Extensions of Regular Tree Grammars

and their relation to Tree-Adjoining Grammars

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
- Cf. CFTGs
- Useful for ling?
- Prob version is interesting

Preliminaries

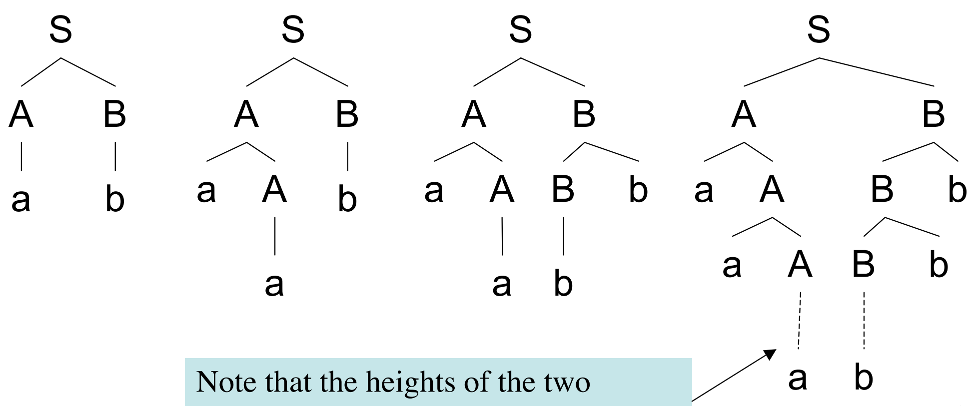
Strong vs. Weak Generative Capacity

- A property of a formal grammar, e.g. of a regular grammar or a CFG
- **Weak Generative Capacity** of a grammar is the set of strings or the *string language*
- **Strong Generative Capacity** of a grammar is the set of structures (usually the set of trees) produced by the grammar or the *tree language*

Tree Languages

S → A B
A → a A | a
B → B b | b

This grammar generates the tree language informally shown below



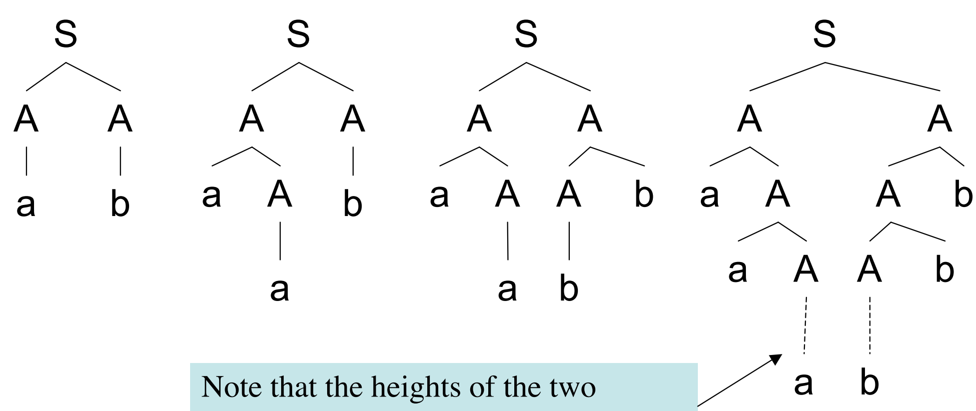
Note that the heights of the two branches do not have to be equal

Grammars that generate trees

A Tree Language with no CFG

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

~~S → A A~~
~~A → a A | a~~
~~A → A b | b~~



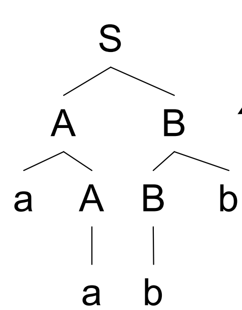
Note that the heights of the two branches do not have to be equal

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) (Brainerd, 1969) and (Rounds, 1970) provided such a generative device

S → A B
A → a A | a
B → B b | b

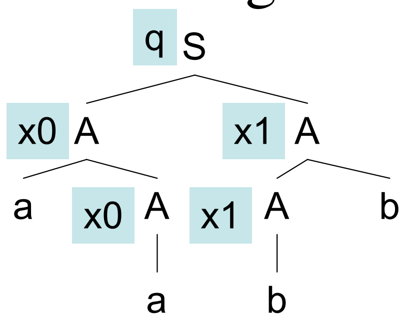
Local set: tree sets from CFGs



Map B to A

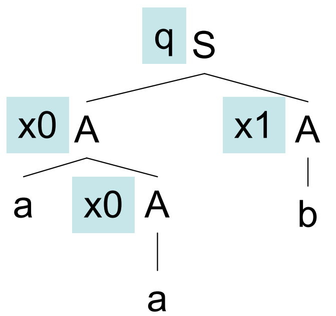
Recognizable set: local set closed under homomorphism

