Recognition and Parsing for Tree Adjoining Grammars

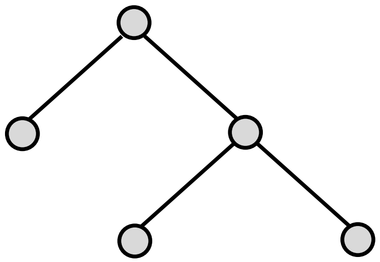
Giorgio Satta
University of Padua
satta@dei.unipd.it

Outline

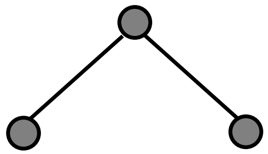
- preliminaries
- basic tabular techniques
- selected topics
- other advanced techniques
- computational/generative complexity trade-off

Sample TAG

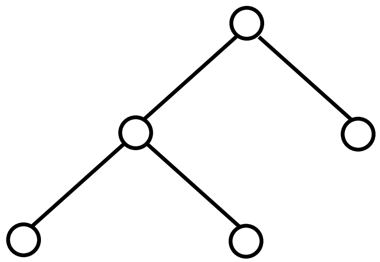
$$G = (V_N, V_\Sigma, S, I, A)$$



Grey $\in$ *Init*

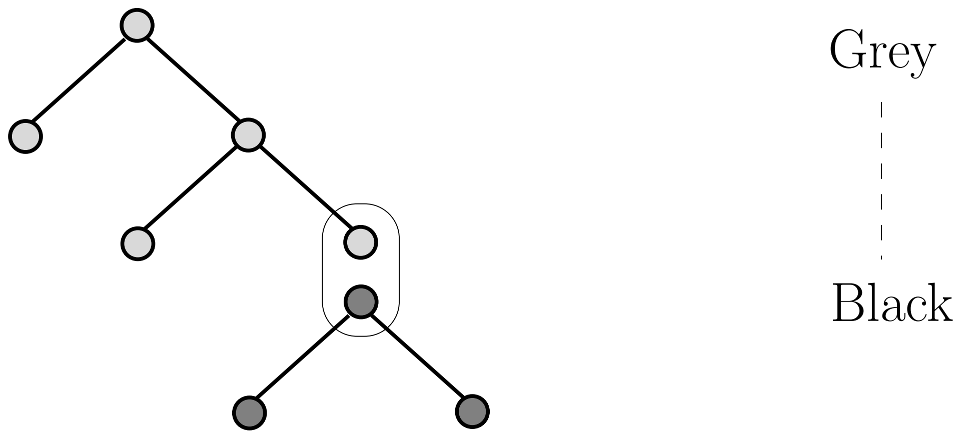


Black $\in$ *Init*

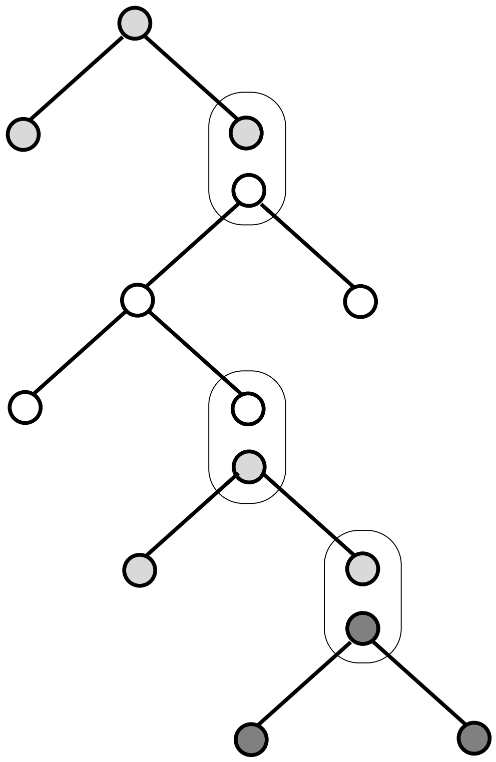


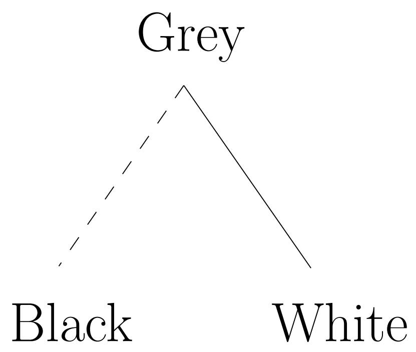
White $\in$ *Aux*

Sample derivation



Sample derivation (cont'd)





Outline

- preliminaries
- **basic tabular techniques**
  - bottom up, binary TAG
  - bottom up, general case
  - bottom up (derived tree)
  - top down
- selected topics
- other advanced techniques
- computational/generative complexity trade-off

Bottom Up on Derivation Trees

(Vijay-Shanker & Joshi, 1985)

Algorithm:

- visit each elementary tree bottom up
- visit derivation tree bottom up

In the presentation we assume:

- binary trees
- no node labeled by ε

BU: Representation

Let the input string be

$$w = a_1 \; a_2 \; a_3 \; \cdots \; a_n$$

An **index** for w is an integer i, $0 \leq i \leq n$, denoting a cut of w

$$[0] \; a_1 \; [1] \; a_2 \; [2] \; a_3 \; \cdots \; a_{n-1} \; [n-1] \; a_n \; [n]$$

BU: Representation (cont'd)

