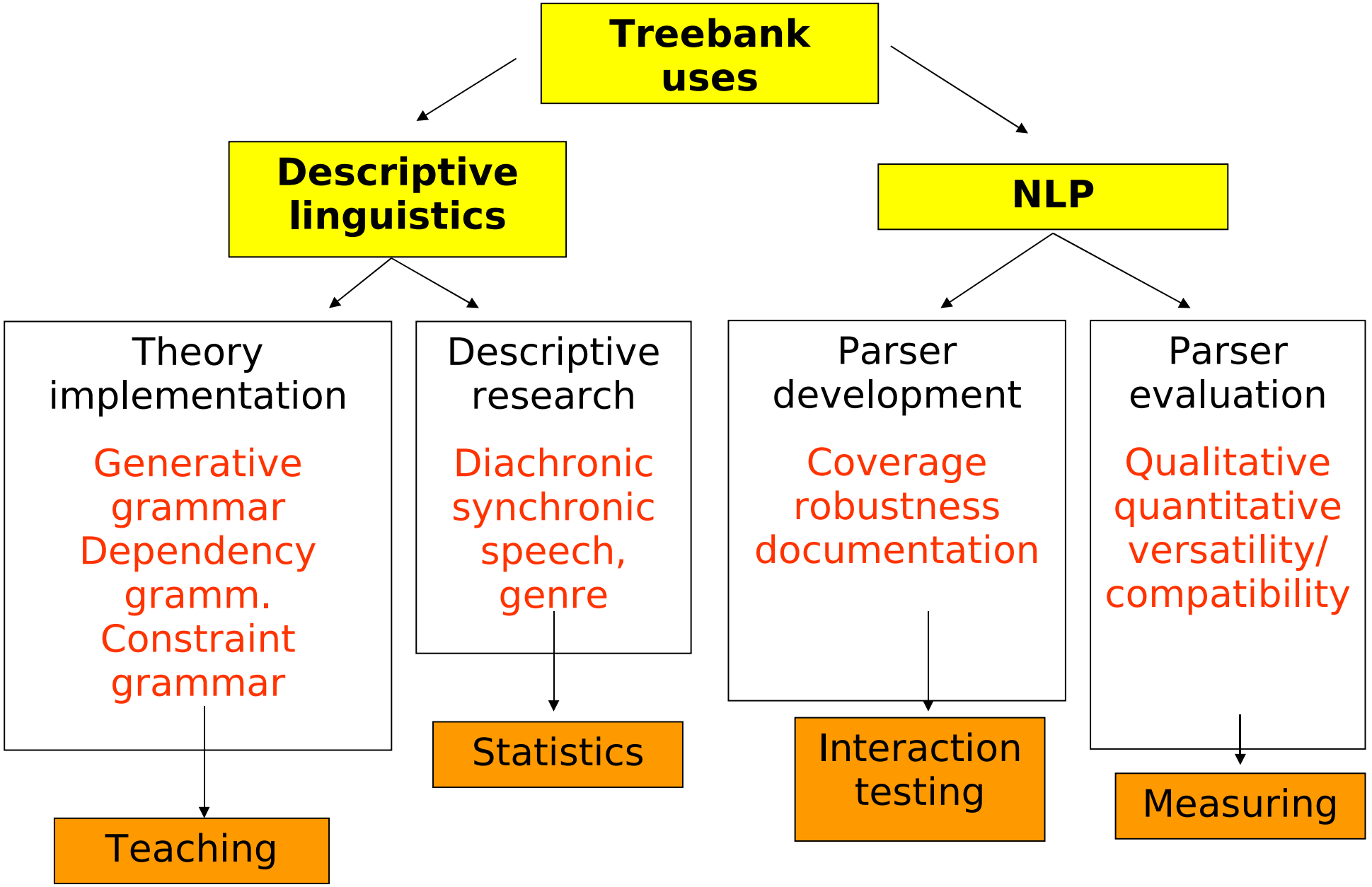
Computational syntax

Eckhard Bick

The task

- produce automatic sentence analyses from morphologically analysed data
  - to allow users to discuss/teach/inspect syntactic structures in live, non-text book examples
  - as a means to quickly annotate corpora
  - to facilitate the manual construtction of treebanks
- while doing so, disambiguate both morphological and syntactic ambiguity in a context-based way
- do all this in a robust way, so as to
  - allow heuristic analysis in the face of new lexical data
  - provide parses even for rare, wrong or unforeseen constructions
  - minimize manual linguistic revision work
- implement a modular, progressive system, that allows for the addition of more semantic "syntax" later (finer distinctions, anaphora, semantic roles, discourse analysis ...)

