Natural Language Processing

Class 06:
Introduction to Fluid Construction Grammar

VRIJE UNIVERSITEIT BRUSSEL

30 March 2018

Katrien Beuls/ Luc Steels

Overview

What is a computational grammar and why do we need it?

Language processing as problem solving

Construction grammar

Schema’s in Fluid Construction Grammar

Processing

Examples

Why do we need it?

Develop a scientific description of a language requires formal objective notation and definition of processing steps

Languages are highly complex objects that require a team to develop

Computational applications: information retrieval, translation, question-answering, human-robot interaction, etc.

Models of human language processing – but not realistic from the viewpoint of neuroscience

Language as problem solving

Production requires a very large number of choices

- What meanings can achieve a certain goal?
- What words can express these meanings?
- What additional grammatical devices have to be added (e.g. word order, intonation structure)
- What kind of syntactic and semantic structures are to be conveyed?
- How to pronounce chosen words? ...

Language as problem solving

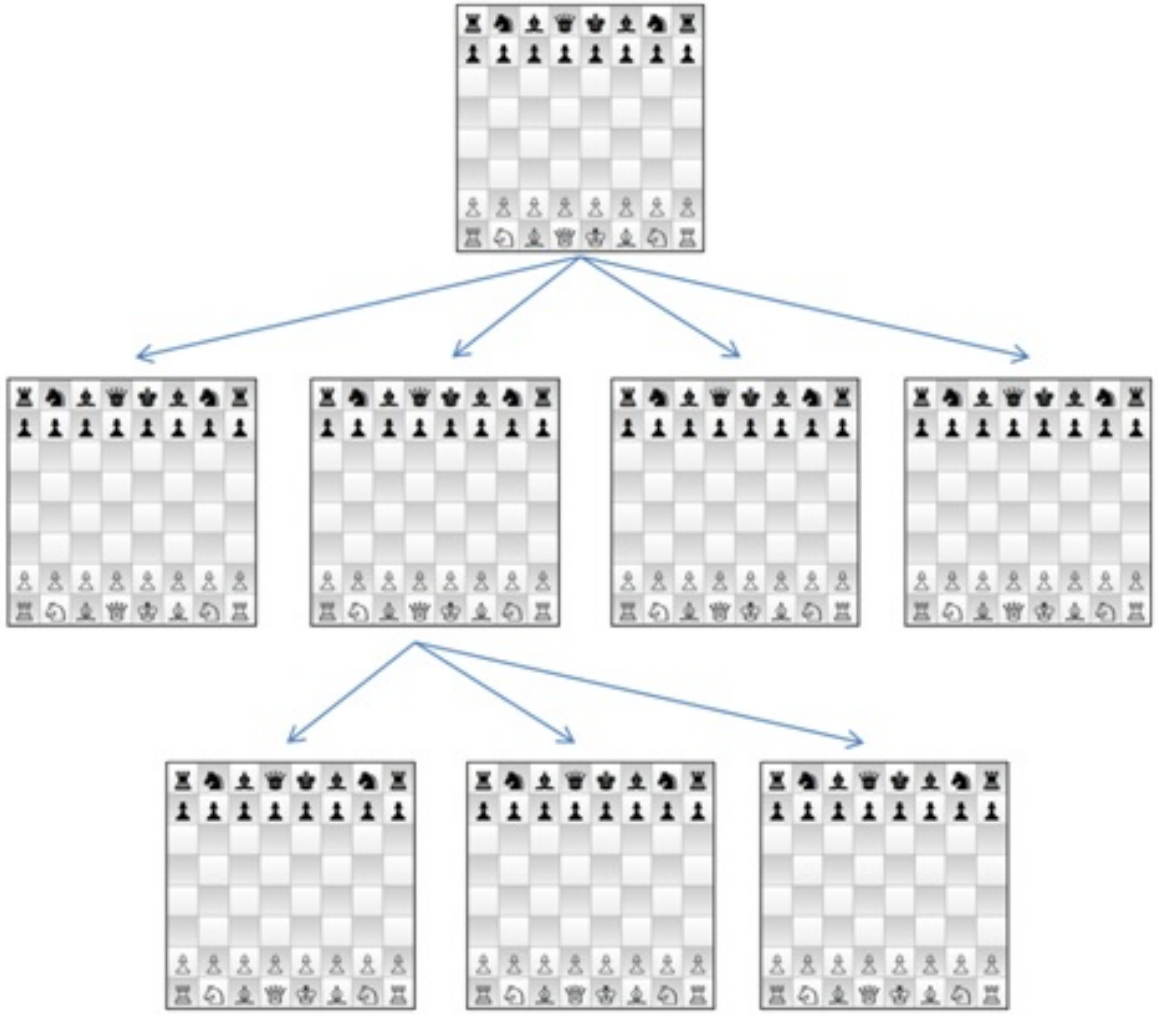
Comprehension requires a similar set of choices going in the other direction

Often there are are alternatives and it is not clear which one should be chosen looking only at the local context:

- The sheep owned by farmer Joe escaped from the stable and ran to the street until it was caught by farmer Bill.

Problem spaces

Allen Newell + Herb Simon
General Problem Solver, 1959



Search algorithms

Heuristic search

Complexity theory

Learning through Chunking

Components problem spaces

Problem state contains all information about the state of the world relevant for the problem domain

Operators transform one state into another

Initial state what is known when problem solving starts

Final state defined by a goal function.

Pathway sequence of states

Selection function decides which state to expand next

Components of a linguistic problem space

Problem State = Transient structure

- Contains all that is known about an utterance being parsed or produced

Operators: Steps to expand transient structure

Initial state:

- Parsing: What is observable from an utterance.
- Production: The meaning to be expressed

Final state:

- Parsing: The meaning of an utterance
- Production: An utterance expressing the meaning

Pathway: Sequence of steps from initial to final state

Selection function: Decides which operator to apply first, typically heuristic depth-first

Transient structure

Represented using units with features and values

Units correspond to lexical units (words, morphemes) and phrasal units

Features concern:

- Form: what the utterance looks like; described explicitly
- Meaning: predicate calculus, frame-based, grounded procedural semantics
- Syntactic categorizations and structures
- Semantic categorizations and structures

Everything is described explicitly

Linguistic structures

Parts of speech (noun, verb, etc.)

