Grammar in Performance and Acquisition: recognition

E Stabler, UCLA

ENS Paris • 2008 • day 3

Q1 How are utterances interpreted 'incrementally'?

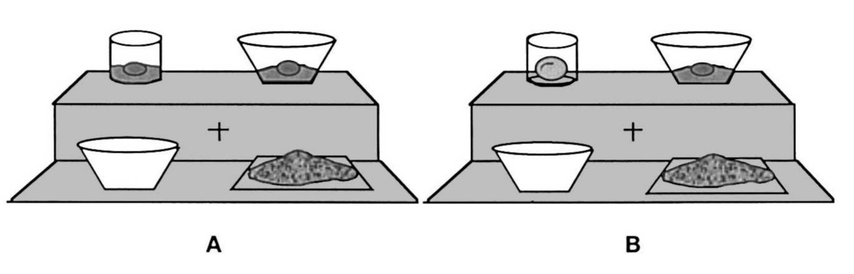
Q2 How is that ability acquired, from available evidence?

Q3 Why are some constituent orders unattested across languages?

Q4 What kind of grammar makes copying a natural option?

- we don't need to start from zero (start from grammar!)
- frame explanations supported by convergent evidence

(Chambers et al., 2004)



'Pour the egg in the bowl over the flour'

- Recognition: sequences $\rightarrow \{true, false\}$
- Parsing: sequences $\rightarrow$ Trees $\cup \{false\}$

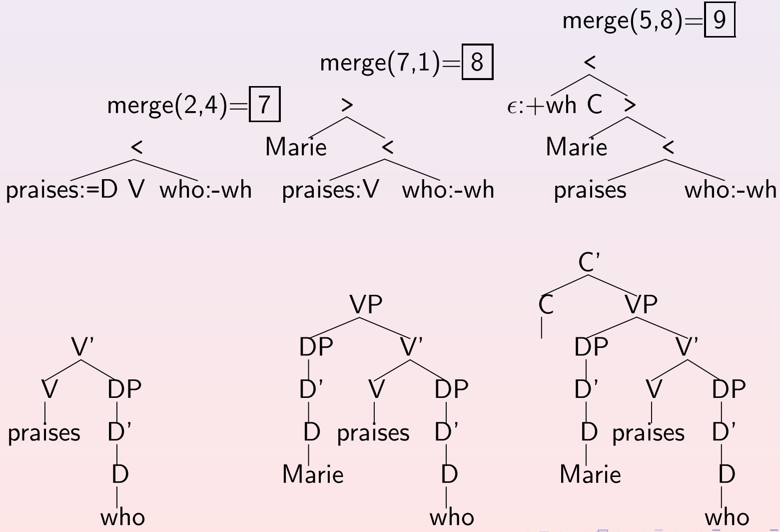
How to design a recognition/parsing strategy:

- Understand what you are parsing!
  - separate grammar definition from procedural issues
  - in parser, stay as close to grammar mechanisms as possible
  - consider time+memory after finding sound+complete algorithm
- Lessons from well-understood problems, esp. CFG parsing:
  - $\nexists$algorithm $\Rightarrow$ separate representations of each derivation
    (We can maybe exclude grammars with infinite ambiguity
    . . . But in human languages, as in CFLs, the number of derivations/string not bounded by polynomial)
  - Two main strategies (can both be used at once!):
    - Store all trees built, sharing structure ('chart','packed forest')
    - Carefully select steps that look like they're building the desired trees, backtracking and reanalyzing when necessary

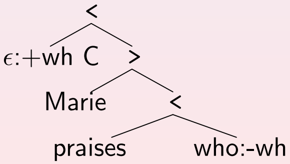
review

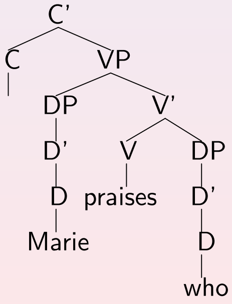
$\boxed{0}$	Pierre::D	who::D -wh	$\boxed{4}$
$\boxed{1}$	Marie::D	ϵ::=V +wh C	$\boxed{5}$
$\boxed{2}$	praises::=D =D V	and::=C =C C	$\boxed{6}$
$\boxed{3}$	ϵ::=V C		

review: steps 1,2,3

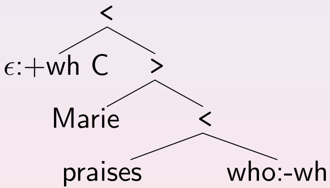


step 3: what do derived structures represent?