Regular Tree Grammars



start state: q
q → S(x0 x1)
x0 → A(a x0)
x1 → A(x1 b)
x0 → A(a)
x1 → A(b)

note: rhs 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** (May & Knight, 2006)

Regular Tree Grammars

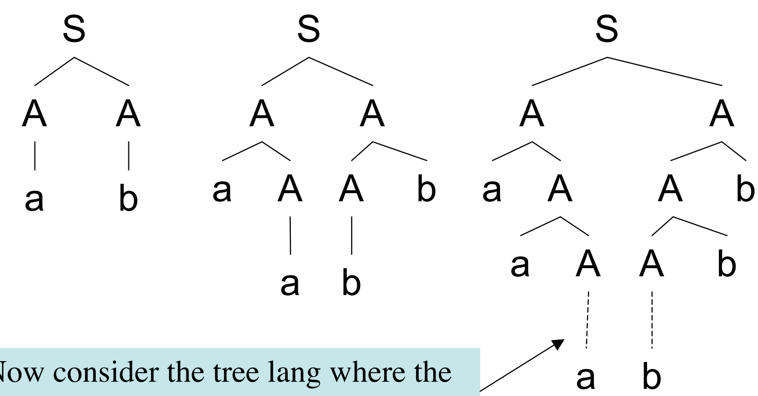
- RTGs generate tree languages
- The yield of each tree in this language produces a string
- *yield*(RTG) provides a string language
- For each RTG: *yield*(RTG) = CFL
- But the set of tree languages of CFGs is contained within that of RTGs

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(a)~~
~~x1 → A(x1 b) | A(b)~~

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

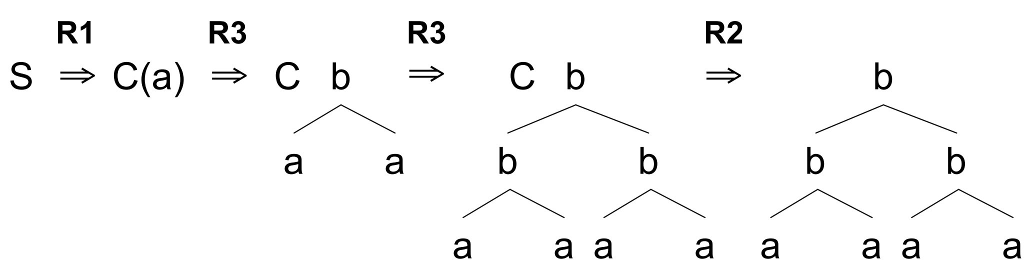


Now consider the tree lang where the depth of the two branches is **equal**

(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: does not have the constant growth property
- Also, recognition algorithm is NP-complete (Rounds, 1973)
- Perhaps there is a tree grammar formalism between RTG and CFTG?
- How much context-sensitivity over RTGs should this tree grammar have?

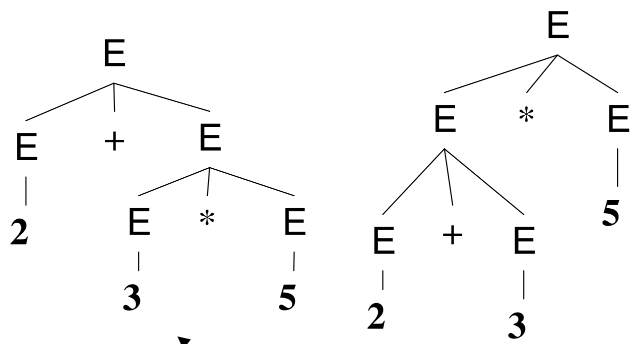
Context-sensitive predicates on trees bear less fruit than you think*

* borrowed from a title of a paper by A. Joshi

Tree Languages: Another Example

A more practical example

E → E + E
E → E * E
E → (E)
E → **N**



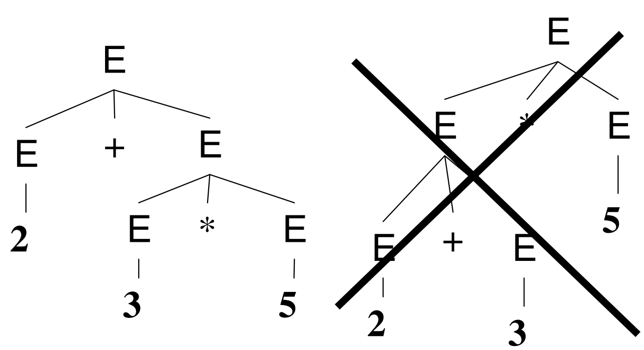
2+3*5 is ambiguous
either **17** or **25**

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!

