CFLs :: RTLs
TALs :: ??

Anoop Sarkar
School of Computing Science
Simon Fraser University
http://natlang.cs.sfu.ca/
anoop@cs.sfu.ca

Context Free Languages

Regular Tree Languages

CFLs :: RTLs
TALs :: ??

Tree Adjoining Languages

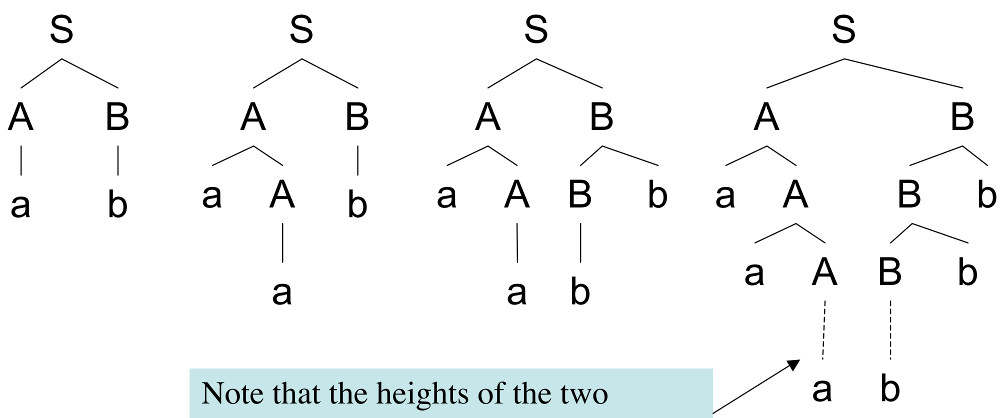
ATLs:
- Tree language defn for TALs
- Rid TAG of adj constraints!
- Useful for ling?
- Prob version is interesting

Anoop Sarkar
School of Computing Science
Simon Fraser University
http://natlang.cs.sfu.ca/
anoop@cs.sfu.ca

Tree Languages

$S \rightarrow A\ B$
$A \rightarrow a\ A \mid a$
$B \rightarrow B\ b \mid b$

This grammar generates the tree language informally shown below

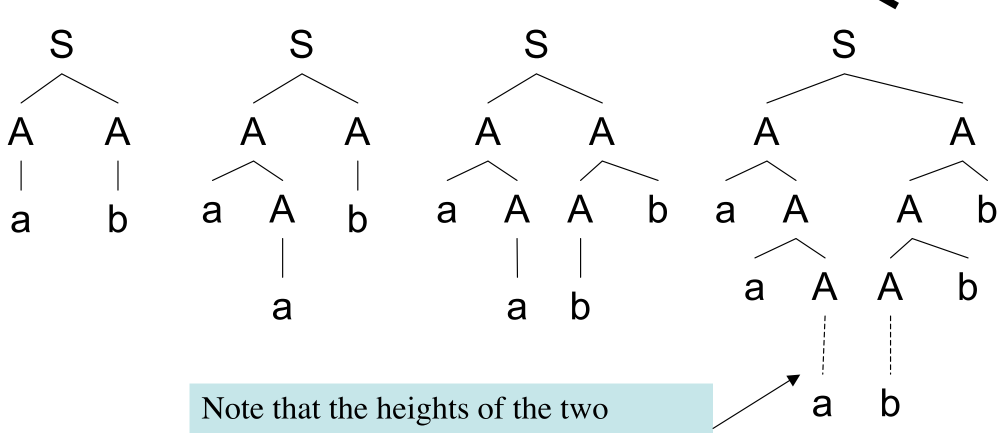


Note that the heights of the two branches do not have to be equal

A Tree Language with no CFG

Claim: There is no CFG that can produce the tree lang below:

~~$S \rightarrow A\ A$~~
~~$A \rightarrow a\ A \mid a$~~
~~$A \rightarrow A\ b \mid b$~~

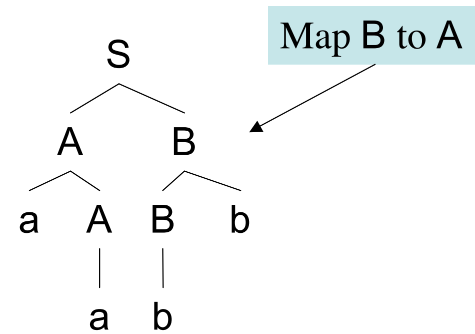


Note that the heights of the two branches do not have to be equal

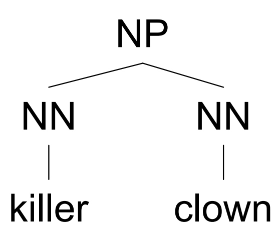
Grammars for Tree Languages

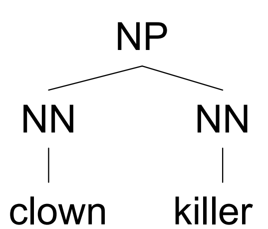
- A simple trick: start with a CFG that almost works
- Then re-label the node labels, map B to A to get the desired tree set
- But how can we directly generate the tree sets?
- We need a **generative device** that generates *trees*, not *strings*
- (Thatcher, 1967) and (Rounds, 1970) provided such a generative device

S → A B
A → a A | a
B → B b | b



Regular Tree Grammars (Thatcher 1967)