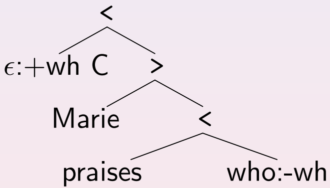




trees and labeled bracketing

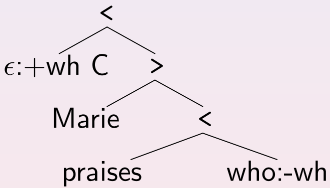


trees and labeled bracketing



$$[_{<}\ [\epsilon\text{:+wh C}]\ [_{>}[\text{Marie}]\ [_{<}[\text{praises}]\ [\text{who:-wh}]]]]$$

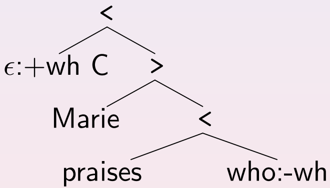
trees and labeled bracketing



$[_< [\epsilon\text{:+wh C}]\ [_>[\text{Marie}]\ [_<[\text{praises}]\ [\text{who:-wh}]]]]$

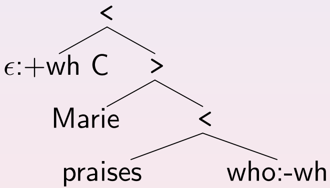
$[_< [\epsilon\text{:+wh C}]\ [_>\text{Marie}\ [_<\text{praises}\ (\text{who:-wh})]]]$

trees and labeled bracketing



$[_<$ $[\epsilon$:+wh C] $[_>$[Marie] $[_<$[praises] [who:-wh]]]]
$[_<$ $[\epsilon$:+wh C] $[_>$Marie $[_<$praises (who:-wh)]]]
$[_<$ $[\epsilon$:+wh C] $[_>$Marie praises (who:-wh)]]

trees and labeled bracketing



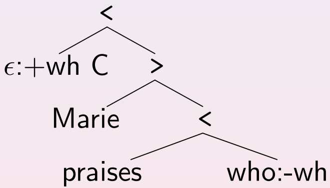
$[_< \; [\epsilon\text{:+wh C}] \; [_>[\text{Marie}] \; [_<[\text{praises}] \; [\text{who:-wh}]]]]$

$[_< \; [\epsilon\text{:+wh C}] \; [_>\text{Marie} \; [_<\text{praises (who:-wh)}]]]$

$[_< \; [\epsilon\text{:+wh C}] \; [_>\text{Marie praises (who:-wh)}]]$

$[_< \; (\epsilon\text{:+wh C}) \text{ Marie praises (who:-wh)}]$

trees and labeled bracketing



$[_<$ [ϵ:+wh C] $[_>$[Marie] $[_<$[praises] [who:-wh]]]]

$[_<$ [ϵ:+wh C] $[_>$Marie $[_<$praises (who:-wh)]]]

$[_<$ [ϵ:+wh C] $[_>$Marie praises (who:-wh)]]

$[_<$ (ϵ:+wh C) Marie praises (who:-wh)]

((Marie praises:+wh C),(who:-wh))

EM: two different cases

praises::=D =D V $+$ Pierre::D $\Rightarrow$

$<$

praises:=D V   Pierre

praises::=D =D V $+$ who::D -wh $\Rightarrow$

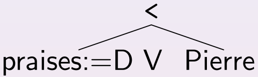
$<$

praises:=D V   who:-wh

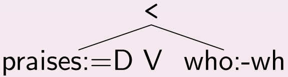
unlike *Pierre*, the DP *who* is a **mover**

so second result has 2 active ‘chains’

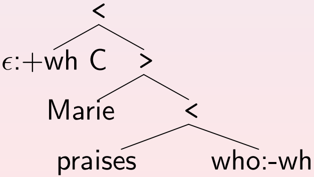
what the syntax must see



$\equiv$ praises Pierre:=D V



$\equiv$ praises:=D V, who:-wh



$\equiv$ Marie praises:+wh C, who:-wh

example:

$\boxed{1}$	`Marie::D`	`who::D -wh`	$\boxed{4}$
$\boxed{2}$	`praises::=D =D V`	ϵ`::=V +wh C`	$\boxed{5}$

example:

$\boxed{1}$	`Marie::D`	`who::D -wh`	$\boxed{4}$
$\boxed{2}$	`praises::=D =D V`	ϵ`::=V +wh C`	$\boxed{5}$

merge($\boxed{2}$,$\boxed{4}$)=praises:=D V, who:-wh $\boxed{A}$

example:

$\boxed{1}$	`Marie::D`	`who::D -wh`	$\boxed{4}$
$\boxed{2}$	`praises::=D =D V`	ϵ`::=V +wh C`	$\boxed{5}$

merge($\boxed{2}$,$\boxed{4}$)=praises:=D V, who:-wh $\boxed{A}$

merge($\boxed{A}$,$\boxed{1}$)=Marie praises:V, who:-wh $\boxed{B}$

example:

$\boxed{1}$	`Marie::D`	`who::D -wh`	$\boxed{4}$
$\boxed{2}$	`praises::=D =D V`	ϵ`::=V +wh C`	$\boxed{5}$

merge($\boxed{2}$,$\boxed{4}$)=praises:=D V, who:−wh $\boxed{A}$

merge($\boxed{A}$,$\boxed{1}$)=Marie praises:V, who:−wh $\boxed{B}$

merge($\boxed{5}$,$\boxed{B}$)=Marie praises:+wh C, who:−wh $\boxed{C}$

example:

$\boxed{1}$	`Marie::D`	`who::D -wh`	$\boxed{4}$
$\boxed{2}$	`praises::=D =D V`	ϵ`::=V +wh C`	$\boxed{5}$

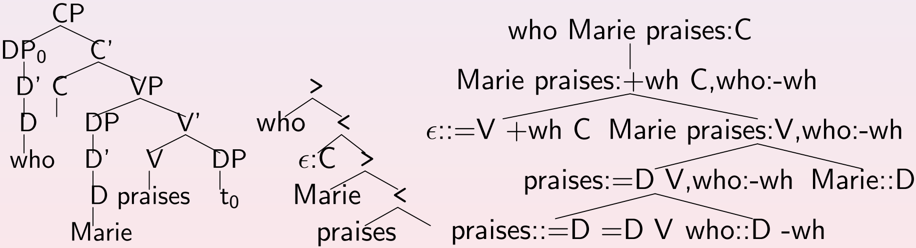
merge($\boxed{2}$,$\boxed{4}$)=praises:=D V, who:−wh $\boxed{A}$

merge($\boxed{A}$,$\boxed{1}$)=Marie praises:V, who:−wh $\boxed{B}$

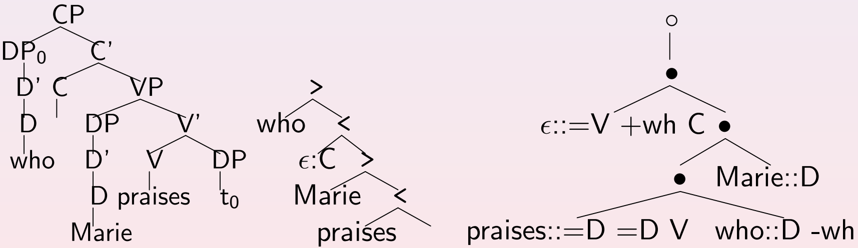
merge($\boxed{5}$,$\boxed{B}$)=Marie praises:+wh C, who:−wh $\boxed{C}$

move($\boxed{C}$)=who Marie praises:C

derived structures and derivation trees



derived structures and derivation trees



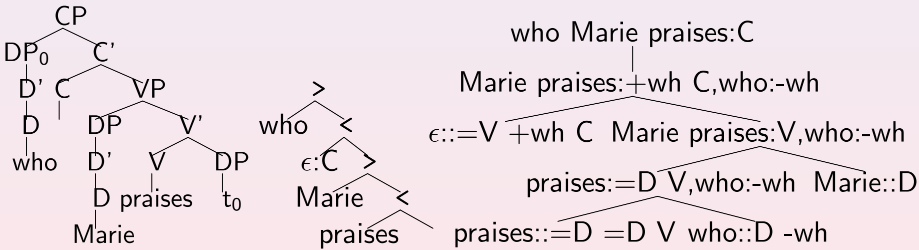
minimalist grammar G=$\langle Lex, \mathcal{F}\rangle$ reformulated

- **vocabulary** $\Sigma = \{\texttt{every},\texttt{some},\texttt{student},\ldots\}$
- **types** $T = \{\texttt{::},\ \texttt{:}\}$ "lexical" and "derived"
- **syntactic features** F:

`C, T, D, N, V, P,...`	(selected categories)
`=C, =T, =D, =N, =V, =P,...`	(selector features)
`+wh, +case, +focus,...`	(licensors)
`-wh, -case, -focus,...`	(licensees)

