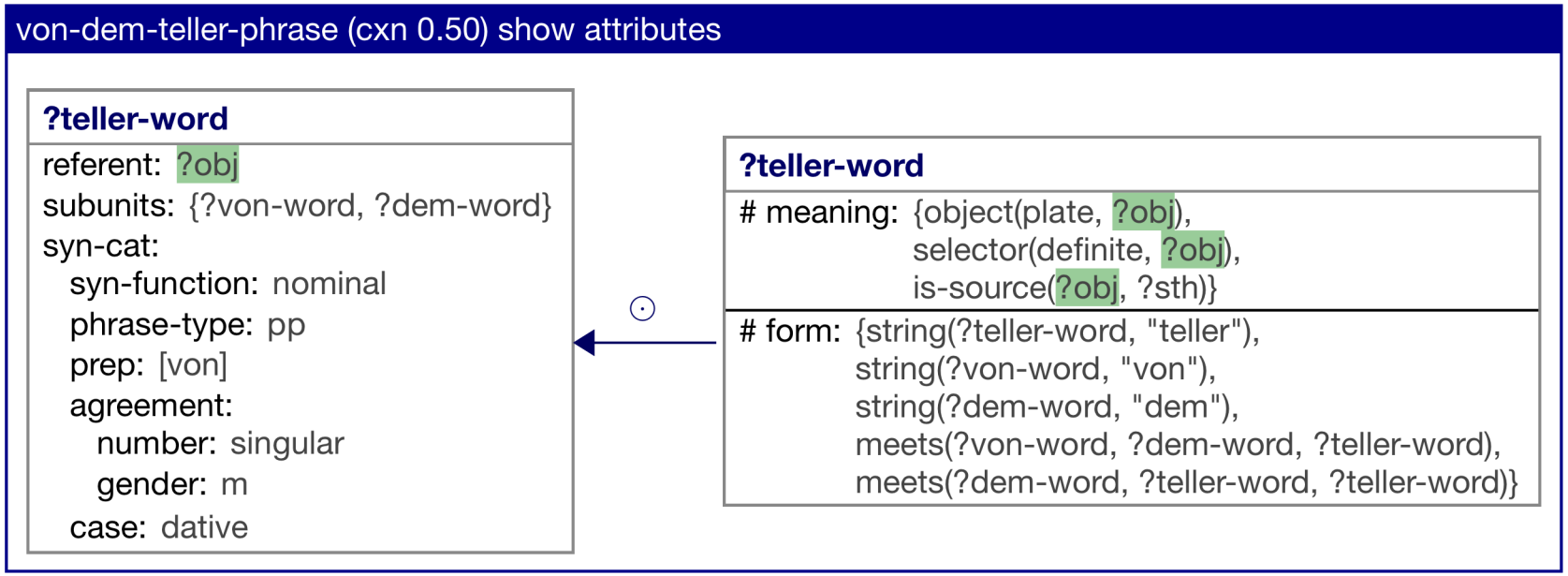
An English lexical cxn

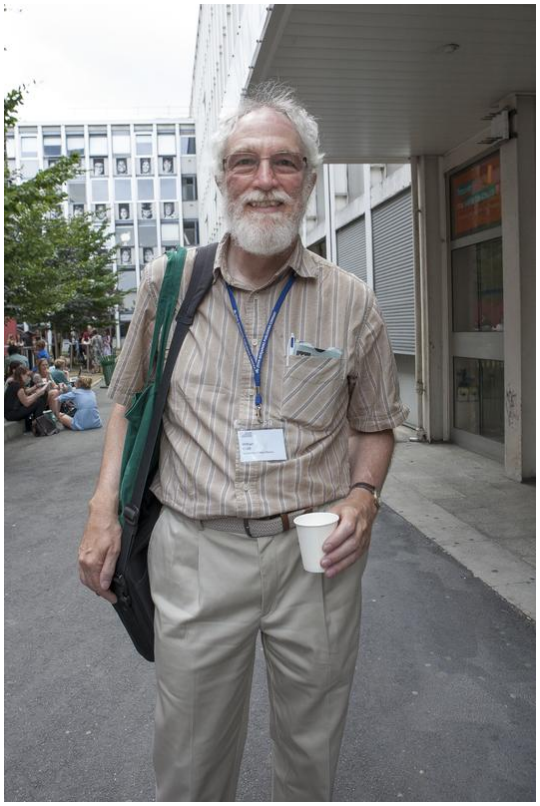


A Dutch cat cxn



A German phrasal cxn

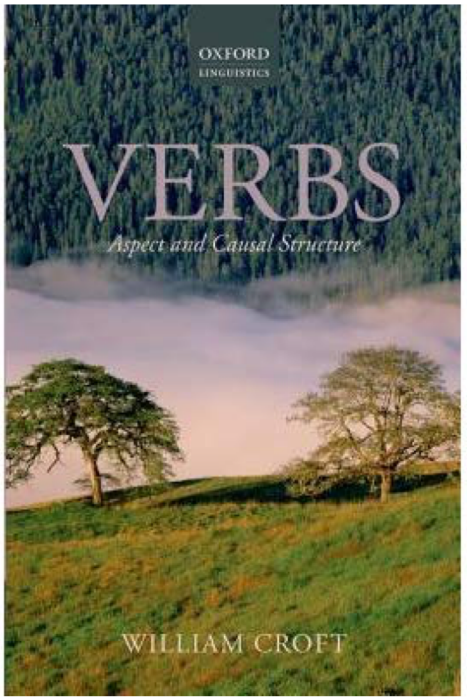




TWO-DIMENSIONAL ANALYSIS OF ASPECTUAL TYPES (CROFT)

Aspectual types in *Verbs* (Croft 2012)

- Typological analysis, but worked out for English.
- Chapters 2-4: aspectual structure
- Chapters 5-9: causal structure



Aspect can be studied on at least two different levels

1. Lexical Aspect (LA): Aktionsart of verbs
2. Grammatical Aspect (GA): Inflectional or periphrastic distinctions

