LING 696G
Computational Linguistics Seminar

Lecture 2

2/2/04

Administrivia

- No meeting next Monday (2/09)

The PAPPI System

- PAPPI:
  - **P**rinciples-**a**nd-**P**arameters **P**arser **I**nterface
  - (*real demo this time*…)

Outline

- General Introduction:
  - What is a Parser?
  - The Principles-and-Parameters Framework: Government-and-Binding Theory
- Introduction to the PAPPI System:
  - User Interface
  - Debugging
  - Project Scope

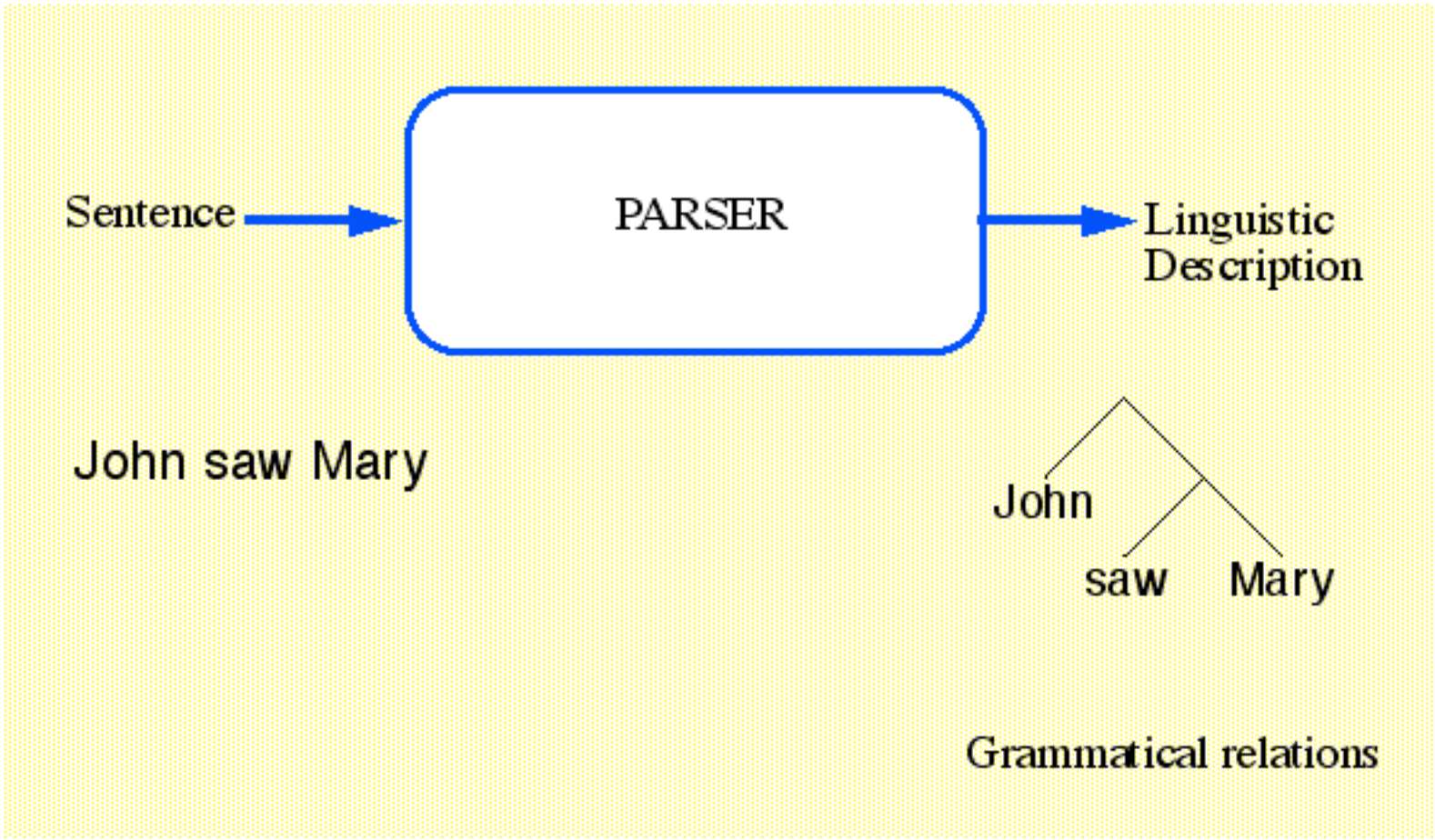
Outline

- Architecture Overview
- Components:
  - Simple Morphology
  - Recovery of Phrase Structure
  - Recovery of Movement: Determination of chain features
  - Free Indexation
  - Conditions on Trees and Domains

Outline

- **General Introduction:**
  - **What is a Parser?**
  - **The Principles-and-Parameters Framework: Government-and-Binding Theory**
- Introduction to the PAPPI System:
  - User Interface
  - Debugging
  - Project Scope

What is a Parser?



Immediate Issues

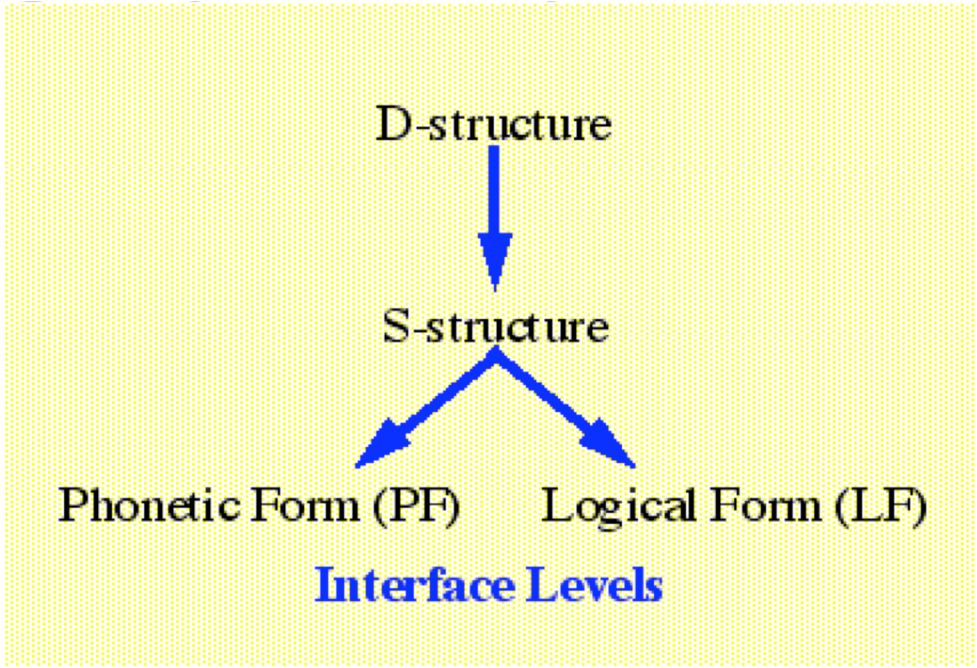
- **Parser Knowledge:**
  - Grammar
    - Which linguistic theory?
  - How to produce LDs?
    - Which parsing algorithm?
- **Other Issues:**
  - Language Parameterization?
    - Software viewpoint:
      - How do we re-use subcomponents of grammar for other languages?
    - Linguistic theory viewpoint:
      - Re-use for other language => supporting evidence for principle or module
  - Lexical representation?