- **Chains** $C = \Sigma^* \times T \times F^*$
- **expressions** $E = C^+$
- **lexicon** $Lex \subseteq C^+$, a finite subset of $\Sigma^* \times \{::\} \times F^*$

derived structures and derivation trees



(remember why this reformulation is being considered now: we need to decompose derivations to share structure)

em: $(E \times E) \rightarrow E$ is the union of the following 3 functions,
for $\cdot \in \{:, ::\}$, $\gamma \in F^*$, $\delta \in F^+$

$$\frac{s :: =f\gamma \qquad t \cdot f, \alpha_1, \ldots, \alpha_k}{st : \gamma, \alpha_1, \ldots, \alpha_k}$$ em1: lexical item selects non-mover

$$\frac{s : =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f, \iota_1, \ldots, \iota_l}{ts : \gamma, \alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l}$$ em2: non-lex selects non-mover

$$\frac{s \cdot =f\gamma, \alpha_1, \ldots, \alpha_k \qquad t \cdot f\delta, \iota_1, \ldots, \iota_l}{s : \gamma, \alpha_1, \ldots, \alpha_k, t : \delta, \iota_1, \ldots, \iota_l}$$ em3: any item selects mover

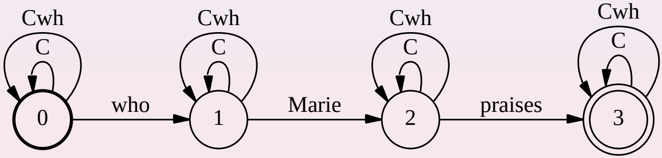
(Here, $\alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l$ $(0 \leq k, l)$ are any chains) *im*: $E \rightarrow E$ is the union of the following 2 functions, for $\gamma \in F^*$, $\delta \in F^+$,

$$\frac{s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \ldots, \alpha_k}{ts : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k} \text{ im1: final move}$$

$$\frac{s : +f\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f\delta, \alpha_{i+1}, \ldots, \alpha_k}{s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k} \text{ im2: nonfinal move}$$

(SMC) none of the chains $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k$ has $-f$ as its first feature,

Given a minimalist grammar G, how can we tell if string $s \in L(G)$?
0. Represent input by a finite state machine, with all possible empty elements:



(0,0)::=VC	(1,1)::=VC	(2,2)::=VC	(3,3)::=VC
(0,0)::=VC+wh	(1,1)::=VC+wh	(2,2)::=VC+wh	(3,3)::=VC+wh
(0,1)::who	(1,2)::Marie	(2,3)::praises	

1. Replace each lexical item by its features, in a matrix m(input).

(0,0)::=VC	(1,1)::=VC	(2,2)::=VC	(3,3)::=VC
(0,0)::=VC+wh	(1,1)::=VC+wh	(2,2)::=VC+wh	(3,3)::=VC+wh
(0,1)::D-wh	(1,2)::D	(2,3)::=D =D V	

2. Close m(input) with respect to merge,
where each string is given now by the matrix indices,
(enforcing adjacency reqs for em1,em2,im1; none for em3,im2)

3. <u>Success</u> if $(0, |\text{input}|) \cdot$ Start

With the 'deductive parsing' implementation of **Shieber, Schabes & Pereira (1994)**, only a small bit of code is needed, available from the webpage. The method is called CKY because it is based on early work of Cocke, Kasami, and Younger on parsing context free languages (Aho and Ullman, 1972; Sikkel and Nijholt, 1997).

Example:

1. (0,0)::=VC (1,1)::=VC (2,2)::=VC (3,3)::=VC
(0,0)::=VC+wh (1,1)::=VC+wh (2,2)::=VC+wh (3,3)::=VC+wh
(0,1)::D-wh (1,2)::D (2,3)::=D =D V

2. Now, close w.r.t merge. First, using em3:

$$\frac{(2,3)::=D\ =D\ V \qquad (0,1)::D\ -wh}{(2,3):=D\ V\ ,\ (0,1):-wh}$$

So we add this to the matrix. . .

Example: continuing 2. . .

(0,0)::=V C	(1,1)::=V C	(2,2)::=V C	(3,3)::=V C
(0,0)::=VC+wh	(1,1)::=VC+wh	(2,2)::=VC+wh	(3,3)::=VC+wh
(0,1)::D-wh	(1,2)::D	(2,3)::=D =D V	
		(2,3):=D V,(0,1):−wh	

Now we can use em2:

$$\frac{(2,3)\text{:=D V , }(0,1)\text{::−wh} \qquad (1,2)\text{::D}}{(1,3)\text{:V , }(0,1)\text{:−wh}}$$

We add this to the matrix. . .

Example: continuing 2. . .