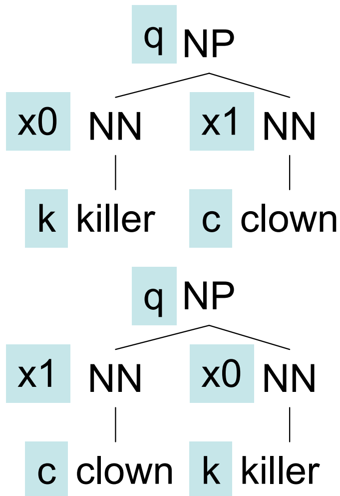




Example from (May & Knight, 2006)

- Consider a simple tree set with two trees for the strings { *killer clown*, *clown killer* }
- No CFG can recognize this simple tree set without also recognizing trees for *clown clown* and *killer killer*
- A Regular Tree Grammar recognizes this tree set (analogy with regular grammars on strings)

Regular Tree Grammars



start state: q
q → NP(x0 x1)
q → NP(x1 x0)
x0 → NN(k)
x1 → NN(c)
k → killer
c → clown

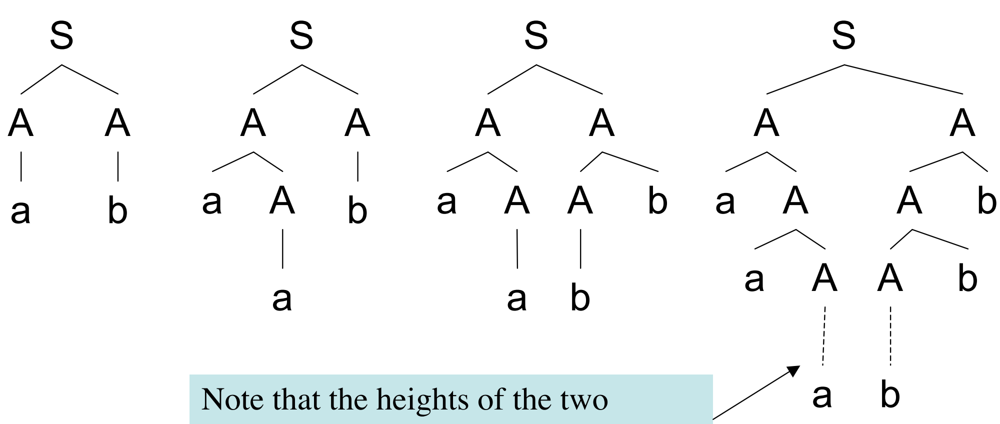
note: can be a tree of any size!

- RTGs = Top-down tree automata
- **Can generate infinite tree sets**
- **Found useful in syntax-based statistical machine translation**

Example from (May & Knight, 2006)

Another RTG Example

q → S(x0 x1)
x0 → A(a x0) | a
x1 → A(x1 b) | b



Note that the heights of the two branches do not have to be equal

Regular Tree Grammars

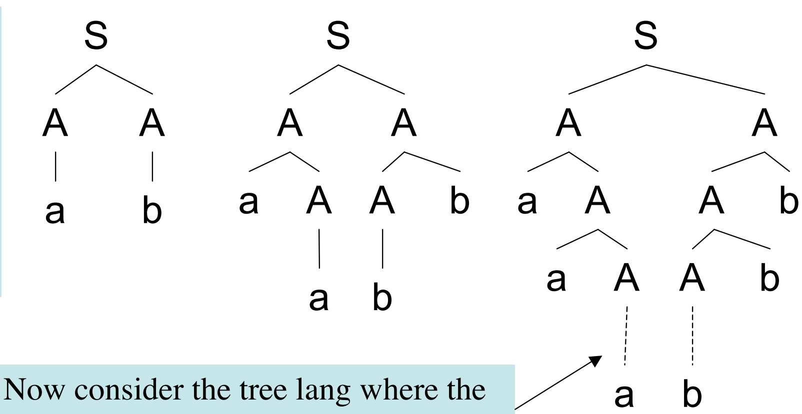
- RTGs generate tree languages
- The yield of each tree in this language produces a string
- *yield*(RTG) provides a string language
- For each RTG: *yield*(RTG) = CFL
- But as we saw, set of tree languages of CFGs is contained within that of RTGs

A Tree Language with no RTG

Claim: There is no RTG that can produce the tree language below:

~~q → S(x0 x1)~~
~~x0 → A(a x0) | a~~
~~x1 → A(x1 b) | b~~

RTG is like a finite-state machine, the state cannot count how many times it was reached

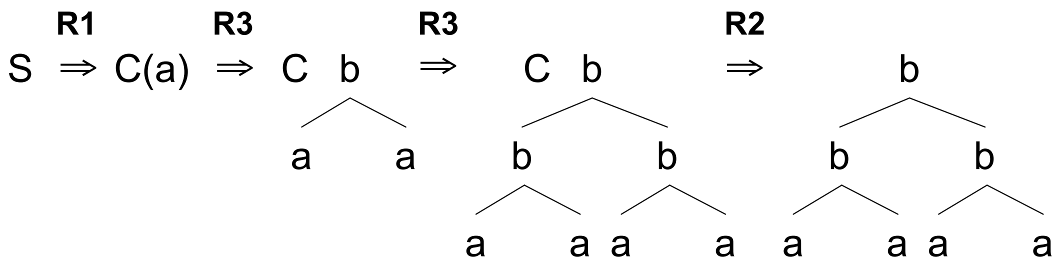


Now consider the tree lang where the depth of the two branches is **equal**

(Rounds 1970)