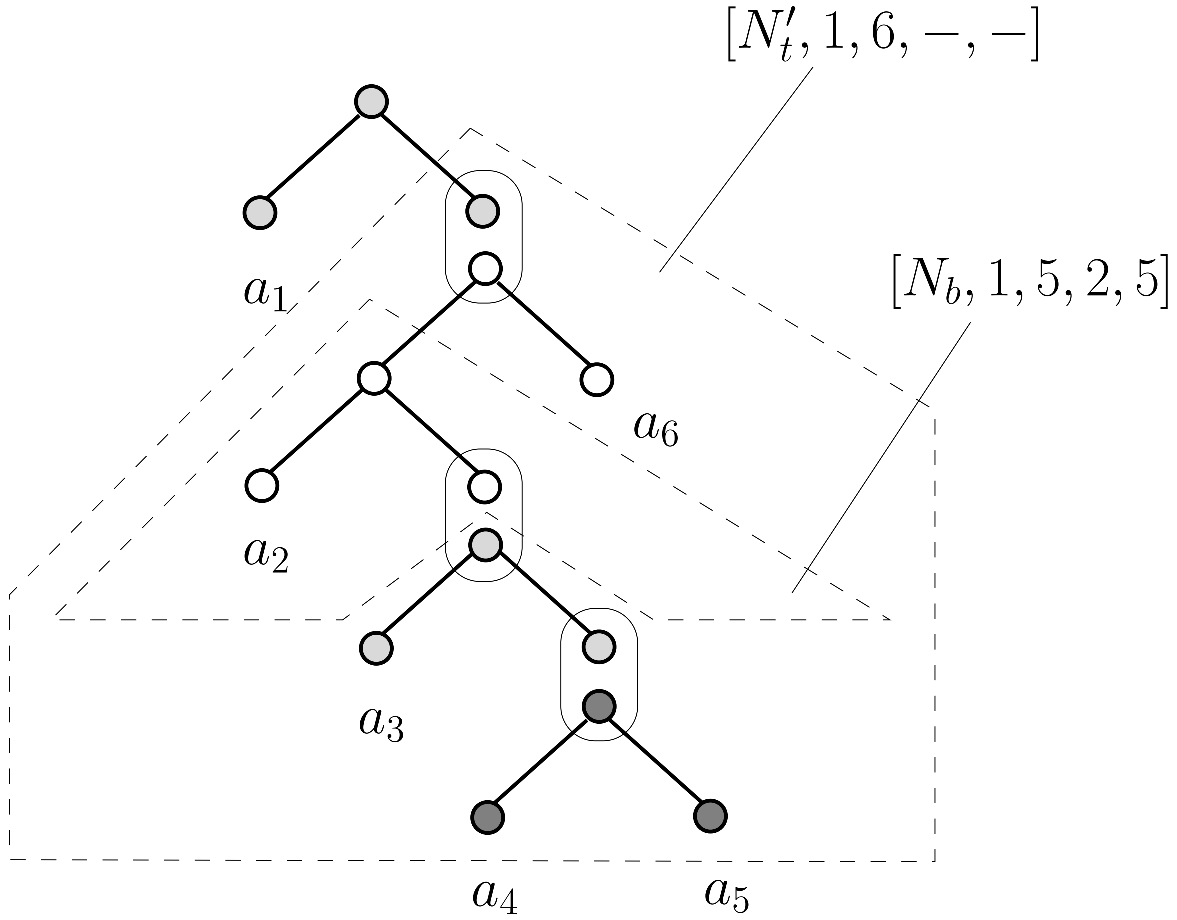
An **item** is a tuple

$$[N_X, i, j, f_1, f_2]$$

denoting a complete subtree of an elementary tree (with possible substitutions and adjunctions):

- N is the root of the subtree, spanning from index i to index j
- foot of subtree spans from index f_1 to index f_2; $f_1 = f_2 = -$ if there is no foot
- $X \in \{b, t\}$ records whether adjunction has been already checked ($X = t$) or not ($X = b$)