(0,0)::=VC	(1,1)::=VC	(2,2)::=VC	(3,3)::=VC
(0,0)::=VC+wh	(1,1)::=VC+wh	(2,2)::=VC+wh	(3,3)::=VC+wh
(0,1)::D-wh	(1,2)::D	(2,3)::=D =D V	
		(2,3):=D V,(0,1):−wh	
	(1,3):V,(0,1):−wh		

Now we can use em1:

$$\frac{(1,1)\text{:=V C} \qquad (1,3)\text{:V,(0,1):−wh}}{(1,3)\text{:+wh C, (0,1):−wh}}$$

We add this to the matrix. . .

Example: continuing 2...

(0,0)::=VC	(1,1)::=VC	(2,2)::=VC	(3,3)::=VC
(0,0)::=VC+wh	(1,1)::=VC+wh	(2,2)::=VC+wh	(3,3)::=VC+wh
(0,1)::D-wh	(1,2)::D	(2,3)::=D =D V	
		(2,3):=D V,(0,1):-wh	
	(1,3):V,(0,1):-wh		
	(1,3):+wh C,(0,1):-wh		

Now we can use im1:

$$\frac{(1,3){:}{+}\text{wh C},\ (0,1){:}{-}\text{wh}}{(0,3){:}\text{C}}$$

We add this to the matrix...

Example: continuing 2. . .

(0,0)::=VC	(1,1)::=VC	(2,2)::=VC	(3,3)::=VC
(0,0)::=VC+wh	(1,1)::=VC+wh	(2,2)::=VC+wh	(3,3)::=VC+wh
(0,1)::D-wh	(1,2)::D	(2,3)::=D =D V	
		(2,3):=D V,(0,1):-wh	
	(1,3):V,(0,1):-wh		
	(1,3):+wh C,(0,1):-wh		
(0,3):C			

We can now answer:

Step 3. Does the table contain $(0,3) \cdot C$? **Yes**

$_0$ who $_1$ Marie $_2$ praises $_3$

	0	1	2	3
0	(=V C) (=V +wh C)	(D -wh)		
1		(=V C) (=V +wh C)	(D)	
2			(=V C) (=V +wh C)	(=D =D V)
3				(=V C) (=V +wh C)

(Look up how to compute closures in Cormen et al'92,§26.2, or other text on algorithms)

$_0$ who $_1$ Marie $_2$ praises $_3$

	0	1	2	3
0	(=V C) (=V +wh C)	(D -wh)		
1		(=V C) (=V +wh C)	(D)	
2			(=V C) (=V +wh C)	(=D =D V) (=D V,(0,1):-wh)
3				(=V C) (=V +wh C)

$_0$ who $_1$ Marie $_2$ praises $_3$

	0	1	2	3
0	(=V C) (=V +wh C)	(D -wh)		
1		(=V C) (=V +wh C)	(D)	(V,(0,1):-wh)
2			(=V C) (=V +wh C)	(=D =D V) (=D V,(0,1):-wh)
3				(=V C) (=V +wh C)

$_0$ who $_1$ Marie $_2$ praises $_3$

	0	1	2	3
0	(=V C) (=V +wh C)	(D -wh)		
1		(=V C) (=V +wh C)	(D)	(V,(0,1):-wh) (+wh C,(0,1):-wh)
2			(=V C) (=V +wh C)	(=D =D V) (=D V,(0,1):-wh)
3				(=V C) (=V +wh C)

$_0$ who $_1$ Marie $_2$ praises $_3$

	0	1	2	3
0	(=V C) (=V +wh C)	(D -wh)		(C)
1		(=V C) (=V +wh C)	(D)	(V,(0,1):-wh) (+wh C,(0,1):-wh)
2			(=V C) (=V +wh C)	(=D =D V) (=D V,(0,1):-wh)
3				(=V C) (=V +wh C)

(matrix guarantees a completed derivation, which we can now collect. . .)

Soundness, completeness, complexity:

- (sound) for every G, item derived only if licensed by G
- (complete) for every G, if licensed by G, item derived
- The number of possible entries in any cell is finitely bounded.
- No more than $\mathcal{O}(n^{4m+4})$ steps (Harkema'00), m a constant depending on the number of licensees in the grammar.
- Generalizes to copying (P-MCFG translation)
- Generalizes to arbitrary semi-rings (probabilities, weights)

Incremental?