The user perspective



The methods: how to get there

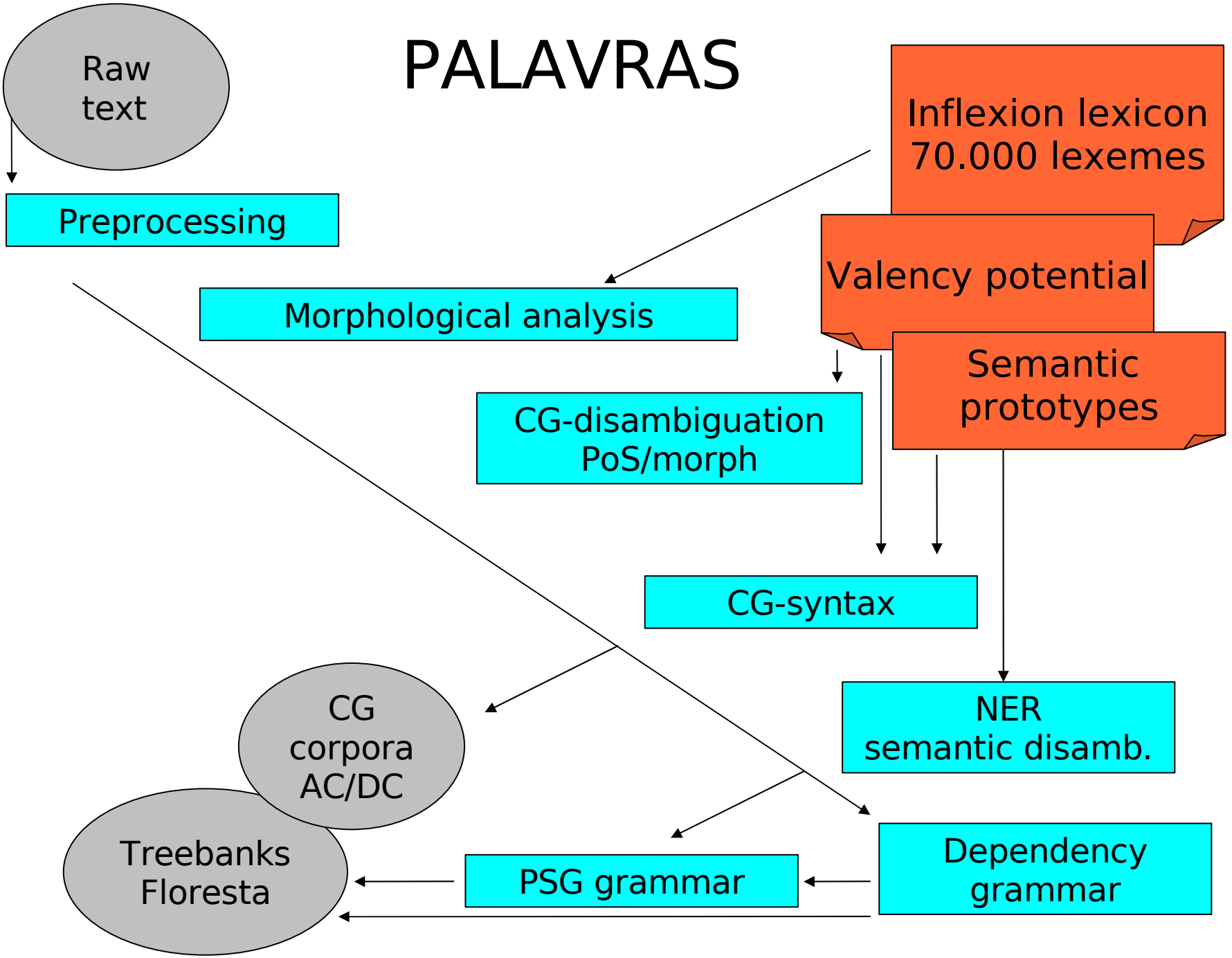
- parsing vs. tagging
  - HMM part of speech tagging is not enough
  - a generative Chomskyan grammar is overkill
- practicable methods
  - rule-based CFGs with heuristic and partial fixes
  - probabilistic CFGs
  - probabilistic dependency grammars
  - Constraint Grammar (CG) syntactic function tagging
  - rule-based dependency tagging

Interdependence between method and descriptive issues

- Focus on syntactic form
  - Phrase structure grammar (PSG) -> labelled brackets
  - Dependency grammar (DG) -> labelled arcs
- Focus on syntactic function
  - Constraint grammar (CG) -> dependency pointers
- Focus on semantic function
  - Case roles taggin (Filmore)
  - separating surface syntax and semantic layers: Lexical Functional Grammar (LFG)

Parsers for Portuguese

- **PALAVRAS**: CG-based robust DG & PSG parser http://visl.hum.sdu.dk/itwebsite/port/portgram.html (Bick 2000)
- **Curupira**: Robust syntactic parser, based on ranked and constrained ReGra PSG rewriting rules http://www.nilc.icmc.usp.br/nilc/tools/curupira.html (Martins, Hasegawa & Nunes 2002)
- **GojolParser** http://www.linguateca.pt/Repositorio/GojolParser.txt, DG & PSG, commercial, calls itself the best (error rate < 1%)
- **LX-Suite** - lemmatizer and PoS tagger, parser (LX-Gram planned for syntax - http://lxsuite.di.fc.ul.pt/ (NLX group, University of Lisbon)
- **CoNLL parsers**: probabilistic, machine-learned dependency parsers, trained on the Floresta Sintá(c)tica (10-15 different systems)


PALAVRAS
Raw text
Preprocessing
Inflexion lexicon 70.000 lexemes
Valency potential
Semantic prototypes
Morphological analysis
CG-disambiguation PoS/morph
CG-syntax
NER semantic disamb.
CG corpora AC/DC
Treebanks Floresta
PSG grammar
Dependency grammar

PALAVRAS output format

- native CG (token based tags)
  - lemma, PoS, inflexion, derivation
  - syntactic function label
  - semantic prototype
  - NER classes
  - semantic roles (upcoming)
- tree formats
  - VISL-psg, native or graphical
  - dependency trees
  - export formats: TIGER xml, MALT, CQP database

Syntactic models

1. The flat classical model: word function, no form

O meu hipopótamo não come peixe.