The Vendler classes

- Croft's aspectual classes are based on the Vendler classes [Vendler: 1957]
- These four classes are based on a triple binary distinction:

	stative	durative	telic
States	+	+	-
Activities	-	+	-
Achievements	-	-	+
Accomplishments	-	+	+

The Vendler classes

The Vendler classes were intended for lexical aspectual types. For English, they should be attributed to predicates, rather than to verbs.

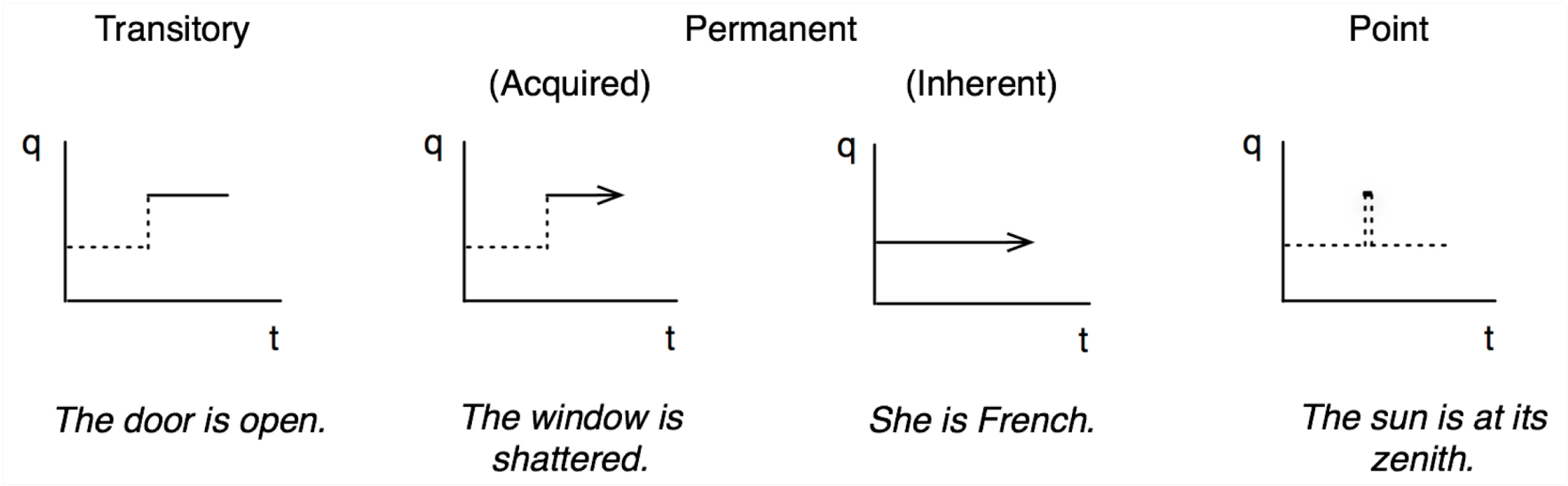
States	be Polish, be polite, love
Activities	sing, dance, read
Achievements	reach the summit
Accomplishments	read the book