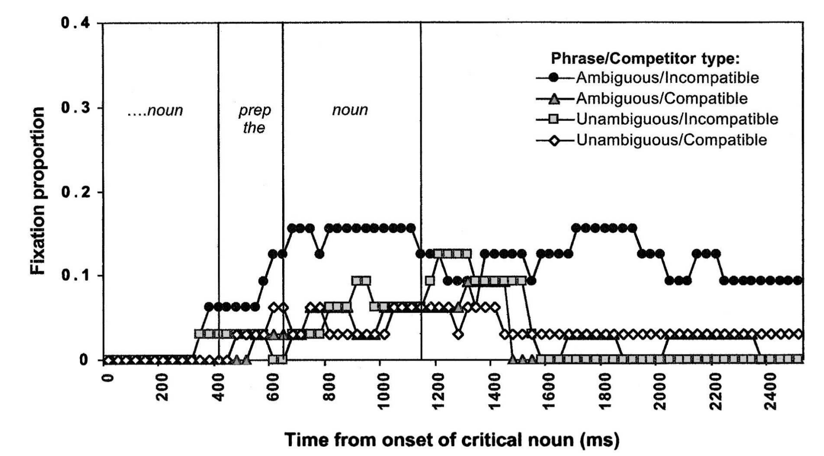
- This CKY-like method is bottom-up and all-paths-at-once.
- Given

  '(you) pour the egg in the bowl over the flour'

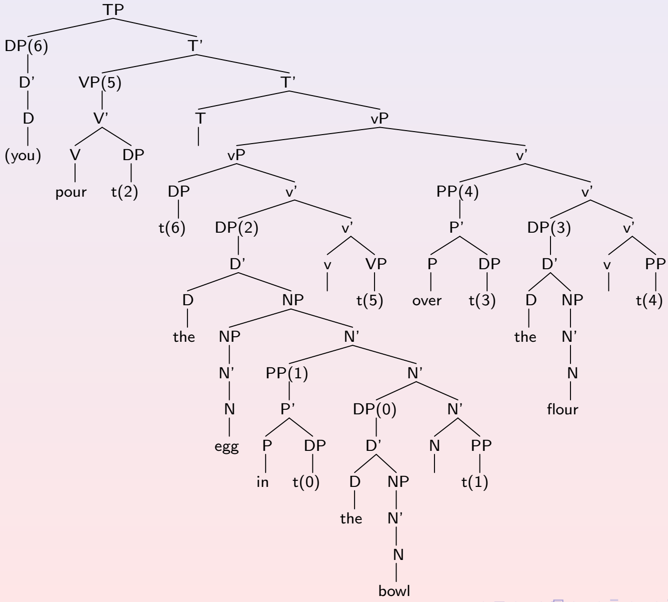
  when is egg or egg in the bowl related to the object position of pour?

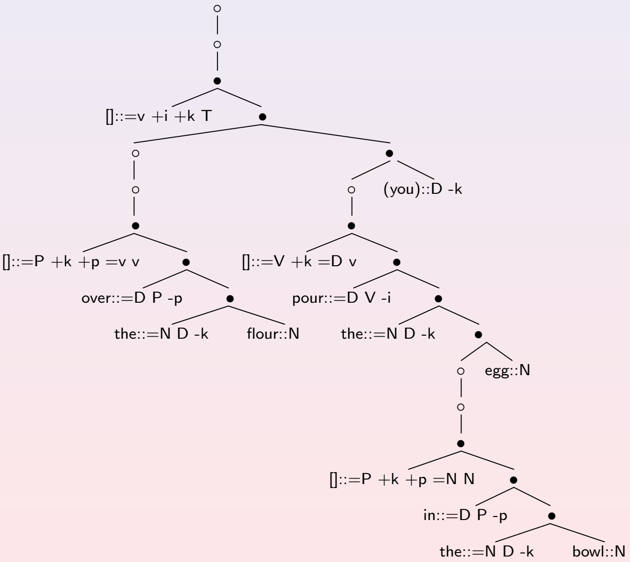
(NB: question not clear!)