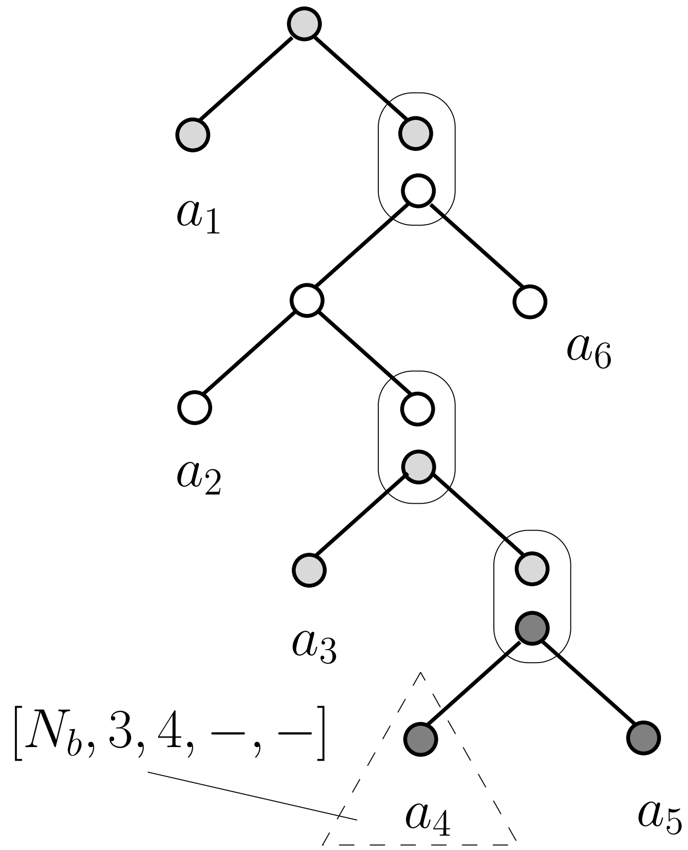
Example



BU: Steps

Init$_1$

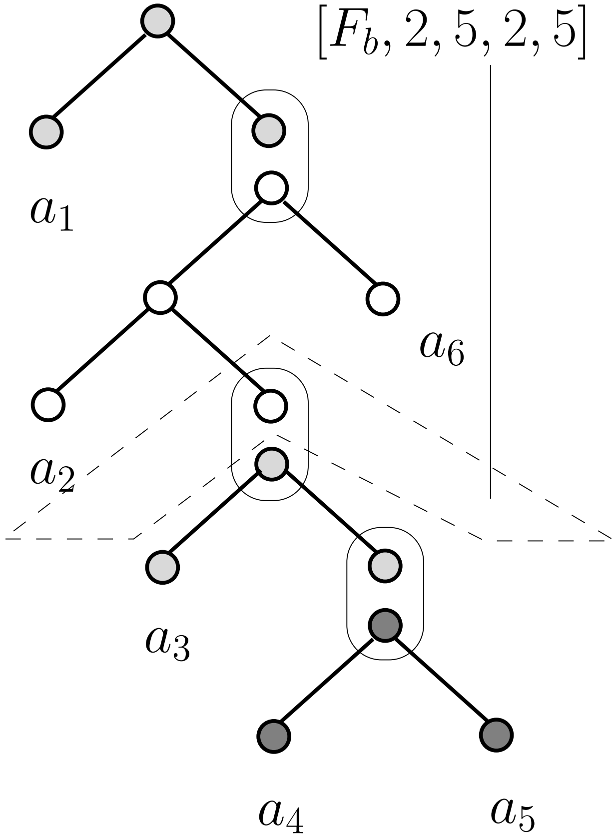
$$\frac{a_i = Labl(N)}{[N_b, i-1, i, -, -]}$$



BU: Steps (cont'd)

Init$_2$

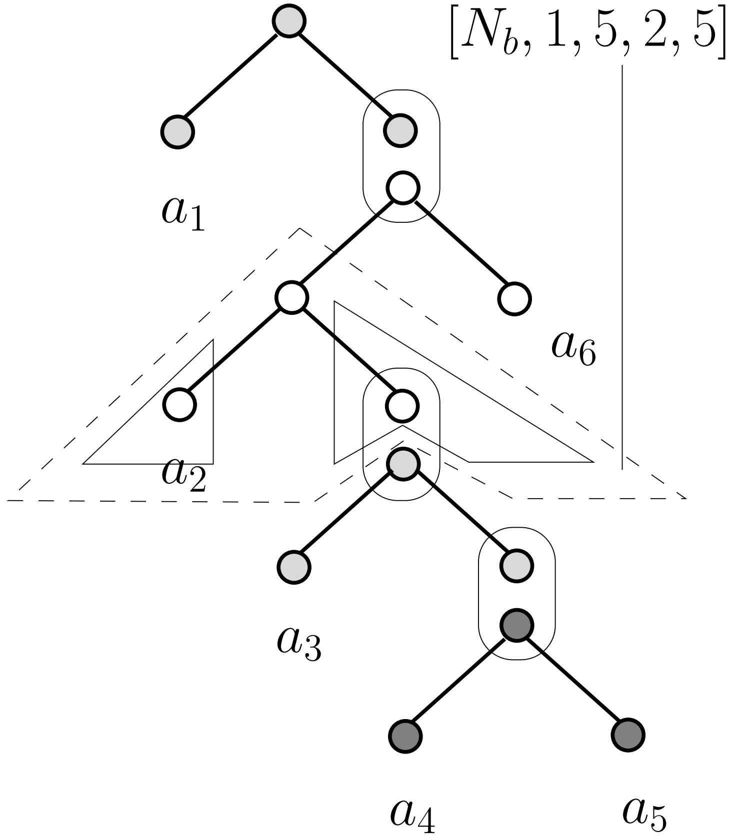
$$\frac{F = Foot(\beta) \quad \forall i, j \in \{0..n\},\ i < j}{[F_b, i, j, i, j]}$$



BU: Steps (cont'd)

Complete$_1$ (3 cases)

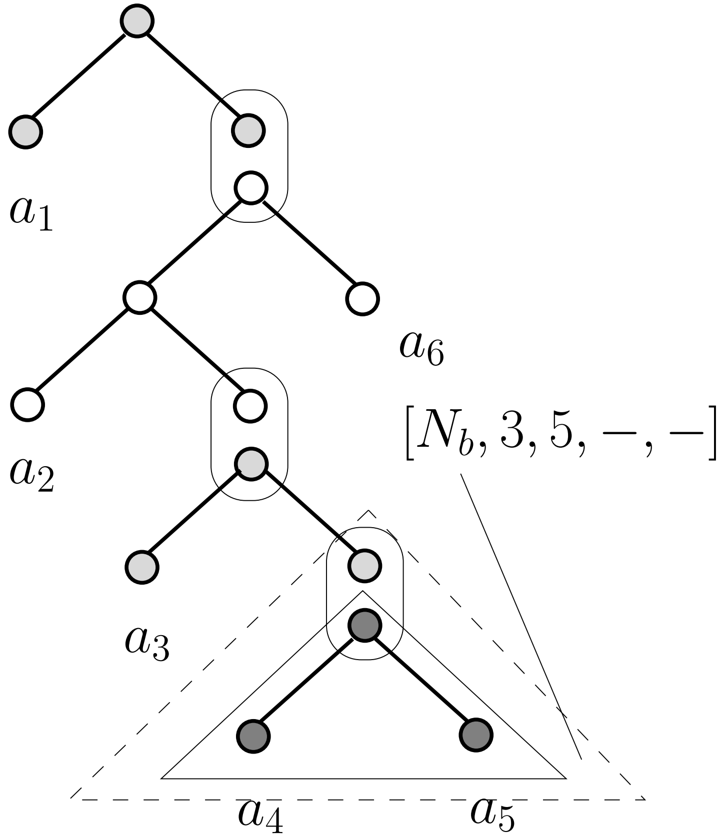
$$\frac{\begin{array}{c} Childr(N) = N'N'' \\ [N'_t, i, k, -, -] \\ [N''_t, k, j, f_1, f_2] \end{array}}{[N_b, i, j, f_1, f_2]}$$



