Computational Linguistics
A.Y. 2018/19 - C. Chesi

Lecture 12-14

ADVANCED PARSING: FROM RULES, TO P&P… AND MINIMALISM

Index

References

- **Essential references**
  - Stabler E. (1997) Derivational minimalism. in Retoré, ed. *Logical Aspects of Computational*. Linguistics. Springer
- **Extended references**
  - Chesi C. (2015) On directionality of phrase structure building. *Journal of Psycholinguistic Research.* 44:65–89
  - Fong S. (1991) *Computational Properties of Principle-Based Grammatical Theories*. tesi Ph.D. MIT

From Rules to Principles and Parameters

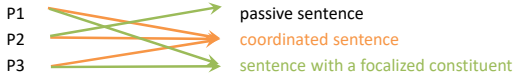
- **rules** (Language specific) — **Principle & Parameters (P&P)** (linguistic universals + parameters settings)
- P&P aims at a better explicative adequacy (other than descriptive)
- **Goal**: linguistic universals capture the limited syntactic variability across languages
- **Principle-based parsers** (Barton 1984, Berwick e Fong 90, Stabler 92) are inspired by this intuition:
  - Grammatical principles are parser **axioms**
  - the parser operates as a **deductive system** inferring grammatical structures applying the axioms to the input

From Rules to Principles and Parameters

- **Rules**

passive transformation	→	passive sentence
coordination transformation	→	coordinated sentence
focalization transformation	→	sentence with a focalized constituent
...		

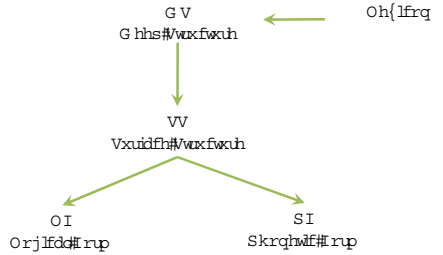
- **P&P**



- Few principles + few parameters = thousands of rules!

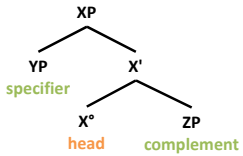
P&P

- "T" model



P&P

- Some principles
  - **X' theory**



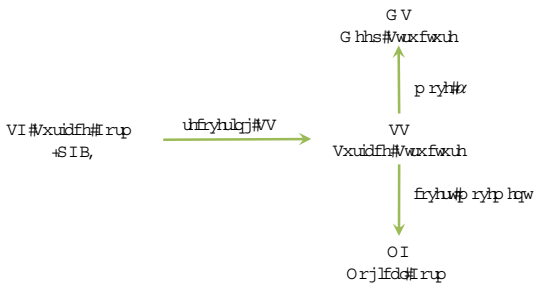
  - **θ - criterion**
    every argument must receive one and only one thematic (θ) role (and every thematic role is assigned to just one argument)
  - **Case filter**
    every lexical NP must receive case (P e V_{finite} are case assigners)

P&P

- **Generators**
  principles producing more structures than the ones in input:
  - **Move α**
  - **Free indexation**
  - **...**
- **Filters**
  principles selecting fewer structures than the ones received as input:
  - **X' theory**
  - **θ - criterion**
  - **Case filter**

PAPPI (Fong 1991)

- Grammatical principles are expressed on a "**high order**" language (first order logics) then compiled in the parsed (this is a strategy for improving performance)



PAPPI (Fong 1991)

- **Case Filter**
  - **definition**
    "Every lexical NP must receive a case at SS"
  - **(pseudo) formalization**
    $\forall$NP: NP $\subset CF_{SS}$, NP → case
  - **Implementation in Prolog**

```
:- caseFilter in_all_configuration CF where
        lexicalNP(CF) then assignedCase(CF).

lexicalNP :- cat(NP, np), +\ ec(NP).
assignedCase(X) :- X has_features case(Case), assigned (Case).
```

PAPPI (Fong 1991)

Operation	Encoded Using
θ-role assignment	in_all_configurations
θ-criterion	in_all_configurations
D-structure θ-condition	in_all_configurations
Subjacency	compositional_cases_on
Inherent Case assignment	in_all_configurations
Structural Case assignment	in_all_configurations
$\overline{S}$-deletion	in_all_configurations
Case filter	in_all_configurations
Case condition on traces	in_all_configurations
Trace theory (chain formation at S-structure)	compositional_cases_on
Wh-movement in syntax	in_all_configurations
Coindex Subject and INFL	in_all_configurations
Free Indexation	compositional_cases_on
Functional Determination	compositional_cases_on
Condition A	in_all_configurations (transformed)
Condition B	in_all_configurations (transformed)
Condition C	in_all_configurations (transformed)
Empty Category Principle (ECP)	in_all_configurations (transformed)
Control	compositional_cases_on
LF movement	Hand-coded
ECP at LF	in_all_configurations (transformed)
License operator-variables	in_all_configurations
License syntactic adjuncts	in_all_configurations
Wh-Comp requirement at LF	in_all_configurations

PAPPI (Fong 1991)

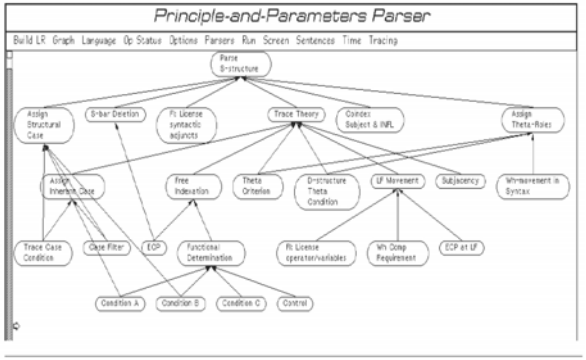


Figure 1.6 Operational dependency graph.

