

An Introduction to Tree Adjoining Grammars

GUO, Yuqing
NCLT, DCU
15 Nov. 2006





- Introduction
 - Basic Mechanisms
 - Properties
 - Variants

- Applications
 - TAG Resource Induction
 - Parsing with TAG



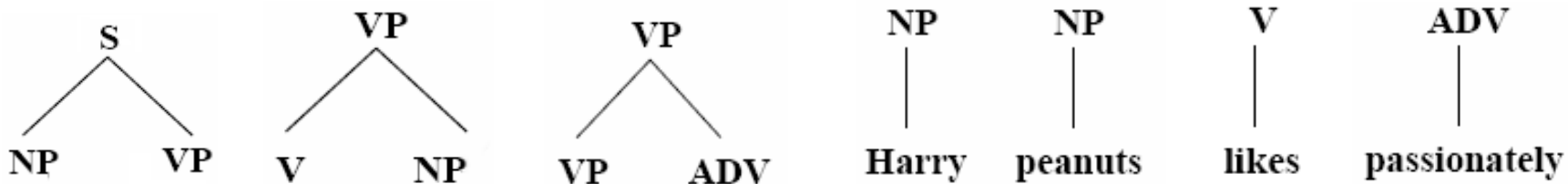
- Tree Adjoining Grammar
 - Introduced in [Joshi & Takahashi, 1975]
 - A formal **tree** rewriting system
- Basics of TAG Formalism
 - Primitive elements: elementary trees
 - Initial trees
 - Auxiliary trees
 - Operations
 - Substitution
 - Adjoining
 - Derivations
 - Derived trees
 - Derivation trees





- One level tree corresponding to a rule
- Not every rule is lexicalized

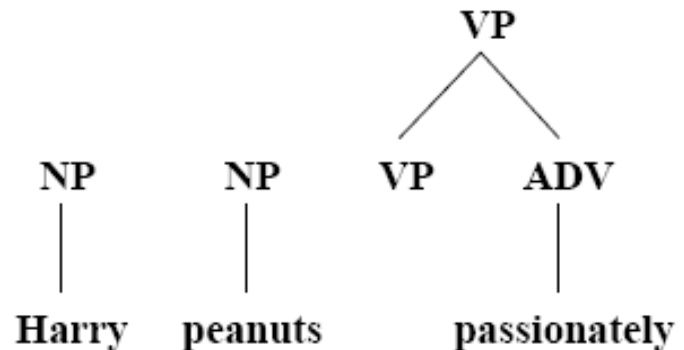
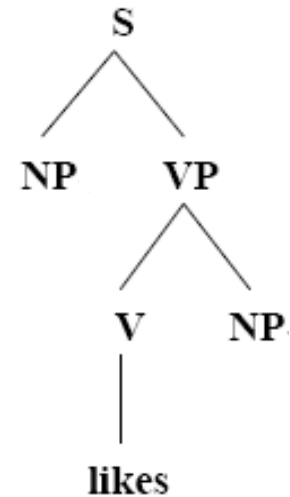
<i>Syntactic Rules</i>	<i>Lexical Rules</i>
1. $S \rightarrow NP VP$	4. $NP \rightarrow \text{Harry}$
2. $VP \rightarrow VP ADV$	5. $NP \rightarrow \text{peanuts}$
3. $VP \rightarrow V NP$	6. $V \rightarrow \text{likes}$
$S \rightarrow NP V NP$	7. $ADV \rightarrow \text{passionately}$



- Is a grammar capable of
 - Lexicalization of each elementary domain
 - Encapsulation of the arguments of the lexical anchor



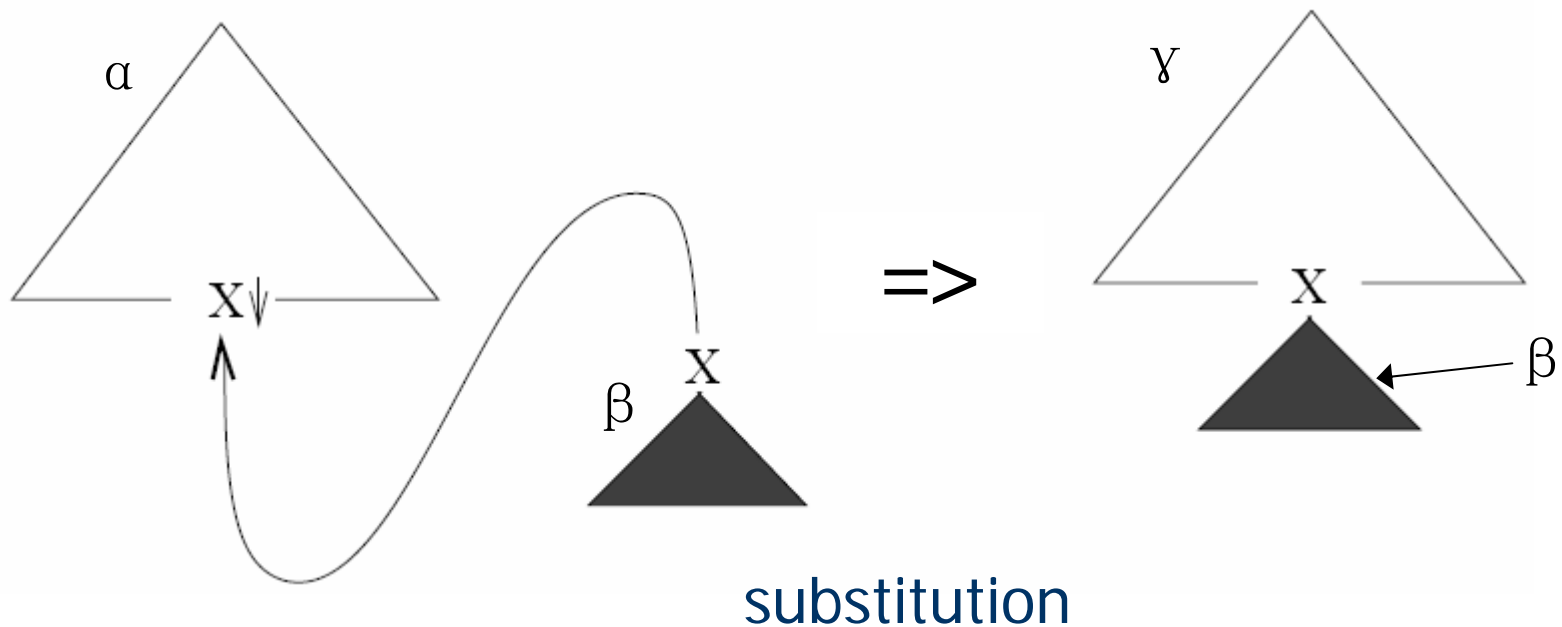
- Elementary objects are elementary trees
- Each tree contains at least one frontier node labeled with a terminal symbol (lexical anchor)