BU: Steps (cont'd)

Substitute

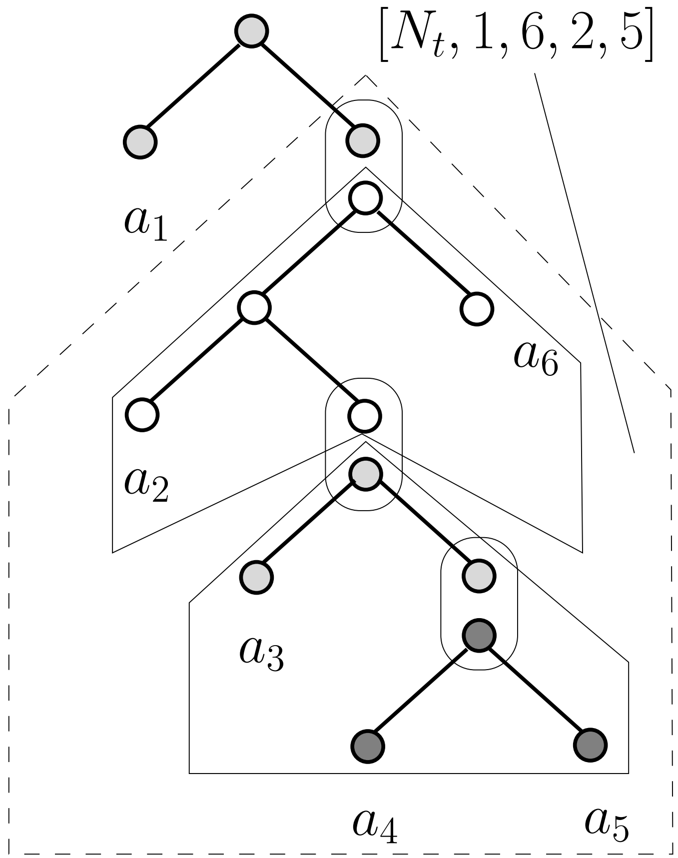
$$\frac{\begin{array}{c} R = Root(\alpha) \\ \alpha \in Sbst(N) \\ [R_t, i, j, -, -] \end{array}}{[N_t, i, j, -, -]}$$



BU: Steps (cont'd)

Adjoin$_1$

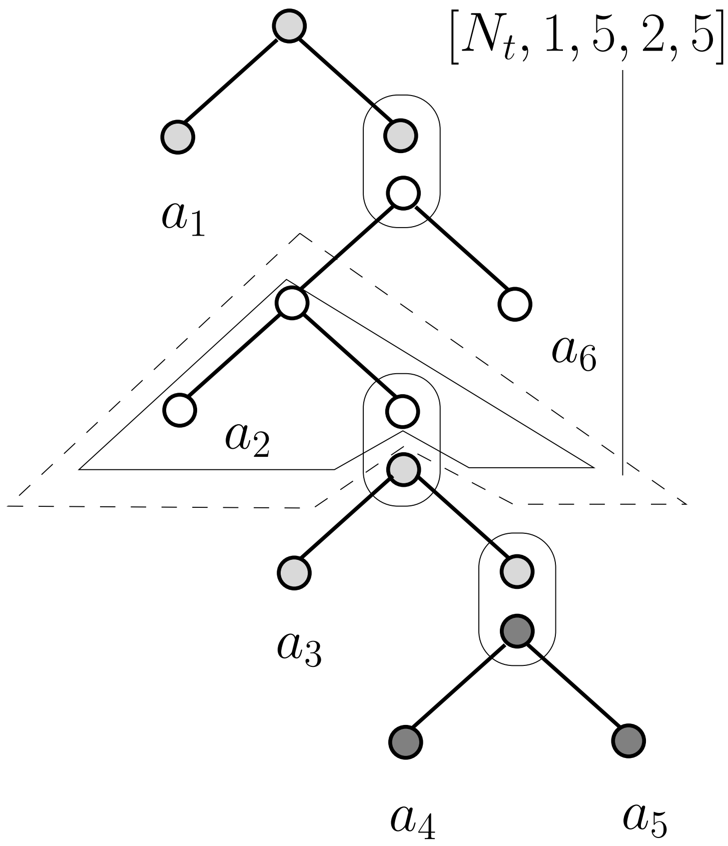
$$\frac{\begin{array}{c} R = Root(\beta) \\ \beta \in Adj(N) \\ [R_t, i, j, f_1, f_2] \\ [N_b, f_1, f_2, f_1', f_2'] \end{array}}{[N_t, i, j, f_1', f_2']}$$



BU: Steps (cont'd)

Adjoin$_2$

$$\frac{\begin{array}{c}\text{nil} \in Adj(N) \\ [N_b, i, j, f_1, f_2]\end{array}}{[N_t, i, j, f_1, f_2]}$$



BU: Data structure

Table:
a 4-dimensional table to retrieve items from their indices i, j, f_1 and f_2

Agenda:
a heap to order items w.r.t. some priority score

BU: Control

Init$_{1,2}() \rightarrow$ *Agenda*
while $I \leftarrow$ *Agenda* **do**
    $I \rightarrow$ *Table*
    Substitute$(I) \rightarrow$ *Agenda*;   **Adjoin**$_2(I) \rightarrow$ *Agenda*
    foreach I' in *Table* (adjacent to I)
        Complete$_{1,2,3}(I, I') \rightarrow$ *Agenda*
        Adjoin$_1(I, I') \rightarrow$ *Agenda*
    end
end
accept iff $[R_t, 0, n, -, -]$ in *Table* for some R root of an initial tree labeled by S