Context-sensitive Grammars

- Rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$ where γ cannot be the empty string, also written as $A \rightarrow \gamma \,/\, \alpha\_\_\beta$
- CSGs are very powerful: they can generate languages like $\{ 1^p : p \text{ is prime} \}$
- This kind of computational power is unlikely to be needed to describe natural languages
- Like other grammar formalisms in the Chomsky hierarchy CSGs generate string sets
- What if they are used to recognize tree sets?

Context-sensitive predicates

- Consider each CSG rule $A \rightarrow \gamma \,/\, \alpha\_\_\beta$ to be a predicate (i.e. either true or false)
- Apply all the rules in a CSG as predicates on an input tree
- If all predicates are true then *accept* the tree, else *reject* the tree
- Can be easily extended to a set of trees and used to accept a tree set
- Can we precisely describe this set of tree languages?

Peters-Ritchie Theorem

- The Peters-Ritchie Theorem (Peters & Ritchie, 1967) states a surprising result about the generative power of CSG predicates
- Consider each tree set accepted by CSG predicates
- Theorem: The string language of this tree set is a context-free language
- Each CSG when applied as a set of predicates can be converted into a weakly equivalent CFG
- See also: (McCawley, 1967) (Joshi, Levy & Yueh, 1972) (Rogers, 1997)

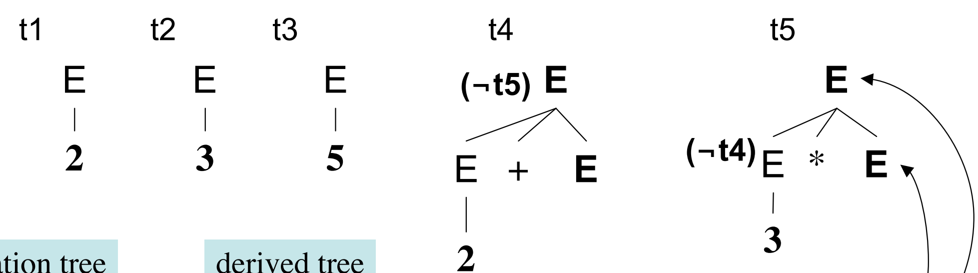
Local Transformations

- This theorem was extended by (Joshi & Levy, 1977) to handle arbitrary boolean combinations and sub-tree / domination predicates
- Proof involves conversion of all CSG predicates into top-down tree automata that accept tree sets
- (Joshi & Levy, 1977) showed transformations used in transformational grammar can be written in this way
- Important caveat: we assume some source GEN generating trees which are then validated. (connection to Optimality Theory)

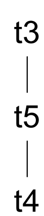
Tree-Adjoining Grammars

- Construct a tree set out of tree fragments
- Each fragment contains only the structure needed to express the locality of various CSG predicates
- Each tree fragment is called an elementary tree
- In general we need to expand even those non-terminals that are not leaf nodes: leads to the notion of adjunction

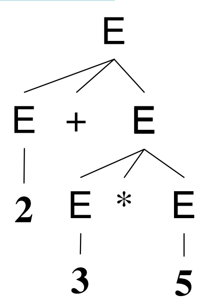
Tree-Adjoining Grammars



derivation tree



derived tree

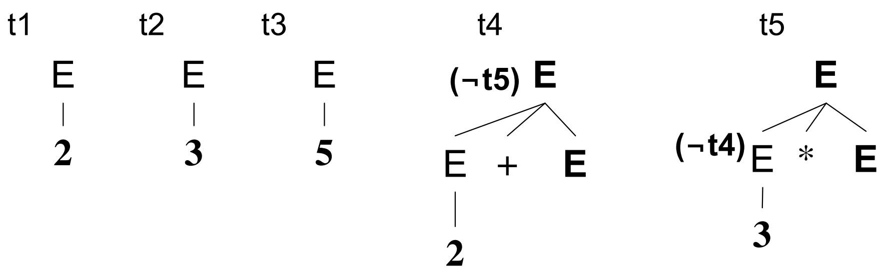


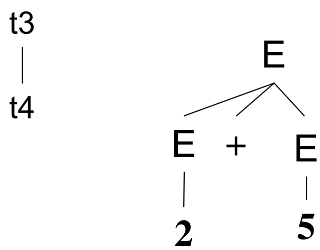
notice how the root node and frontier node are the same label