- An initial tree

- all interior nodes are labelled with non-terminal symbols
- the nodes on the frontier are either labelled with terminal symbols, or with non-terminal symbols marked for substitution (\downarrow)





- Are CFG, G and TSG, G' strongly equivalent?
Not only the string languages but also the sets of structure descriptions are the same.

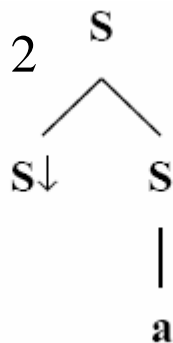
CFG G: $S \rightarrow S S$ (non-lexical)
 $S \rightarrow a$ (lexical)

TSG G':

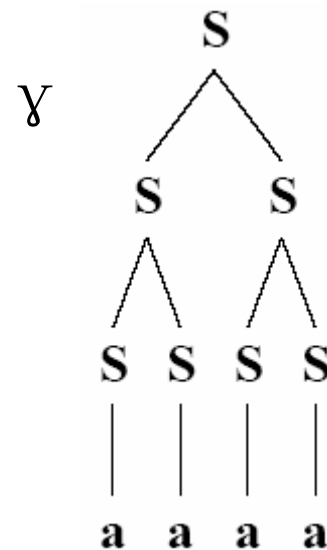
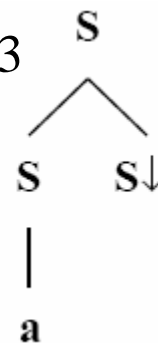
α_1