Context-free Tree Languages

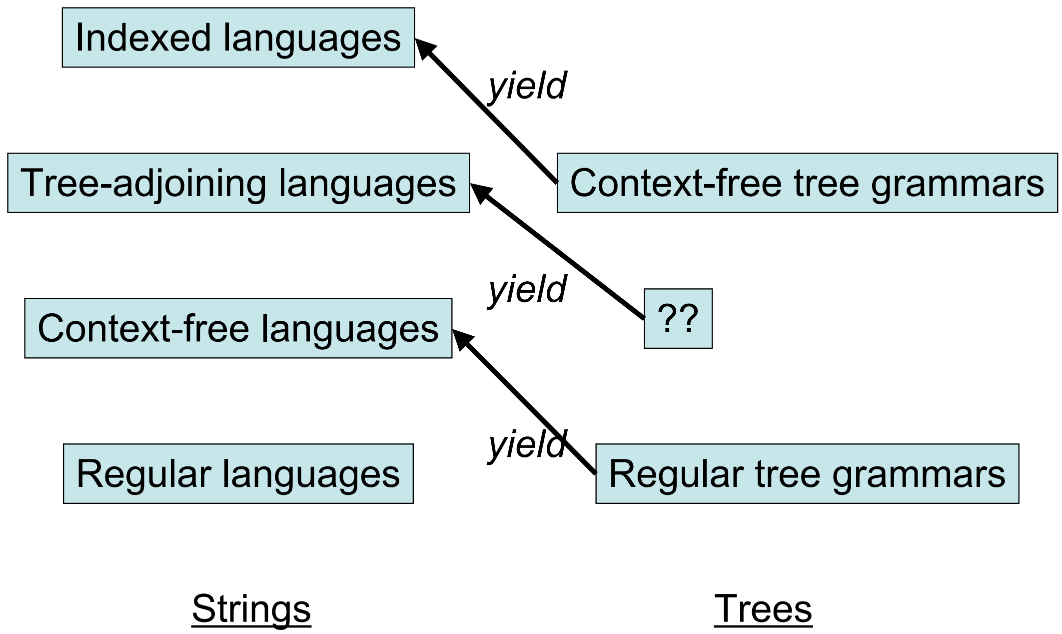
R1: S → C(a)
R2: C(x1) → x1
R3: C(x1) → C(b(x1 x1))



String language $= \{a^{2^n} \mid n \geq 0\}$

Context-free Tree Languages

- *yield*(CFTLs) = Indexed Languages
  (Fischer 1968)
- Indexed languages: not constant growth, recognition algorithm is NP-complete
  (Rounds, 1973)
- Perhaps we can obtain a tree language between RTG and CFTG by adding constraints to CFTGs?
- Ideally, this tree language should be weakly equivalent to TAGs



Modifying CFTGs

- Simple CFTG = linear and non-deleting
- Tree language of TAGs is contained within monadic simple CFTGs
- String language of TAGs is equal to that of monadic simple CFTGs
  (Fujiyoshi & Kasai, 2000; Mönnich 1997)
- Another alternative approach: (Lang, 1994)

Tree Languages: Another Example

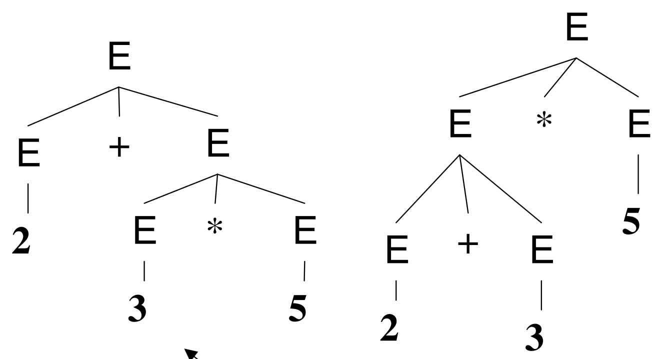
A more practical example

E → E + E

E → E * E

E → (E)

E → **N**



2+3*5 is ambiguous
either **17** or **25**

Ambiguity resolution: * has precedence over +
cannot use RTGs!

Tree Languages: Context-sensitivity

Eliminating ambiguity

E → E + E

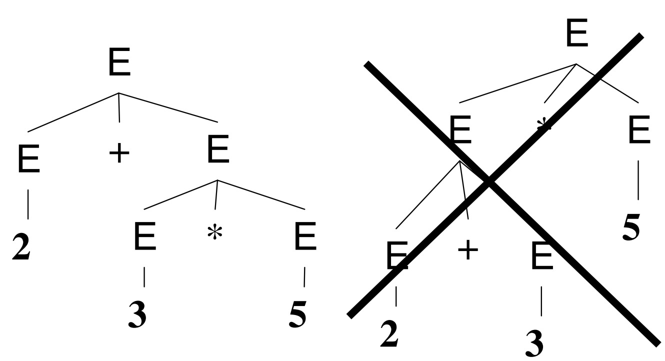
¬(+__)∧¬(*__)∧¬(__*)

E → E * E

¬(*__)

E → (E)