all constraints are checked locally in the tree where adjunction occurs

Input string: 2 + 3 * 5

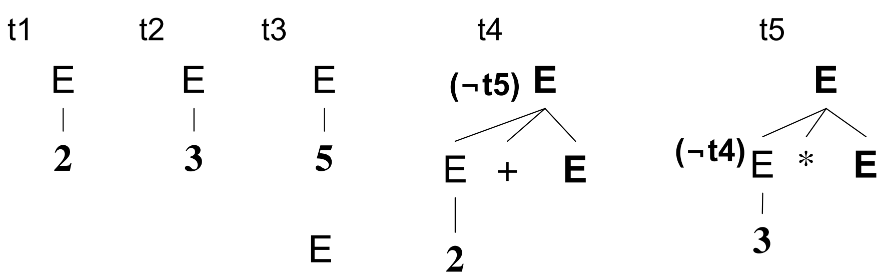
Tree-Adjoining Grammars

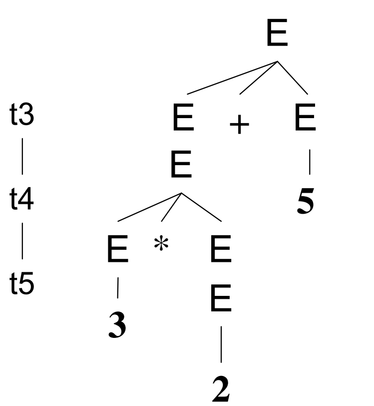




derivation for **3*2+5**

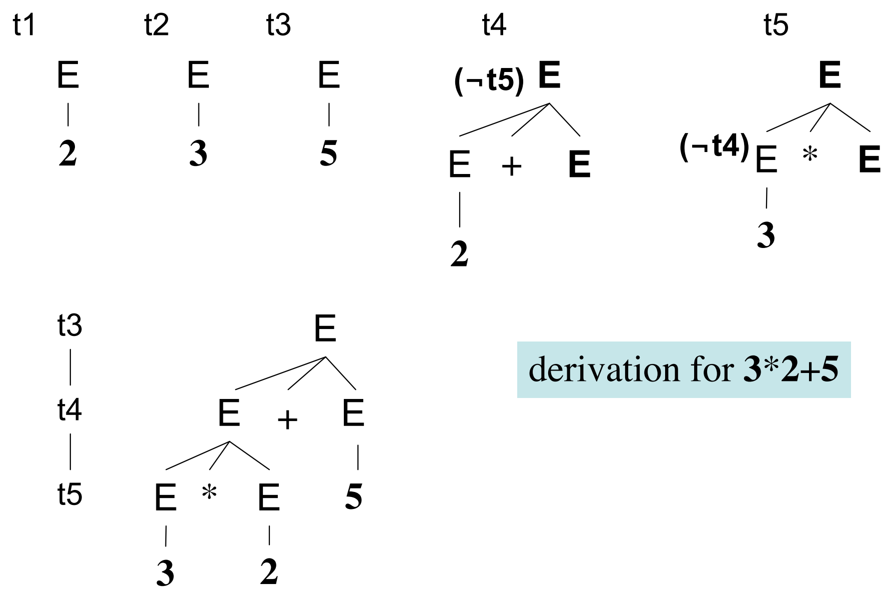
Tree-Adjoining Grammars





derivation for **3*2+5**

Tree-Adjoining Grammars



Tractable Descriptions

- Why not use context-sensitive grammars?
- For G, given a string *x* what is the complexity of an algorithm for the question: is *x* in L(G)?
  - Unrestricted Grammars/Turing machines: undecidable
  - Context-sensitive: NSPACE[n] linear non-deterministic space
  - Indexed Grammars: NP-complete
  - Tree-Adjoining Grammars: $O(n^6)$
  - Context-free: $O(n^3)$
  - Regular: $O(n)$

Other motivations for TAG

- Strong **lexicalization** of CFGs leads to the adjunction operation
- NL is probably not weakly context-free, but also possibly not fully context-sensitive and so NL could be **mildly context-sensitive**
- Some NLs clearly show **crossing dependencies**, but maybe of limited variety
- Many issues of **locality** in linguistics, e.g. constraints on long-distance dependencies, can be encoded within the notion of elementary tree

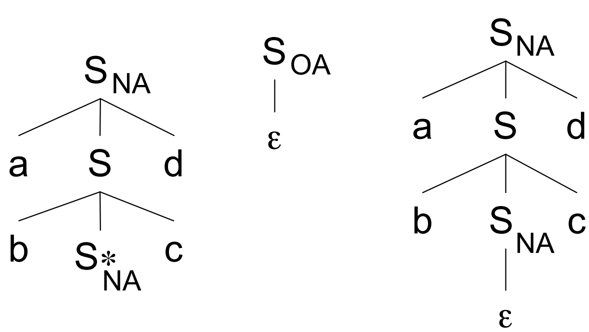
Adjunction Constraints

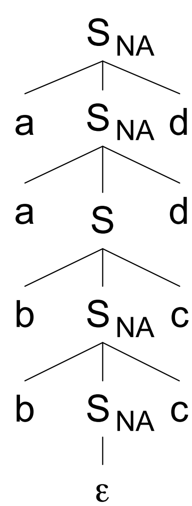
- Adjunction is the rewriting of a non-terminal in a tree with an auxiliary tree
- We can think of this operation as being “context-free”
- Constraints are essential to control adjunction: both in practice for NL syntax and for formal closure properties