Principles-and-Parameters Framework

- Government-and-Binding Theory:
  - *Lectures on Government and Binding* (LGB) (Chomsky, 1981)
  - *A Course In GB Syntax* (Lasnik & Uriagereka) MIT Press
    - Good introduction

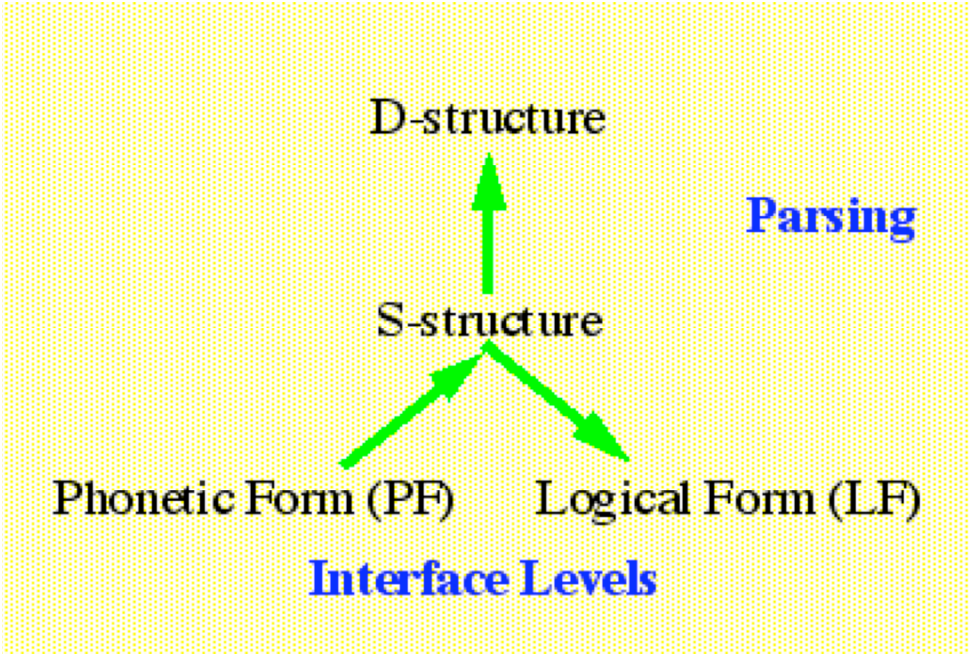
Levels of Representation

- D-structure:
  - Basic structure
- S-structure:
  - visible movement, e.g. passivization, *wh*-movement, raising
- LF:
  - Quantifier scope, covert *wh*-movement



Levels of Representation

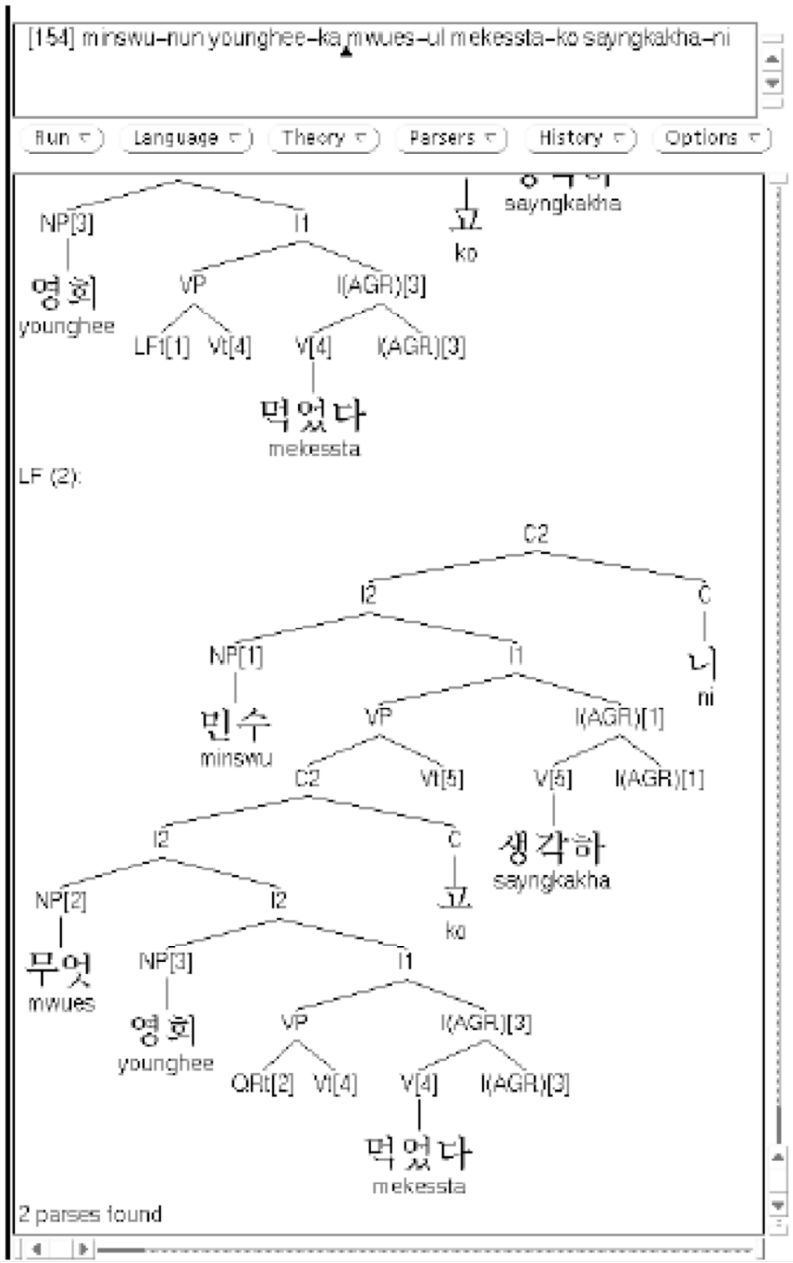
- Parsing Issues:
  - Recovery of empty categories
  - Recovery of "hidden" levels of representation
  - Relation between levels: re-creation of movement
  - Computational complexity



Outline

- General Introduction:
  - What is a Parser?
  - The Principles-and-Parameters Framework: Government-and-Binding Theory
- **Introduction to the PAPPI System:**
  - **User Interface**
  - **Debugging**
  - **Project Scope**

Interface



Info ...

Demo ...

Filters

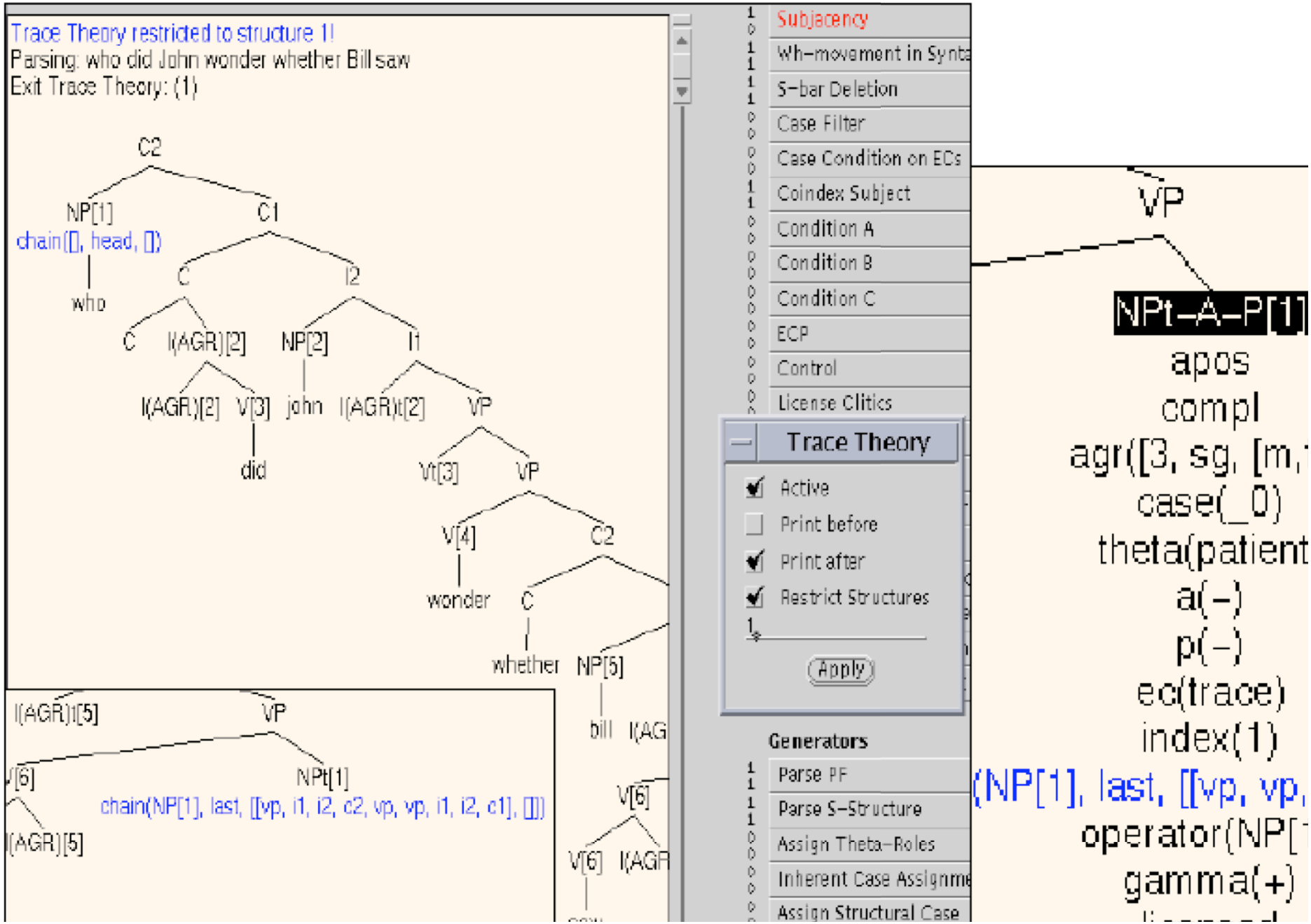
- Theta Criterion
- D-structure Theta Condition
- Subjacency
- Wh-movement in Syntax
- S-bar Deletion
- Case Filter
- Case Condition on ECs
- Coindex Subject
- Condition A
- Condition B
- Condition C
- ECP
- Control
- License Clitics
- License Object pro
- ECP at LF
- FI: License operator/variables
- FI: Quantifier Scoping
- FI: Reanalyze Bound Proforms
- License Clausal Arguments
- License Syntactic Adjuncts
- Wh Comp Requirement