PAPPI (Fong 1991)
Ordering Principles

- **16 principles = 16! Possible orders…**
  nearly 21.922.789.888.000 possibilities (which one is the… best?)
- Each order **different performance** with respect to the same input
- **No universal order** guarantee absolute efficiency (best performance ever)
- Fong suggests using cheap cues for pre-ordering generators and filters.

PAPPI (Fong 1991)
Control strategies

- **Analysis-by-synthesis**
  produces everything, then test the input (parallel top-down approach)
- **Generate and Test**
  one principle at time, generate structures, then test the input (serial top-down, depth first)
- **Coroutining, freezing, clause selection, ordering**
  re-organize the problem space, trying to minimize the exploration needs
- **Covering**
  off-line grammar compile; groups of states collapsing whenever possible, reducing time complexity (number of steps)

PAPPI (Fong 1991):
Architecture

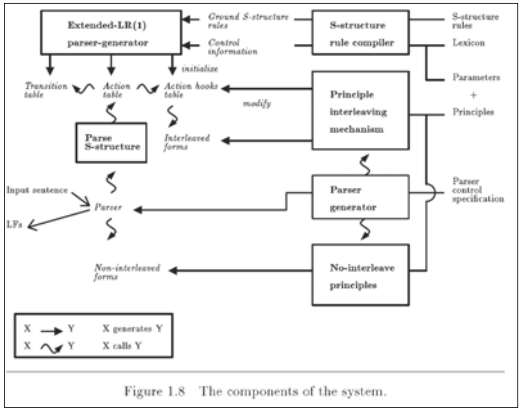


Figure 1.8 The components of the system.

Minimalist Grammars

- Stabler's (1997) formalization of a Minimalist Grammar, MG (Chomsky 1995) as a 4-tuple {V, Cat, Lex, F} such that:

  V is a finite set of non-syntactic features, (P ∪ I) where
  P are phonetic features and I are semantic ones;
  Cat is a finite set of syntactic features,
  Cat = (base ∪ select ∪ licensors ∪ licensees) where
  base are standard categories {comp, tense, verb, noun ...},
  select specify a selection requirement {=x | x base}
  licensees force phrasal movement {–wh, –case ...},
  licensors satisfy licensee requirements {+wh, +case ...}
  Lex is a finite set of expressions built from V and Cat (the lexicon);
  F is a set of two partial functions from tuples of expressions to expressions : {merge, move};

Minimalist Grammars

V = *P* = {/what/, /did/, /you/, /see/},
I = {[what], [did], [you], [see]}

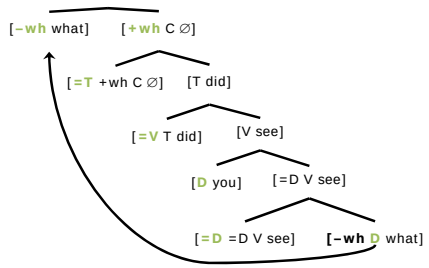
Cat = *base* = {D, N, V, T, C}
select = {=D, =N, =V, =T, =C}
licensors = {+wh}
licensees = {–wh}

Lex = { [–wh D what], [=V T did], [D you], [=D =D V see], [=T +wh C ∅] }

F = {*merge*, *move*} such that:
merge ([=*F* X] , [*F* Y]) = [$_X$ X Y]
("simple merge" on the right, "complex merge" on the left)
move ([+*g* X] , [W [–*g* Y]]) = [[$_X$ Y X] W, t_Y]

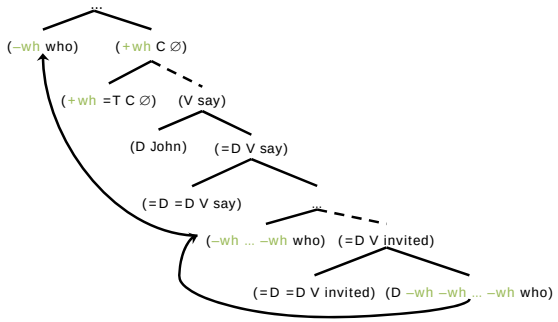
Minimalist Grammars

1. merge ([=D =D V see], [-wh D what]) → [$_{see}$ =D V see, –wh what]
2. merge ([D you], [=D V see, -wh what]) → [$_{see}$ you, [$_{see}$ V see, –wh what]]
3. merge ([=V T did], [$_{see}$ you, [$_{see}$ V see, -wh what]]) →
([$_{did}$ T did, [$_{see}$ you, [$_{see}$ see, –wh what]]]
4. merge ([=T +wh C ∅], [$_{did}$ T did, [$_{see}$ you, [$_{see}$ see, –wh what]]]) →
([$_C$ +wh C ∅, [$_{did}$ did, [$_{see}$ you, [$_{see}$ see, –wh what]]]])
5. move ([$_C$ +wh C ∅, [$_{did}$ did, [$_{see}$ you, [$_{see}$ see, –wh what]]]]) →
[$_C$ What C ∅, [$_{did}$ did, [$_{see}$ you, [$_{see}$ see, t_{what}]]]]



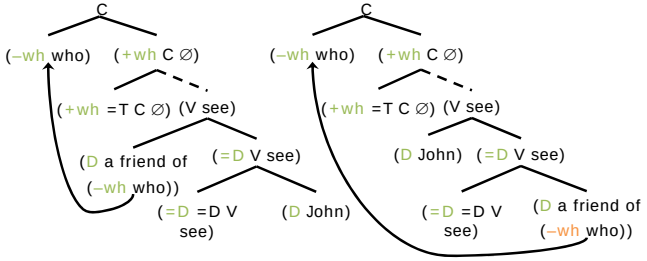
Minimalist Grammars: a problem

- *Wh*- successive cyclic movement (how many features needed? Infinite!)