Ordering information

Phonetic structures (accent, intonation, phonemes)

Phrase structure (noun phrase, verb phrase)

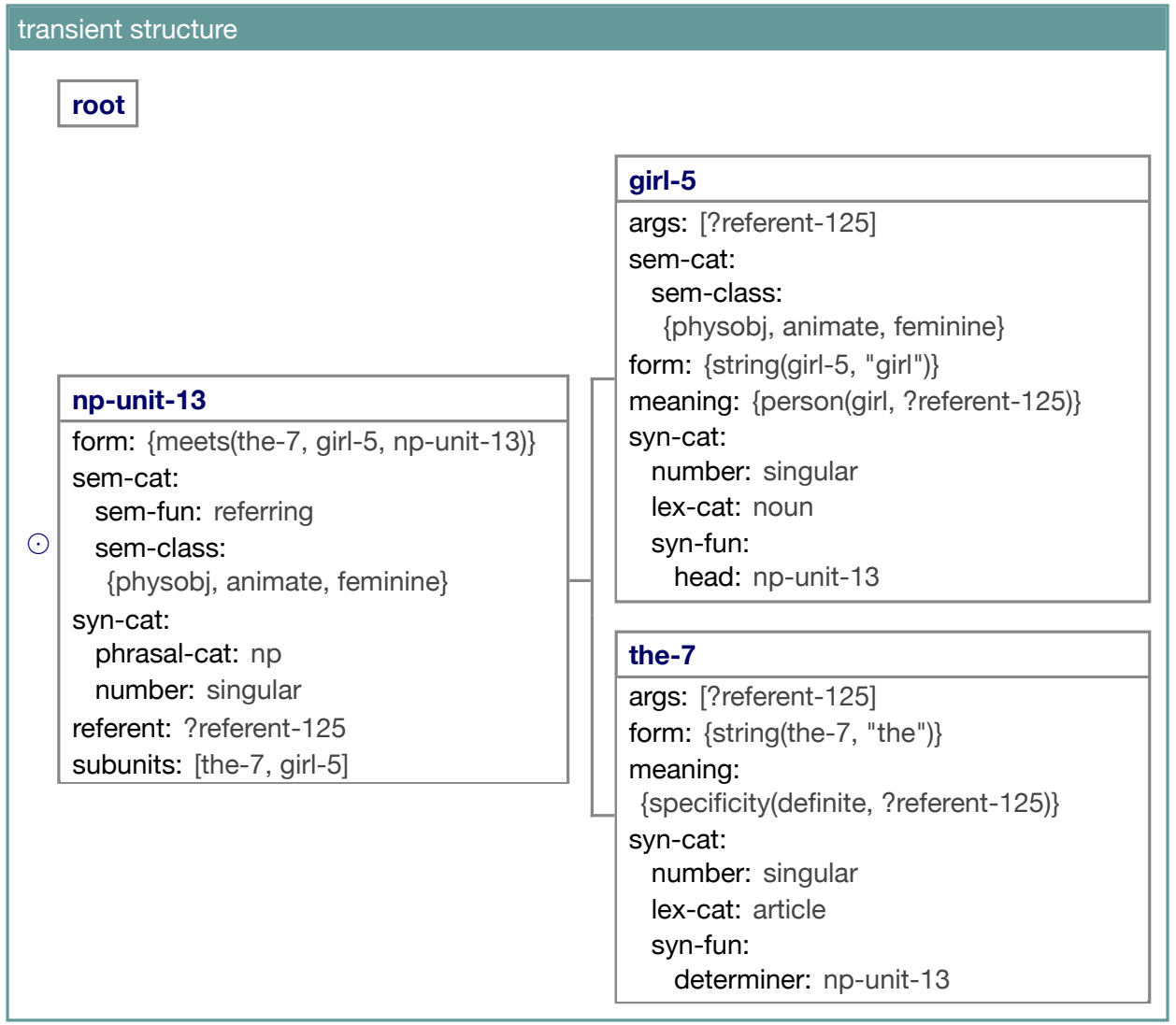
Functional structure (dependencies): subject, predicate, object