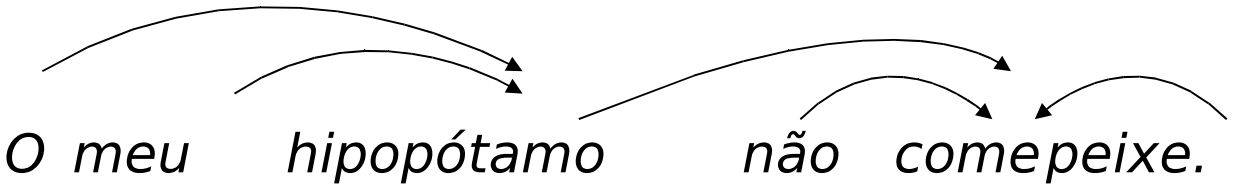
S A V O

- word-based
- psychologically easy to grasp
- function markers attached to semantically heavy words
- easy to turn into tags:

O	article	PRE-N
meu	determiner	PRE-N
hipopótamo	noun	S
não	adverb	A
come	verb	V
peixe	noun	O

2. Dependency grammar

- strictly token based
  - expresses syntactic form as asymmetrical relations (“arcs”) between head tokens and dependent tokens
  - no zero tokens, no nonterminal nodes
- each dependent is allowed 1 head (exc. secondary arcs)
- directed acyclic graphs
- projective or non-projective (crossing branches / discontinuity)