Adjunction Constraints

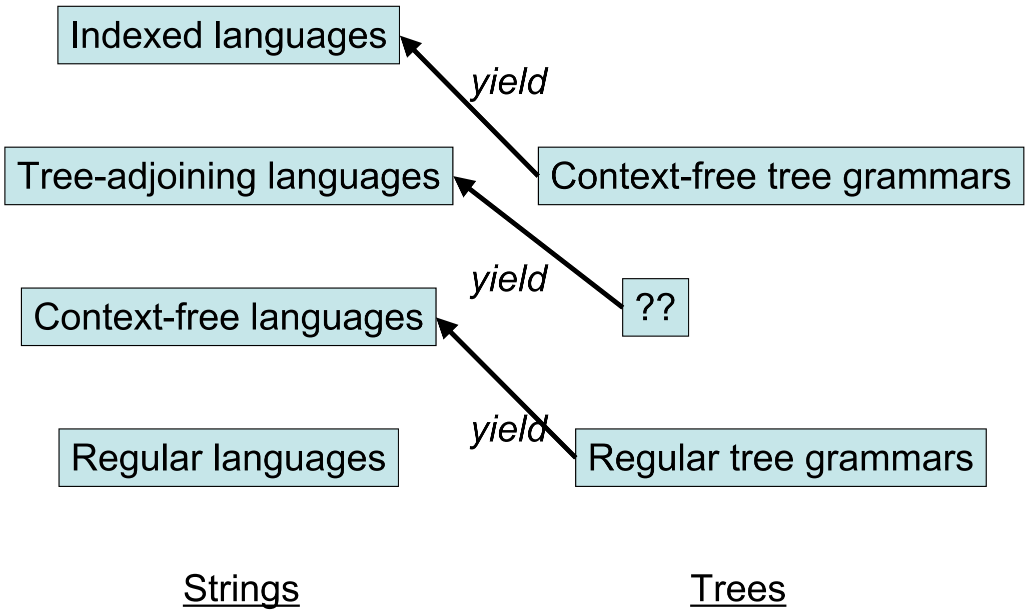
- Three types of constraints:
  - null adjunction (NA): no adjunction allowed at a node
  - obligatory adjunction (OA): adjunction must occur at a node
  - selective adjunction (SA): adjunction of a pre-specified set of trees can occur at a node

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



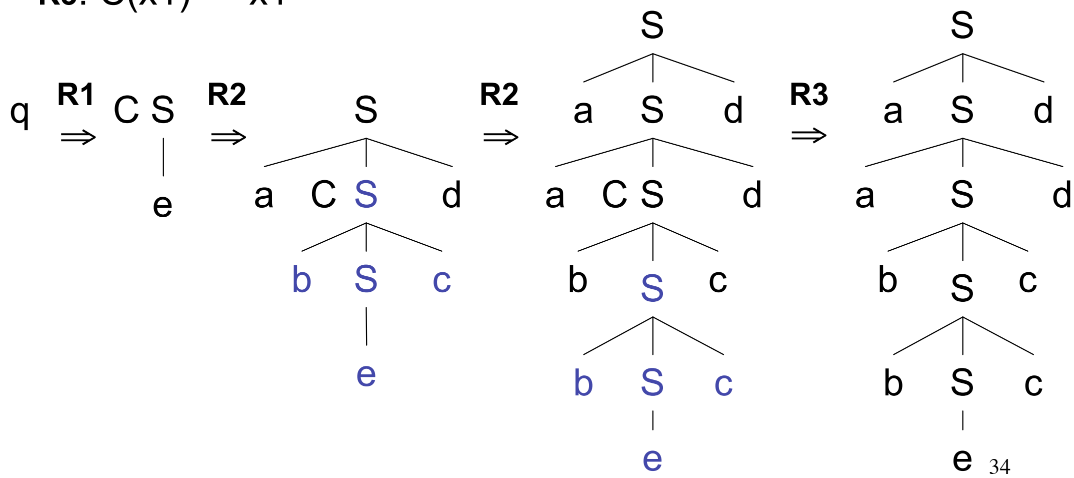
Limiting the power of Context-free tree grammars

Modifying Context-free Tree Grammars

- Simple CFTG = linear and non-deleting
- Linear = tree variables shalt not multiply
- Non-deleting = tree variables shalt not be matched on the lhs and dropped in the rhs
- The non-deleting condition can be dropped (Fujiyoshi, 2005)
- Monadic CFTG = only one subtree can be matched on the lhs of any rule, $A(x) \rightarrow T$