Case structure: nominative, accusative, dative, …

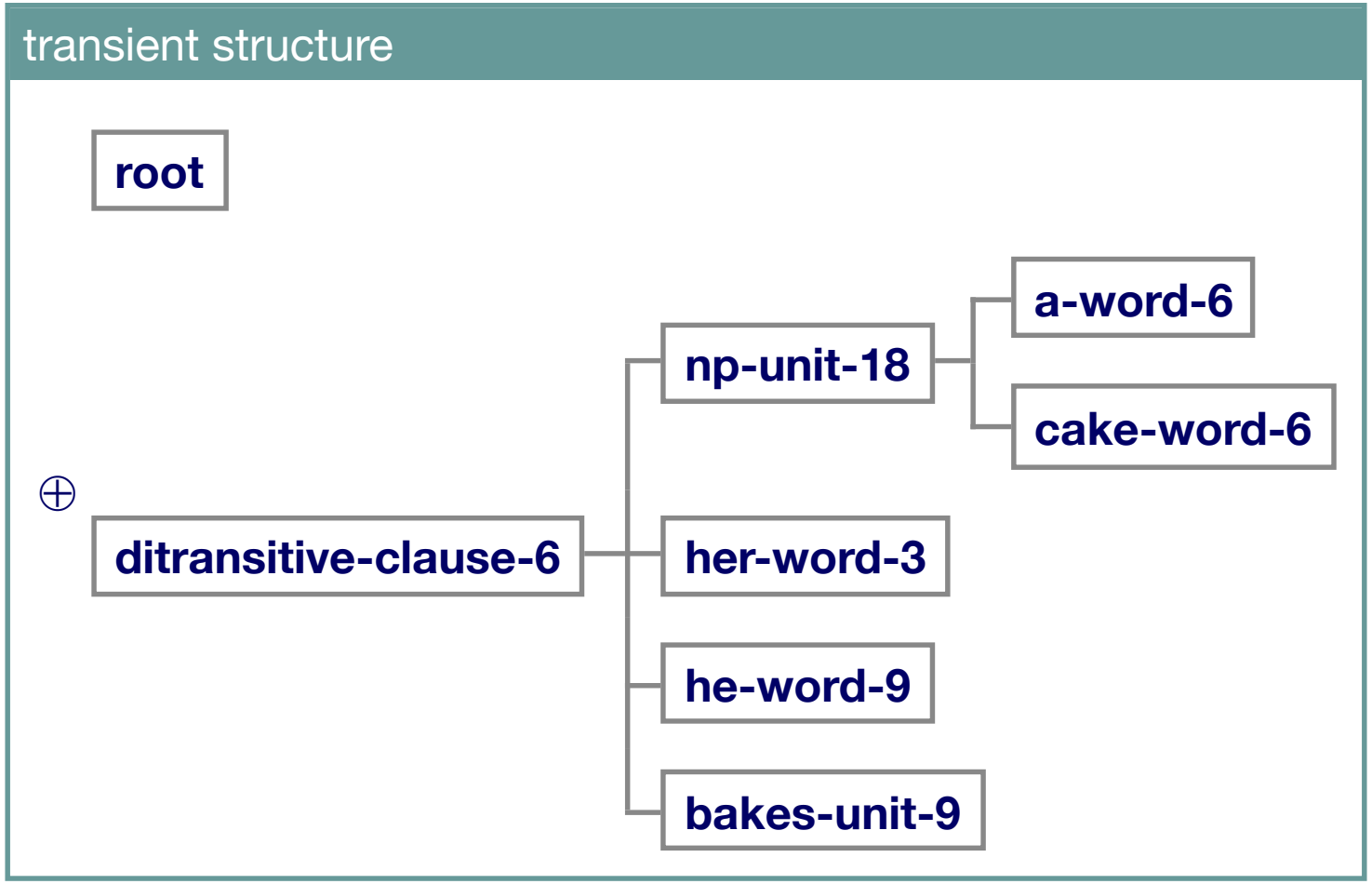
Information structure: topic, comment

Transient structures

All the information that is known about at a certain point in processing



Transient structures



Construction schemas

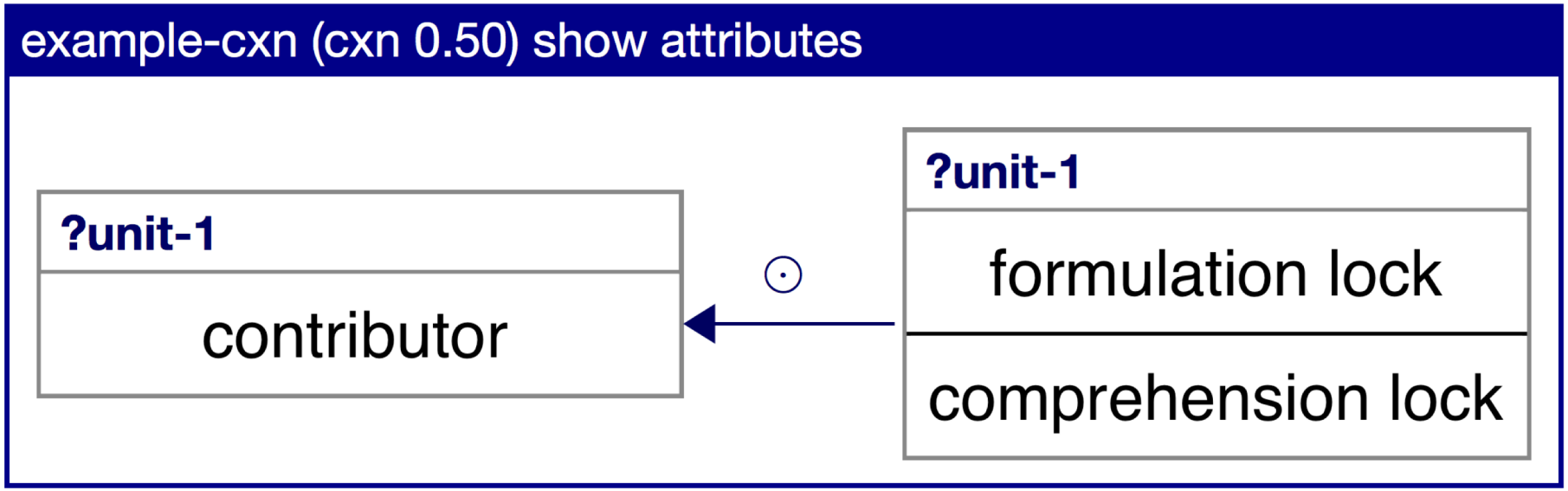
A coupling of any form with whatever meaning it has

Abstract schema that can be used to expand any aspect of a transient structure

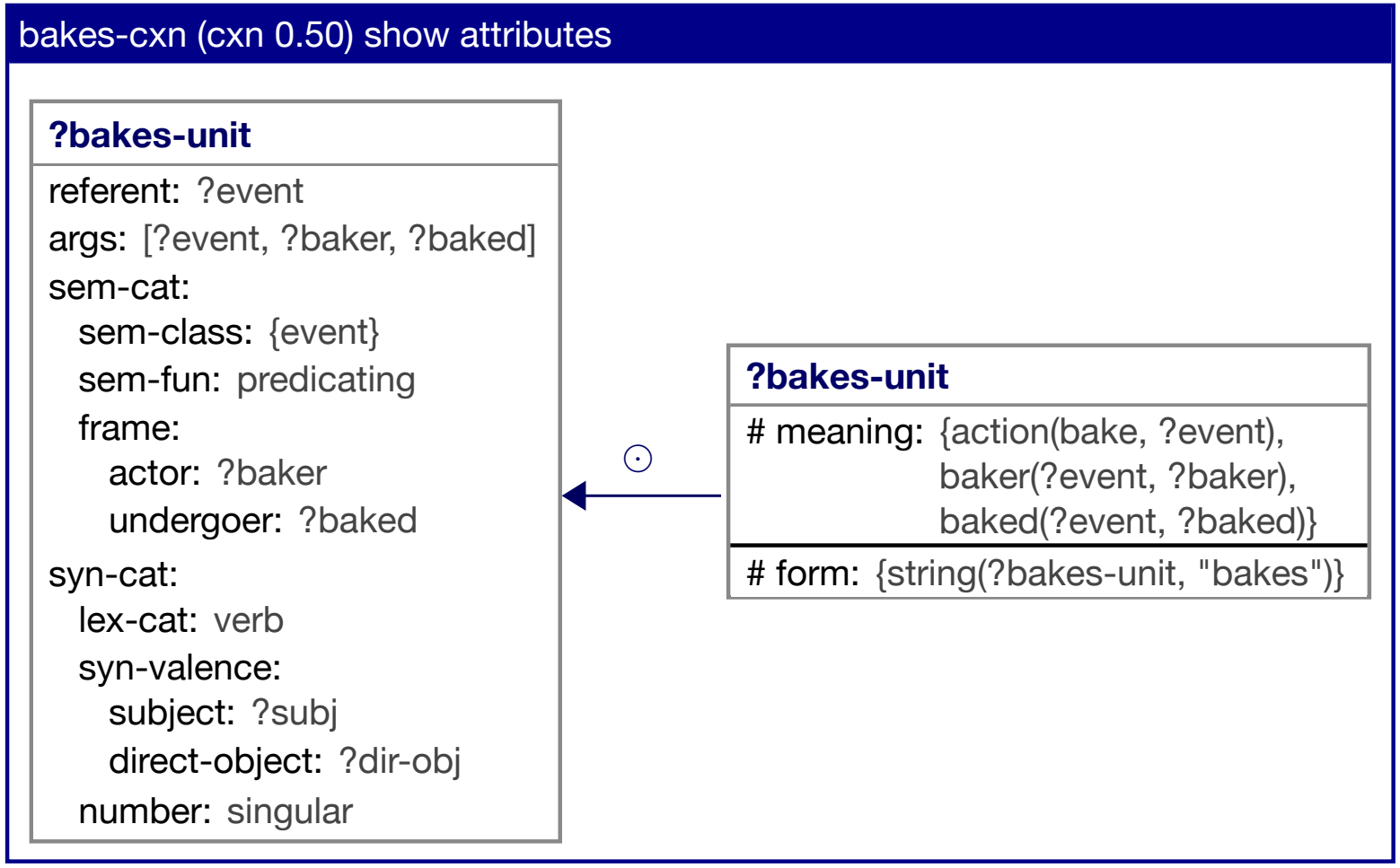
A construction can consult any aspect of the transient structure to decide how to do so