Minimalist Grammars: another problem!

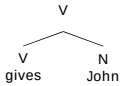
- How to capture (**subject) islandhood** (same for **RCs** e **Adjuncts**)



PMGs SBO: Merge Right (Phillips 1996)

Lexicon: {[$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+K\ (N)}$ to], [$_{+D\ N}$ John], [$_{+D\ N}$ children], [$_{+D\ N}$ candies]}

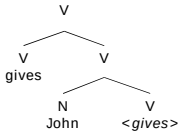
1. merge ([$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+D\ N}$ John])



PMGs SBO: Merge Right (Phillips 1996)

Lexicon: {[$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+K\ (N)}$ to], [$_{+D\ N}$ John], [$_{+D\ N}$ children], [$_{+D\ N}$ candies]}

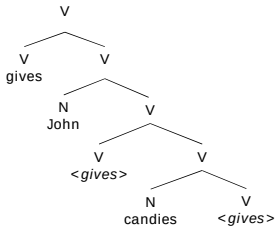
1. merge ([$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+D\ N}$ John])



PMGs SBO: Merge Right (Phillips 1996)

Lexicon: {[$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+K\ (N)}$ to], [$_{+D\ N}$ John], [$_{+D\ N}$ children], [$_{+D\ N}$ candies]}

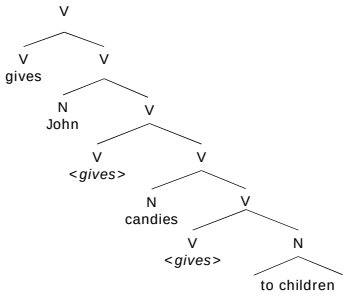
1. merge ([$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+D\ N}$ John])
2. merge ([$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+D\ N}$ candies])



PMGs SBO: Merge Right (Phillips 1996)

Lexicon: {[$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+K\ (N)}$ to], [$_{+D\ N}$ John], [$_{+D\ N}$ children], [$_{+D\ N}$ candies]}

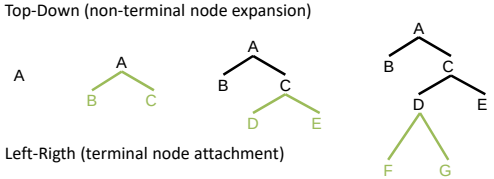
1. merge ([$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+D\ N}$ John])
2. merge ([$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+D\ N}$ candies])
3. merge ([$_{=DP\ =DP\ =PP\ V}$ gives], [$_{+K\ +D\ N}$ to children])



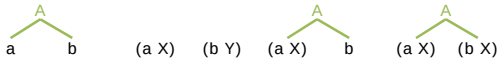
Proposal: Phase-based MGs

(PMGs, Chesi 2004-07)

- Constraints on Structure building operations
  - Top-Down (non-terminal node expansion)



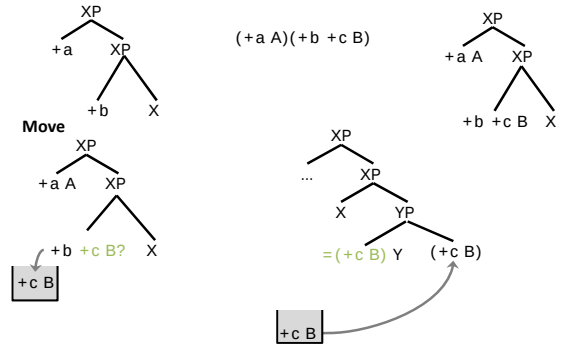
  - Left-Rigth (terminal node attachment)



Proposal: Phase-based MGs

(PMGs, Chesi 2004-07)

- Structure building operations:
  - **Insert** (Unification? (Shieber 1986))

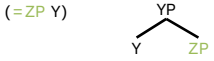


  - **Move**

Proposal: Phase-based MGs

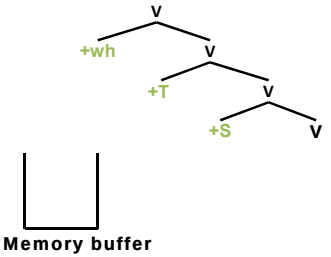
(PMGs, Chesi 2004-07)

- Structure building operations:
  - **Phase Projection**



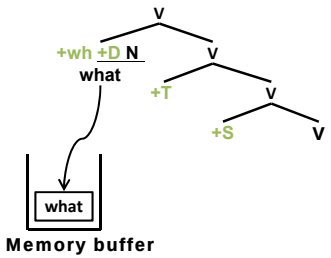
PMG: Sample derivation of wh-question

- *(default)* Expand(Lex: CP_{wh} = **(+wh +T +S V)**)



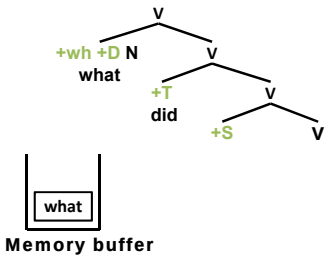
PMG: Sample derivation of wh-question

- *(default)* Expand(Lex: CP_{wh} = **(**+wh +T +S **V)**)
- Insert(Lex: (+wh +D N what))