How do events evolve over time?

- Two-dimensional geometric representation
  - X-axis: Time dimension (continuous)
  - Y-axis: Qualitative State dimension (discrete or continuous)
- Essentially a subcategorization of the Vendler classes:
  - Four kinds of states
  - Three kinds of achievements
  - Two kinds of activities
  - Two kinds of accomplishments

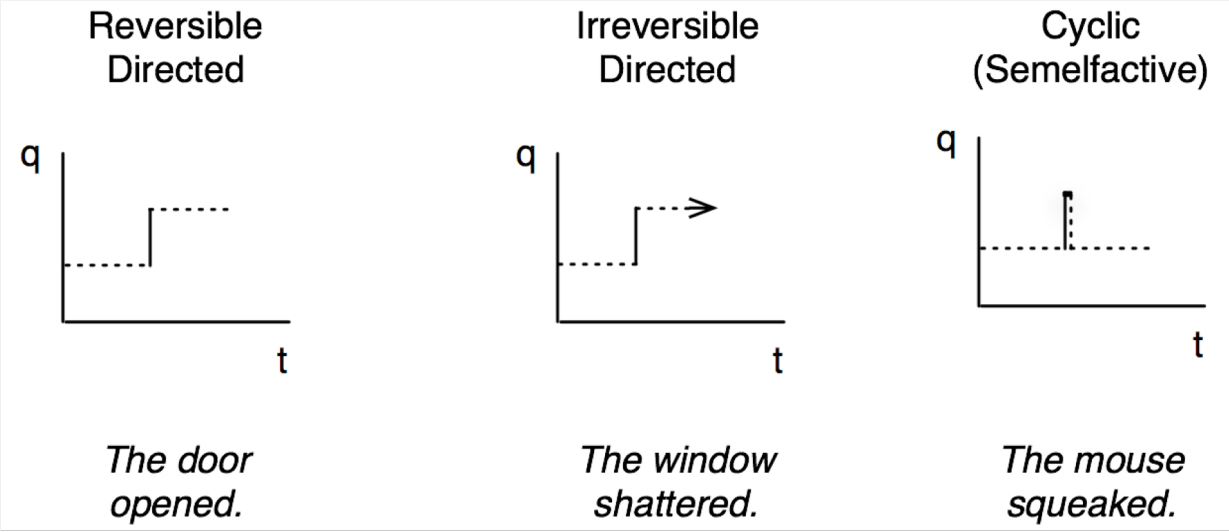
Four kinds of states

For states, the profiled phase is a single point in the q dimension.