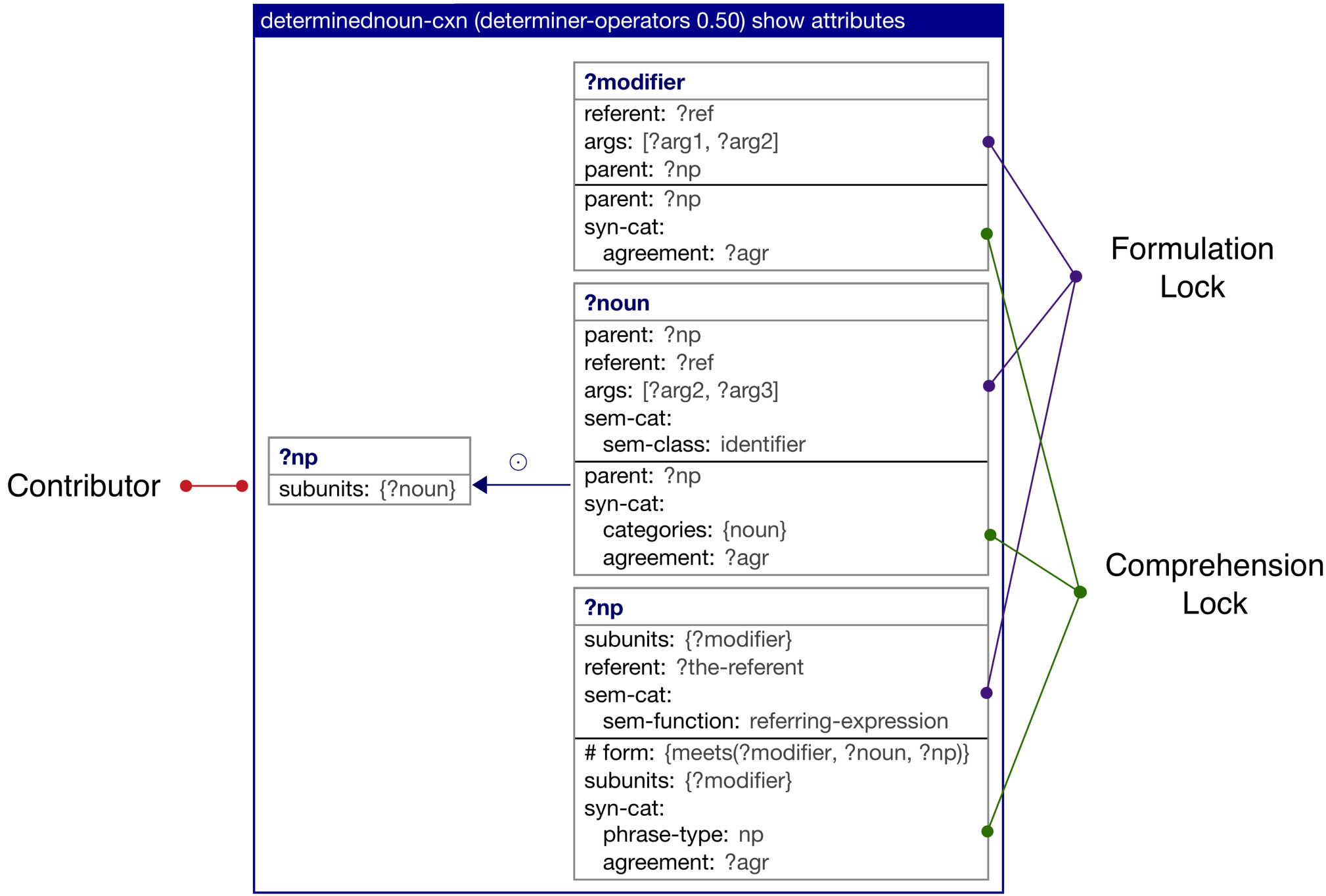
Not merely descriptive, but computational entity usable for processing