'Pour the | egg | in the | bowl over the flour'

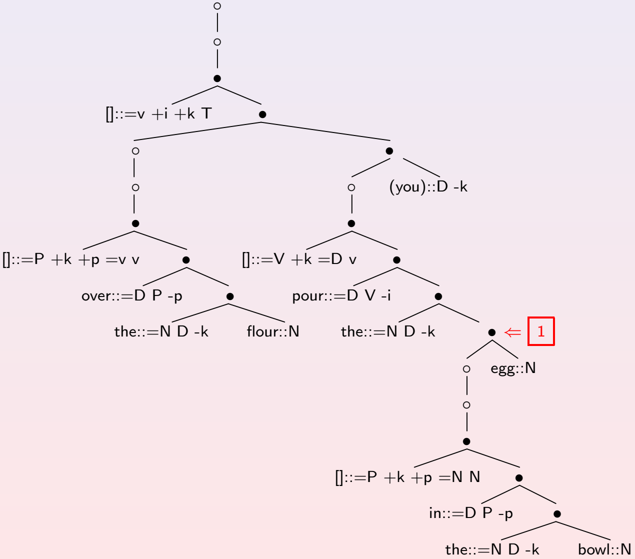


(Chambers et al., 2004)

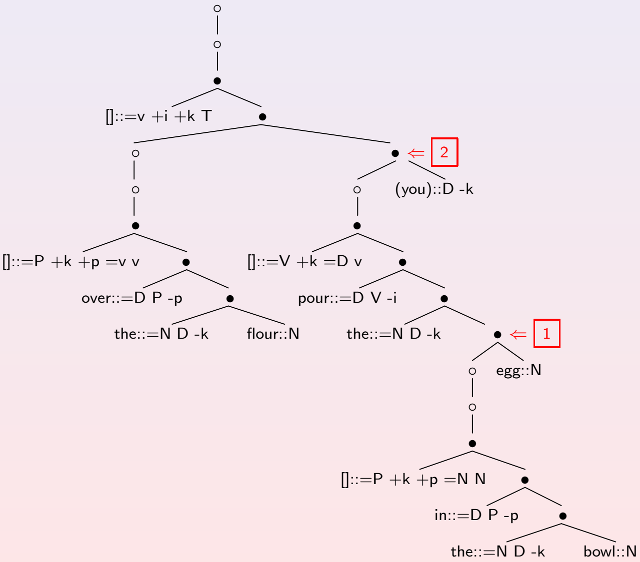

TP
DP(6)
T'
D'
VP(5)
T'
D
V'
T
vP
(you)
V
DP
vP
v'
pour
t(2)
DP
v'
PP(4)
v'
t(6)
DP(2)
v'
P'
DP(3)
v'
D'
v
VP
P
DP
D'
v
PP
D
NP
t(5)
over
t(3)
D
NP
t(4)
the
NP
N'
the
N'
N'
PP(1)
N'
N
N
P'
DP(0)
N'
flour
egg
P
DP
D'
N
PP
in
t(0)
D
NP
t(1)
the
N'
N
bowl


[]::=v +i +k T
[]::=P +k +p =v v
(you)::D -k
[]::=V +k =D v
over::=D P -p
pour::=D V -i
the::=N D -k
flour::N
the::=N D -k
egg::N
[]::=P +k +p =N N
in::=D P -p
the::=N D -k
bowl::N

x


[]::=v +i +k T
[]::=P +k +p =v v
(you)::D -k
[]::=V +k =D v
over::=D P -p
pour::=D V -i
the::=N D -k
flour::N
the::=N D -k
1
egg::N
[]::=P +k +p =N N
in::=D P -p
the::=N D -k
bowl::N

x


[]::=v +i +k T
2
(you)::D -k
[]::=P +k +p =v v
[]::=V +k =D v
over::=D P -p
pour::=D V -i
the::=N D -k
flour::N
the::=N D -k
1
egg::N
[]::=P +k +p =N N
in::=D P -p
the::=N D -k
bowl::N

TD Motivations:

- CKY method not incremental(?) Lacks ‘prefix property’:
  - A parser has the prefix property iff it halts on any prefix of the input that cannot be extended to a successful parse.
- Marcus’80 proposes a bottom-up method to minimize local ambiguity. He proves that even then, and even with lookahead, backtracking cannot be avoided. He shows,

  English is not LR(k) for any k.
- Recently, a different idea is to build structures that are fully connected, so that they can be interpreted incrementally.
- Top-down (TD) parsing builds fully connected trees at every point. (+proposed to explain grammatical facts, Chesi et al)

TD Motivations:

- CKY method not incremental(?) Lacks 'prefix property':
  - A parser has the prefix property iff it halts on any prefix of the input that cannot be extended to a successful parse.
- Marcus'80 proposes a bottom-up method to minimize local ambiguity. He proves that even then, and even with lookahead, backtracking cannot be avoided. He shows,

  English is not LR(k) for any k.
- Recently, a different idea is to build structures that are fully connected, so that they can be interpreted incrementally.
- Top-down (TD) parsing builds fully connected trees at every point. (+proposed to explain grammatical facts, Chesi et al) For CFGs, TD is **non-terminating**, and even when terminating is **intractable**. TD for MGs has these same problems **and more**. . . (discussed in Harkema'01)

Earley: A TD recognition strategy that works

- Earley'68 showed how TD and BU methods can be combined to avoid TD nontermination. The basic idea is simply:
  - Constituents are predicted TD, using chart representation
  - Predicted elements are completed BU, and then new predictions are generated TD
- Method extends to TAGs and MGs, with 'prefix property' (for MGs, see Harkema'01; for TAGs, Vijay-Shanker'87)
- For incremental parsing, one idea is: semantically analyze one TD prediction path from the chart at a time, while completing BU and storing completed elements as required by Earley, so that reanalysis is feasible.