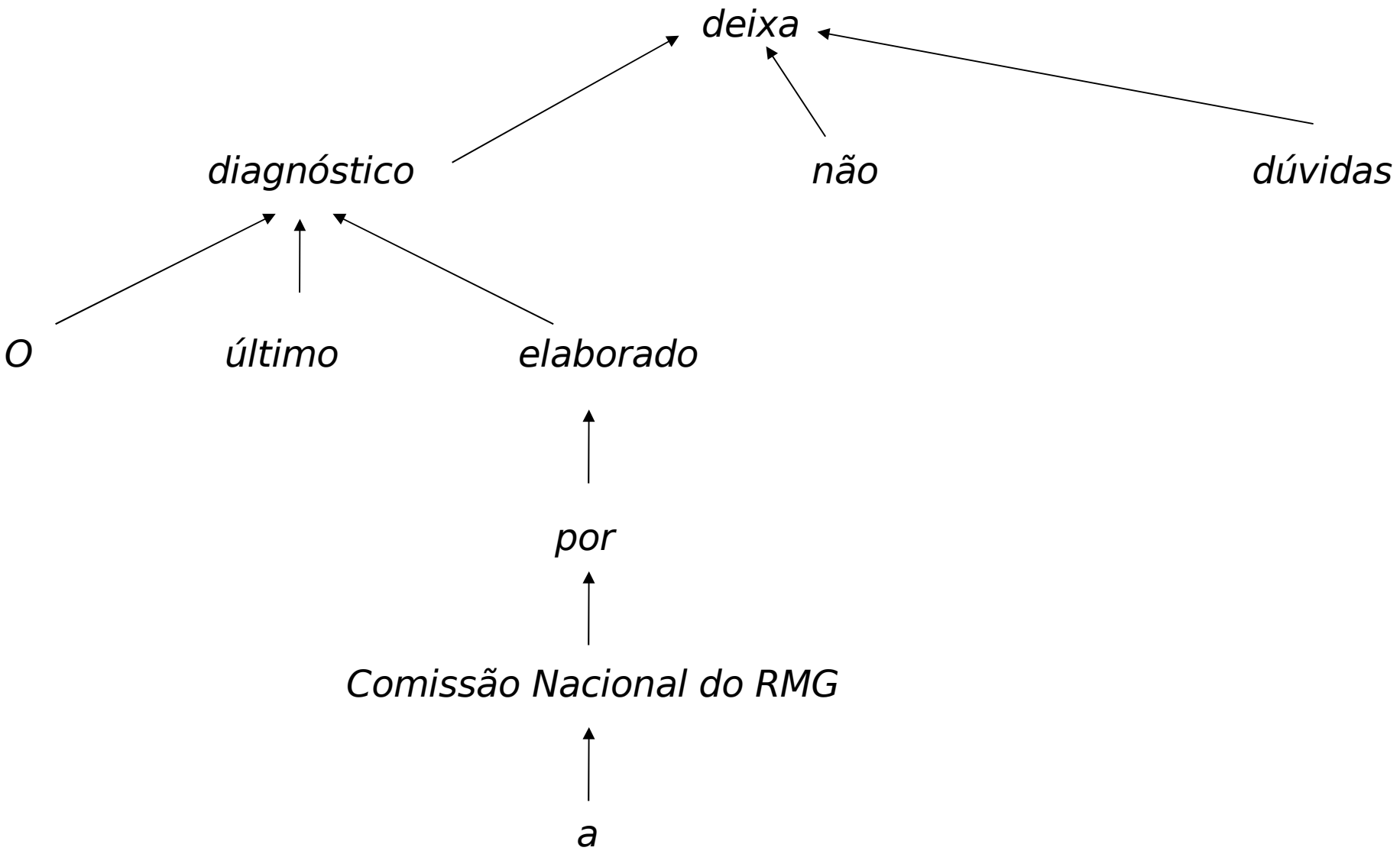


Dependency grammar annotation

O	#1->3
último	#2->3
diagnóstico	#3->9
elaborado	#4->3
por	#5->4
a	#6->7
Comissão=Nacional=do=RMG	#7->5
não	#8->9
deixa	#9->0
ROOT	
dúvidas	#10->9

O	<id=1> <head=3>
último	<id=2> <head=3>
diagnóstico	<id=3> <head=9>
elaborado	<id=4> <head=3>
por	<id=5> <head=4>
a	<id=6> <head=7>
Comissão=Nacional=do=RMG	<id=8> <head=5>
não	<id=8> <head=9>
deixa	<id=9> <head=0> ROOT
dúvidas	<id=10><head=9>