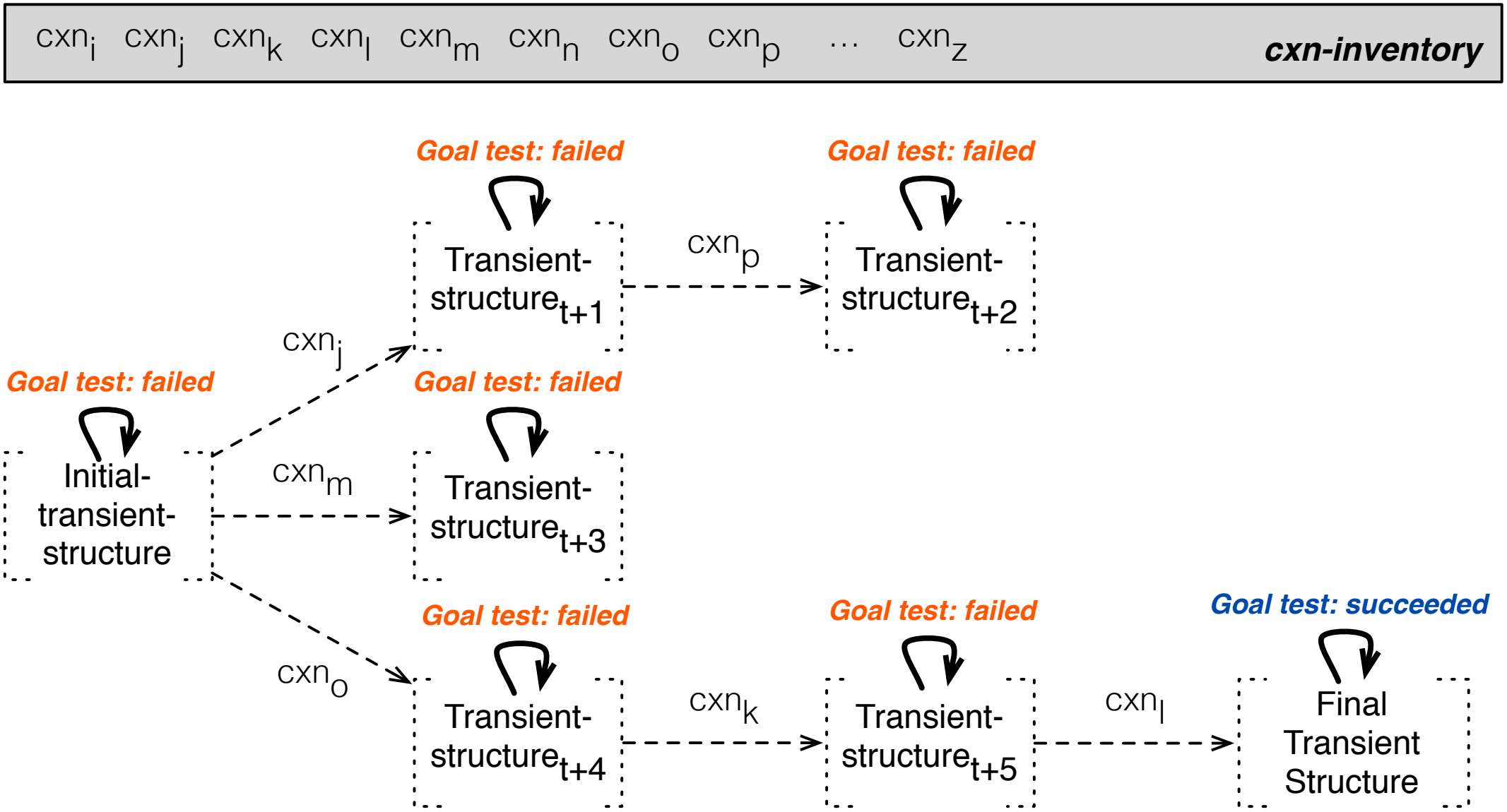
Construction schemas



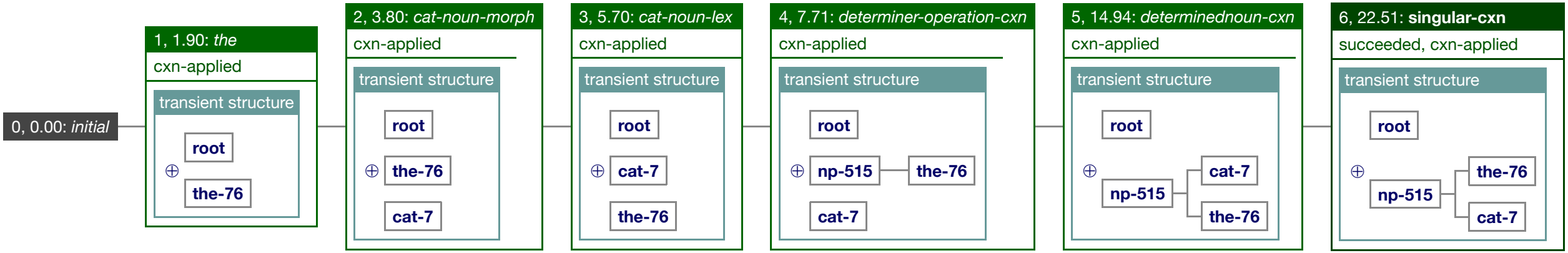
Construction schemas



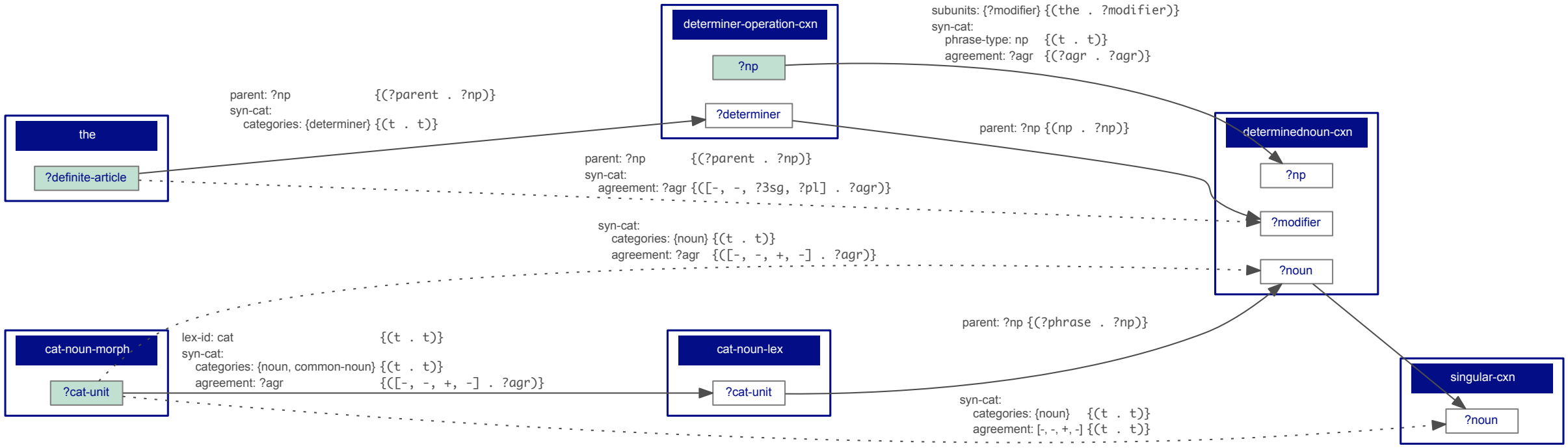

determinednoun-cxn (determiner-operators 0.50) show attributes
?modifier
referent: ?ref
args: [?arg1, ?arg2]
parent: ?np
parent: ?np
syn-cat:
agreement: ?agr
?noun
parent: ?np
referent: ?ref
args: [?arg2, ?arg3]
sem-cat:
sem-class: identifier
parent: ?np
syn-cat:
categories: {noun}
agreement: ?agr
?np
subunits: {?modifier}
referent: ?the-referent
sem-cat:
sem-function: referring-expression
form: {meets(?modifier, ?noun, ?np)}
subunits: {?modifier}
syn-cat:
phrase-type: np
agreement: ?agr
?np
subunits: {?noun}
Contributor
Formulation Lock
Comprehension Lock


cxn_i cxn_j cxn_k cxn_l cxn_m cxn_n cxn_o cxn_p … cxn_z
cxn-inventory
Goal test: failed
Goal test: failed
Transient-structure_t+1
cxn_p
Transient-structure_t+2
cxn_j
Goal test: failed
Goal test: failed
Initial-transient-structure
cxn_m
Transient-structure_t+3
Goal test: failed
Goal test: failed
Goal test: succeeded
cxn_o
Transient-structure_t+4
cxn_k
Transient-structure_t+5
cxn_l
Final Transient Structure