BU: Complexity

Step	time	space
Init	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Complete	$\mathcal{O}(n^5)$	$\mathcal{O}(n^4)$
Substitute	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Adjoin	$\mathcal{O}(n^6)$	$\mathcal{O}(n^4)$

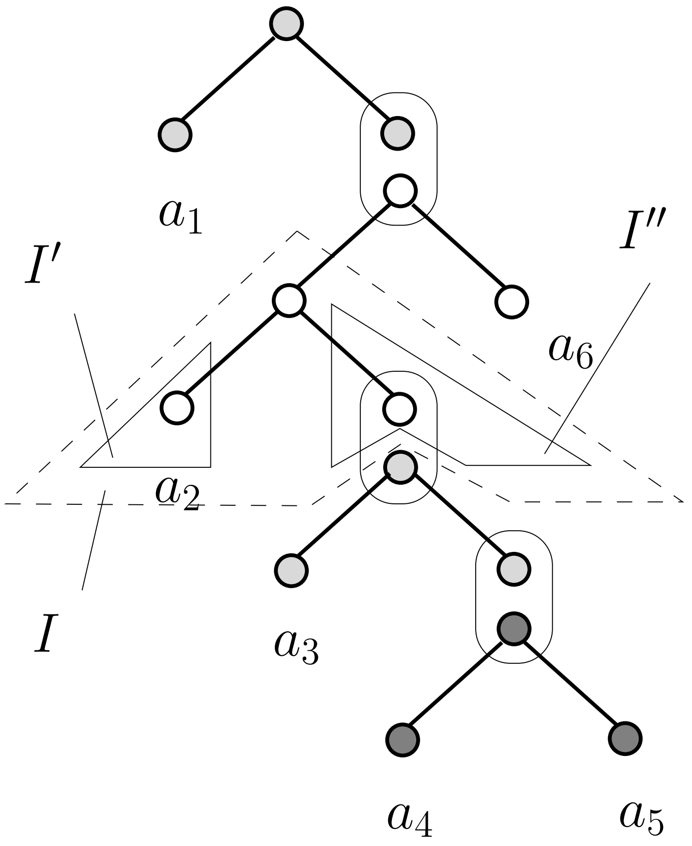
BU: Parsing

A **parse forest** for w and G is a representation of all derived trees for w in G

We construct a parse forest P of size $\mathcal{O}(n^6)$ from which a single derived tree can be extracted in time $\mathcal{O}(n)$

To build P, whenever item I is obtained from the application of some **Step**(I', I''), store with I pair of links $[(I, I'), (I, I'')]$ (similar condition for application of unary steps)

Example



$$I \longrightarrow \langle [(I, I'), (I, I'')], \ldots \rangle$$

BU: General Grammars

If N has children $M_1, \ldots, M_q$, $q \geq 1$, use additional items

$$[N \longrightarrow M_1 \cdots M_h \bullet M_{h+1} \cdots M_q, i, j, f_1, f_2]$$

and split **Complete** into

$$\frac{\begin{array}{c}[N \longrightarrow M_1 \cdots M_h \bullet M_{h+1} \cdots M_q, i, k, f_1, f_2] \\ [(M_{h+1})_t, k, j, -, -]\end{array}}{[N \longrightarrow M_1 \cdots M_{h+1} \bullet M_{h+2} \cdots M_q, i, j, f_1, f_2]}$$

$$\frac{[N \longrightarrow M_1 \cdots M_q \bullet, i, j, f_1, f_2]}{[N_b, i, j, f_1, f_2]}$$

BU: General Grammars (cont'd)

If N is labeled by ε extend **Init**$_1$

$$\frac{\forall i \in \{0..n\}}{[N_b, i, i, -, -]}$$

Bottom Up on Derived Trees

(Vijay-Shanker & Weir, 1993)