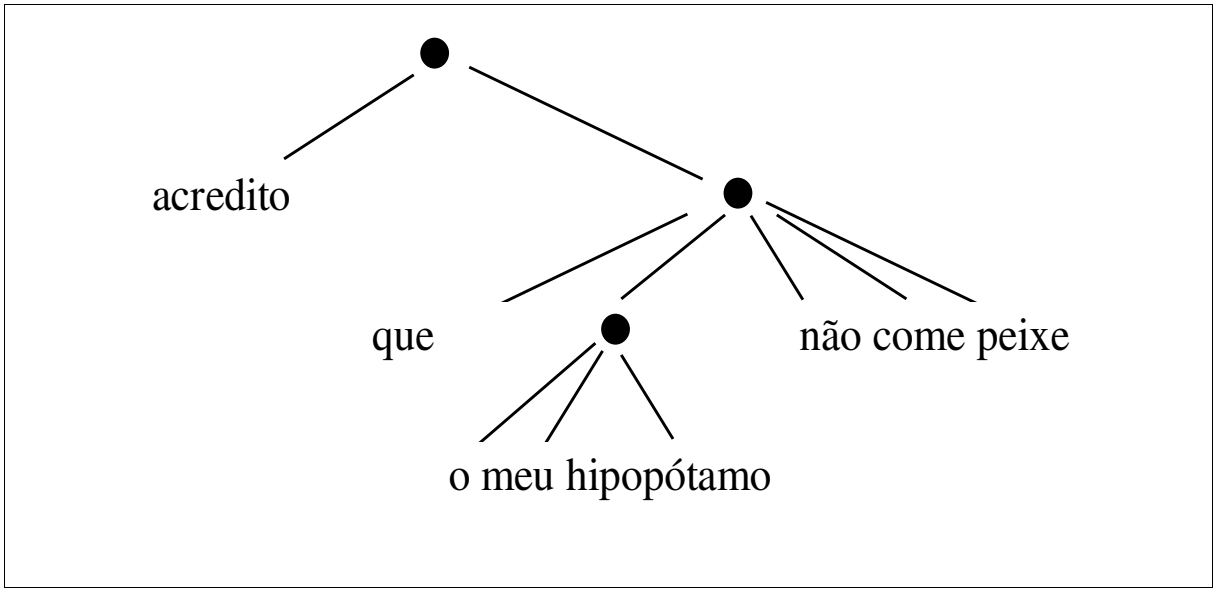
Dependency grammar as trees



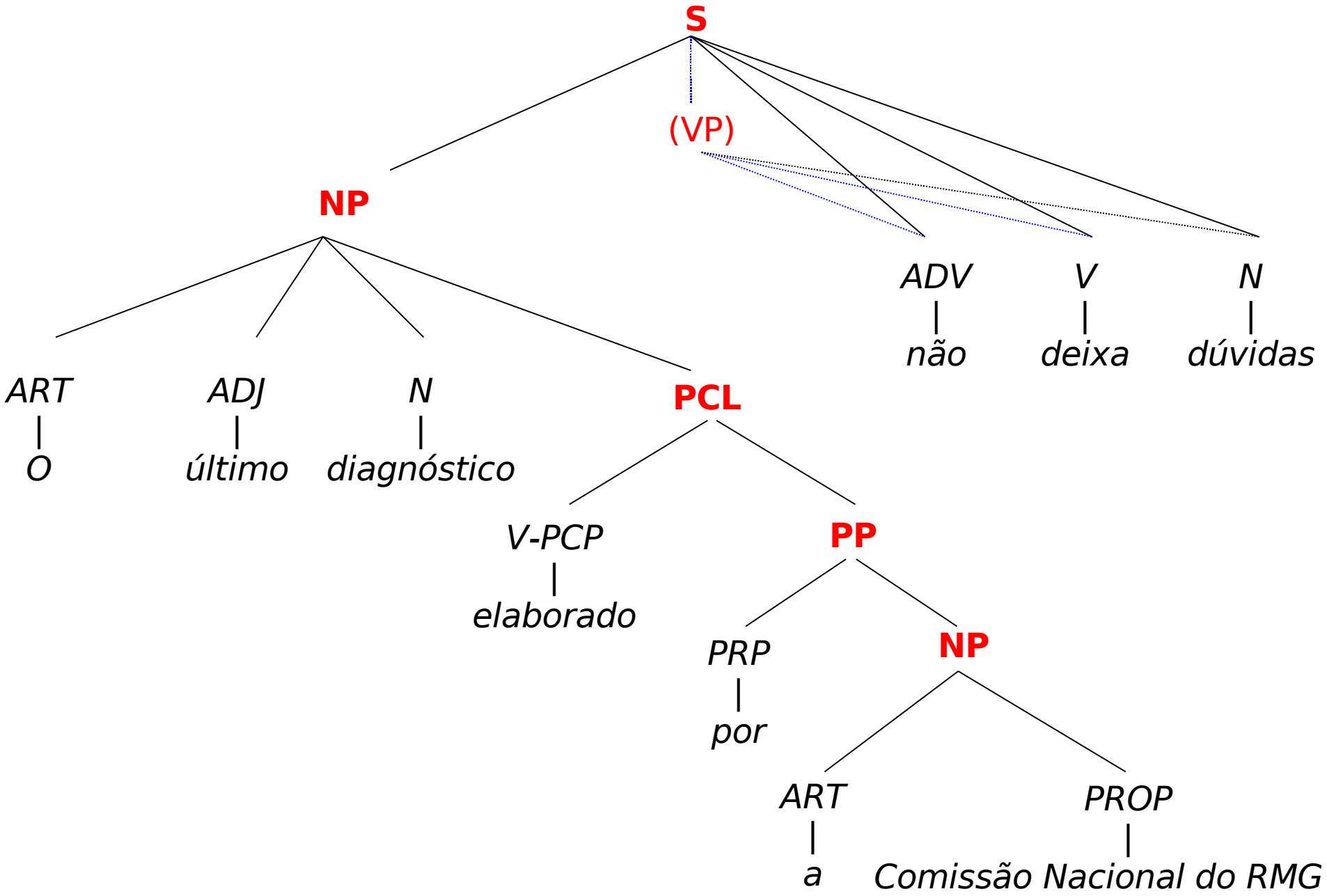
3. Constituent Grammar

- hierarchical word grouping with non-terminals
- syntactic form, no (or implicit) function
- expressed by rewriting rules, where a nont-terminal node is rewritten as a sequence of non-terminals and terminals (words or word classes)
  - e.g. CLAUSE -> SUBJ V OBJ ADVL

Pure Constituent Grammar:



Classical PSG with phrase labels

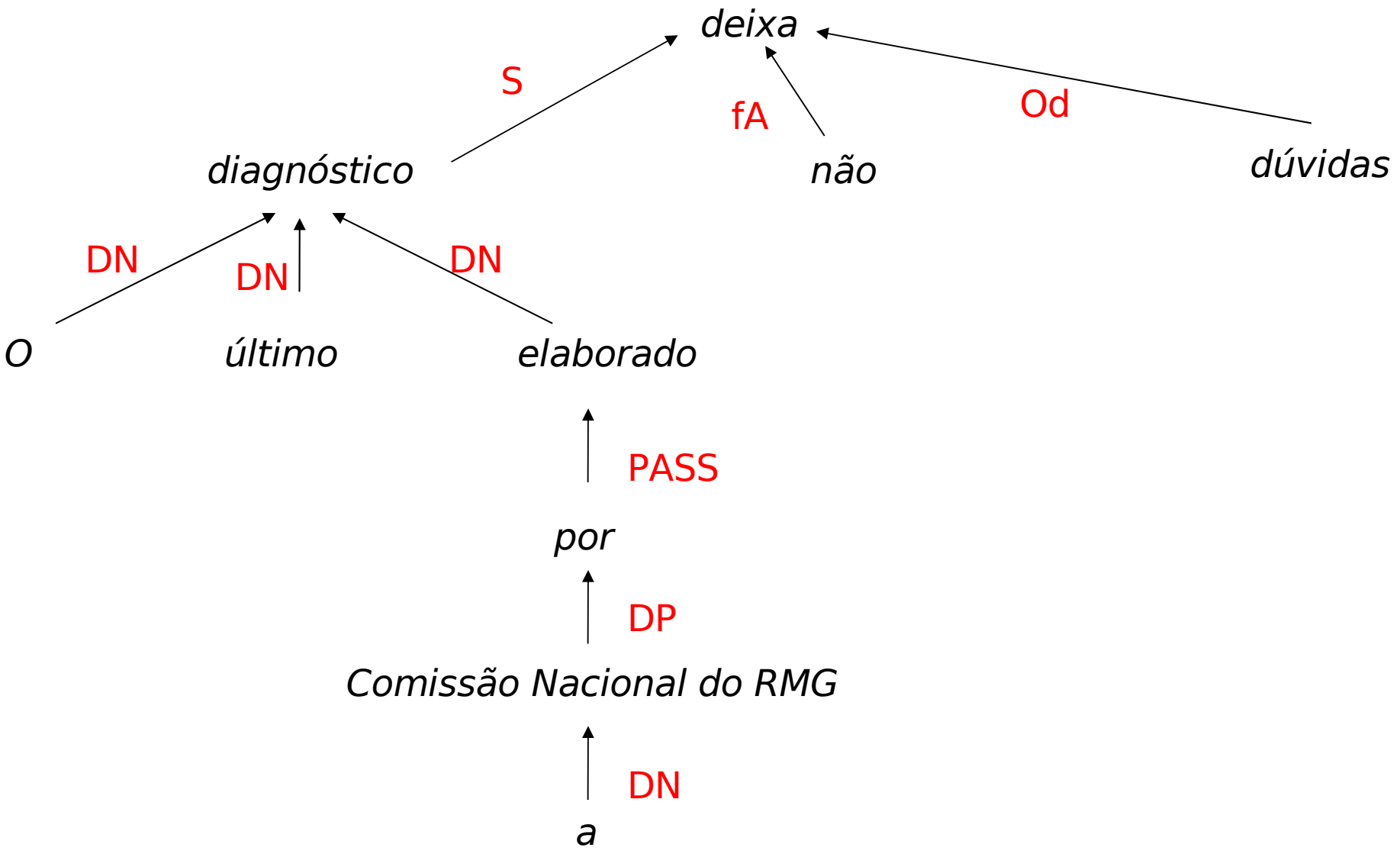


PSG annotation

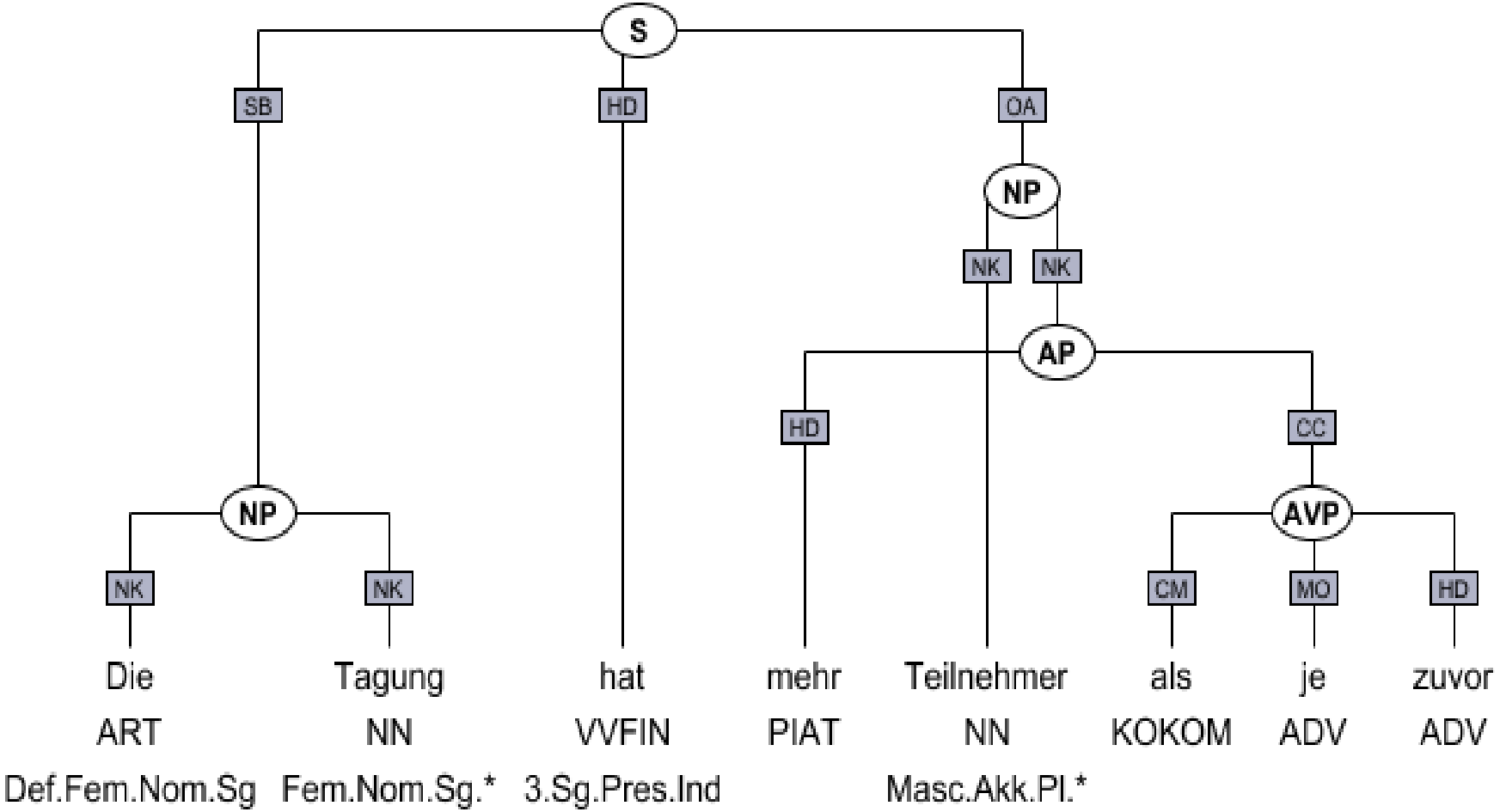
- **Penn Treebank bracketing**: Labeling opening brackets
  - [NP A minha irmã] [VP não fala [PP com [NP as amigas]]]
- **SUSANNE Treebank bracketing**: Labeling all brackets (cf. EAGLES)
  - [NP A minha irmã NP] [VP não fala [PP com [NP as amigas NP] PP] VP]
- **Vertical indented** (her with part of speech on one line):
  - [NP
    [Art A]
    [Det minha]
    [N irmã]
    NP]
    [VP
    [Adv não]
    [V fala]
    [PP
    [Prp com]
    [NP
    [Art as]
    [N amigas]
    NP]
    PP]
    VP]