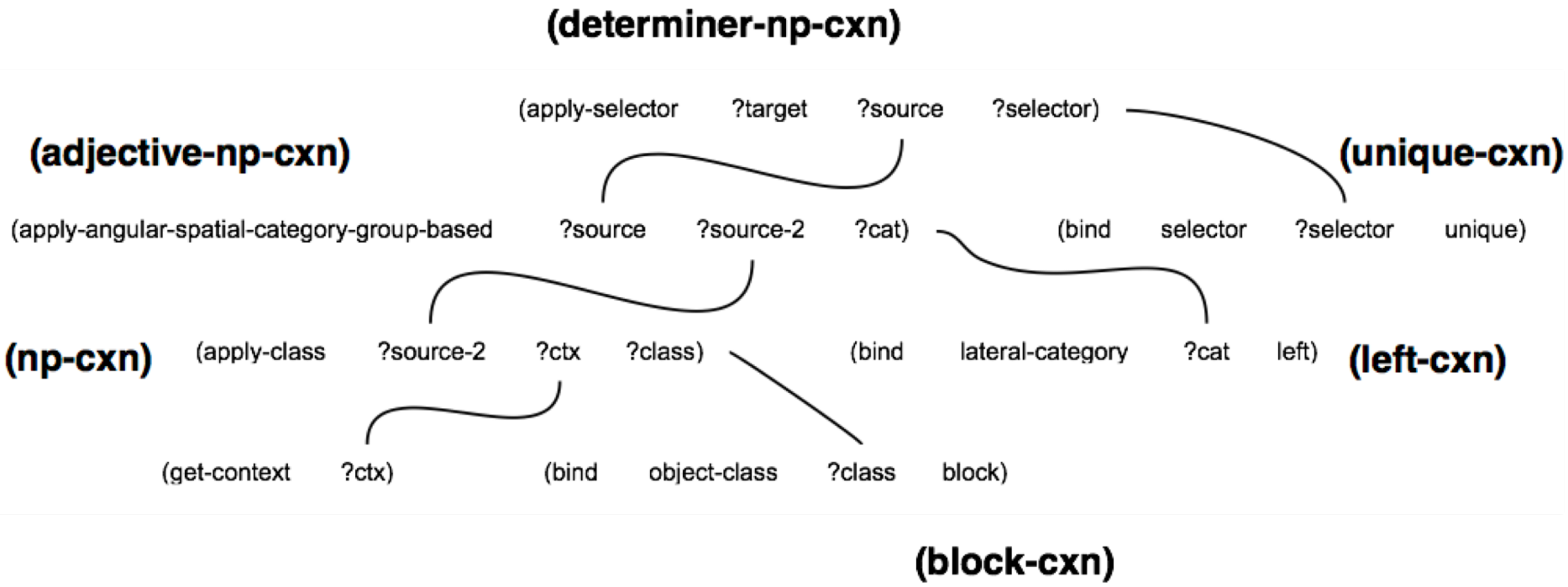
Search process



Constructional dependencies

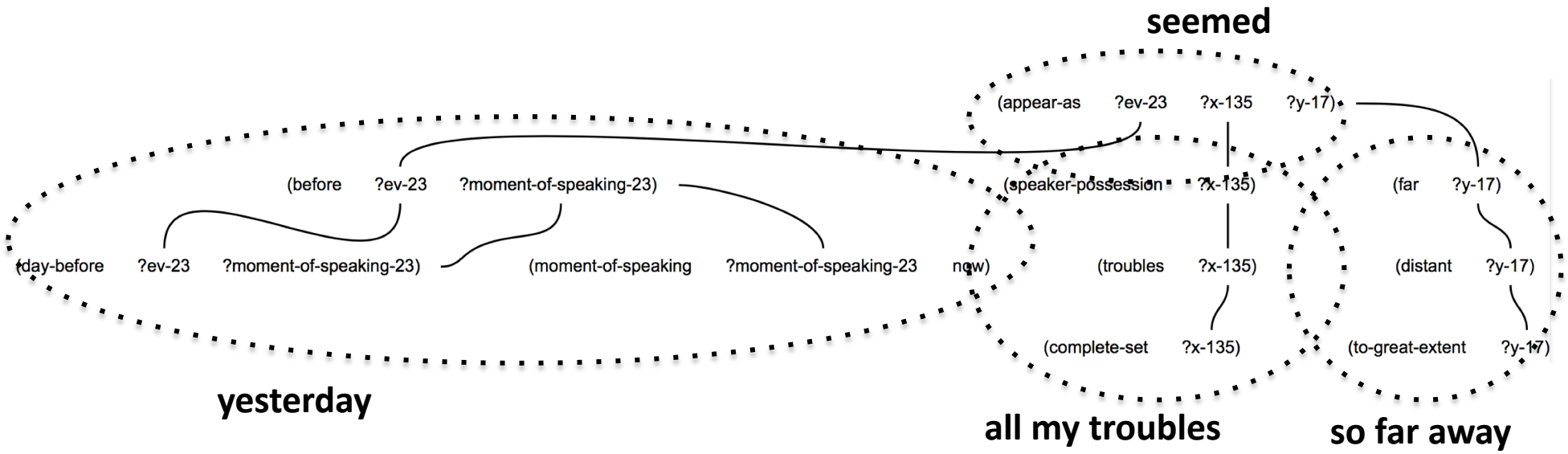


Semantics



(Figure adapted from Spranger & Steels 2012)

Semantics



Example by Paul Van Eecke

FCG interactive

https://www.fcg-net.org/fcg-interactive/

Natural language understanding

Pragmatic Inference Engine (PIE)