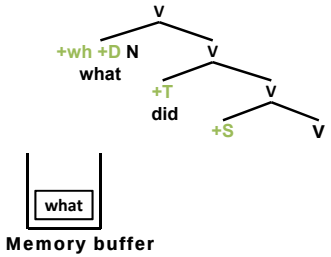


PMG: Sample derivation of wh-question

- Insert(Lex: (**+T** did))

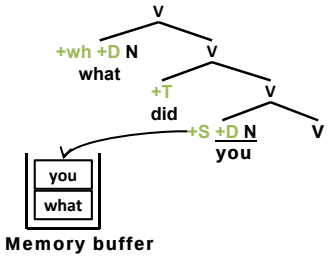


PMG: Sample derivation of wh-question



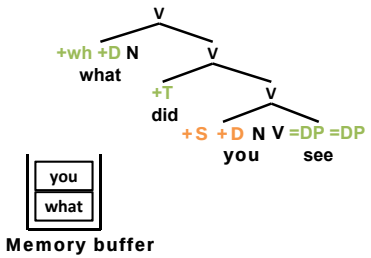
PMG: Sample derivation of wh-question

- Insert(Lex: (+S **+D N** you))



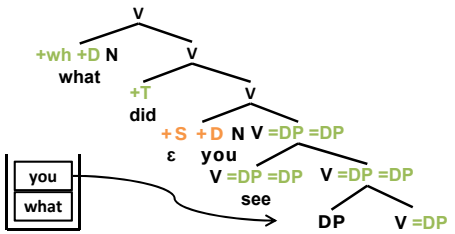
PMG: Sample derivation of wh-question

- Insert(Lex: (**V** =DP =DP see))



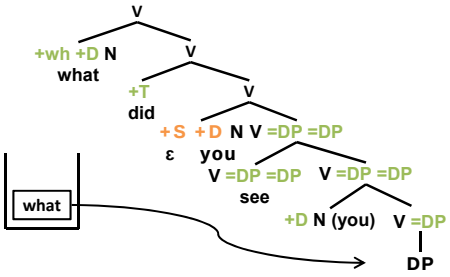
PMG: Sample derivation of wh-question

- Expand((V **=DP**))
- Move((+D N you))



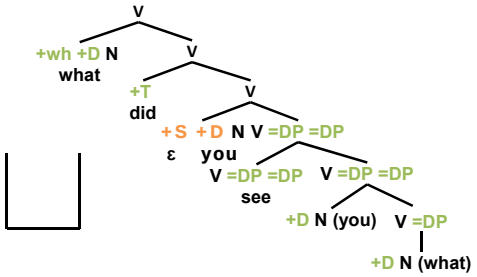
PMG: Sample derivation of wh-question

- Expand((V **=DP**))
- Move((+D N what))



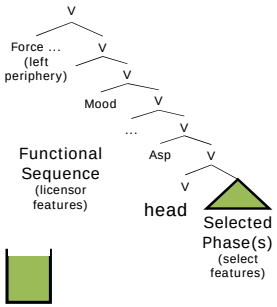
PMG: Sample derivation of wh-question

- Expand((V **=DP**))
- Move((+D N what))



Two more Structure Building Operations

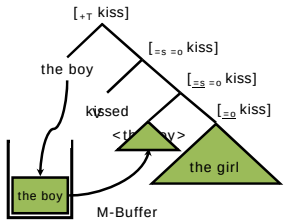
PHASE (PROJECTION)



MOVE

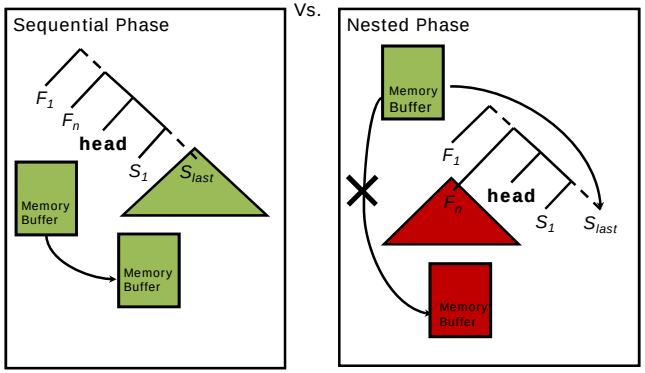
Linearization Principle (inspired by Kayne's LCA) if A immediately dominates B, then either
a. <A, B> if A selects B as an argument, or
b. <B, A> if B is in a functional specification of A

e.g. "the boy kissed the girl"

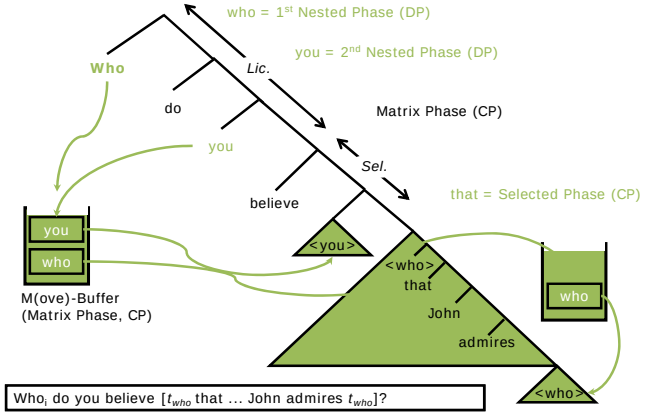


Success condition: M-buffer(s) must be empty at the end of the computation

"Internal" Merge

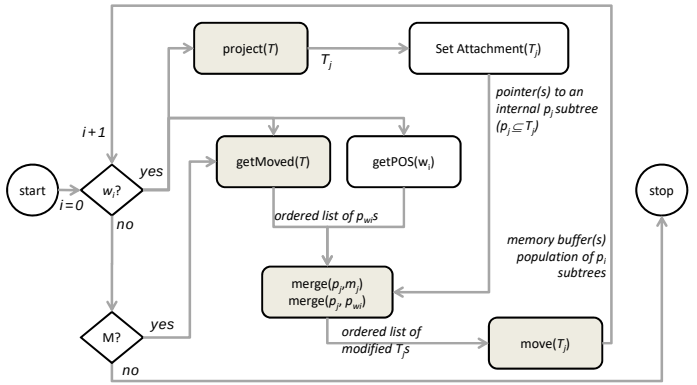


Successive Cyclic A'-movement



Who$_i$ do you believe [t_{who} that … John admires t_{who}]?