Generators

- Parse PF
- Parse S-Structure
- Assign Theta-Roles
- Inherent Case Assignment
- Assign Structural Case
- Trace Theory
- Functional Determination
- Free Indexation
- Expletive Linking
- LF Movement

Korean
Head-final
Case particles
Allow
scrambling

Debugging


Trace Theory restricted to structure 1!
Parsing: who did John wonder whether Bill saw
Exit Trace Theory: (1)
C2
NP[1]
chain([], head, [])
who
C1
C
I2
C
I(AGR)[2]
NP[2]
I1
I(AGR)[2]
V[3]
did
john
I(AGR)t[2]
VP
Vt[3]
VP
V[4]
C2
wonder
C
whether
NP[5]
bill
I(AG
V[6]
V[6]
I(AGR
I(AGR)t[5]
VP
V[6]
NPt[1]
chain(NP[1], last, [[vp, i1, i2, c2, vp, vp, i1, i2, c1], []])
I(AGR)[5]
Subjacency
Wh-movement in Synta
S-bar Deletion
Case Filter
Case Condition on ECs
Coindex Subject
Condition A
Condition B
Condition C
ECP
Control
License Clitics
Trace Theory
Active
Print before
Print after
Restrict Structures
Apply
Generators
Parse PF
Parse S-Structure
Assign Theta-Roles
Inherent Case Assignme
Assign Structural Case
VP
NPt-A-P[1]
apos
compl
agr([3, sg, [m,
case(_0)
theta(patient
a(-)
p(-)
ec(trace)
index(1)
(NP[1], last, [[vp, vp,
operator(NP[
gamma(+)

Project Scope

- English
  - Classic GB: A Course in GB Syntax, Lasnik & Uriagereka
  - Plug-ins: VPS, Double Objects (Ch5) Zero Syntax Pesetsky
- Japanese
  - On the Nature of Proper Government, (Lasnik & Saito), Head-final, pro-Drop, LF Wh-movement
  - Scrambling: Some Asymmetries in Japanese... (Saito 85)
  - On Long-Distance Scrambling (Saito 92). WCO, A/A-bar distinction, Binding
  - Kanda Project: o/ni-causatives, Double-o constraint, Anti-superiority...

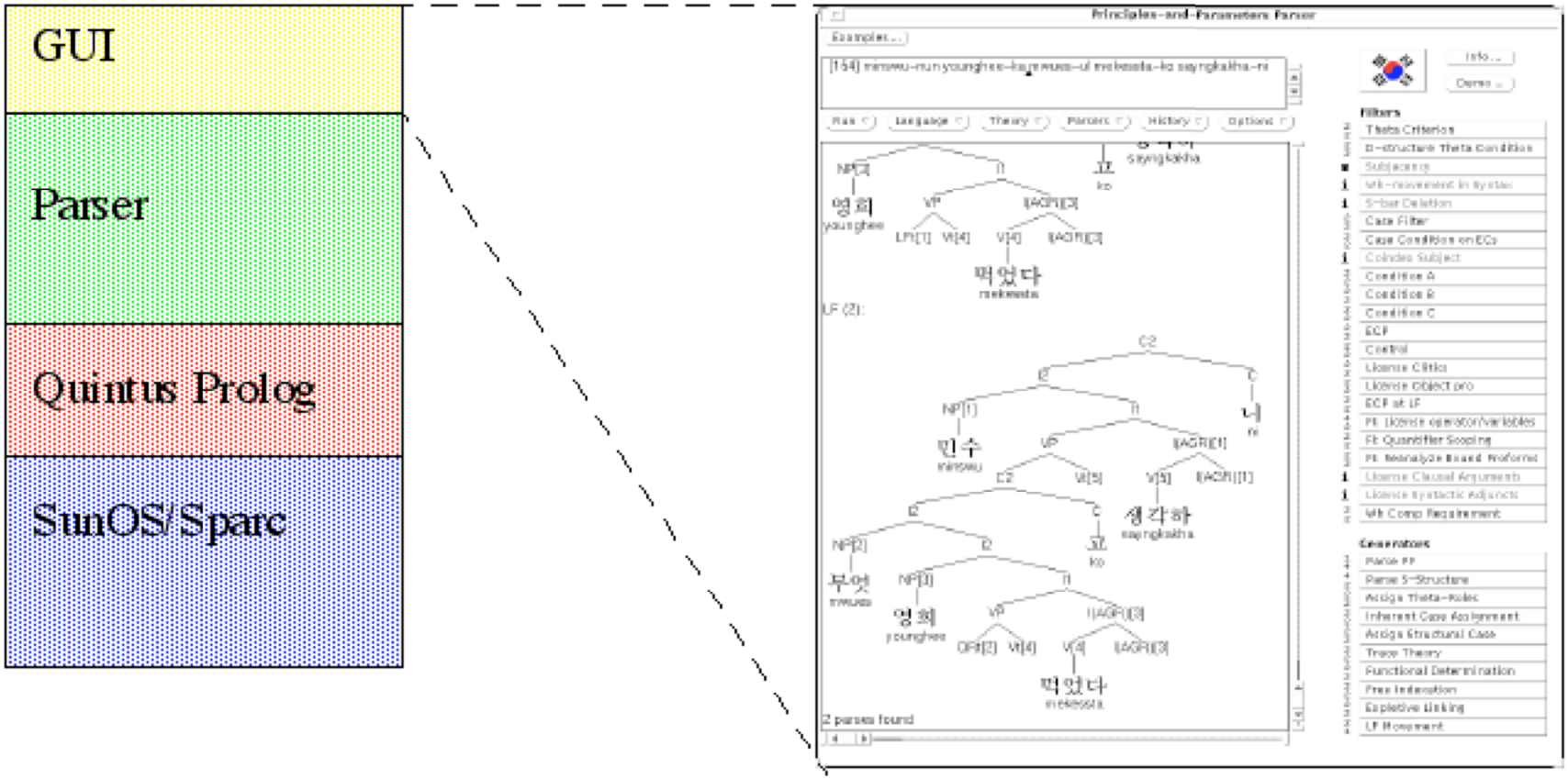
Project Scope

- Dutch
- V2 (also German), VR, Rightwards clausal extraposition.
- French
  - Verb Movement, UG, and the Structure of IP, Pollock. Clitics (also Spanish)
- Korean
  - Scrambling as Case-Driven Obligatory Movement, (Ch2) Lee, Reconstruction
- Turkish, Hungarian
  - Basic sentence structure, morphology, causatives.
- Arabic
  - VSO/SVO weak/strong verb agreement features