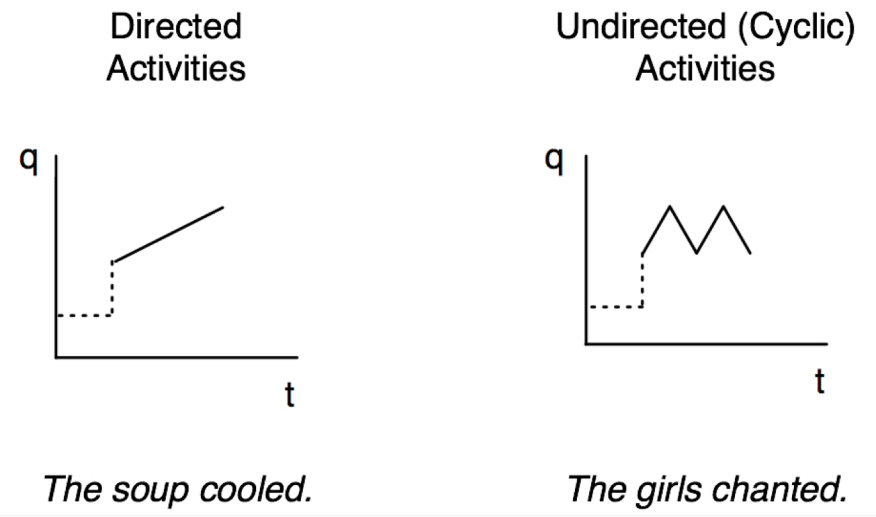
Three kinds of achievements

For achievements, the profiled phase is a single point in the t dimension and a transition in the q dimension.



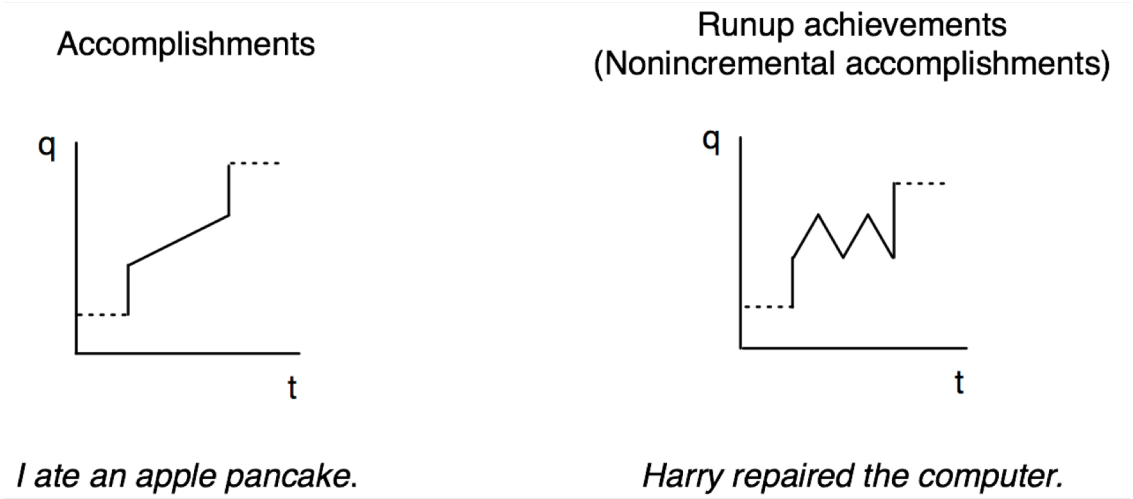
Two kinds of activities

For activities, the profiled phase continuous in both the q and t dimensions. However, there is no transition into a resulting state, representing a completed action.



Two kinds of accomplishments

For accomplishments, the profiled phase continuous in both the q and t dimensions. At the end, there is a transition into a resulting state, representing a completed action.