E → **N**

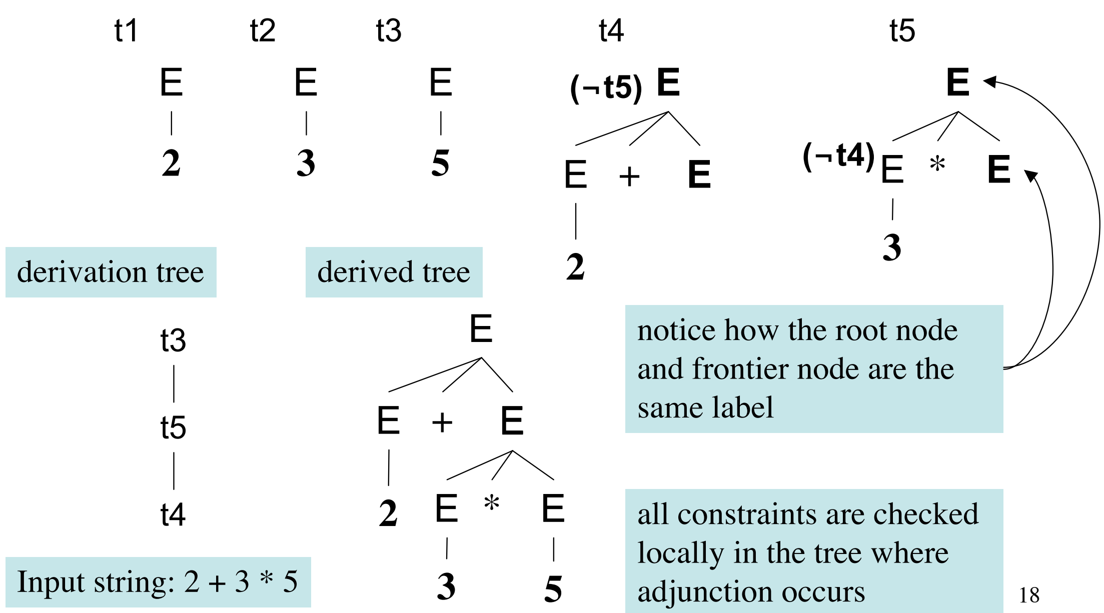


similar to context-sensitive grammars!

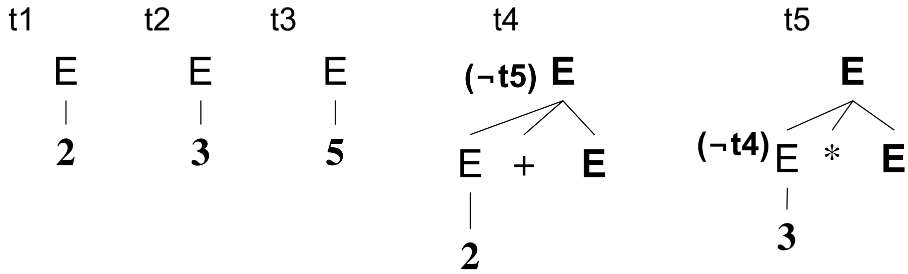
Locality of CSG predicates

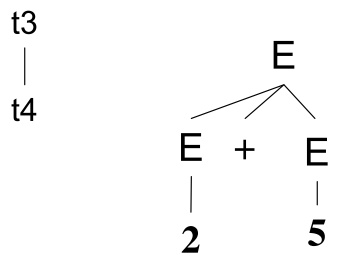
- An analysis of the kinds of CSG grammars used to define linguistic analyses in practice showed an interesting fact
- All the CSG predicates were very local
- They did not include in the context various parts of the tree that were arbitrarily far apart
- Long distance dependencies were expressed by chaining together many local CSG predicates
- This insight can be used to generate trees from an input string and validate them using CSG predicates