Parsing Algorithm with PMGs (PMG-pa)



Parsing States: CFG-Earley Vs. PMG-pa

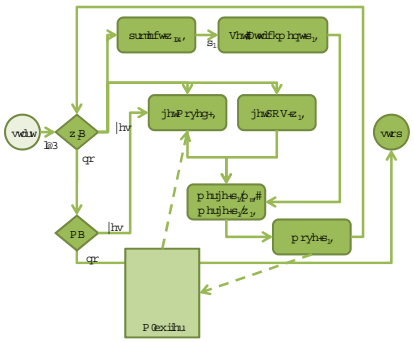
CFG-Earley	PMG-pa
Input position *(e.g. the man • runs)*	*Input position* *(the man • runs)*
Grammar rule (e.g. S → NP VP)	*Phase inspected* (e.g. Verbal)
Dot-position (e.g. S → NP • VP)	*Constituent completion status* (is the phase headed? **yes** are all thematic requirements satisfied? **yes** are non-thematic dependencies licensed? **yes** are non-thematic dependencies unique? **yes** further non-thematic dependencies available? **yes** status: **potentially complete**)
Leftmost edge of the substring this rule generates (e.g. **the** man runs)	*Constituent prefix* (e.g. **the man** runs)
	Memory buffer status (e.g. one N(ominal), determined, potentially complete phase)

Parsing Operations: CFG-Earley Vs. PMG-pa

CFG-Earley	PMG-pa
Predict top-down expansion of non-terminal rules in the grammar; Result: new rules (NP, VP, PP, AP…) added	Phase Projection new constituents insertion based on selection features in the lexicon; Result: Rooted Trees (RTrees) decorated with empty NP, VP or AP
Scan bottom-up inspection of the lexicon given a word; Result: PoS list to be integrated in the rules	Move / Lexical insertion unselected lexicalized sub-trees (moved LTrees) are available in this list, plus lexicalized sub-trees projected from a Top-Down inspection of the processed word-token; Result: ordered list (the pending list) of lexicalized sub-trees (LTrees) to be integrated in the structure
Complete top-down expansion of non-terminal rules in the grammar	Merge unification algorithm among pending rooted structures (RTrees) and lexicalized sub-trees in the pending list (LTrees)

Getting asymmetries with PMG-pa

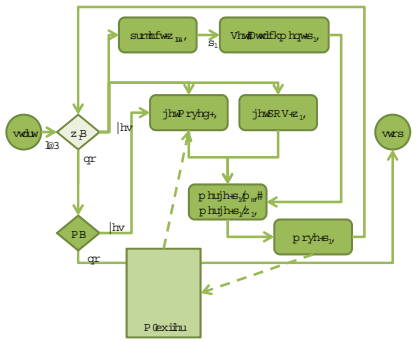
Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

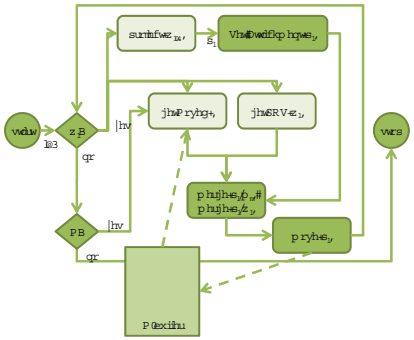
1. w_0 = "the"



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

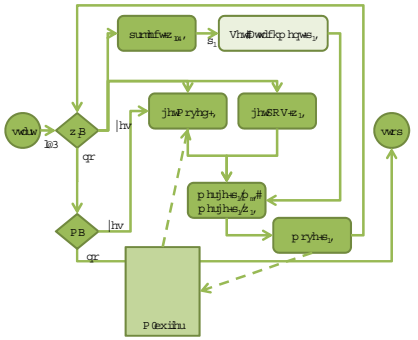
1. w_0 = "the"
2. project(*default*): [$_{VP}$], [$_{NP}$]
   getMoved(): nothing
   getPOS(*the*): [$_{NP\ +D}$ *the*]



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

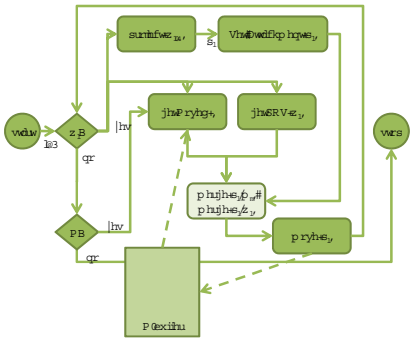
1. w_0 = "the"
2. project(*default*): [$_{VP}$], [$_{NP}$]
   getMoved(): nothing
   getPOS(*the*): [$_{NP\ +D}$ *the*]
3. setAttachment([$_{VP}$]): [$_{VP}$]
   setAttachment([$_{NP}$]): [$_{NP}$]