α_2



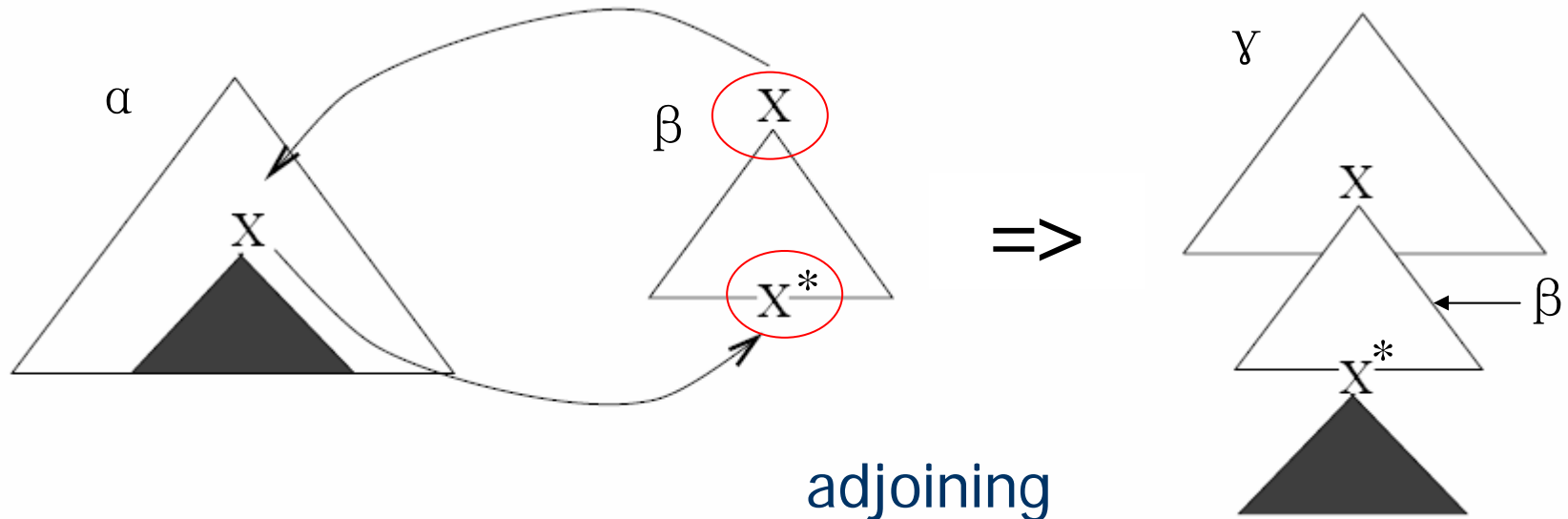
α_3

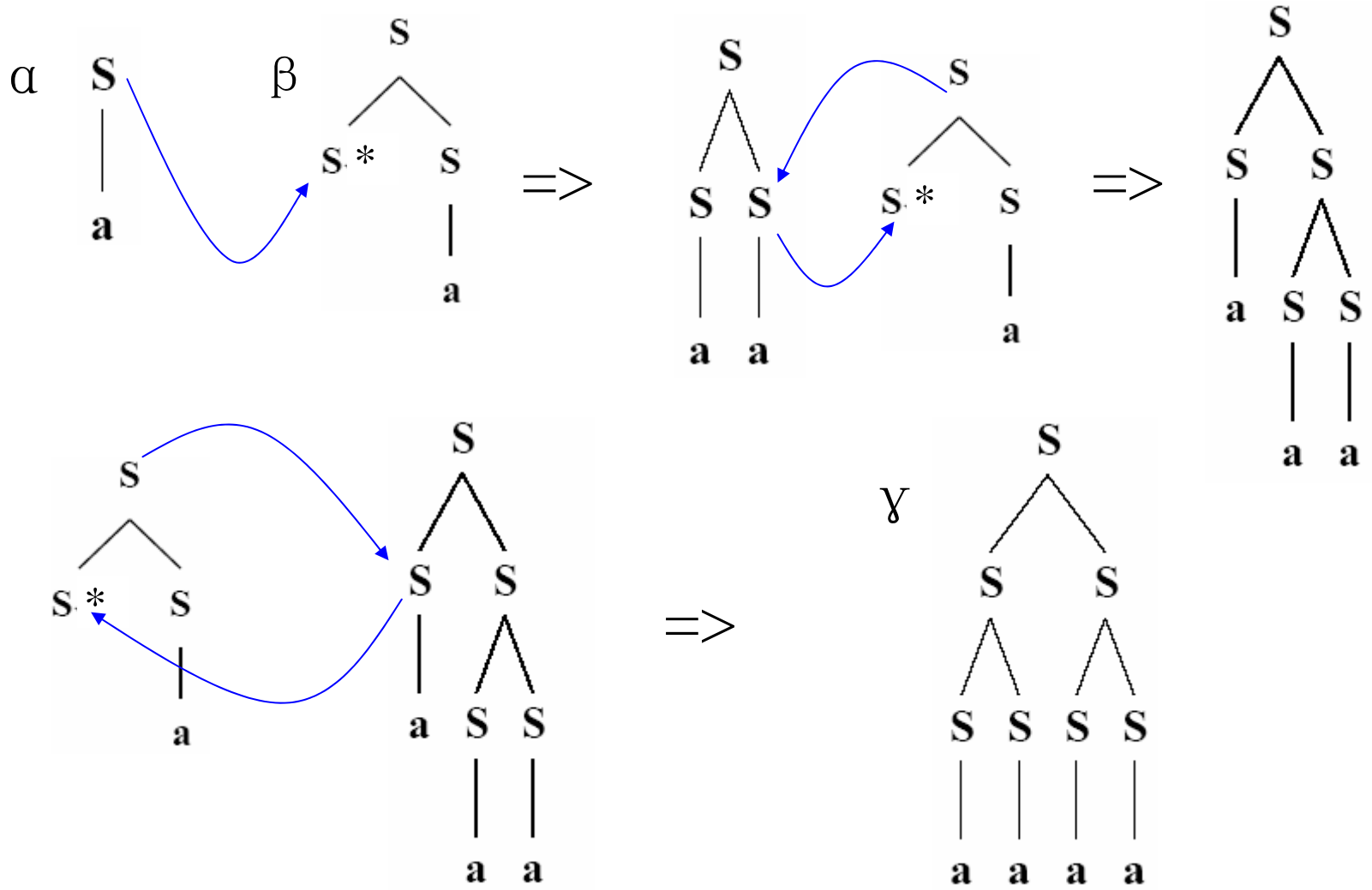


Can't be derived in TSG

- An auxiliary tree

- one of its frontier nodes must be marked as foot node (*)
- the foot node must be labeled with a non-terminal symbol which is identical to the label of the root node.