Tree-Adjoining Grammars



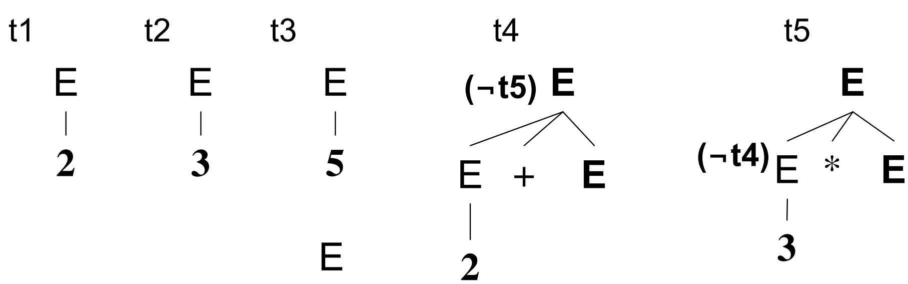
Tree-Adjoining Grammars

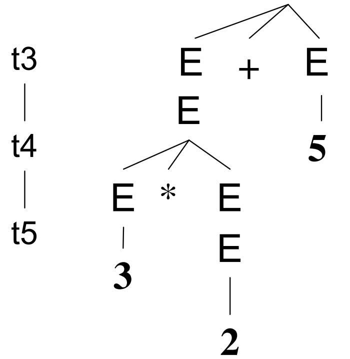




derivation for **3*2+5**

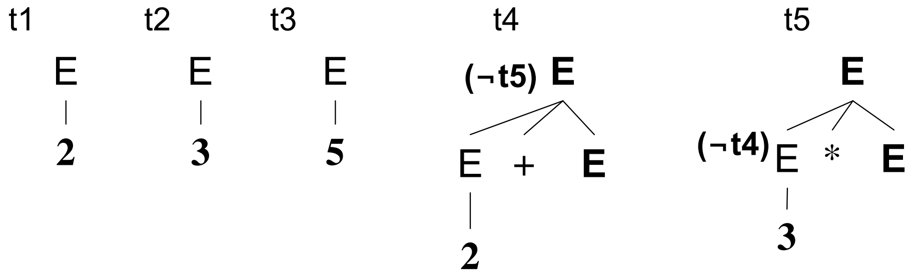
Tree-Adjoining Grammars

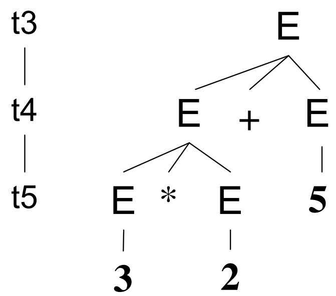




derivation for **3*2+5**

Tree-Adjoining Grammars



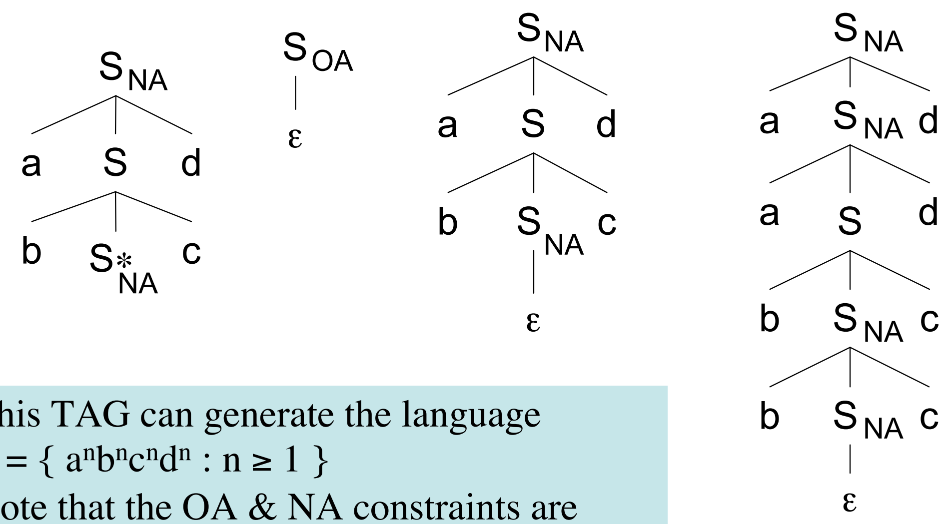


derivation for **3*2+5**

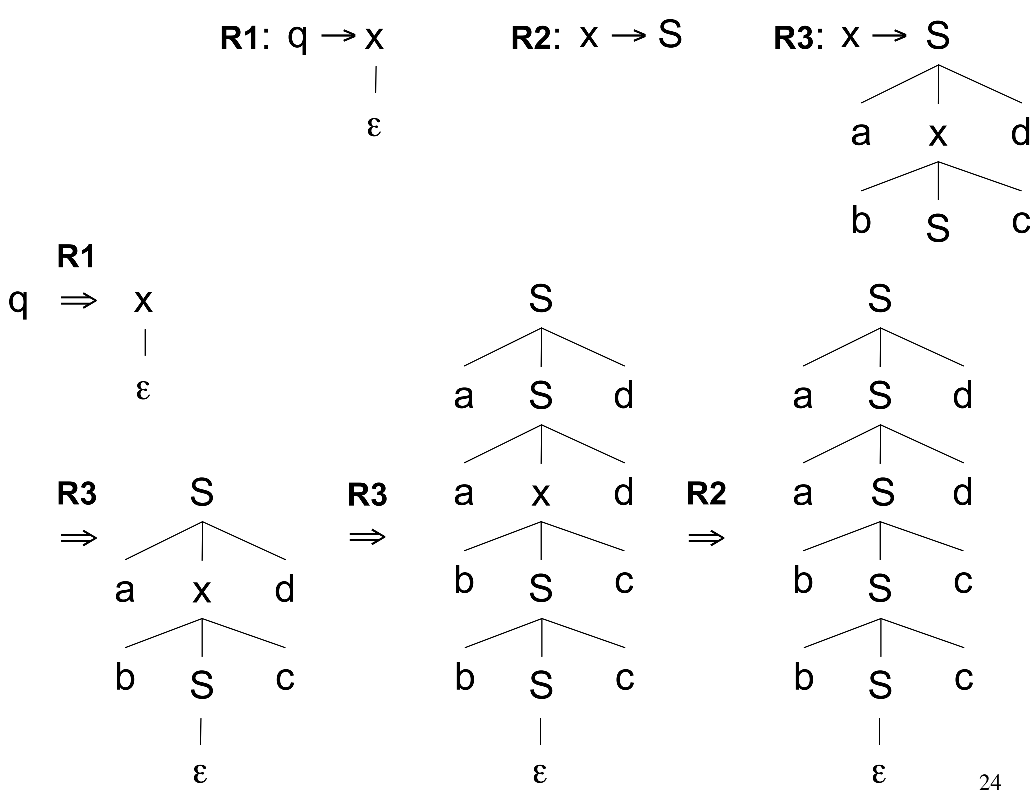
Adjunction Constraints

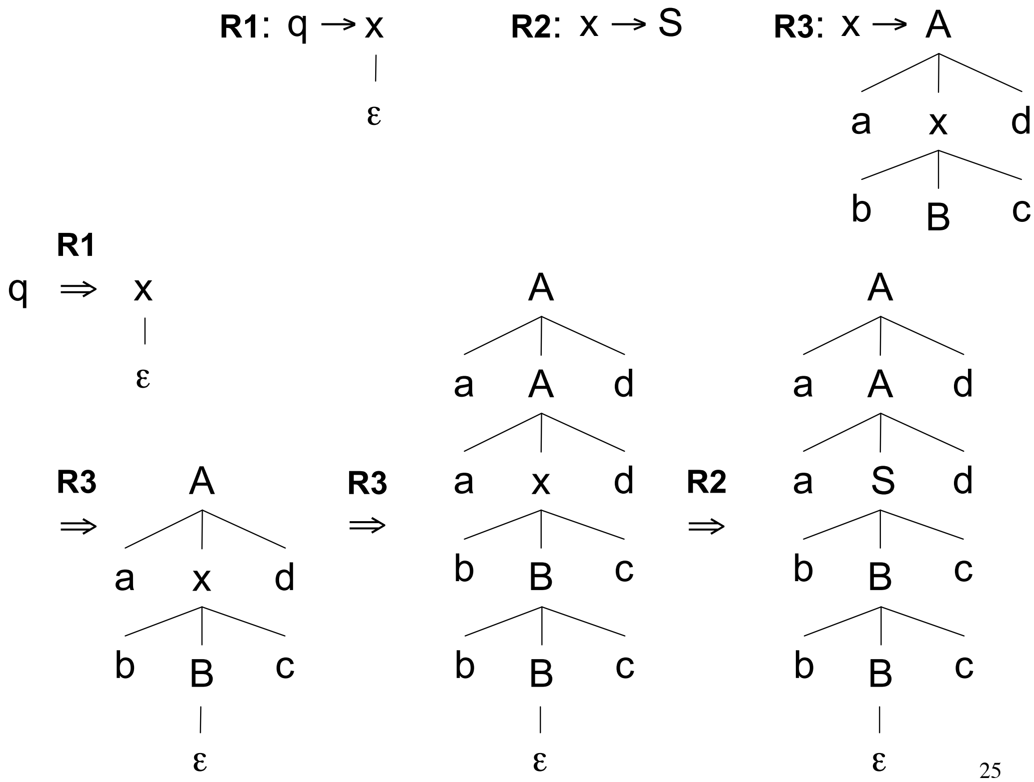
- Adjunction is the rewriting of a non-terminal in a tree with an auxiliary tree
- We can think of this operation as being “context-free”
- Constraints are essential to control adjunction: both in practice for NLP and for formal closure properties
- Three types of constraints:
  - null adjunction (NA): no adjunction allowed at a node
  - obligatory adjunction (OA): adjunction must occur at a node
  - selective adjunction (SA): adjunction of a pre-specified set of trees can occur at a node

Adjunction Constraints



This TAG can generate the language
$L = \{ a^n b^n c^n d^n : n \geq 1 \}$
Note that the OA & NA constraints are crucial to obtain the correct language