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

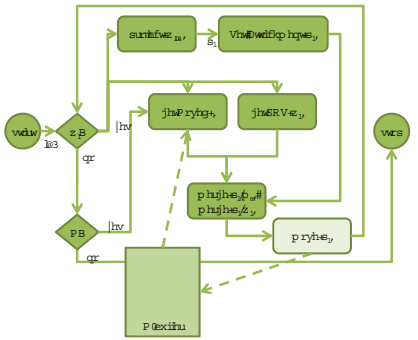
1. w_0 = "the"
2. project(*default*): [$_{VP}$], [$_{NP}$]
   getMoved(): nothing
   getPOS(*the*): [$_{NP\ +D}$ *the*]
3. setAttachment([$_{VP}$]): [$_{VP}$]
   setAttachment([$_{NP}$]): [$_{NP}$]
4. merge([$_{VP}$], [$_{NP\ +D}$ *the*]): [$_{VP}$ [$_{NP\ +D}$ *the*]]
   merge([$_{NP}$], [$_{NP\ +D}$ *the*]): [$_{NP\ +D}$ *the*]



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

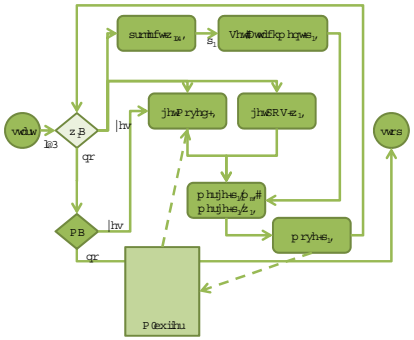
1. w_0 = "the"
2. project(*default*): [$_{VP}$], [$_{NP}$]
   getMoved(): nothing
   getPOS(*the*): [$_{NP\ +D}$ *the*]
3. setAttachment([$_{VP}$]): [$_{VP}$]
   setAttachment([$_{NP}$]): [$_{NP}$]
4. merge([$_{VP}$], [$_{NP\ +D}$ *the*]): [$_{VP}$ [$_{NP\ +D}$ *the*]]
   merge([$_{NP}$], [$_{NP\ +D}$ *the*]): [$_{NP\ +D}$ *the*]
5. move([$_{VP}$ [$_{NP\ +D}$ *the*]]): nothing
   move([$_{NP\ +D}$ *the*]): nothing



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

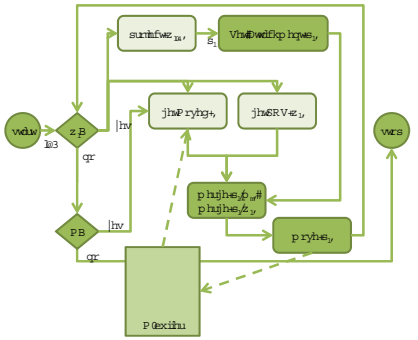
1. w_1 = "reporter"



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

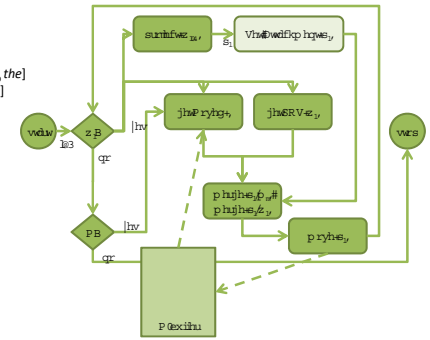
1. w_1 = "reporter"
2. project(*the*): nothing
3. getMoved(): nothing;
   getPOS(*reporter*): [$_{NP\ N}$ *reporter*]



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

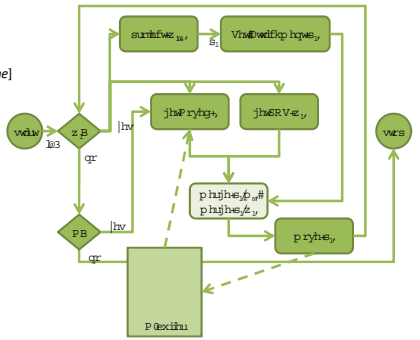
1. w_1 = "reporter"
2. project(*the*): nothing
3. getMoved(): nothing;
   getPOS(*reporter*): [$_{NP\ N}$ *reporter*]
4. setAttachment([$_{VP}$ [$_{NP\ +D}$ *the*]]): [$_{NP\ +D}$ *the*]
   setAttachment([$_{NP\ +D}$ *the*]): [$_{NP\ +D}$ *the*]



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

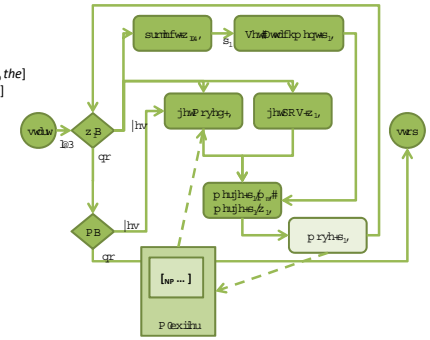
1. w_1 = "reporter"
2. project(*the*): nothing
3. getMoved(): nothing;
   getPOS(*reporter*): [$_{NP\ N}$ *reporter*]
4. setAttachment([$_{VP}$ [$_{NP\ +D}$ *the*]]): [$_{NP\ +D}$ *the*]
   setAttachment([$_{NP\ +D}$ *the*]): [$_{NP\ +D}$ *the*]
5. merge([$_{NP\ +D}$ *the*], [$_{NP\ N}$ *reporter*]):
   [$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter*]]
   merge([$_{NP\ +D}$ *the*], [$_{NP\ N}$ *reporter*]):
   [$_{NP\ +D}$ *the* $_{N}$ *reporter*]