So far 3

- simple formalisms can model many linguistic proposals
- a straightforward semantics values every constituent

Q1 What performance models allow incremental interpretation (and remnant movement, doubling constructions)?

- CKY efficiently parses every MGC
- Earley efficiently parses every MGC,
  with (factored) representation of TD derivations
- Probabilistic Earley may model TD choice
- Can we interpret TD partial constituents like [DP [V. . .]]?
  Yes, but many open questions!
  (Hale'08,Shieber&Johnson'94,Stabler'91,Steedman'89)

Q2 How is this ability acquired, from available evidence?

Aho, Alfred V. and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling. Volume 1: Parsing*. Prentice-Hall, Englewood Cliffs, New Jersey.

Becker, Tilman, Owen Rambow, and Michael Niv. 1992. The derivational generative power of formal systems, or, scrambling is beyond LCFRS. IRCS technical report 92-38, University of Pennsylvania.

Chambers, Craig G., Michael K. Tanenhaus, Kathleen M. Eberhard, Hana Filip, and Greg N. Carlson. 2004. Actions and affordances in syntactic ambiguity resolution. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 30(3):687–696.

Chesi, Cristiano. 2007a. Five reasons for building phrase structures top-down from left to right. *Nanzan Linguistics, Special Issue 3,*, 1:71–105.

Chesi, Cristiano. 2007b. An introduction to phase-based minimalist grammars: Why *move* is top-down from left-to-right. Technical report, Centro Interdepartmentale di Studi Cognitivi sul Linguaggio.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1991. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts.

Goodman, Joshua. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Hale, John. 2003. *Grammar, Uncertainty, and Sentence Processing*. Ph.D. thesis, Johns Hopkins University.

Hale, John. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30(1):609–642.

Harkema, Henk. 2000. A recognizer for minimalist grammars. In *Sixth International Workshop on Parsing Technologies, IWPT'00*.

Kuich, Werner and Arto Salomaa. 1986. *Semirings, Automata, Languages*. Springer-Verlag, NY.

Marcus, Mitchell. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, Massachusetts.

Nederhof, Mark-Jan and Giorgio Satta. 2003. Probabilistic parsing strategies. In *Proceedings of the 3rd AMAST Workshop on Algebraic Methods in Language Processing (AMiLP 2003)*, pages 305–314, Verona, Italy.

Rambow, Owen and Giorgio Satta. 1994. A two-dimensional hierarchy for parallel rewriting systems. IRCS technical report 94-02, University of Pennsylvania.

Satta, Giorgio. 1994. Tree adjoining grammar parsing and boolean matrix multiplication. *Computational Linguistics*, 20:173–232.

Shieber, Stuart and Mark Johnson. 1994. Variations on incremental interpretation. *Journal of Psycholinguistic Research*, 22:287–318.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1993. Principles and implementation of deductive parsing. Technical Report CRCT TR-11-94, Computer Science Department, Harvard University, Cambridge, Massachusetts.

Sikkel, Klaas and Anton Nijholt. 1997. Parsing of context free languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 2: Linear Modeling*. Springer, NY, pages 61–100.

Stabler, Edward P. 1991. Avoid the pedestrian's paradox. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-based Parsing: Computation and Psycholinguistics*. Kluwer, Boston, pages 199–238.

Steedman, Mark J. 1989. Grammar, interpretation, and processing from the lexicon. In William Marslen-Wilson, editor, *Lexical Representation and Process*. MIT Press, Cambridge, Massachusetts, pages 463–504.

Valiant, Leslie G. 1975. General context free recognition in less than cubic time. *Journal of Computer and System Sciences*, 10:308–315.

Vijay-Shanker, K. and David Weir. 1994. Parsing some constrained grammar formalisms. *Computational Linguistics*, 15:591–636.

Vijayashanker, K. 1987. *A Study of Tree Adjoining Languages*. Ph.D. thesis, University of Pennsylvania.