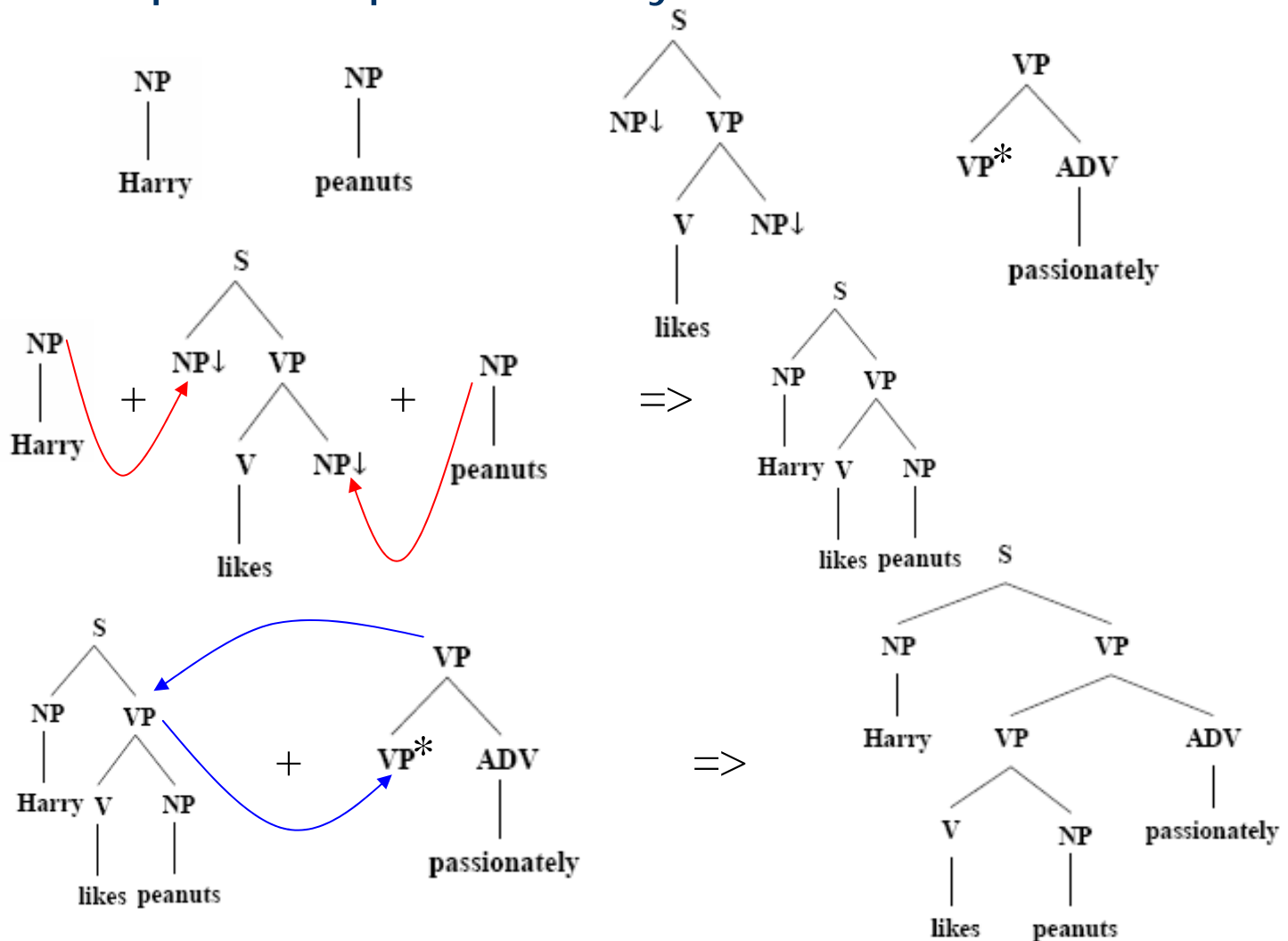






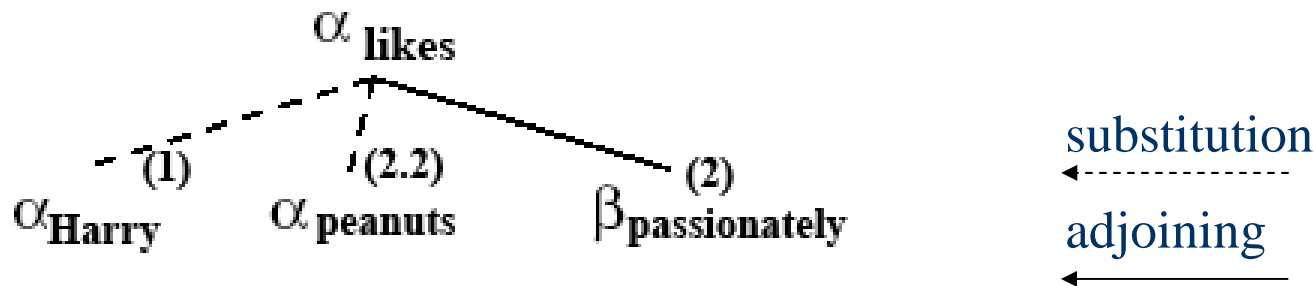
e.g. 'Harry likes peanuts passionately'

TAG G





- A derivation tree -- Specifies how a derived tree was constructed
 - The root node is labeled by an S-type initial tree
 - Other nodes are labeled by auxiliary trees in the case of adjoining or initial trees in the case of substitution
 - A tree address of the parent tree is associated with each node (Gorn address is used as tree address: the root of a tree has address 0, and the jth child of the node with address i has address i.j)