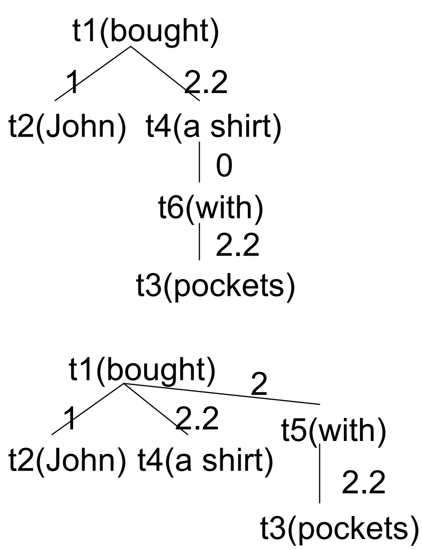
Adjoining Tree Grammars

- Similar to defn by (Lang, 1994)
- No adjoining constraints required
- Weakly equivalent to TAGs
- Set of tree languages for TAGs contained within that for ATGs
- Is ATG attractive for simplifying some TAG-based linguistic analysis?
  - Analyses that use adjoining constraints (feature structures)
  - Analyses that require different labels on rootnode and footnode

Adjoining Tree Grammars

- Closure properties for TALs (union, concat, homomorphism, substitution) can be shown using ATGs instead of TAGs.
  - By taking yield of the tree language
  - Without using adjunction constraints
- Intersection with regular languages (Lang, 1994)
- What about pumping lemma? cf. (Kanazawa, 2006)
- Polynomial time parsing algorithm provided by (Lang, 1994) = takes a string as input **not** a tree.
- Is ATG strongly equivalent to monadic simple CFTGs?

Ambiguity Resolution



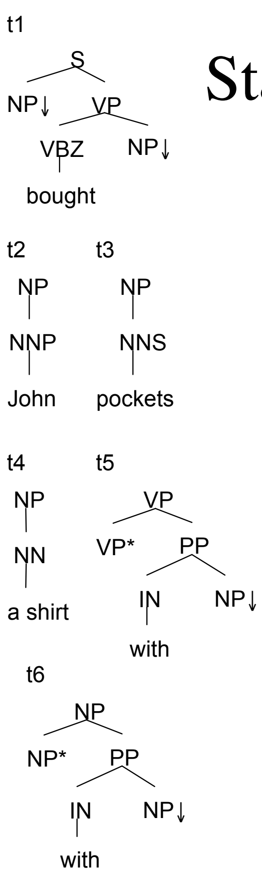
- Two possible derivations for ***John bought a shirt with pockets.***
- One of them is more plausible than the other.
- Statistical parsing is used to find the most plausible derivation.