Adding function:

- Dependency Grammar with function: adding function ("edge labels") to dependency arcs



- Constituent Grammar with function:
  - NEGRA, TIGER: cat labels (mother) vs. edge label (daughter

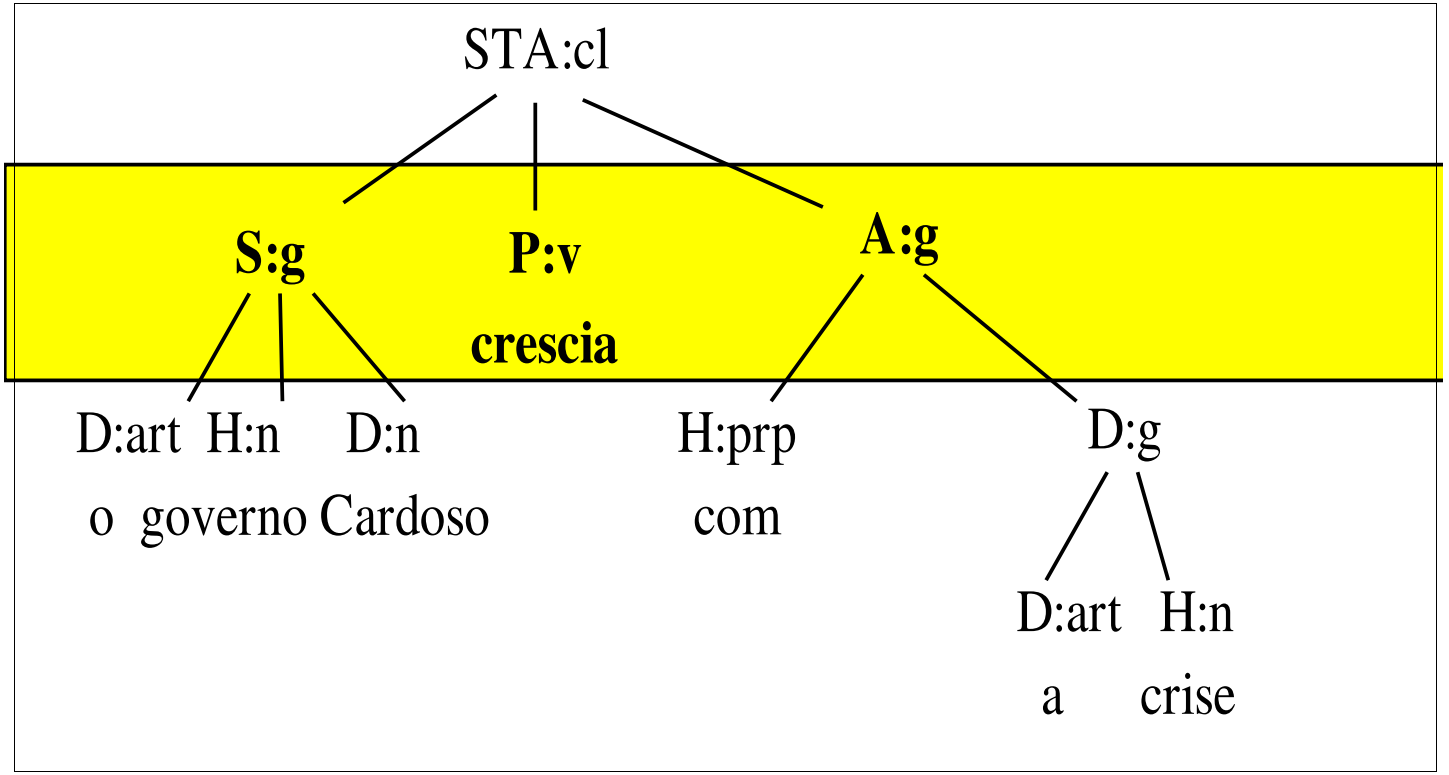


- Constituent Grammar with function:
  - VISL (function:form labels for each node)

Vertical Notation:
(Floresta native)

```
STA:cl
=SUBJ:np
==>N:art    O
==H:n       governo
==N<:prop   Cardoso
=P:v-fin
=ADVL:pp
==H:prp     com
==P<:np
===>N:art   a
===H:n      crise
```