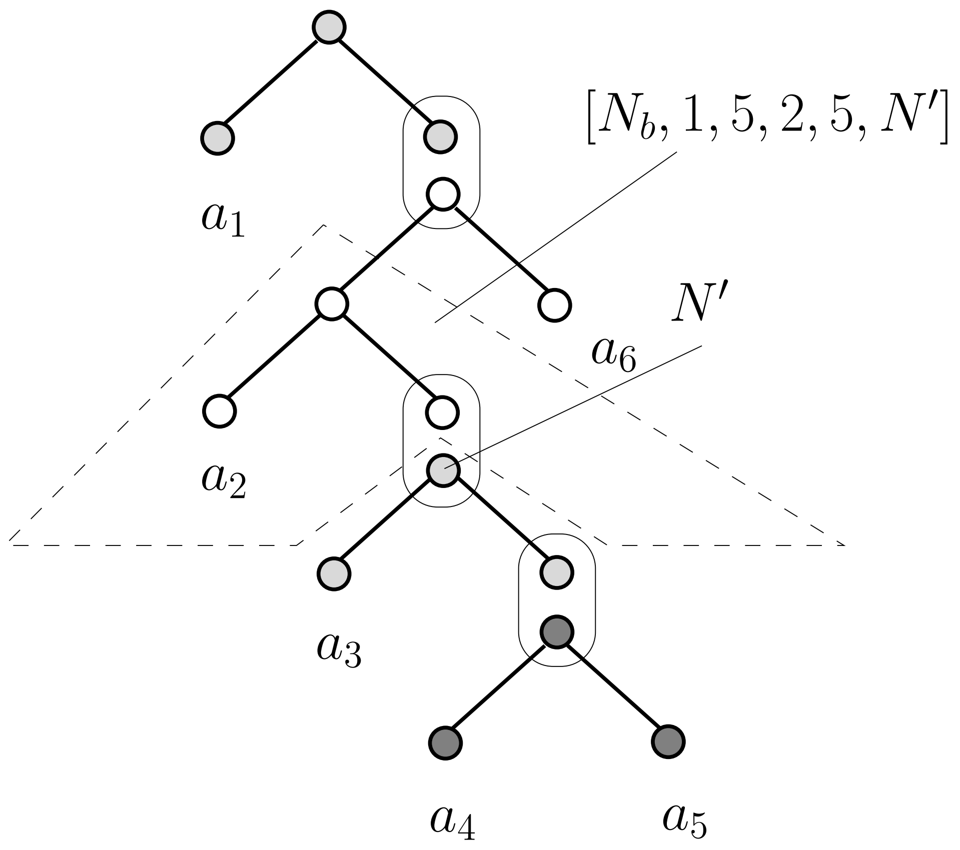
Use items

$$[N_X, i, j, f_1, f_2, N'],$$

where:

- N' is the node at which current subtree has been adjoined, if $f_1, f_2 \neq -$
- $N' = -$ if $f_1 = f_2 = -$

Example



Top Down on Derived Trees + Valid Prefix Property

(Schabes & Joshi, 1988), (Nederhof, 1997)

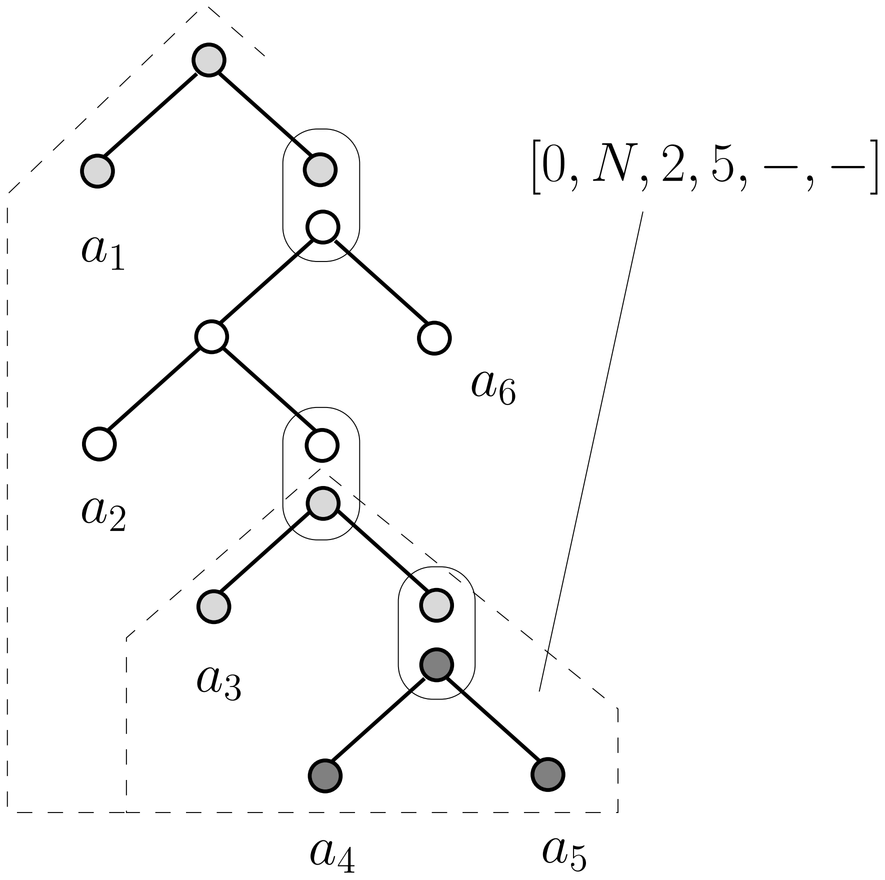
Parse input from left-to-right and detect an error as early as its position is reached

Use items

$$[h, N, i, j, f_1, f_2],$$

representing a subtree of an elementary tree γ such that γ spans the input from index h

Example



Outline

- preliminaries
- basic tabular techniques
- **selected topics**
  - guided parsing
  - supertagging
  - automata tabulation
- other advanced techniques
- computational/generative complexity trade-off

Guided Parsing

(Harbush, 1990),(Poller, 1994; Poller & Becker, 1998)

Construct a kernel context-free grammar from TAG:

- one production for each internal node
- one production for each substitution/adjunction

Algorithm:

- parse input using kernel grammar
- re-parse input BU using TAG and kernel parse forest as an oracle

Guided Parsing (cont'd)

(Barthélemy et al., 2001)

Construct a context-free grammar approximation of TAG *via* transformation to Range Concatenation Grammar (RCG):

- one production for each initial tree
- two productions for each auxiliary tree

Algorithm:

- use existing RCG guided parser
- on XTAG, three times faster than non-guided RCG parser

Supertagging

(Bangalore & Joshi, 1999)

Phase I: for each word in input, select lexicalized elementary tree (supertag) that is most likely:

- use language modeling techniques (3-grams)
- use error-driven method

Phase II: produce dependency linkages and partial parse (Lightweight Dependency Analyzer)

Supertagging (cont'd)

(Boullier, 2003)

Supertag selection using non-local generative models

- superset approximations, 100% recall
- LTAG to CFG
- LTAG to FA (dynamically constructed)

Automata tabulation

(Alonso et al., 2000)