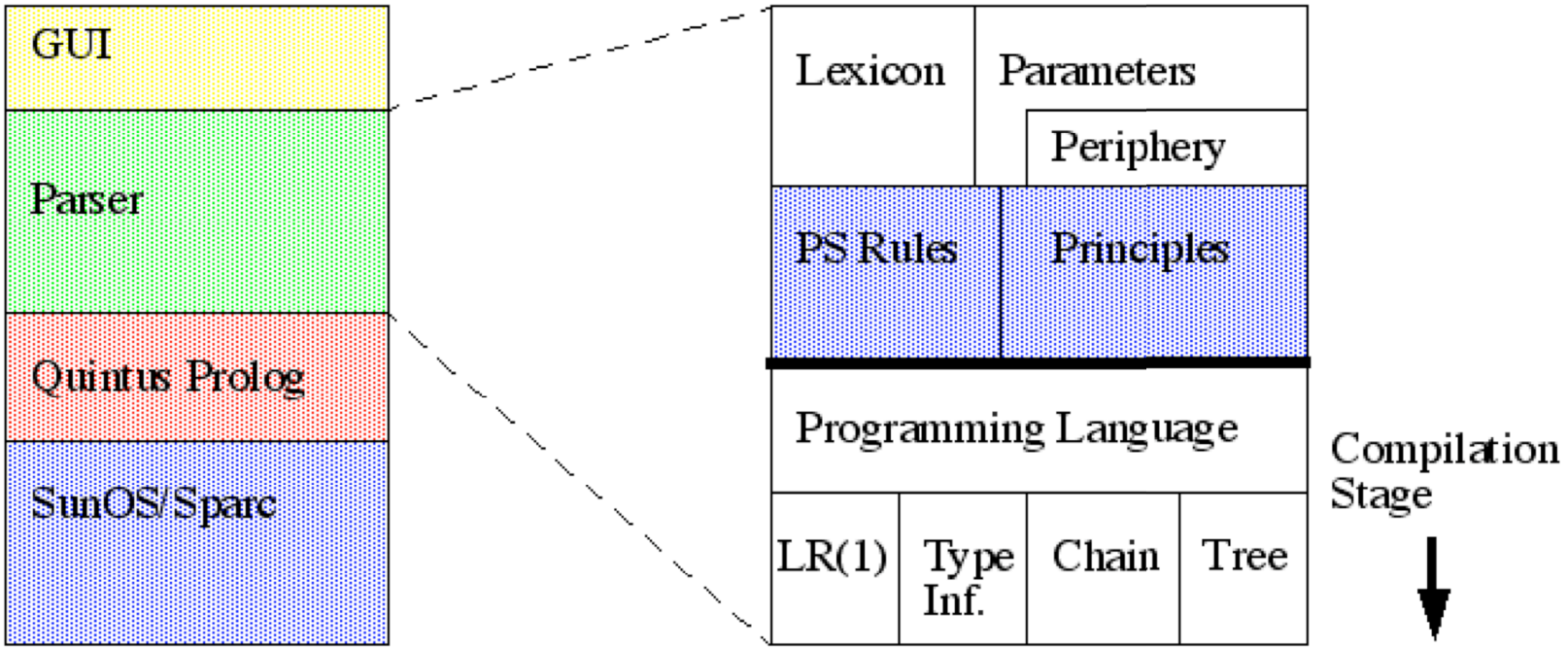
Outline

- **Architecture Overview**
- Components:
  - Simple Morphology
  - Recovery of Phrase Structure
  - Recovery of Movement: Determination of chain features
  - Free Indexation
  - Conditions on Trees and Domains

Architecture

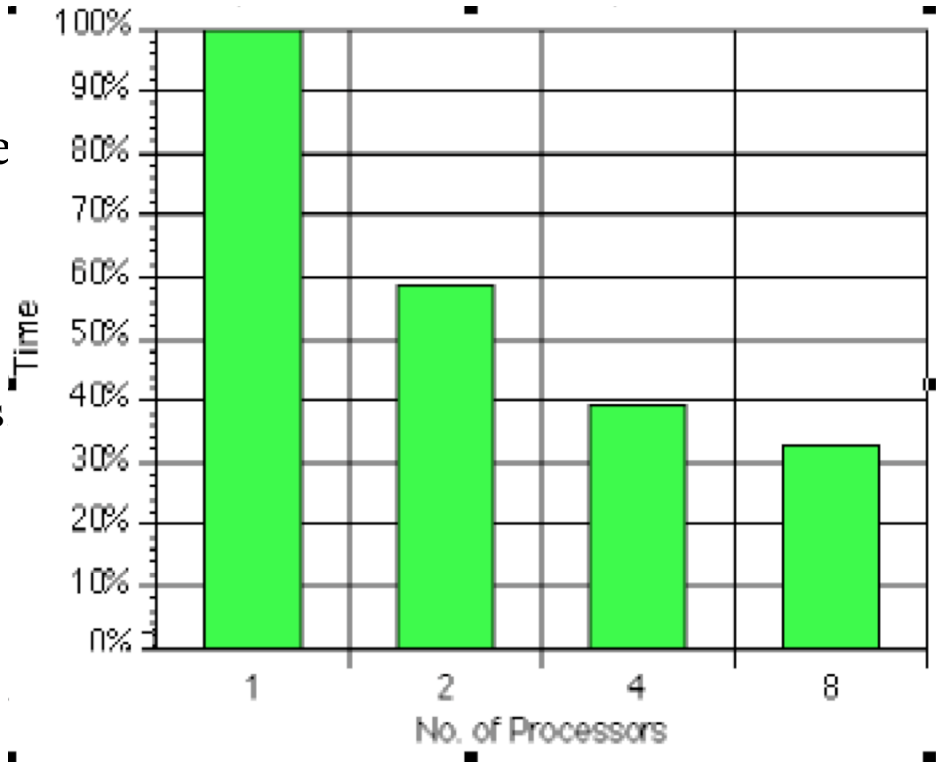


Architecture

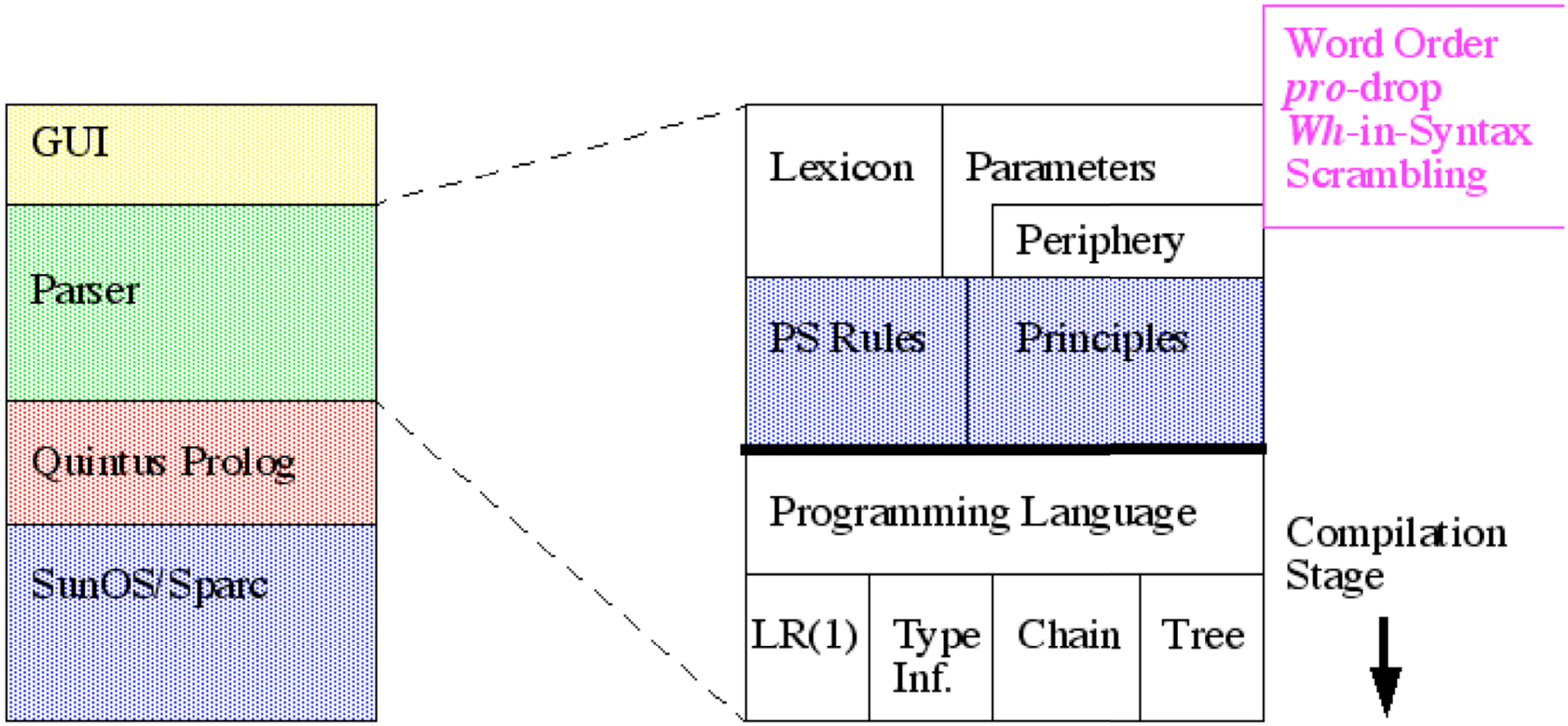


Architecture

- Salient Points:
  - Avoid hardwiring
  - Logic-based inference engine
  - Unification. Choice points: single thread, parallel
  - Abstraction:
    - conditions on trees, features domains
  - Direct interpretation: infeasible.
    - Specialization permits re-targeting
  - Sicstus Prolog (native: fastcode) port: factor of 0.425