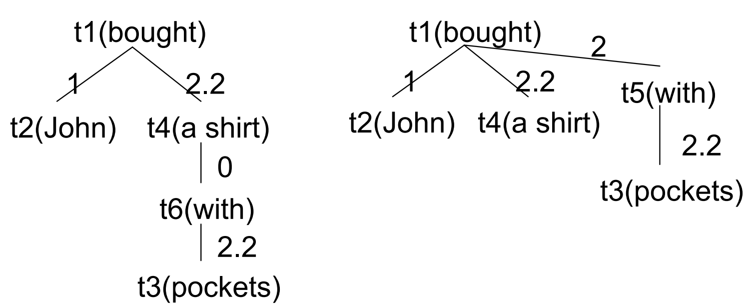
Statistical parsing

- Statistical parsing = ambiguity resolution using machine learning
- S = sentence, T = derivation tree
- Find best parse: $\underset{T}{\arg\max}\, P(T,S)$

P(T,S) is a generative model: it contains parameters that generate the input string

Statistical Parsing with TAG





P(tree1, *John bought a shirt with pockets*)=
P(t1) * P(t1#0, NONE | t1#0) *
P(t1#1, t2 | t1#1) *
P(t1#2.2, t4 | t1#2.2) *
P(t1#2, NONE | t1#2) *
P(t2#0, NONE | t2#0) *
P(t4#0, t6 | t4#0) *
P(t6#0, NONE | t6#0) *
P(t6#2, NONE | t6#2) *
P(t6#2.2, t3 | t6#2.2) *
P(t3#0, NONE | t3#0)

P(tree2, *John bought a shirt with pockets*)=
P(t1) * P(t1#0, NONE | t1#0) *
P(t1#1, t2 | t1#1) *
P(t1#2.2, t4 | t1#2.2) *
P(t1#2, t5 | t1#2) *
P(t2#0, NONE | t2#0) *
P(t4#0, NONE | t4#0) *
P(t5#0, NONE | t5#0) *
P(t5#2, NONE | t5#2) *
P(t5#2.2, t3 | t5#2.2) *
P(t3#0, NONE | t3#0)

Statistical Parsing with TAG

$$\underset{T}{\arg\max}\, P(T,S)$$

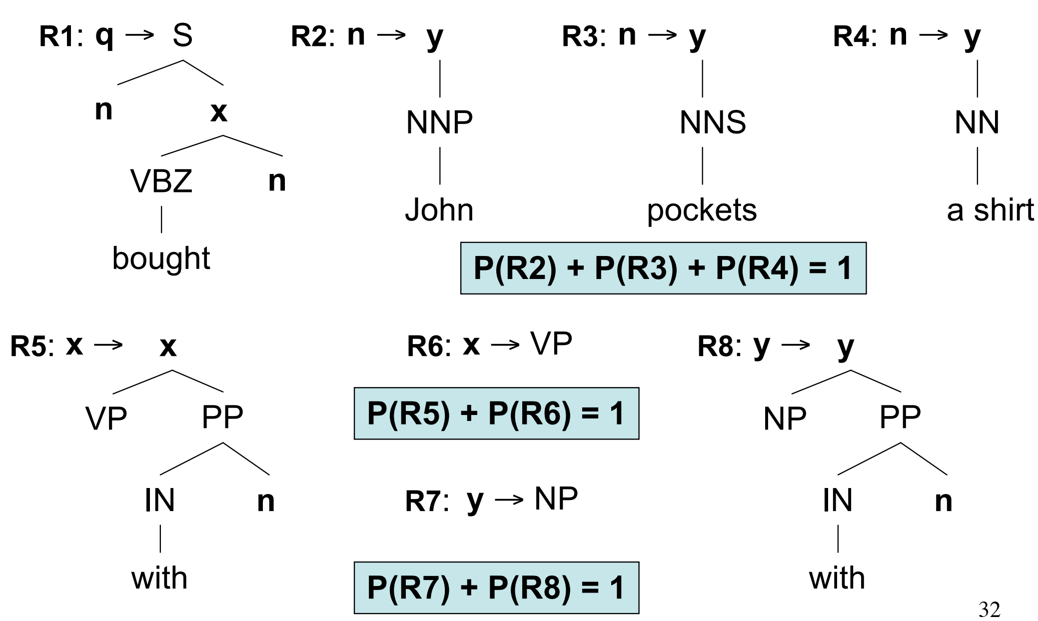
- PCFG
- Let tree T be built out of *r* CFG rules
- Note that in both PCFG and Prob. TAG, T is the *derivation tree*
- (in contrast with DOP models)
- Find all T for given S in $O(G^2n^3)$
- For lexicalized CFG: $O(n^5)$

$$P(T,S) = \prod_{i=1}^{r} P(LHS_i \rightarrow RHS_i \mid LHS_i)$$

- Prob. TAG
- Let tree T be built using *r* elementary trees, $t_1 \ldots t_r$
- Let there be *s* nodes where substitution can happen
- And *a* nodes where adjunction can happen
- Find all T for given S in $O(n^6)$

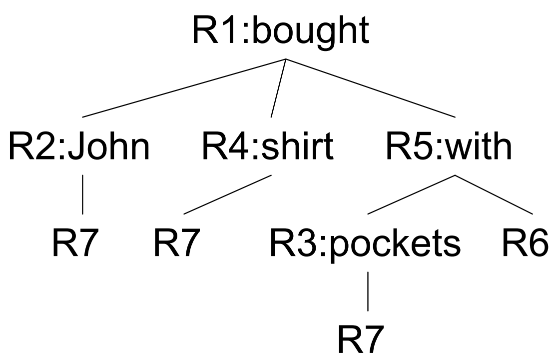
$$P(T,S) = p(i) \times \prod_{i=1}^{s} P(i,t \mid i) \times \prod_{j=1}^{a} P(j, \{t, \text{NONE}\} \mid j)$$

Statistical Parsing with ATGs

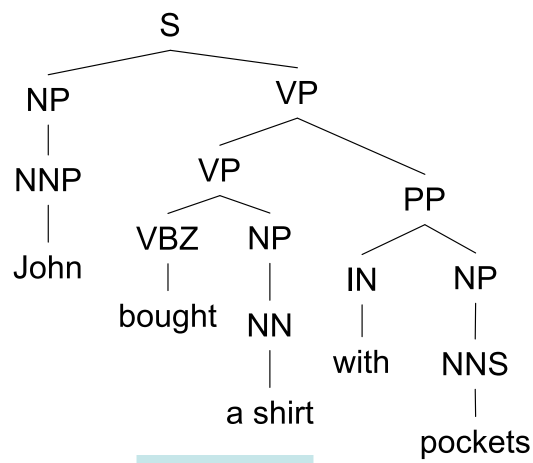


Probabilities are not bi-lexical!