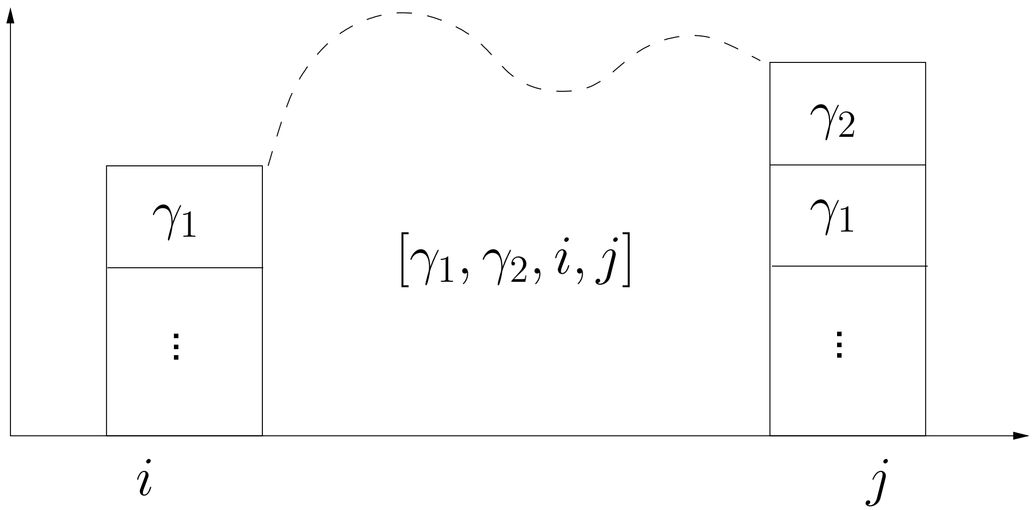
Separation between

- parsing strategy specification through operational automata
- algorithm specification through tabulation

$$2SA \longrightarrow \boxed{\text{Tabulation}} \longrightarrow \text{Parser}$$

Automata tabulation (cont'd)

- 2-stack automata abstractly specify parsing strategy
- nondeterminism is simulated through dynamic programming



Outline

- preliminaries
- basic tabular techniques
- selected topics
- **other advanced techniques**
- computational/generative complexity trade-off

Advanced Techniques

LR parsing for TAG (Schabes & Vijay-Shanker, 1990), (Nederhof, 1998), (Prolo, 2004)

TAG parse forest represented by means of TAG (Lang, 1994) or context-free grammars (Vijay-Shanker & Weir, 1993)

Bidirectional parsing for TAG (Lavelli & Satta, 1991), (van Noord, 1994)

Parsing for 2-lexical TAG (Eisner & Satta, 2000)

Advanced Techniques (cont'd)

TAG parsing through Range Concatenation Grammars (Boullier, 2000)

TAG parsing through Boolean matrix multiplication (Rajasekaran & Yooseph, 1998)

TAG parsing through synchronous context-free grammar parsing (Guan, 1992)

TAG parsing also used to parse other formalisms, as HPSG (Kasper *et al.*, 1995), (Becker & Lopez, 2000)

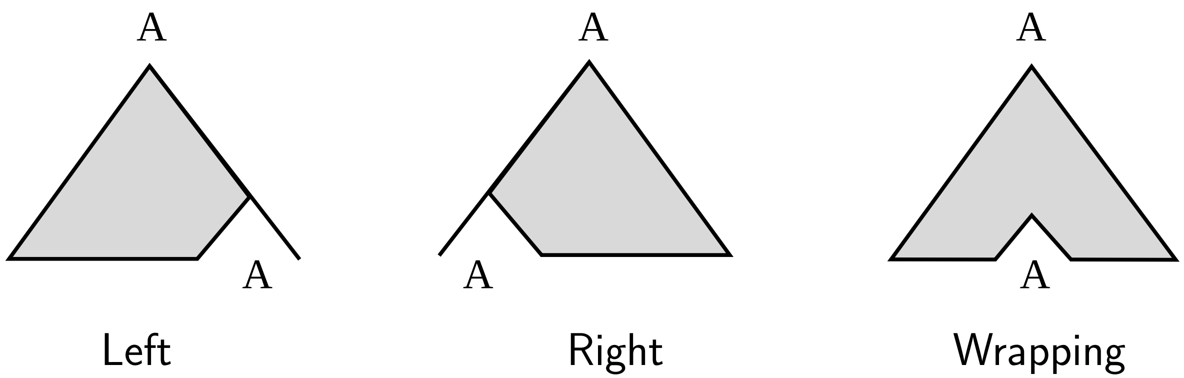
Outline

- preliminaries
- basic tabular techniques
- selected topics
- other advanced techniques
- **computational/generative complexity trade-off**

Computational/Generative Complexity Trade-Off

A **left tree** is an auxiliary tree with all nodes in its yield at the left of the foot (a **right tree** is symmetrically defined)

A **wrapping tree** is an auxiliary tree that is neither left nor right



Trade-Off (cont'd)

Parsing in $\mathcal{O}(n^3)$ (Schabes & Waters, 1993; 1995):

- no wrapping trees
- no adjunction of left (right) trees at the spine of right (left) trees

Parsing in $\mathcal{O}(n^3)$ (Rogers, 1994):

- no adjunction at the spine if this causes "recursion" of some nonterminal

These formalisms are weakly equivalent to context-free grammars

Trade-Off (cont'd)

Parsing in $\mathcal{O}(n^5)$ (Satta & Schuler, 1998):

- no more than two nodes on spine of left/right trees
- at most one adjunction of a wrapping tree at the spine of a wrapping tree

Trade-Off (cont'd)

TAG parsing asymptotically faster than $\mathcal{O}(n^6)$ implies Boolean matrix multiplication asymptotically faster than $\mathcal{O}(m^3)$ (Satta, 1994)

Result holds whenever we can adjoin at more than one site in the spine of an auxiliary tree

References

[1] A. V. Aho and J. D. Ullman. Translations on a context-free grammar. *Information and Control*, 19:439–475, 1971.