Architecture



Outline

- Architecture Overview
- **Components:**
  - **Simple Morphology**
  - **Recovery of Phrase Structure**
  - **Recovery of Movement: Determination of chain features**
  - **Free Indexation**
  - **Conditions on Trees and Domains**

Simple Morphology: Features and Markers

Of-Insertion, Chomsky [KofL,86]:

... proud *of* him

case(gen)

Generators

1 1	Parse PF
1 1 2	Parse S-Structure

Markers

```
lex(to,mrkr,[right(v,morph(_,[]),inf([]))]).
lex(of,mrkr,[right(np,case(gen),[])]).
```

Contraction Rules

```
contraction([],can,'t,[can,neg]).
```

読まない
yo-manai

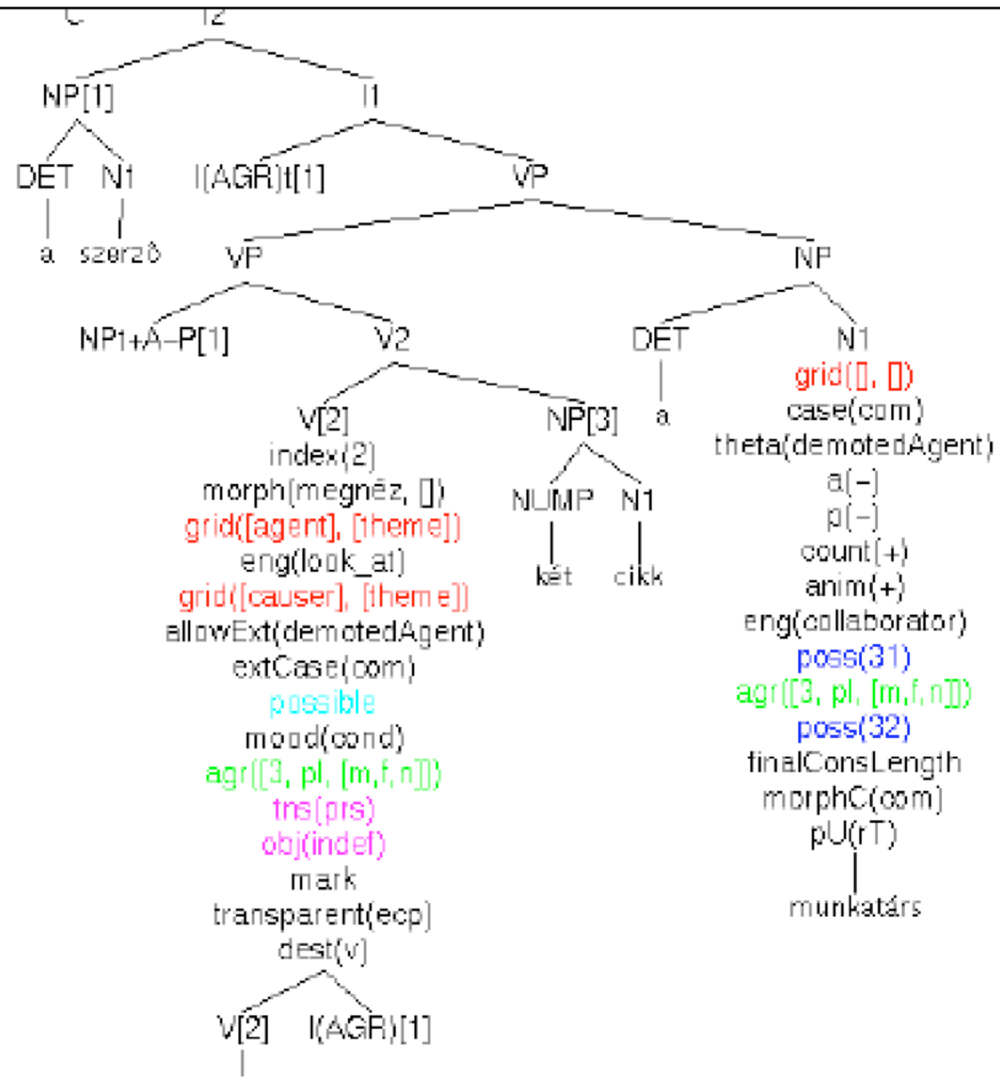
```
contraction(vEnd,X+nai,[X=pf([require(vNStem)]),negnpast]).
contraction(vNStem,X+ma,[X=word(base(mu))$$prefix(ma)]]).
```

Hungarian

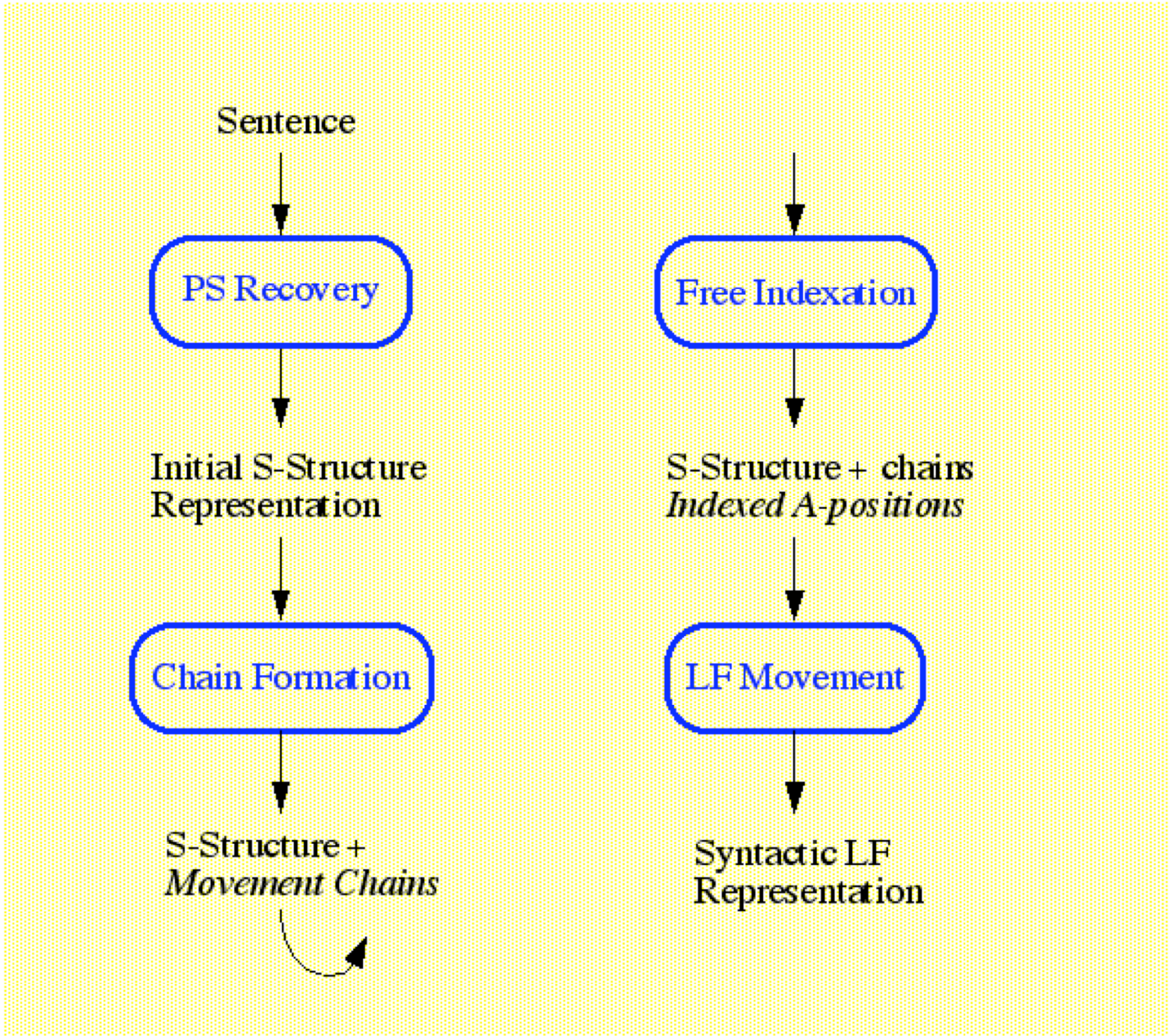
EXAMPLE:

a **szerzô-k** **megnéz-et---het---∅----né---nek --∅** **két** **cikk --et**
the author-Agr3Pl look_at---Caus-Possib-tns(prs)-Cond-Agr3Pl-Obj(indef) two article-Acc

a **munkatárs-a------ik -----------kal**
the colleague----Poss3Sg-Agr3Pl+Poss3Pl-LengdFC+Com