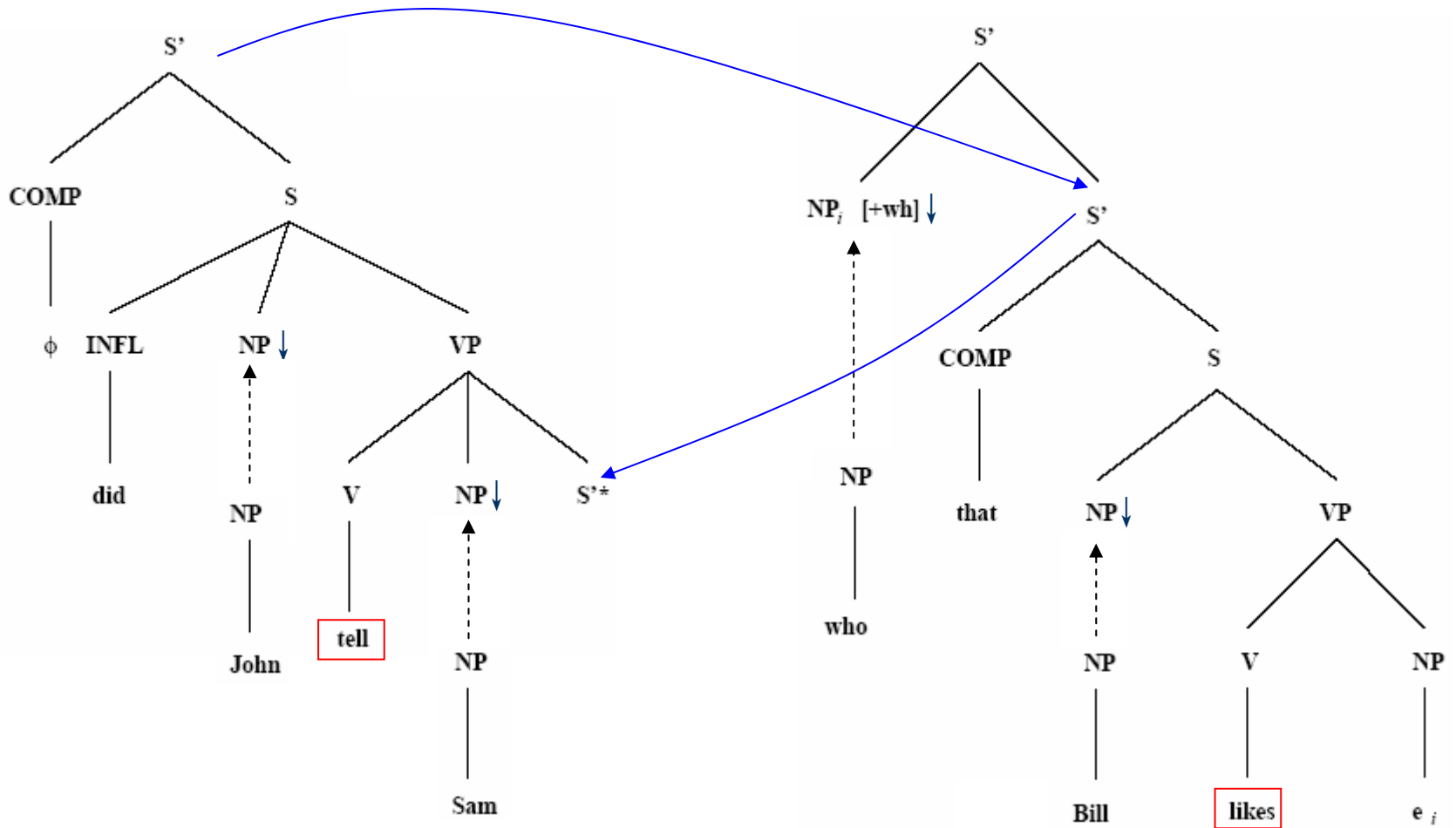


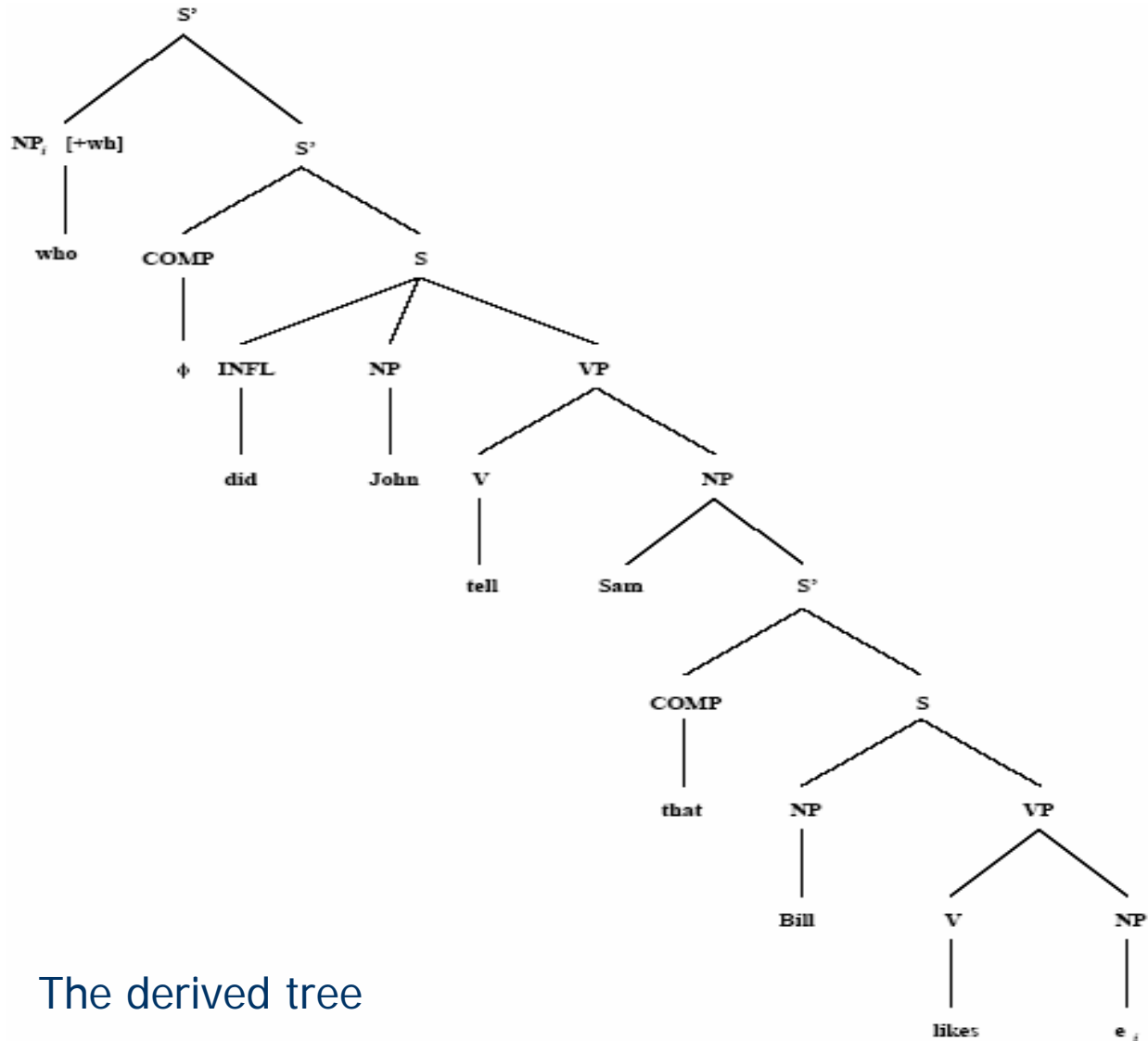


- LFG
 - Stated on f-structures
 - Resolved by functional uncertainties
 - ↑ TOPIC = ↑ COMP*OBJ
- TAG [Joshi & Vijay-Shanker, 1998]
 - Localized in the elementary trees
 - Functional uncertainty is *not* necessary



e.g. 'Who_i did John tell Sam that Bill likes e_i'





The derived tree



- Definition of TAG G
 - I : a finite set of initial trees
 - A : a finite set of auxiliary trees
 - $T(G)$: the set of all derived trees starting from S-type initial trees in I whose frontier consists of terminal nodes
 - $L(G)$: the set of all terminal strings on the frontier of the trees in $T(G)$
- Correspondence to embedded pushdown automata (EPDA)

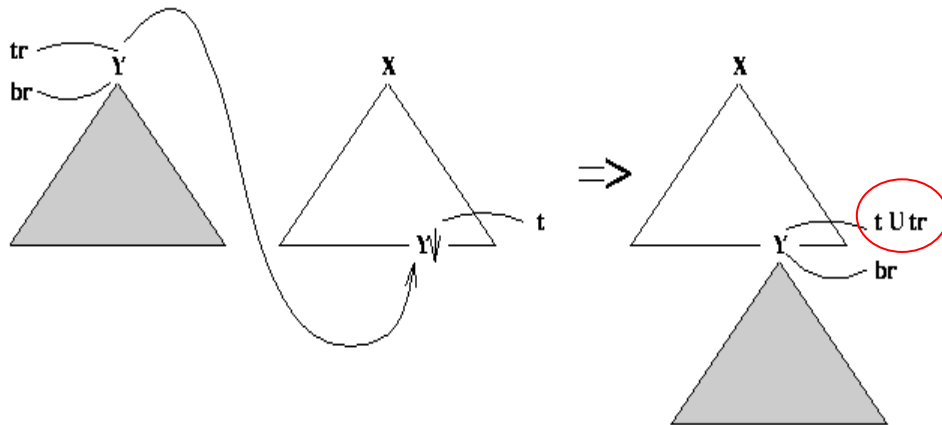


- Important Properties of TAG
 - An extended domain of locality
 - Factoring recursion from the domain of dependencies
- Localize the dependencies
- Mildly context-sensitive: $CFL \subset TAL \subset CSL$
 - strong generative capacity
 - efficiently parsable (in polynomial time)