Monadic Simple Context-free Tree Languages

R1: q → C(S(e))
R2: C(x1) → S(a C(S(b x1 c)) d)
R3: C(x1) → x1



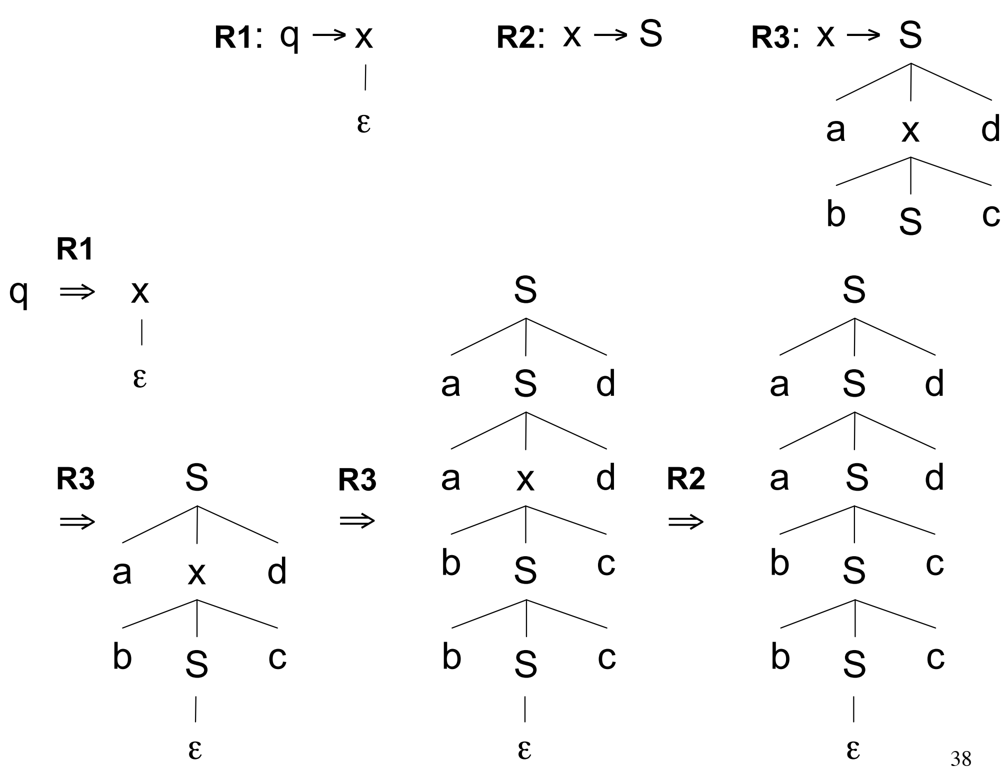
Monadic Simple CFTGs

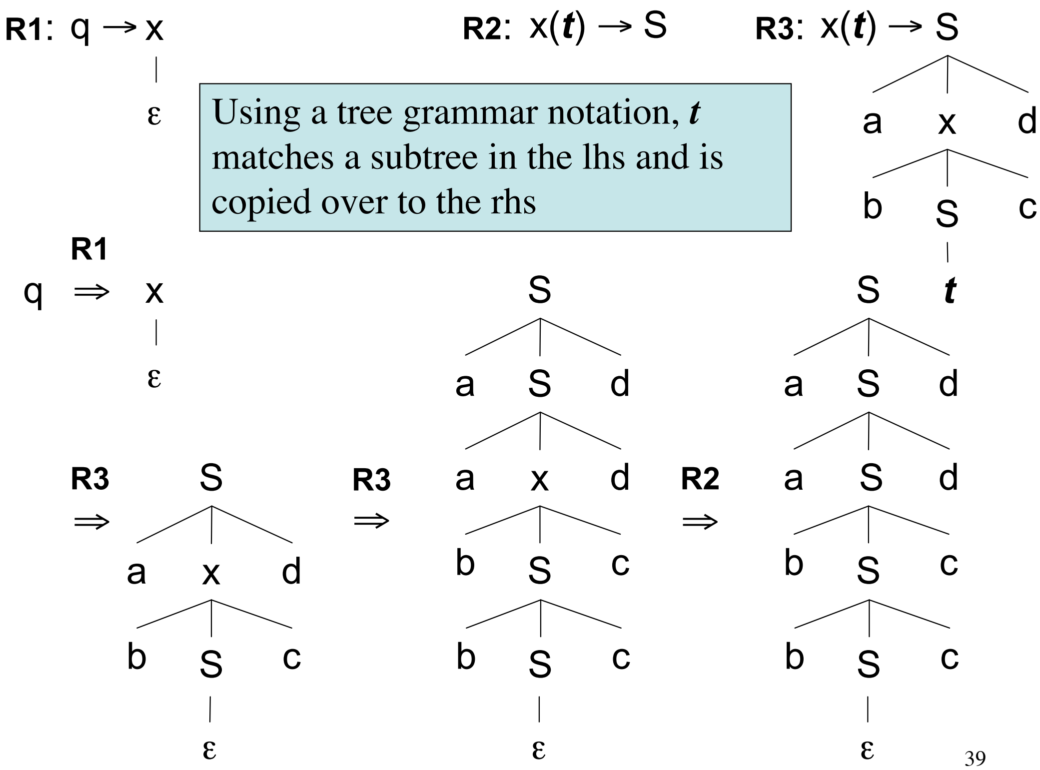
- Tree language of TAGs is contained within *monadic simple CFTGs*
- TAGs are weakly equivalent to CFTGs (Fujiyoshi & Kasai, 2000; Mönnich 1997)
- Focus of this talk: how about extending RTGs instead? (Lang, 1994)
- Another way to limit CFTGs is the so-called *spinal form CFTG* (Fujiyoshi & Kasai, 2000)