Stages of Processing

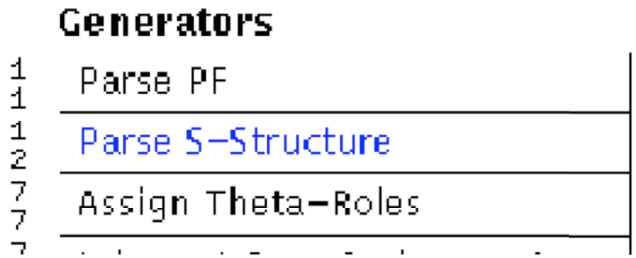


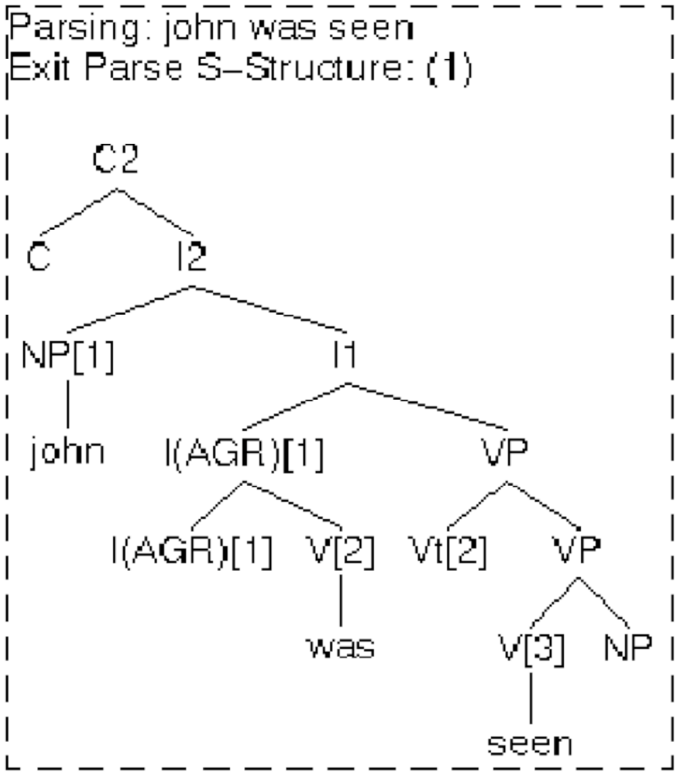
Initial Phrase Structure

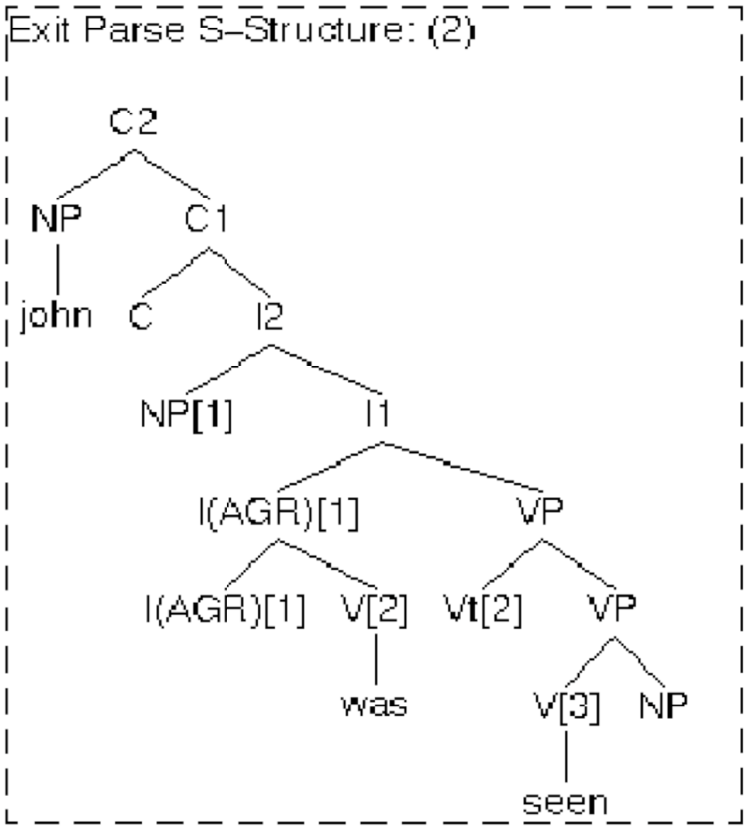
Example: John was seen

One source of structural ambiguity: phrase valency

V: intransitive/transitive
IP/CP: Specifier-position





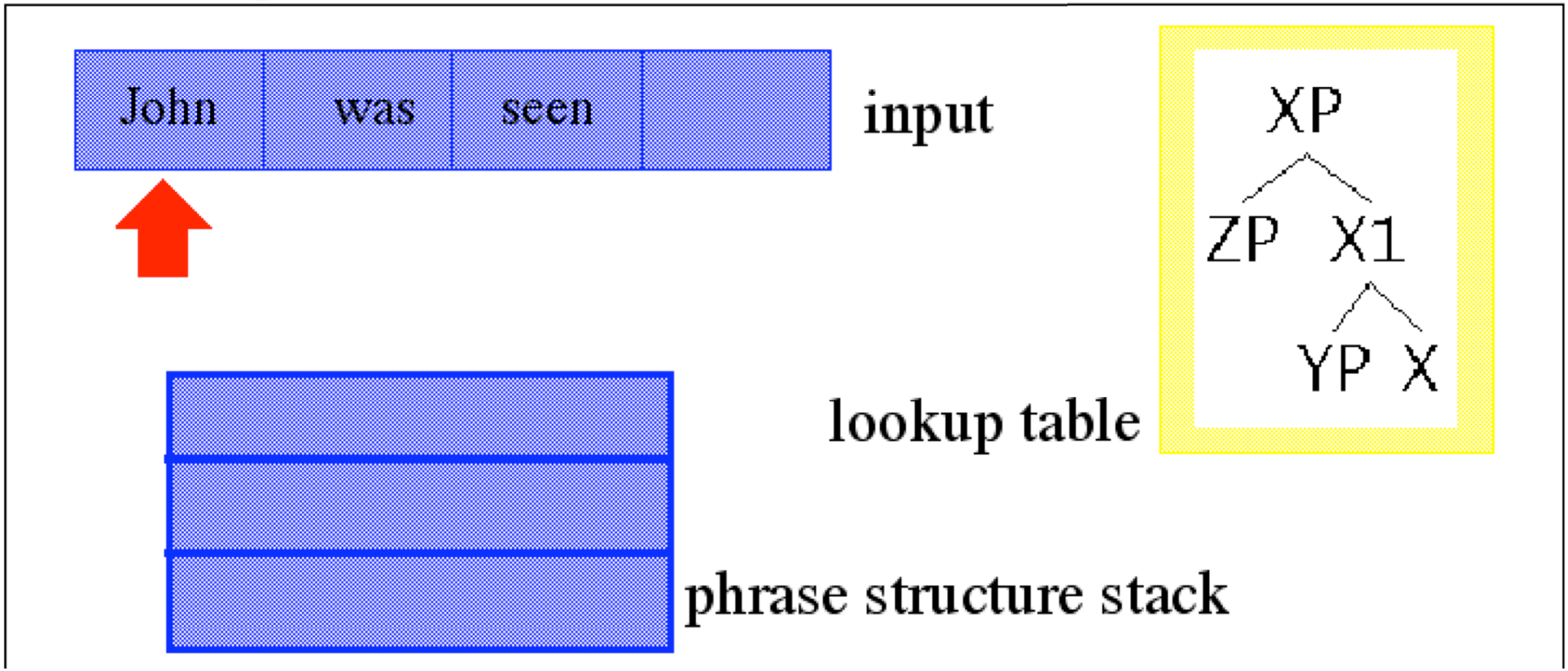


Recovery of Phrase Structure

$\overline{X}$ + X^0-Movement Rules

```
rule XP -> [XB|spec(XB)] ordered specFinal st max(XP), proj(XB,XP).
rule XB -> [X|compl(X)] ordered headInitial(X)
                                st bar(XB), proj(X,XB), head(X).
rule v(V) moves_to i provided agr(strong), finite(V).
rule v(V) moves_to i provided agr(weak), V has_feature aux.
```

Backtracking Canonical LR(1)-based Shift/Reduce Parser



Recovery of Phrase Structure

- X-bar + X^0-Movement Rules:
  - **rule** XP -> [XB|spec(XB)] **ordered** specFinal **st** max(XP), proj(XB,XP).
  - **rule** XB -> [X|compl(X)] **ordered** headInitial(X) **st** bar(XB), proj(X,XB), head(X).
  - **rule** v(V) **moves_to** i **provided** agr(strong), finite(V).
  - **rule** v(V) **moves_to** i **provided** agr(weak), V has_feature aux.