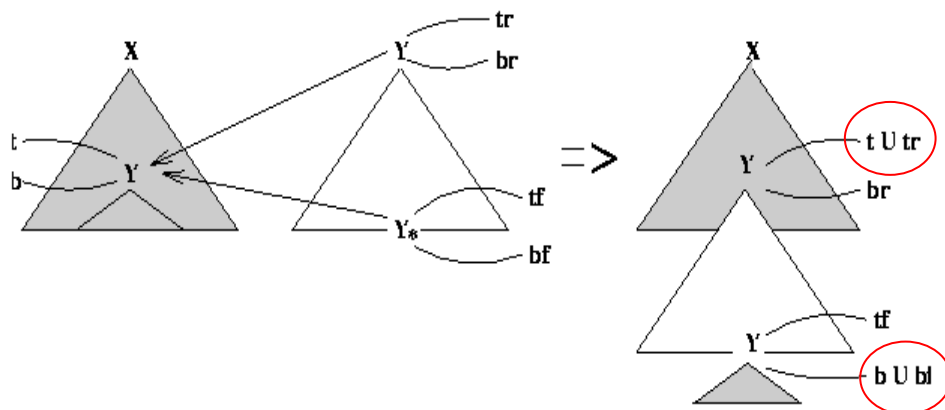


- Feature Structure Based TAG (FTAG)

each of the nodes of an elementary tree is associated with two feature structures: top & bottom



Substitution with features



Adjoining with features



- Synchronous TAG
 - A pair of TAGs characterize correspondences between languages
 - Semantic interpretation, language generation and translation
- Multi-component TAG (MCTAG)
 - A set of auxiliary tree can be adjoined to a given elementary tree
- Probabilistic TAG (PTAG)
 - Associating a probability with each elementary tree
 - Compute the probability of a derivation



- Lexical Functional Grammar
 - Translate LFG to LTAG [Kameyama, 1984]
 - Replace c-structure by LTAG structures
 - Map them into LTAG-like f-structures
- Head-Driven Phrase Structure Grammar
 - Compile HPSG to LTAG [Kasper, 1995]
- Combinatory Categorical Grammar
 - Encapsulate the arguments in categorial grammar framework
 - Construct a categorial system with LTAG-like properties [Joshi etc, 1997 & 1999]



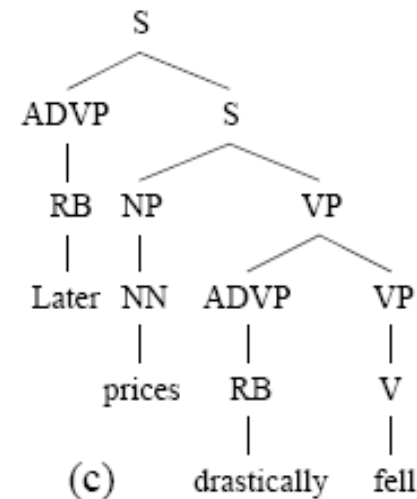
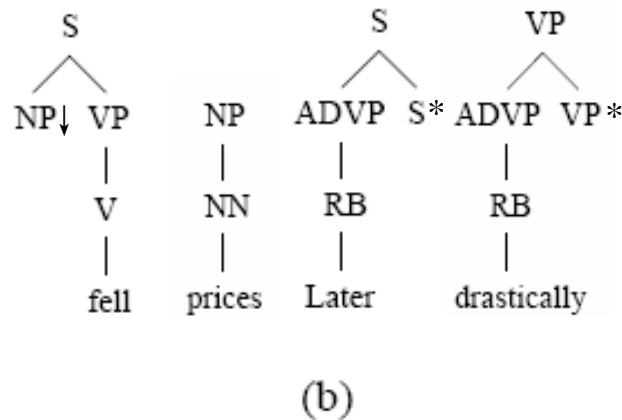
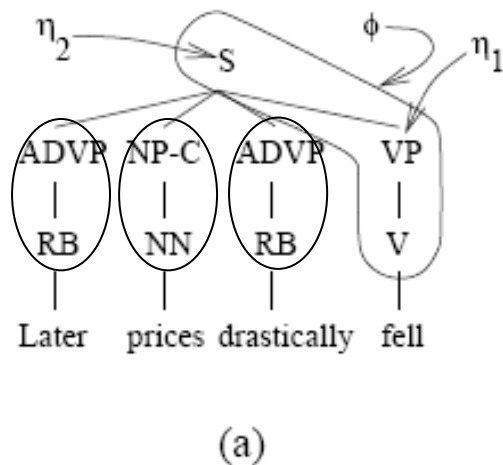
- The XTAG Project

(<http://www.cis.upenn.edu/~xtag>)

- An on-going project to develop a wide-coverage grammar for English using the LTAG formalism.
- An system for the development of TAGs and consists of a parser, an X-windows grammar development interface and a morphological analyzer.
- Contains diverse linguistic resources
 - Subcategorization information
 - Syntax of various constructions
 - Frequency information
 - Morphological information