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

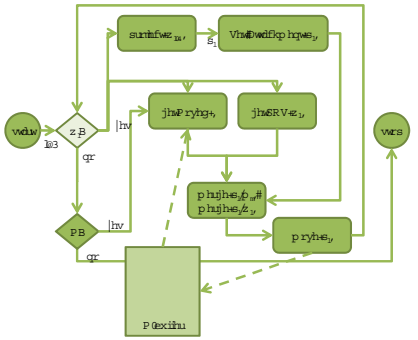
1. w_1 = "reporter"
2. project(*the*): nothing
3. getMoved(): nothing;
   getPOS(*reporter*): $[_{NP\ N}$ *reporter*]
4. setAttachment($[_{VP}\ [_{NP\ +D}$ *the*]]): $[_{NP\ +D}$ *the*]
   setAttachment($[_{NP\ +D}$ *the*]): $[_{NP\ +D}$ *the*]
5. merge($[_{NP\ +D}$ *the*], $[_{NP\ N}$ *reporter*]):
   $[_{VP}\ [_{NP\ +D}$ *the* $_{N}$ *reporter*]]
   merge($[_{NP\ +D}$ *the*], $[_{NP\ N}$ *reporter*]):
   $[_{NP\ +D}$ *the* $_{N}$ *reporter*]
6. move($[_{VP}\ [_{NP\ +D}$ *the* $_{N}$ *reporter*]]):
   A-Buffer[$[_{NP}]$]
   move($[_{NP\ +D}$ *the* $_{N}$ *reporter*]): $[_{NP}$ …]



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

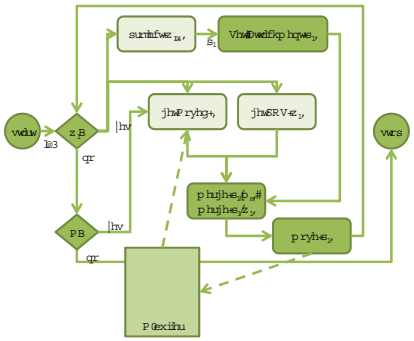
1. w_2= "who"



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

1. w_2= "who"
2. project(*who*): nothing
3. getMoved(): $[_{NP}]$;
   getPOS(*who*): $[_{VP\ +C}$ *who*]

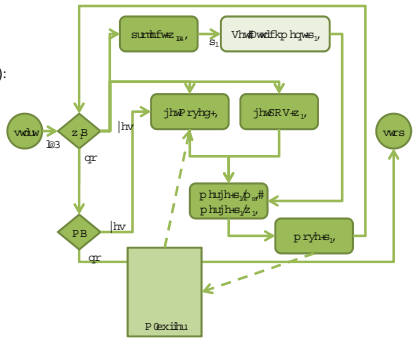


Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

1. w_2= "who"
2. project(*who*): nothing
3. getMoved(): $[_{NP}]$;
   getPOS(*who*): $[_{VP\ +C}$ *who*]
4. setAttachment($[_{VP}\ [_{NP\ +D}$ *the reporter*]]):
   $[_{NP}$ … reporter]

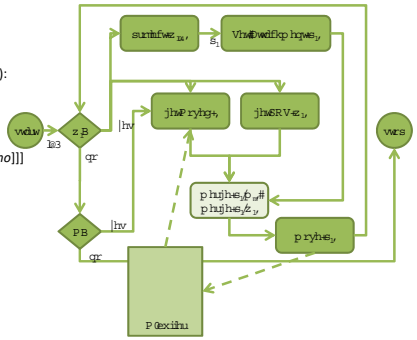
…



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

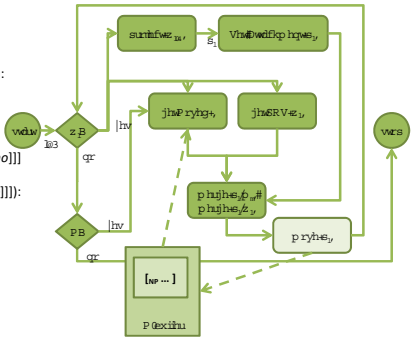
1. w_2 = "who"
2. project(*who*): nothing
3. getMoved(): [$_{NP}$];
   getPOS(*who*): [$_{VP\ +C}$ *who*]
4. setAttachment([$_{VP}$ [$_{NP\ +D}$ *the reporter*]]):
   [$_{NP}$ … reporter]
   …
5. merge([$_{NP}$ … reporter], [$_{NP}$]): failed
   merge([$_{NP}$ … reporter], [$_{VP\ +C}$ *who*]):
   [$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter* [$_{VP\ +C}$ *who*]]]
   …



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

1. w_2 = "who"
2. project(*who*): nothing
3. getMoved(): [$_{NP}$];
   getPOS(*who*): [$_{VP\ +C}$ *who*]
4. setAttachment([$_{VP}$ [$_{NP\ +D}$ *the reporter*]]):
   [$_{NP}$ … reporter]
   …
5. merge([$_{NP}$ … reporter], [$_{NP}$]): failed
   merge([$_{NP}$ … reporter], [$_{VP\ +C}$ *who*]):
   [$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter* [$_{VP\ +C}$ *who*]]]
   …
6. move([$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter* [$_{VP\ +C}$ *who*]]]):
   A-Buffer<[$_{NP}$]>



Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

1. project(*who*): nothing
2. getMoved(): [$_{NP}$];
   getPOS(*attacked*):
   [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]
3. setAttachment([$_{VP}$ [$_{NP}$ … *reporter* [$_{VP}$ *who*]]]): [$_{VP}$ *who*]
   …
4. merge([$_{VP\ +C}$ *who*], [$_{NP}$]): failed
   merge([$_{VP\ +C}$ *who*], [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]):
   [$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter* [$_{VP\ +C}$ *who*
   $_{+T\ V\ =NP\ =NP}$ *attacked*]]]
5. move([$_{VP\ +C}$ *who* $_{+T\ V\ =NP\ =NP}$ *attacked*]):
   A-Buffer<[$_{NP\ who}$]>

1. project(*who*): nothing
2. getMoved(): [$_{NP}$]; getPOS(*the*):
   [$_{NP\ +D}$ *the*]
3. setAttachment([$_{VP}$ [$_{NP}$ … *reporter* [$_{VP}$ *who*]]]):
   [$_{VP}$ *who*]
   …
4. merge([$_{VP\ +C}$ *who*], [$_{NP}$ who]): failed
   merge([$_{VP\ +C}$ *who*], [$_{NP\ +D}$ *the*]):
   [$_{VP\ +C}$ *who* [$_{NP\ +D}$ *the*]]
   …
5. move([$_{VP}$ [$_{NP}$ … [$_{VP\ +C}$ *who* [$_{NP\ +D}$ *the*]]]]):
   A-Buffer<[$_{NP}$ *who*]>

Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

1. project(*who*): nothing
2. getMoved(): [$_{NP}$];
   getPOS(*attacked*):
   [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]
3. setAttachment([$_{VP}$ [$_{NP}$ … *reporter* [$_{VP}$ *who*]]]): [$_{VP}$ *who*]
   …
4. merge([$_{VP\ +C}$ *who*], [$_{NP}$]): failed
   merge([$_{VP\ +C}$ *who*], [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]):
   [$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter* [$_{VP\ +C}$ *who*
   $_{+T\ V\ =NP\ =NP}$ *attacked*]]]
5. move([$_{VP\ +C}$ *who* $_{+T\ V\ =NP\ =NP}$ *attacked*]):
   A-Buffer<[$_{NP\ who}$]>

1. project(*the*): nothing
2. getMoved(): [$_{NP}$]; getPOS(*senator*):
   [$_{NP\ N}$ *senator*]
3. setAttachment([$_{VP}$ [$_{NP}$ … [$_{VP}$ *who* [$_{NP}$ *the*]]]]):
   [$_{NP}$ *the*]
   …
4. merge([$_{NP\ +D}$ *the*], [$_{NP\ N}$ *senator*]):
   [$_{VP\ +C}$ *who* [$_{NP\ +D}$ *the* $_{N}$ *senator*]]]
   …
5. move([$_{VP}$ [$_{NP}$ … [$_{VP}$ … [$_{NP}$ *the senator*]]]]):
   A-Buffer<[$_{NP}$ *who*], [$_{NP}$ *the senator*]>

Getting asymmetries with PMG-pa

Subject relatives in head initial languages
a. The reporter who attacked the senator admitted the error.
b. The reporter who the senator attacked admitted the error.

1. project(*who*): nothing
2. getMoved(): [$_{NP}$];
   getPOS(*attacked*):
   [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]
3. setAttachment([$_{VP}$ [$_{NP}$ … *reporter* [$_{VP}$ *who*]]]): [$_{VP}$ *who*]
   …
4. merge([$_{VP\ +C}$ *who*], [$_{NP}$]): failed
   merge([$_{VP\ +C}$ *who*], [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]):
   [$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter* [$_{VP\ +C}$ *who* $_{+T\ V\ =NP\ =NP}$ *attacked*]]]
5. move([$_{VP\ +C}$ *who* $_{+T\ V\ =NP\ =NP}$ *attacked*]):
   A-Buffer<[$_{NP}$ *who*]>

1. project(*senator*): nothing
2. getMoved(): <[$_{NP}$],[$_{NP}$]>; getPOS(*attacked*):
   [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]
3. setAttachment([[$_{VP}$ [$_{NP}$ … [$_{VP}$ who [$_{NP}$ …]]]]]):
   [$_{VP}$ *who* [$_{NP}$ …]] …
4. merge([$_{VP\ +C}$ *who* [$_{NP}$ …]], [$_{NP}$]): failed
   …
   merge([$_{VP}$ *who* [$_{NP}$ …]],
   [$_{VP\ +T\ V\ =NP\ =NP}$ *attacked*]):
   [$_{VP}$ [$_{NP\ +D}$ *the* $_{N}$ *reporter* [$_{VP\ +C}$ *who*
   [$_{NP\ +D}$ *the* $_{N}$ *senator*] $_{+T\ V\ =NP\ =NP}$ *attacked*]]]
5. move([$_{VP}$ [$_{NP}$ … [$_{VP}$ … [$_{NP}$ *the senator*]]]]):
   A-Buffer<[$_{NP}$ *who*], [$_{NP}$ *the senator*]>

Today's key concepts

- **On CFG-rules inefficiency:**
  - They are language specific
  - Too many rules are difficult to control
  - They can't be possibly learned (explanatory adequacy flaw)
- **Alternatives to CFG-rules:**
  - **Principle and Parameters** (Fong 1991, but principles are inefficient from the computational point of view)
  - **Minimalist Grammars** (Stabler 2007, but merge and move operate in opposition to the parsing direction; deductive parsing is not psycholinguistically plausible)
  - **Phase-based Minimalist Grammars** (Chesi 2004-15, top-down derivations are cognitively plausible and can Merge and Move can be implemented this way; locality can be captured too)