Recovery of Phrase Structure:

Backtracking Canonical LR(1)-based Shift/Reduce Parser

```
machine(accept,[],[SS],_,accept,[],SS,_).
machine([State|CS],[Input,SS,ES,CS2,I2,SS2,ES2]) :-
            Input = [I|Is], ss0Cat(I,C),
            action(State,C,[State|CS],[I|Is],SS,ES,CS1,I1,SS1,ES1),
            machine(CS1,I1,SS1,ES1,CS2,I2,SS2,ES2).
```

(State,Lookahead)

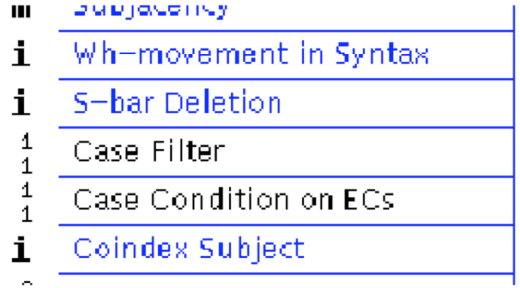
```
% 138:$ no conflict
action(138,$,[_629,_631,_633,_551|_552],_540,[_608,_604|_554],_542,[_549,_551|_552
],_540,[[c2$_561$_562,_604,_608]|_554],_546):-close(_542,_146),specifi-
erR(_608,_604),_608 has_features _562,transi-
tion(_551,c2,_549),c2rgoal([c2$_561$_562,_604,_608]).% reduce:c2-
>[leftc2,specCPwhadv,c1]/#2/2
% 138:c1 ERROR
% 137:_2180 no conflict
action(137,_180,[_290,_292,_199|_200],_188,[_263,_259|_202],_190,[_197,_199|_200
```

LR Actions

Recovery of Phrase Structure:

Other Constraints

- Efficiency: Interleave principles with initial S-structure building



- Type inference alg. determines which reduce actions should have semantic out-calls inserted:

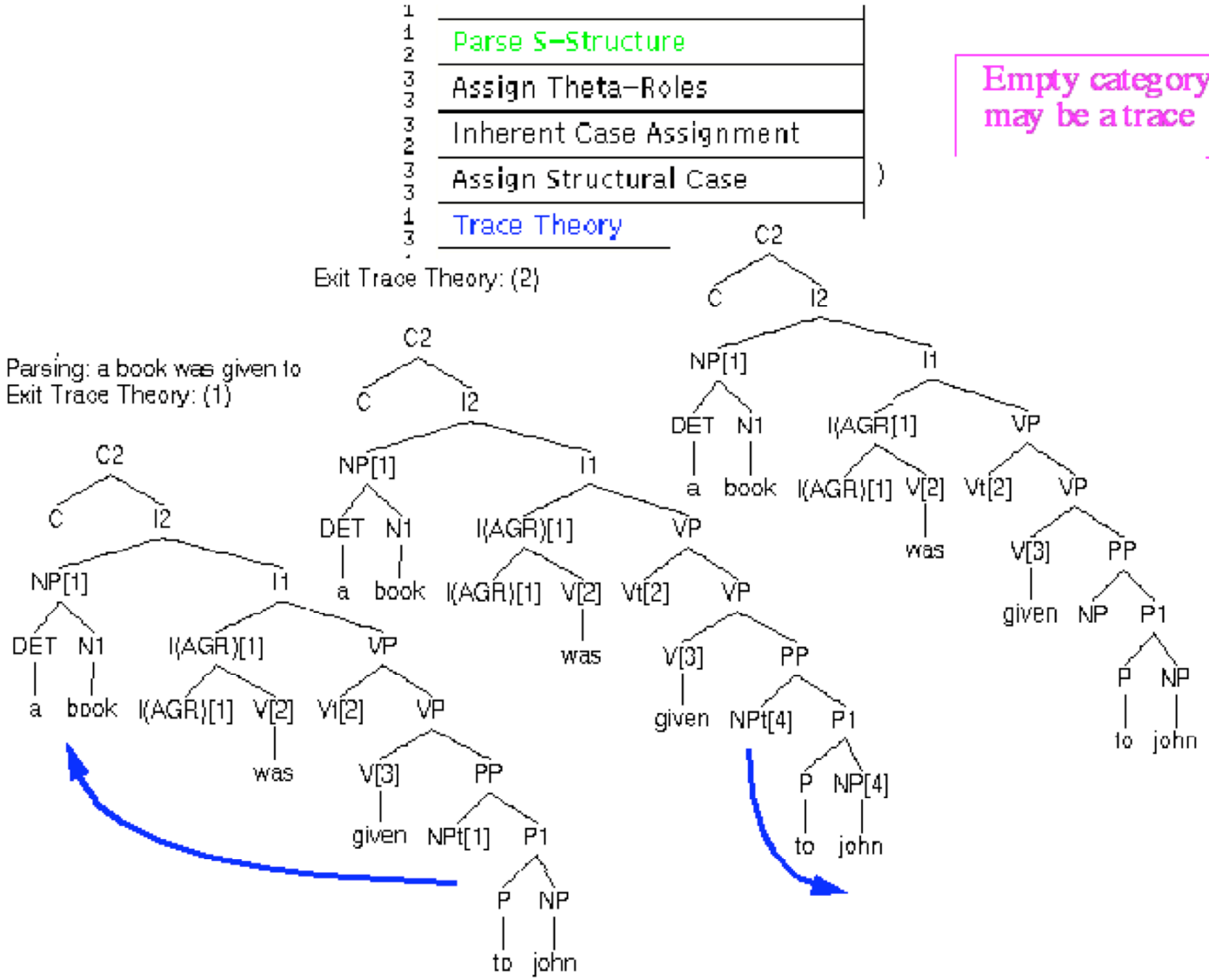
```
Loaded parser j5parser
To interleave operations: licenseClausalArguments, licenseAdjuncts,
sBarDeletion, coindexSubjAndINFL, whInSyntax
licenseClausalArguments interleaved for categories: spec(c2)
licenseAdjuncts interleaved for categories: ap, i2, np, vp
sBarDeletion interleaved for categories: n1, np, ap, v1, vp
coindexSubjAndINFL interleaved for categories: i2
whInSyntax interleaved for categories: c2
```

- Example:

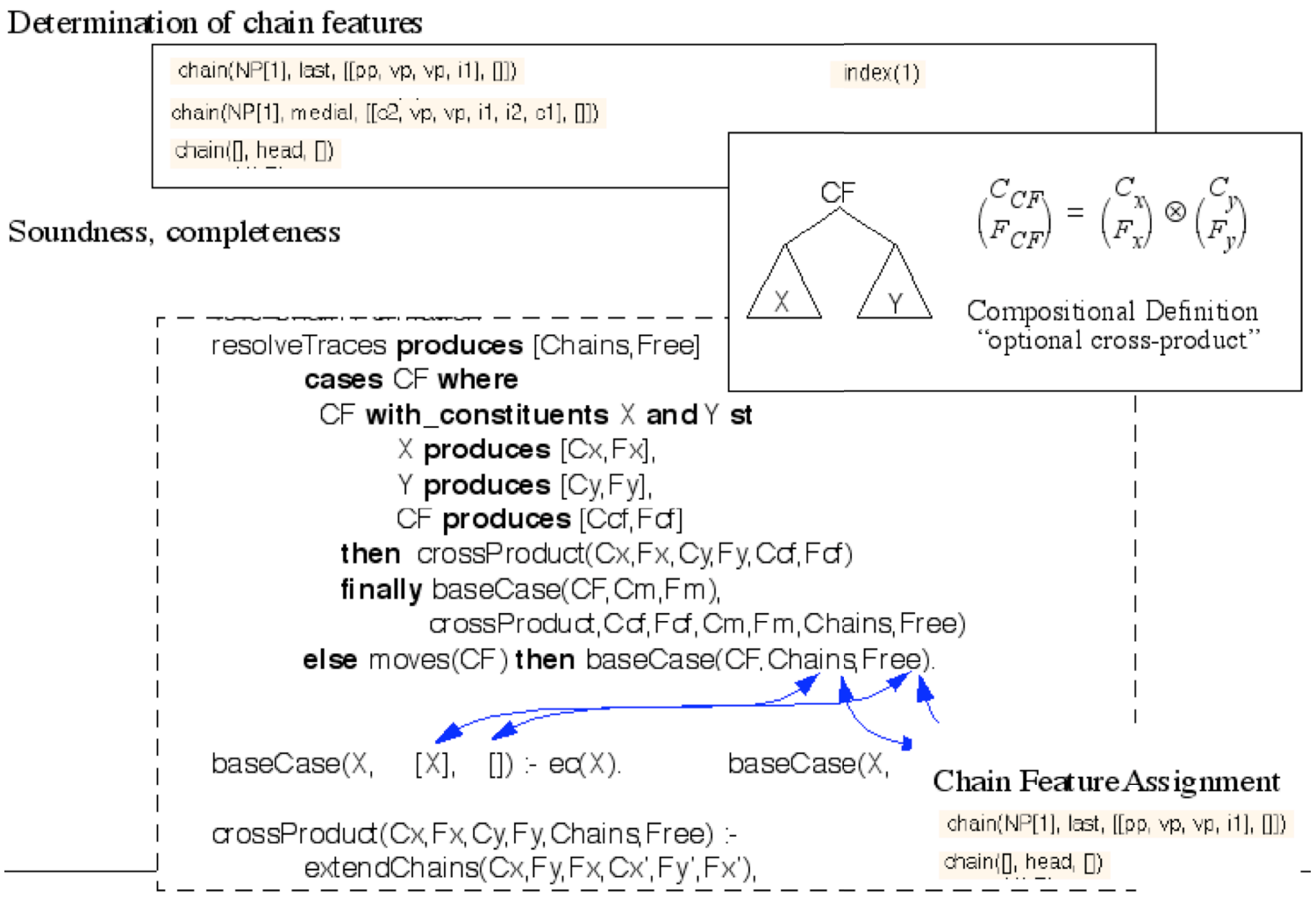
```
coindexSubjAndINFL in_all_configurations CF
        where specIP(CF,Subject) then coindexSI(Subject,CF).

coindexSI(Subj,IP) :- agreeAGR(Subj,IP) if IP has_feature agr(_), coindex(Subj,IP).
```