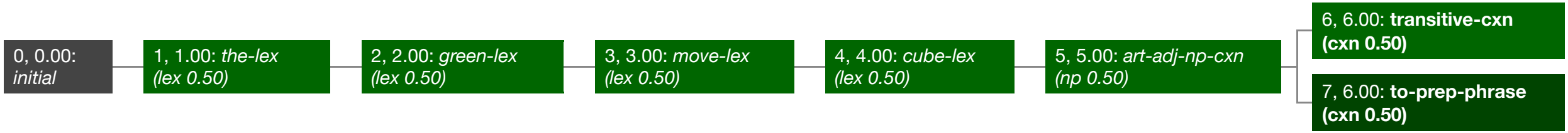
VRIJE
UNIVERSITEIT
BRUSSEL

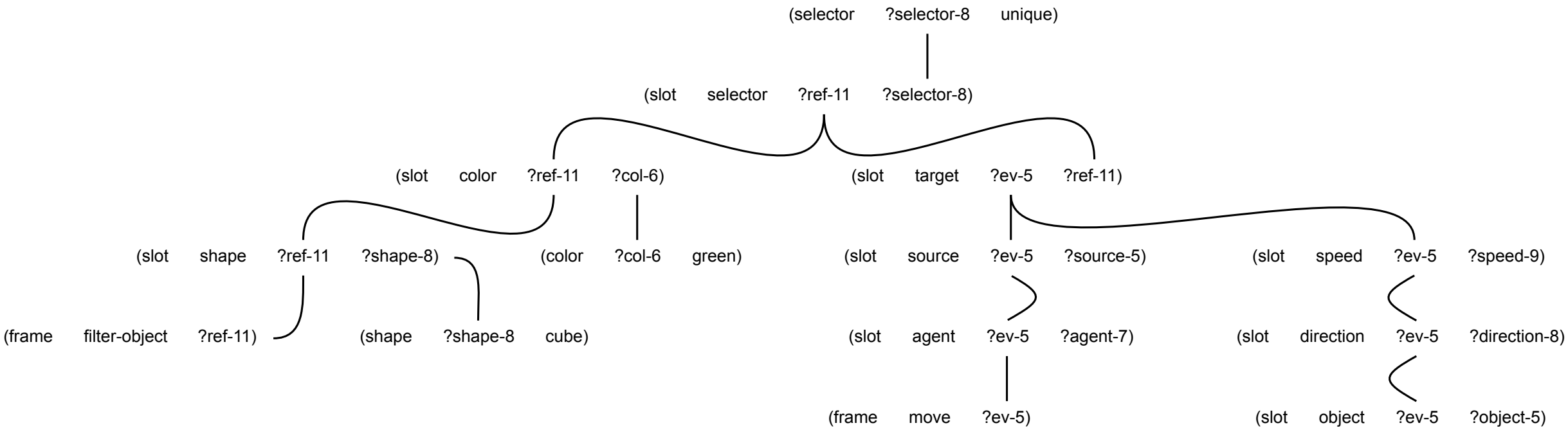
Pragmatic inference engine

Three step process:

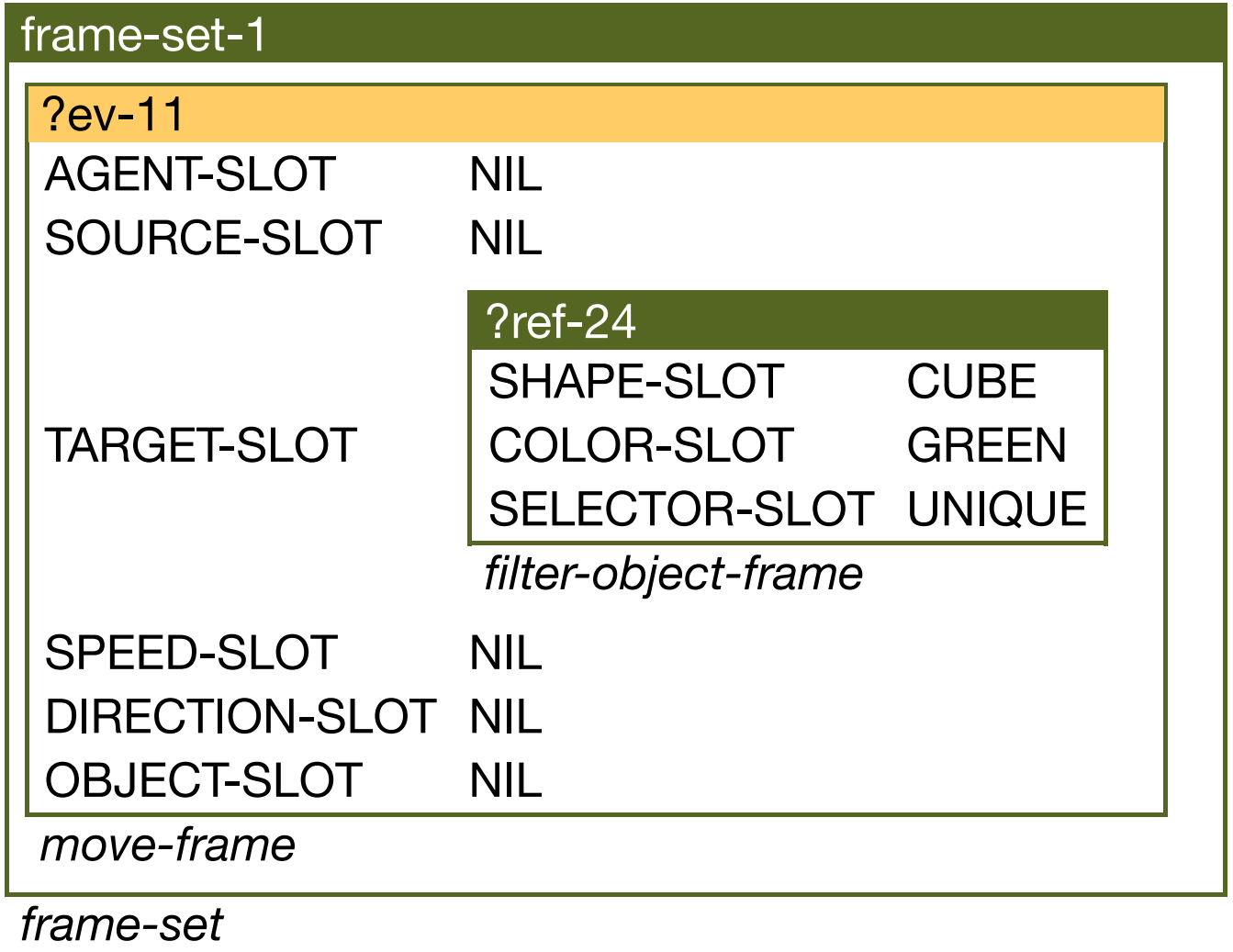
1. Grammatical analysis
2. World model
3. Pragmatic inferences

“move to the green cube”