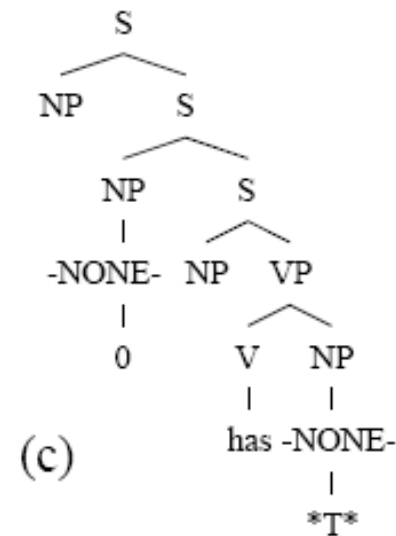
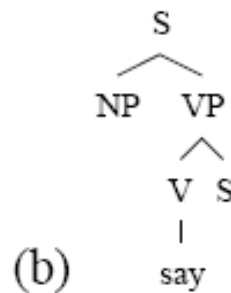
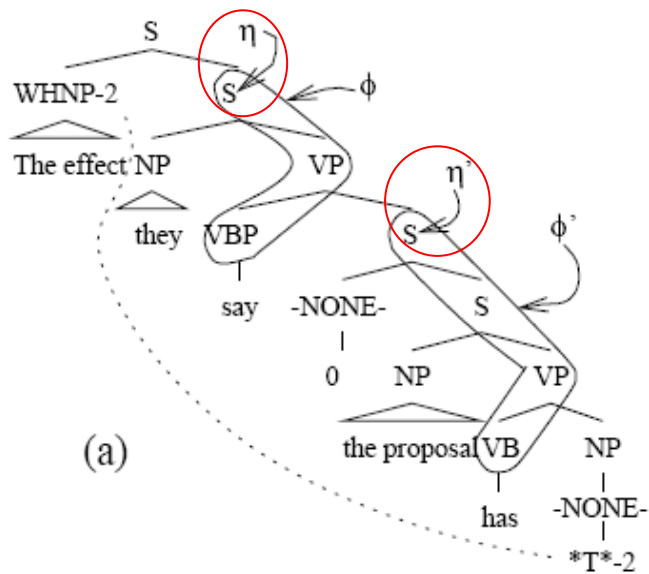


- [Chen, 2000]
 - Find headword for each local tree, the path through the nodes of headword is a trunk
 - Classify the sibling nodes of the headword as complements (arguments) or adjuncts
 - Excise at a complement nodes to form initial trees, leaving behind a substitution node
 - Adjuncts are factored into modifier auxiliary trees





- If node n has a right corner node n' which is an complement with the same label as n , excise the subtree from n down to n' to form a predicate auxiliary trees



- Special strategies for coordinations, punctuations and empty elements etc.



- Data
 - Extract from sections 2-21 of the Penn Treebank
 - Test on section 22
- Different Grammars
 - CA1: use labels and functional tags of the current node and the parent node for argument determination
 - CA2: use labels and functional tags of the current node and the head node and the distance for argument determination
 - ALL: include all empty elements
 - SOME: include some crucial empty elements
 - FULL: use the original Penn Treebank label set
 - MERGED: use a reduced set of labels

Comp Adjunct	Empty Elements	Label Set	Grammar Size	
			Frames	Lexicalized Trees
CA1	ALL	FULL	8996	118333
CA1	ALL	MERGED	5165	111220
CA1	SOME	FULL	8623	117527
CA1	SOME	MERGED	4911	110428
CA2	ALL	FULL	5354	116326
CA2	ALL	MERGED	2632	108370
CA2	SOME	FULL	4936	115335
CA2	SOME	MERGED	2366	107387

Size of various extracted grammars in number of tree frames and number of lexicalized trees

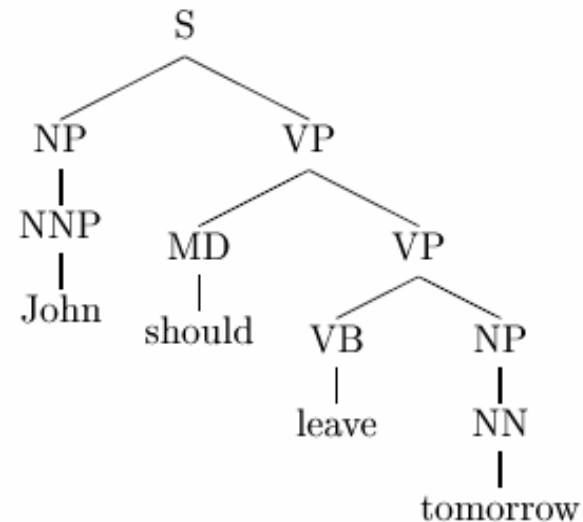
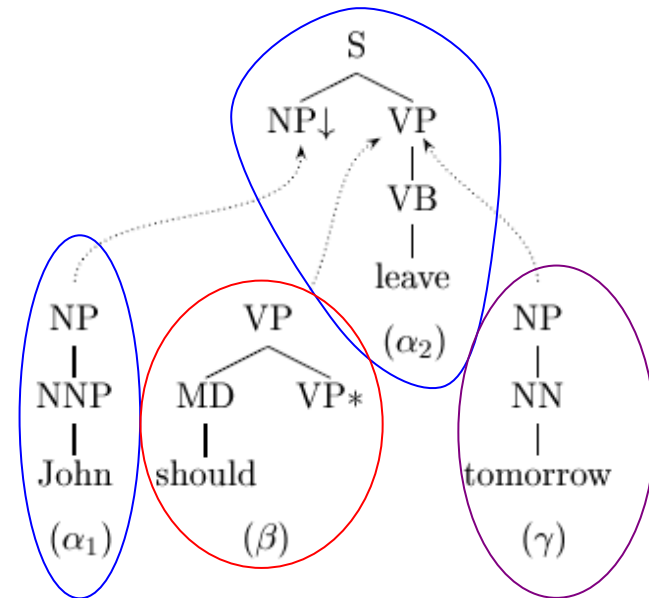


Comp Adj	Empty Elemt	Label Set	% found		% miss		Supertag Accuracy	Cutoff Accuracy
			frames	lex trees	in dict	not in dict		
CA1	ALL	FULL	99.56	91.57	5.67	2.76	77.79	77.85
CA1	ALL	MERGED	99.82	92.18	5.06	2.76	78.70	78.57
CA1	SOME	FULL	99.60	91.66	5.58	2.76	78.00	78.07
CA1	SOME	MERGED	99.83	92.27	4.98	2.76	78.90	78.78
CA2	ALL	FULL	99.80	92.05	5.19	2.76	77.85	77.79
CA2	ALL	MERGED	99.94	92.71	4.53	2.76	78.25	78.25
CA2	SOME	FULL	99.83	92.14	5.10	2.76	78.07	78.08
CA2	SOME	MERGED	99.96	92.80	4.44	2.76	78.55	78.50

Coverage of various extracted grammars and their corresponding supertagging performance. Coverage is in terms of percentage of tree frames and lexicalized trees in the test corpus. Missed coverage is divided into *in dict*—word seen in training corpus and *not in dict*—word not seen in training corpus. Supertagging performance is based on either the full grammar or cutoff grammar, cutoff value $k = 3$.