Graphical Notation:



PSG (Chomsky)	**DG** (Tesnière, Melcuk)
Definite Clause Grammar (DCG) - Prolog *Transformational-Generative Grammar (TGG)* *Head-Driven Phrase Structure Grammar (HPSG) Functional Unification Grammar (FUG),* *Lexical Functional Grammar (LFG)* *Tree Adjoining Grammar (TAG) - Arvid Joshi* *AGFL* ... *Functional Grammar (FG) - Simon Dik* *Systemic Functional Grammar (SFG) - Halliday*	*Constraint Grammar (CG)* *Functional Dependency Grammar (FDG)* *Topological Dependency Grammar (TDG)* *Extensible Dependency Grammar (XDG)* *Link Grammar (D-H symmetry)* *Dependency Grammar Annotator (DGA)*
constituent based structure	token based structure
explanatory-linguistic perspective generative tradition: parsers	descriptive-applicational perspective analytical tradition: taggers
(labeled) brackets	(labeled) arcs
rewriting rules	attachment rules
originally fixed word order languages • English, French	originally free word order languages • Slavonic languages, Finnish, German
Problems: Discontinuity, free word order	Problems: Coordination, ellipsis
declarative programming	procedural programming
linguist-written: AGFL, HPSG, VISL-PSG, PATR, XTAG, ... machine-learned: PCFG e.g. Viterbi, Collins, Bikel	linguist-written: ENGCG, GERCG (Lingsoft), Machinese parsers (Conexor), VISL parsers machine-learned: MALT, Matsumoto, MSTParser,

4. Constraint Grammar (CG)

- CG as a **descriptive paradigm**
  - function-first approach with token-based function tags
  - Classic CG: shallow depedency (attachment direction, head type)
  - depth and constituents only implicitly marked

O	@>N	(pointer to head type: N)
meu	@>N	
hipopótamo	@SUBJ>	(direction pointer without head type)
não	@ADVL>	
come	@FMV	(top node)
peixe	@<ACC	

Adding full **numbered dependency**

- Integrated formalism: FDG
- CG dependency mapping: Vislcg3
- Add-on attachment rules: current PALAVRAS

@<ACC --> (<mv>) IF (L)

@SUBJ> --> (VFIN) IF (R) BARRIER:(@FS)

<np-long> --> (N,PROP,PERS,INDP,¤NP-HEAD)
IF (L) HEADCHILD=(<np-close>)

O <artd> DET M S	@>N	#1->3
último ADJ M S	@>N	#2->3
diagnóstico N M S	@SUBJ>	#3->9
elaborado V PCP2 M S	@ICL-N<	#4->3
por PRP	@<PASS	#5->4
a <artd> DET F S	@>N	#6->7
Comissão=Nacional PROP F S	@P<	#7->5
não ADV	@ADVL>	#8->9
deixa V PR 3S	@FMV	#9->0
dúvidas N F P	@<ACC	#10->9
$.		#11->0

- CG as a **methodological paradigm**
  - reductionist: focus on disambiguation, constraints as to what is *not* allowed in a given context
  - progressive level annotation: same method and tag-based annotation for ever higher linguistic levels
    - lexicon
    - morphology (“Analyzer”, “multitagger”, cohorts)
    - PoS disambiguation (“tagger”)
    - syntactic potential/mapping
    - syntactic disambiguation (“parser”, PALAVRAS syntax)
    - precise attachment (dependency or constituent structure)
    - case roles, clause bounderies, semantic classes, valency instantiation, anaphora resolution, discourse markers ....
      --> add your own module!
  - services many different NLP applications: Corpus research, MT, teaching, spellchecking, QA ...

A Constraint Grammar rules file

DELIMITERS (1 line, defines sentence boundaries)

DELIMITERS = “<.>” “<!>” "<?>" ;

SETS (1 or more sections of set definitions)

LIST N-LOC = <inst> <L> <Lh> <Lciv> <Lwater> <Lpath> <build> <BB> ;
LIST PROP-LOC = <top> <civ> <inst>
SET N/PROP-LOC = N-LOC OR PROP-LOC

MAPPINGS (adding new tags, e.g. syntax)

MAP (@SUBJ>) TARGET N/PROP (*-1 >>> BARRIER NON-PRE-N) (1C VFIN)
MAP (%TOP-PL) TARGET ("em") IF (0 @ADVL) (*1 @P< LINK 0 N/PROP-LOC) ;

CORRECTIONS (replacing tags anywhere in a reading)

SUBSTITUTE (TAG-1) (TAG-2) TARGET (TAG-3) IF (Context1) .. (Context2)

CONSTRAINTS (1 or more sections of REMOVE or SELECT rules, with each section compiled and run seperately)

REMOVE (VFIN) (*-1C CLB-WORD) (*1C VFIN BARRIER CLB OR KC)
SELECT (N) (-1 (<artd>)) (1 (<KOMP>)) (2 (ADJ) OR (PCP)) ;

The interface

corpuseye

standard search interface (old)

user-friendly cqp (new)

Treebanks

Guided tour

VISL credits info copyright publications links

Menu based category search