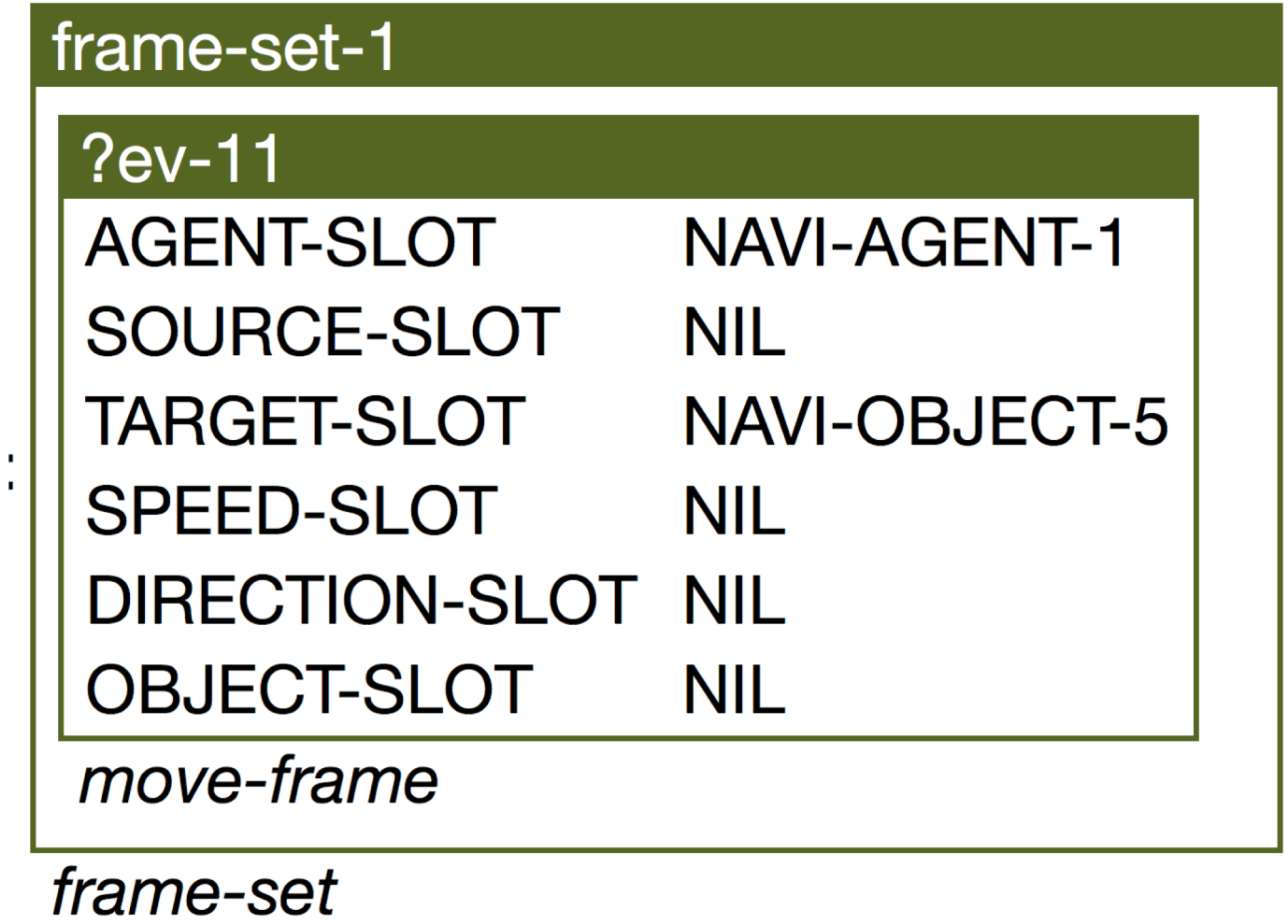




Frames after grammatical processing



Frames after world model check



Frames after pragmatic inferencing

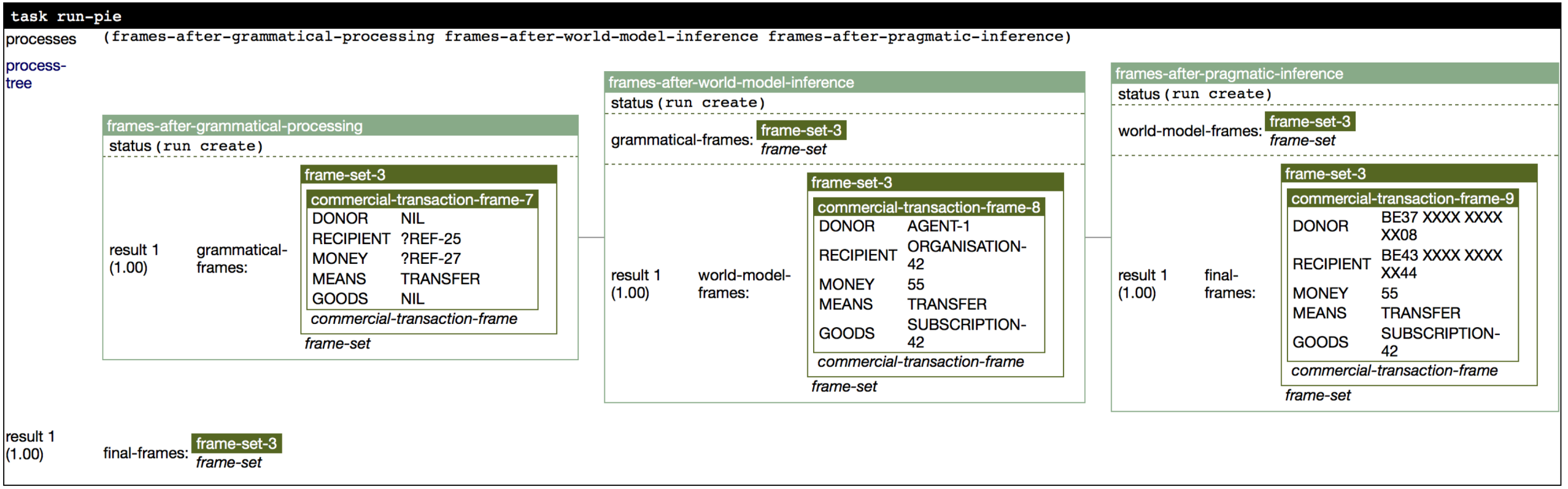
frame-set-1

?ev-11	
AGENT-SLOT	NAVI-AGENT-1
SOURCE-SLOT	(4 . 8)
TARGET-SLOT	NAVI-OBJECT-5
SPEED-SLOT	NORMAL
DIRECTION-SLOT	FORWARD
OBJECT-SLOT	NIL

move-frame

frame-set

“transfer my monthly payment to the health club”



Summary: Fluid Construction Grammar

Fully operational computational grammar formalism

Mechanisms for representing and processing grammars

Bidirectional language processing (comprehension + formulation)

As neutral as possible towards linguistic theories

Construction Grammars

Experiments on language evolution, tutoring, machine translation

www.fcg-net.org (Available on Github)

Next class: 20/4

Fluid Construction Grammar:

- Robustness and learning
- Managing complexity