- A variant of LTAG
 - [Chiang, 2000]
 - Elementary trees
 - Initial tree
 - Predicative auxiliary tree
 - Modifier tree
 - Composition operation
 - Substitution
 - Adjoining (Adjunct)
 - Sister-adjunction





- Parameters of PTAG

$$\sum_{\alpha} P_i(\alpha) = 1 \quad (1)$$

$$\sum_{\alpha} P_s(\alpha | \eta) = 1 \quad (2)$$

$$\sum_{\beta} P_a(\beta | \eta) + P_a(NONE | \eta) = 1 \quad (3)$$

$$\sum_{\gamma} P_{sa}(\gamma | \eta, i, f) + P_{sa}(STOP | \eta, i, f) = 1 \quad (4)$$

- Probability of a derivation

$$\begin{aligned}
 &P_i(\alpha_2) \cdot P_a(NONE | \alpha_2(\epsilon)) \cdot \\
 &P_s(\alpha_1 | \alpha_2(1)) \cdot P_a(\beta | \alpha_2(2)) \cdot \\
 &P_{sa}(\gamma | \alpha_2(2), 1, true) \cdot \\
 &P_{sa}(STOP | \alpha_2(2), 1, false) \cdot \\
 &P_{sa}(STOP | \alpha_2(\epsilon), 0, true) \cdot \dots
 \end{aligned}$$

- A decomposed model

$$\begin{aligned}
 P_i(\alpha) &= P_{i_1}(\tau_{\alpha}) P_{i_2}(w_{\alpha} | \tau_{\alpha}) \\
 P_s(\alpha | \eta) &= P_{s_1}(\tau_{\alpha} | \eta) \cdot \\
 &P_{s_2}(w_{\alpha} | \tau_{\alpha}, t_{\eta}, w_{\eta})
 \end{aligned}$$

$$\begin{aligned}
 P_a(\beta | \eta) &= P_{a_1}(\tau_{\beta} | \eta) \cdot \\
 &P_{a_2}(w_{\beta} | \tau_{\beta}, t_{\eta}, w_{\eta})
 \end{aligned}$$

$$\begin{aligned}
 P_{sa}(\gamma | \eta, i, f) &= P_{sa_1}(\tau_{\gamma} | \eta, i, f) \cdot \\
 &P_{sa_2}(w_{\gamma} | \tau_{\gamma}, t_{\eta}, w_{\eta}, f)
 \end{aligned}$$

α : initial trees β : auxiliary trees

γ : modifier trees η : nodes

$P_i(\alpha)$: beginning a derivation with α

$P_s(\alpha | \eta)$: substituting α at η

$P_a(\beta | \eta)$: adjoining β at η

$P_{sa}(\gamma)$: sister-adjoining γ between the i th and $i+1$ th children of η



- A modified CKY-style parser
- Data
 - Training: sections 2-21 of Penn Treebank
 - Test: section 23

	≤ 40 words					≤ 100 words				
	LR	LP	CB	0 CB	≤ 2 CB	LR	LP	CB	0 CB	≤ 2 CB
(Magerman, 1995)	84.6	84.9	1.26	56.6	81.4	84.0	84.3	1.46	54.0	78.8
(Collins, 1996)	85.8	86.3	1.14	59.9	83.6	85.3	85.7	1.32	57.2	80.8
present model	86.9	86.6	1.09	63.2	84.3	86.2	85.8	1.29	60.4	81.8
(Collins, 1997)	88.1	88.6	0.91	66.5	86.9	87.5	88.1	1.07	63.9	84.6
(Charniak, 2000)	90.1	90.1	0.74	70.1	89.6	89.6	89.5	0.88	67.6	87.7

Parsing results. LR = labeled recall, LP = labeled precision; CB = average crossing brackets, 0 CB = no crossing brackets, ≤ 2 CB = two or fewer crossing brackets. All figures except CB are percentages.



- Joshi, A. K. 2003. Tree-Adjoining Grammars. In R. Mitkov (eds.), *The Oxford Handbook of Computational Linguistics*. 483-498, Oxford University Press, New York.
- Joshi, A. K., L. S. Levy, and M. Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and Systems Sciences*, 10(1), 55-75.
- Joshi, A. K. and Y. Schabes. 1997. Tree-adjoining grammars, In G. Rosenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 3, 69-123, Springer-Verlag, New York, NY.
- Joshi, A. K. and K. Vijay-Shanker. 1998. Treatment of Long Distance Dependencies in LFG And TAG: Functional Uncertainty in LFG is a Corollary in TAG. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 220-227, Vancouver, BC.



- The XTAG project: <http://www.cis.upenn.edu/~xtag>
- Chen, J., K. Vijay-Shanker. 2000. Extraction Of TAGs From The Penn Treebank, In *Proceedings of the 6th International Workshop on Parsing Technologies*, Trento, Italy.
- Neumann, G. 1998. Automatic Extraction of Stochastic Lexicalized Tree Grammars from Treebanks, In *Proceedings of the 4th International Workshop on Tree Adjoining Grammars and Related Frameworks*, 120-123
- Schabes, Y., A. Abeillé, 1988. Parsing Strategies with 'Lexicalized' Grammars: application to tree adjoining grammars, in *Proceedings of the 12th International Conference On Computational Linguistics*, 578-583, Budapest, Hungary.
- Chiang, D. 2000. Statistical Parsing With an Automatically-Extracted Tree Adjoining Grammar, In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, 456-463, Hong Kong.

**Thanks !
&
Questions?**