[2] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[3] M. A. Alonso Pardo, M.-J. Nederhof, and E. Villemonte de la Clergerie. Tabulation of automata for tree-adjoining languages. *Grammars*, 3:89–110, 2000.

[4] S. Bangalore and A. K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265, 1999.

[5] F. Barthélemy, P. Boullier, P. Deschamp, and É. Villemonte de la Clergerie. Guided parsing of range concatenation languages. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*, pages 42–49, University of Toulouse, France, July 2001.

[6] T. Becker and P. Lopez. Adapting HPSG-to-TAG compilation to wide-coverage grammars. In *Proceedings of the 5th Int. Workshop on Tree-Adjoining Grammars and Related Formalisms*, pages 47–54, Paris, France, 2000.

[7] P. Boullier. On TAG parsing. *Traitement Automatique des Langues (T.A.L.)*, 41(3):111–131, 2000. issued June 2001.

[8] P. Boullier. Supertagging: A non-statistical parsing-based approach. In *Proceedings of the 8th Int. Workshop on Parsing Technologies*, pages 55–65, Nancy, France, 2003.

[9] J. Earley. An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery*, 13(2):94–102, 1970.

[10] J. Eisner and G. Satta. A faster parsing algorithm for lexicalized tree-adjoining grammars. In *Proceedings of the 5th International Workshop on Tree Adjoining Grammars*, pages 79–84, Paris, France, 2000.

[11] Y. Guan. An $\mathcal{O}(n^5)$ recognition algorithm for coupled parenthesis rewriting systems. In A. K. Joshi, editor, *Proceedings of the Second International TAG+ Workshop*, Philadelphia, PA, 1992. University of Pennsylvania.

[12] K. Harbusch. An efficient parsing algorithm for tree adjoining grammars. In 28^{th} *Meeting of the Association for Computational Linguistics (ACL'90)*, Pittsburgh, 1990.

[13] R. T. Kasper, B. Kiefer, K. Netter, and K. Vijay-Shanker. Compilation of HPSG to TAG. In *Proc. of the 33*rd *ACL*, pages 92–99, Cambridge, MA, 1995.

[14] B. Lang. Recognition can be harder than parsing. *Computational Intelligence*, 10(4), 1994.

[15] A. Lavelli and G. Satta. Bidirectional parsing of lexicalized tree adjoining grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, Berlin, 1991.

[16] M.-J. Nederhof. Solving the correct-prefix property for TAGs. In T. Becker and H.-U. Krieger, editors, *Fifth Meeting on Mathematics of Language*, pages 124–130, Saarbrücken, Germany, 1997. DFKI.

[17] M.-J. Nederhof. An alternative LR algorithm for TAGs. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 2, pages 946–952, Montreal, Quebec, Canada, August 1998.

[18] P. Poller. Incremental parsing with LD/TLP-TAGs. *Computational Intelligence*, 10(4):549–562, 1994.

[19] P. Poller and T. Becker. Two-step TAG parsing revisited. In *Proceedings of the Fourth International TAG+ Workshop*, Philadelphia, PA, 1998. University of Pennsylvania.

[20] C. Prolo. An efficient LR parser generator for tree adjoining grammars. In H. Bunt, J. Carroll, and G. Satta, editors, *New Developments in Parsing Technology*, pages 125–156. Kluwer, Dordrecht, 2004.

[21] S. Rajasekaran and S. Yooseph. TAL recognition in $\mathcal{O}(M(n^2))$ time. *Journal of Computer and System Sciences*, 56(1):83–89, 1998.

[22] J. Rogers. Capturing CFLs with tree adjoining grammars. In *Proc. of the 32*nd *ACL*, pages 155–162, Las Cruces, New Mexico, 1994.

[23] G. Satta. Tree adjoining grammar parsing and boolean matrix multiplication. *Computational Linguistics*, 20(2), 1993.

[24] G. Satta and W. Schuler. Restrictions on tree adjoining languages. In *Proc. of the 36*th *ACL*, pages 1176–1182, Montreal, Canada, 1998.

[25] Y. Schabes and A. K. Joshi. An Earley-type parsing algorithm for tree adjoining grammars. In *Proc. of the 26*th *ACL*, pages 258–269, 1988.

[26] Y. Schabes and K. Vijay-Shanker. Deterministic left to right parsing of tree adjoining languages. In *28th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 276–283, Pittsburgh, Pennsylvania, USA, June 1990.

[27] Y. Schabes and R. C. Waters. Lexicalized context-free grammars. In *Proc. of the 31st ACL*, Columbus, Ohio, 1993.

[28] Y. Schabes and R. C. Waters. Tree insertion grammar: A cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4):479–515, 1995.

[29] Y. Schabes and K. Vijay-Shanker. Deterministic left to right parsing of tree adjoining languages. In 28^{th} *Meeting of the Association for Computational Linguistics (ACL'90)*, Pittsburgh, 1990.

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

[31] G. van Noord. Head-corner parsing for tag. *Computational Intelligence*, 10(4):525–534, 1994.

[32] K. Vijay-Shanker and Aravind K. Joshi. Some computational properties of tree adjoining grammars. In 23^{rd} *Meeting of the Association for Computational Linguistics*, pages 82–93, Chicago, Illinois, July 1985.

[33] K. Vijay-Shanker and D. J. Weir. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636, 1993.

[34] K. Vijay-Shanker and D. J. Weir. The use of shared forests in tree adjoining grammar parsing. In *Proc. of the 6th EACL*, pages 384–393, Utrecht, The Netherlands, 1993.

[35] É. Villemonte de la Clergerie and M. Alonso Pardo. A tabular interpretation of a class of 2-stack automata. In *Proc. of ACL/COLING'98*, August 1998.

[36] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208, 1967.