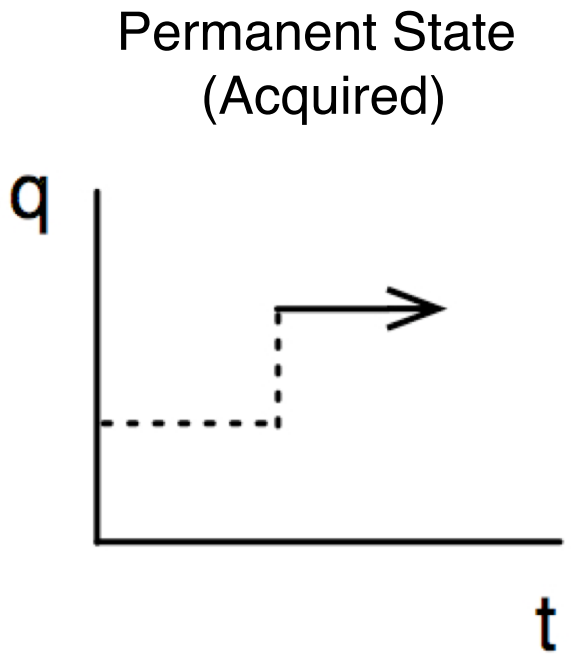
The aspect of an utterance

The aspectual contour of an utterance depends on multiple factors, such as:

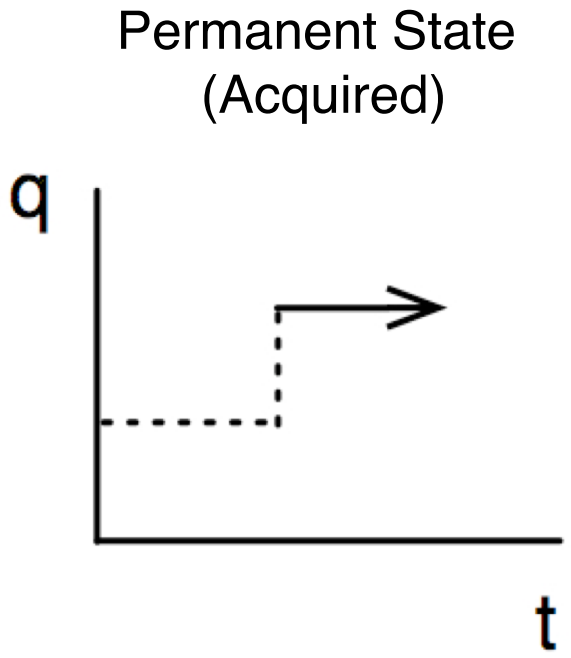
- Semantic properties of the verb
- Verbal Tense-Aspect constructions (e.g. perfect, present, progressive)
- Presence of objects and their type (e.g. non-generic NPs)
- Presence of adverbial groups and their type (e.g. durative adverbs)