Extending the power of Regular tree grammars

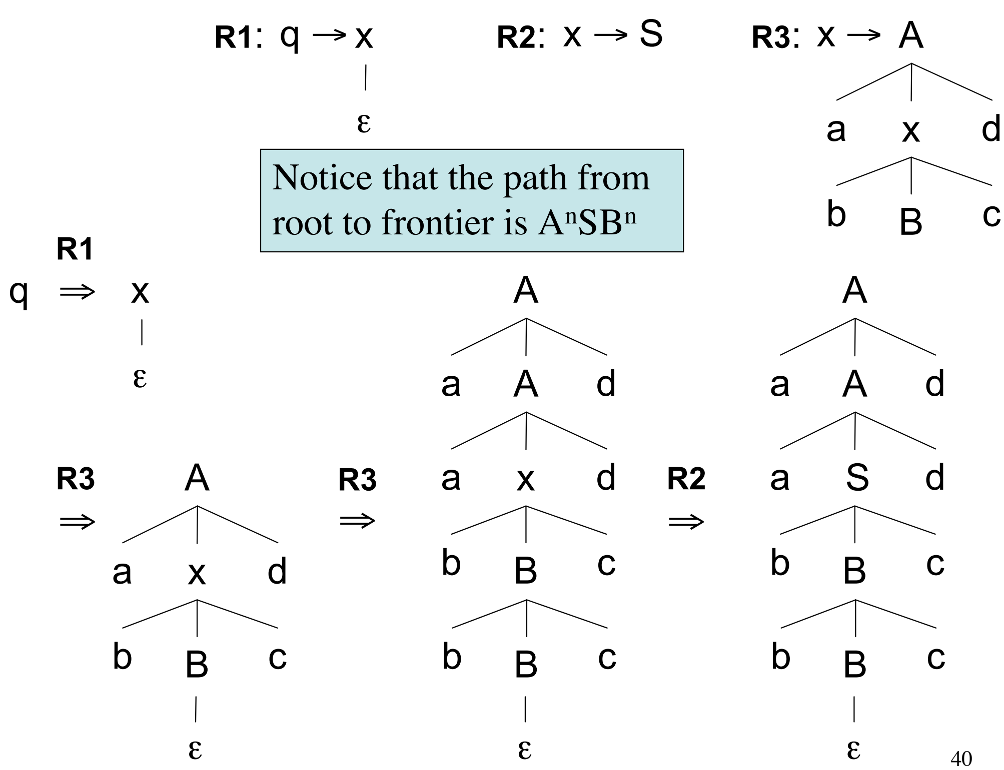
Extending RTGs: Adjoining Tree Grammars

- ATG is a tree grammar formalism
- Rules are of the form x → T; x is tree variable, T is a tree
- The rhs tree T is built with terminals a, b, … and non-terminals A, B, …
- Tree T can also contain tree variables which can be internal nodes dominating a single subtree (unlike RTGs where they occur on the frontier)
- Finally, ATGs have a start tree variable
- An ATG is well-formed if for every sentential form *w* (q ⇒* *w*) is a well-formed tree.