P(R1) * P(R2) * P(R4) * P(R5) * P(R3) * P(R6) * (3 * P(R7))

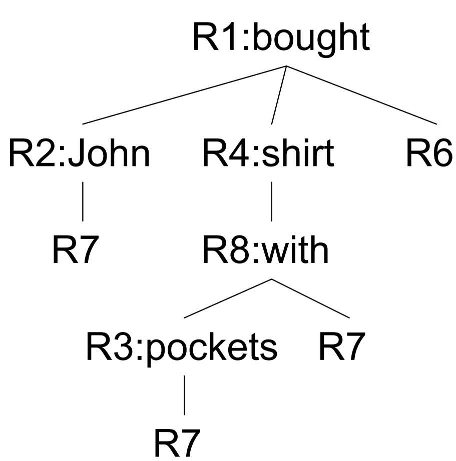


derivation tree

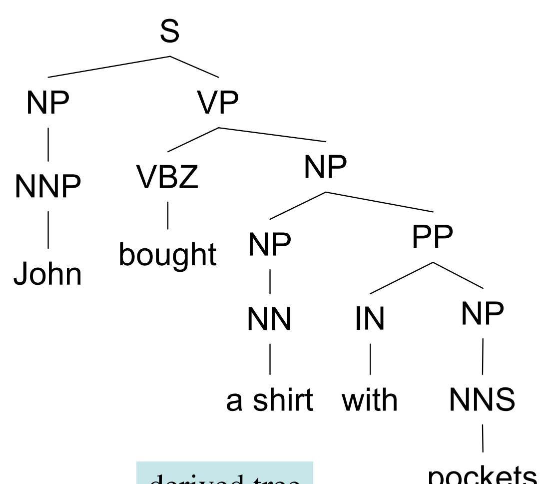


derived tree

P(R1) * P(R2) * P(R4) * P(R8) * P(R3) * P(R6) * (3 * P(R7))



derivation tree



derived tree

Summary

- Adjoining Tree Grammars = tree recognizers
- ATGs are weakly equivalent to TAG
- ATGs generate some tree languages not possible using TAG
- ATGs sit in between regular tree grammars and context-free tree grammars
- ATGs do not have adjoining constraints: makes it much easier to teach TAGs

Summary

- Even though ATGs recognize trees, it is possible to use them to parse strings
- ATGs simplify proofs of TAG closure properties (without constraints!)
- Probabilistic ATG ≠ Probabilistic TAG

Bibliography

(Lang, 1994): B. Lang. Recognition can be harder than parsing. Computational Intelligence Vol 10, No 4, Nov 1994.

(Thatcher, 1967): J. W. Thatcher, *Characterizing derivation trees of context-free grammars through a generalization of finite-automata theory*, J. Comput. Sys. Sci., 1 (1967), pp. 317-322

(Rounds, 1970): W. C. Rounds, *Mappings and grammars on trees*, Math. Sys. Theory 4 (1970), pp. 257-287

(Rounds, 1973): William C. Rounds. Complexity of recognition in intermediate-level languages. In 14th Annual IEEE Symposium on Switching and Automata Theory, pages 145-158. 1973.

(Peters & Ritchie, 1969): P. S. Peters and R. W. Ritchie, *Context sensitive immediate constituent analysis -- context-free languages revisited*, Proc. ACM Symp. Theory of Computing, 1969.

(Joshi & Levy, 1977): A. K. Joshi and L. S. Levy, *Constraints on Structural Descriptions: Local Transformations*, SIAM J. of Comput. 6(2), June 1977.

(May & Knight, 2006): J. May and K. Knight, *Tiburon: a weighted automata toolkit*, In Proc. CIAA, Taipei, 2006.

Bibliography

(Graehl & Knight, 2004): J. Graehl and K. Knight, *Training tree transducers*, In Proc. of HLT-NAACL, Boston, 2004.

(Joshi, 1994): A. K. Joshi, *From Strings to Trees to Strings to Trees ...*, Invited Talk at ACL'94, June 28, 1994.

(Joshi & Schabes, 1997): Joshi, A.K. and Schabes, Y.; Tree-Adjoining Grammars, in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa (eds.), Vol. 3, Springer, Berlin, New York, 1997, 69 - 124.

(Kanazawa, 2006): Makoto Kanazawa. *Mathematical Linguistics*, lecture notes. 2006. http://research.nii.ac.jp/~kanazawa/Courses/2006/MathLing/

(Kepser & Mönnich, 2006): Stephan Kepser and Uwe Mönnich. Closure properties of linear context-free tree languages with an application to optimality theory. Theoretical Computer Science 354, 82-97. 2006.

(Mönnich, 1997): Uwe Mönnich. Adjunction as substitution: An algebraic formulation of regular, context-free and tree adjoining languages. In Proceedings of the Third Conference on Formal Grammar. 1997.

(Fujiyoshi & Kasai, 2000): A. Fujiyoshi and T. Kasai. Spinal-formed context-free tree grammars. Theory of Computing Systems 33, 59-83. 2000.

(Fischer, 1968): Michael J. Fischer. 1968. Grammars with Macro-Like Productions. Ph.D. dissertation. Harvard University.