The man dances

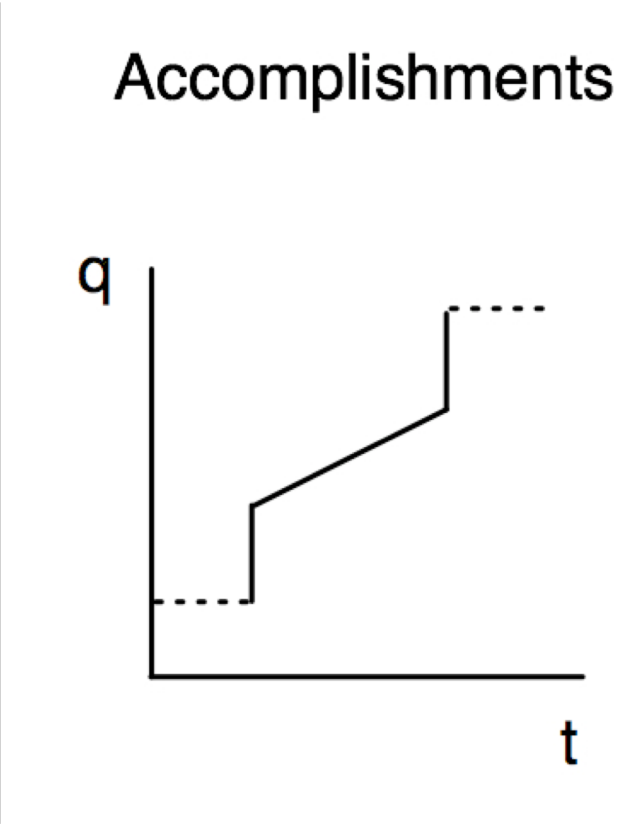
dance = undirected activity



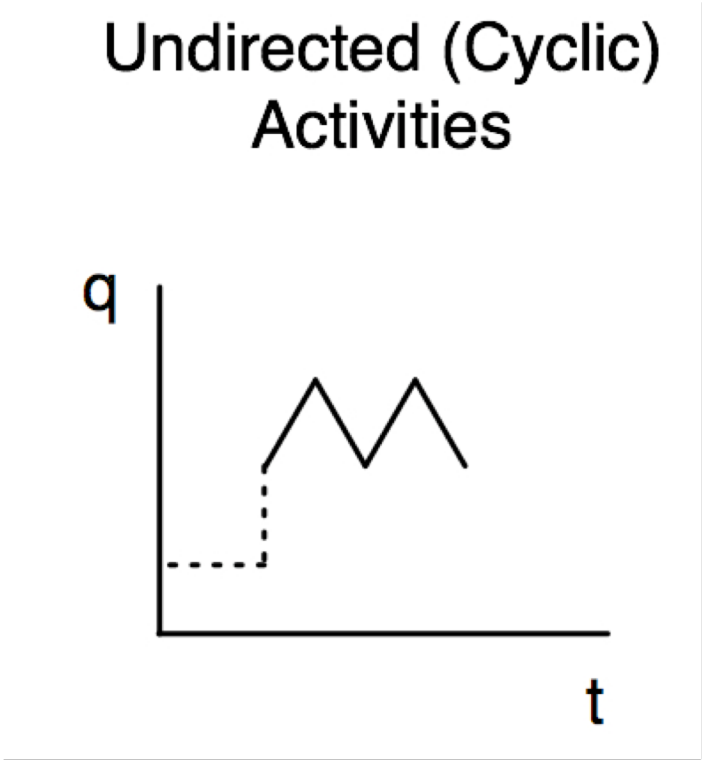
The man dances the sirtaki



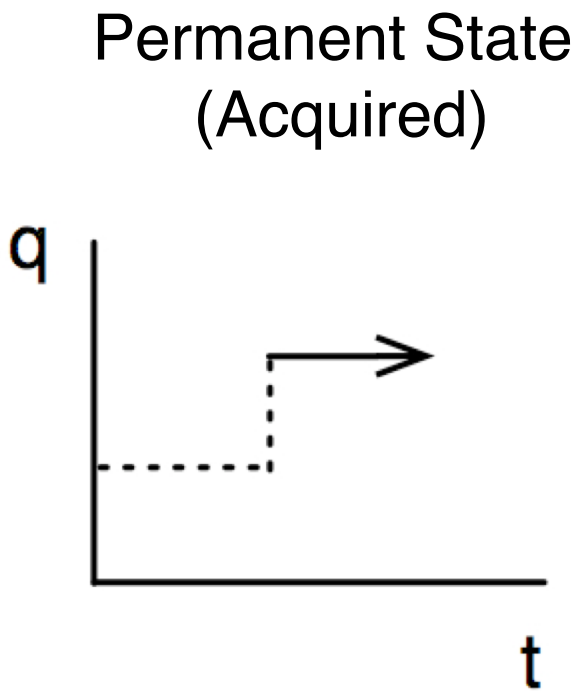
The man danced the sirtaki



The man danced the sirtaki for three hours



The man danced the sirtaki every day



LIVE CODING SESSION

Possible solution

- http://fcg-net.org/demos/iccg-10.zip
- Download and create a folder in your Babel2 folder “course-materials” where you unzip it

Stay in touch

- Follow us on Facebook/Twitter

- Subscribe to our mailing list:

https://ai.vub.ac.be/mailman/listinfo/fcg-mailing