Phrase Structure After Chain Formation



Recovery of Movement:
Computation of Chain Features



Recovery of Movement:
Complexity

Complexity:

NPs	Indexings	NPs	Indexings
1	1	7	877
2	2	8	4140
3	5	9	21147
4	15	10	115975
5	52	11	678570
6	203	12	4123597

Upper-bounded by *Bell's Exponential Number*

$$B_n = \sum_{m=1}^{n} \sum_{k=0}^{m} \frac{(-1)^{m-k}}{(m-k)!\,k!} k^n$$

$$\frac{{m_n}^n e^{m_n - n - \frac{1}{2}}}{\sqrt{\ln n}}$$

$$m_n \ln m_n = n - \frac{1}{2}$$

Partial Solution: Merge constraints

m	Subjacency
m	Lowering Filter
i	Wh-movement in Syntax

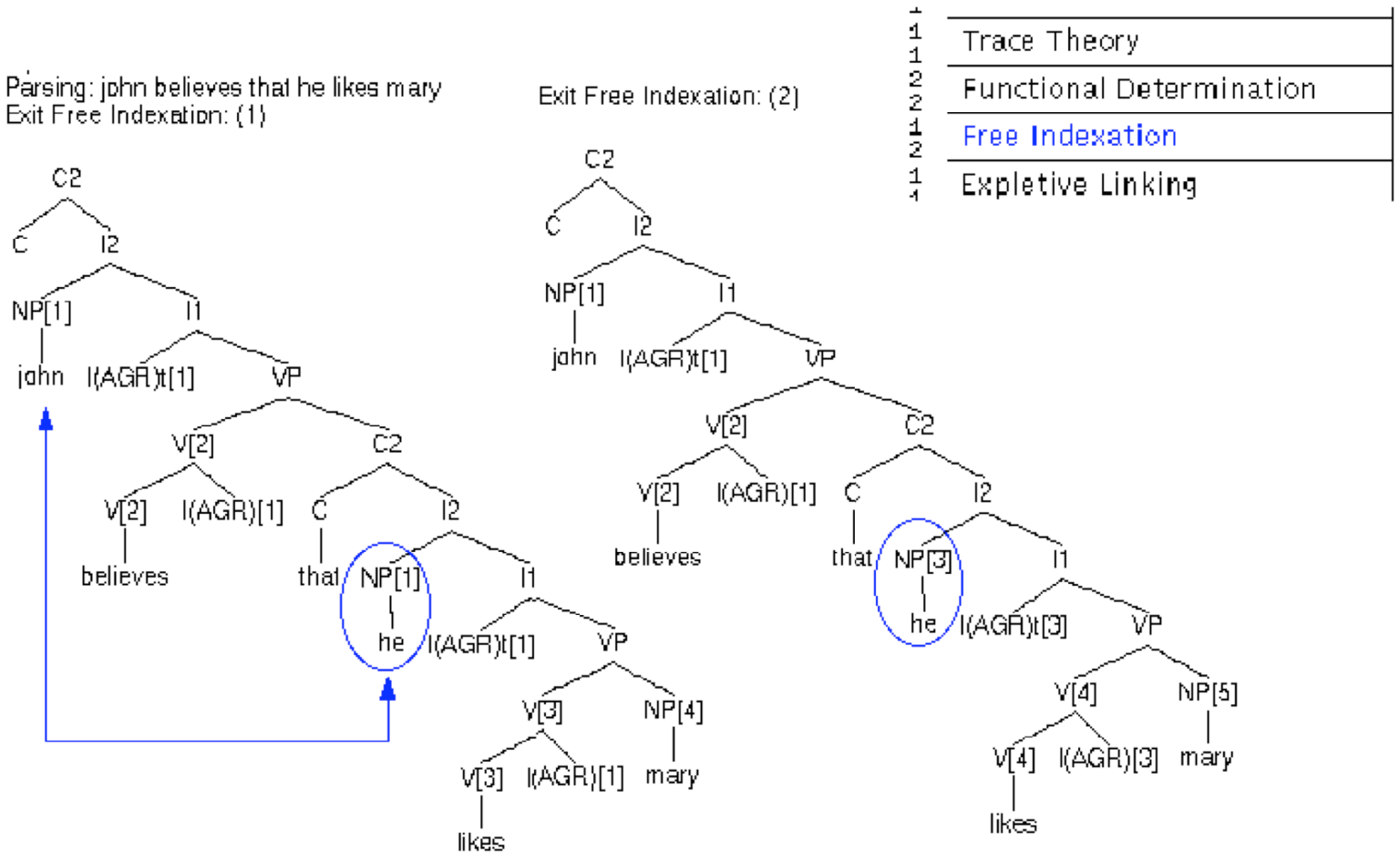
```
subjacency in_all_configurations CF where
        isTrace(CF), upPath(CF,Path) then lessThan2BoundingNodes(Path).

loweringFilter in_all_configurations CF where
        isTrace(CF), downPath(CF,Path) then Path=[].
```

Phrase Structure after Free Indexation

- Chains already indexed: *Chain Formation, Expletive-Argument Linking*

Assign indices freely to NPs in A-positions



Conditions on Trees and Domains

Binding Conditions A and B:

An anaphor must be A-bound in its GC.

A pronominal must not be A-bound in its GC.

Governing Category (GC):

GC(α) is the smallest NP or IP containing:

(A) α, and

(B) a governor of α, and

(C) an accessible SUBJECT for α

```
conditionA(A) :- condA(A,B).
condA(A,B) :-
      A has_constituents C, !,
      condAl(C,D), condAml(D,A,E),
      (anaphor(A)
      -> condAmo(A,E,B)
      ;  B=E).
condA(A,B) :-
      anaphor(A), !,
      initgc(A,C), B=[C].
condA(A,[]).
gc(A,gc(A,B,C),D) :- %% GC(A)
            (var(B) -> (governs(B,A,D
            (var(C) -> (locallyaccSub
completegc(gc(A,B,C),D) :-
            nonvar(A), nonvar(B), nor
            cat(D,E), member(E,[np,i2
initgc(A,gc(A,B,C)).
```

Code: program clichés

```
conditionA in_all_configurations CF where
            anaphor(CF) then gc(CF,GC), aBound(CF,GC)
anaphor(NP) :- NP has_feature apos, NP has_feature a(+).

gc(X) smallest_configuration CF st cat(CF,C), member(C,[np,i2])
            with_components
                        X,
                        G given_by governs(G,X,CF),
                        S given_by accSubj(S,X,CF).
```

Implementation: bottom-up domain instantiation

Phrase Structure After LF Movement

"Why did Bill say that John was fired?"