R1: q → x
R2: x(t) → S
R3: x(t) → S
Using a tree grammar notation, t matches a subtree in the lhs and is copied over to the rhs
R1
R3
R3
R2



R1: q → x
R2: x → S
R3: x → A
Notice that the path from root to frontier is AnSBn
R1
R3
R3
R2

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

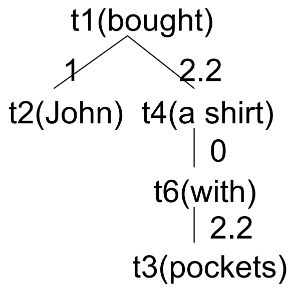
- Closure properties for TALs (union, concat, homomorphism, substitution) can be shown using ATGs instead of TAGs.
  - By taking yield of the tree language
  - Without using adjunction constraints
- Intersection with regular languages (Lang, 1994)
- What about pumping lemma? cf. (Kanazawa, 2006)
- Polynomial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

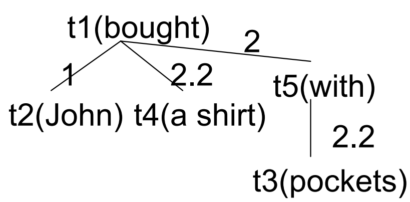
ATGs and monadic simple CFTGs

- Are ATGs strongly equivalent to monadic simple CFTGs?
- First step: what is strong equivalence?
- For each m.s. CFTG construct an ATG that produces the same tree set, and vice versa
- Shown by (Kepser & Rogers, 2007)

Ambiguity Resolution

Ambiguity Resolution



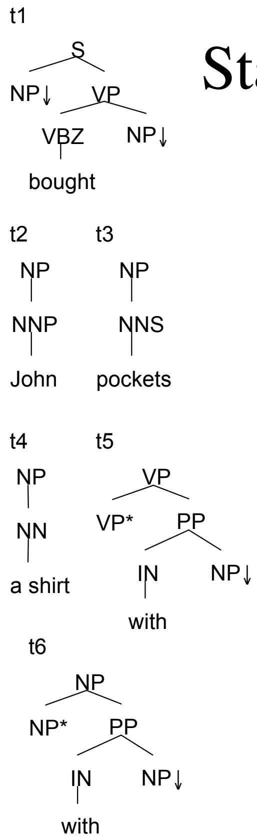


- Two possible derivations for ***John bought a shirt with pockets.***
- One of them is more plausible than the other.
- Statistical parsing is used to find the most plausible derivation.

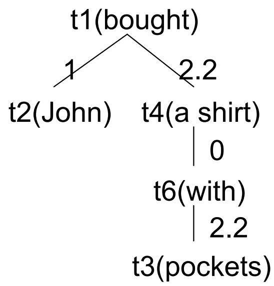
Statistical parsing

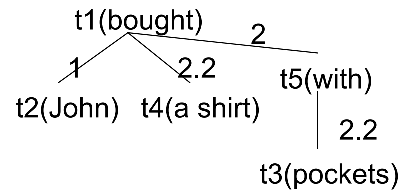
- Statistical parsing = ambiguity resolution using machine learning
- S = sentence, T = derivation tree
- Find best parse: $\underset{T}{\arg\max}\, P(T,S)$

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

$$\arg\max_T P(T,S)$$

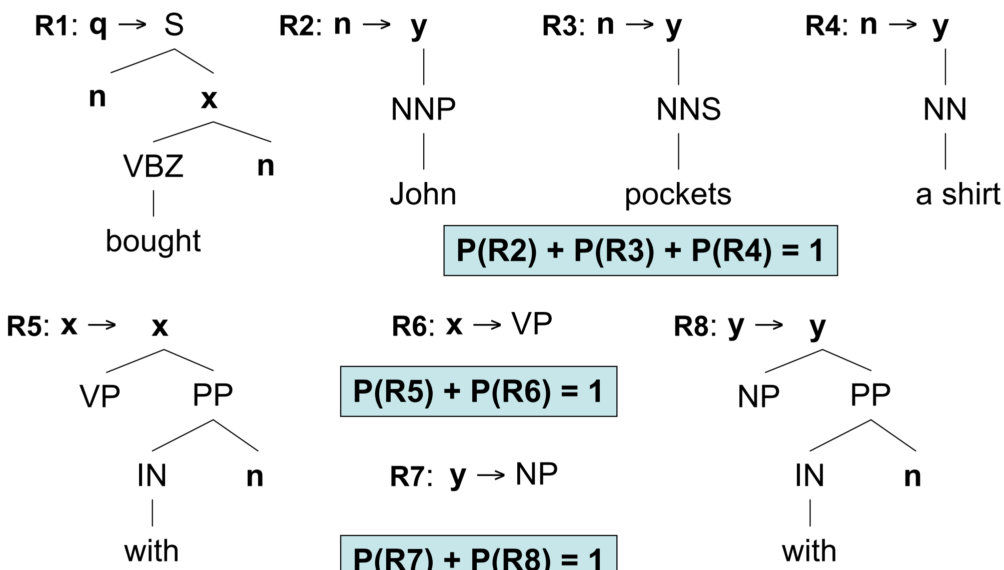
- PCFG
- Let tree T be built out of *r* CFG rules
- Note that in both PCFG and Prob. TAG, T is the *derivation tree*
- (in contrast with DOP models)
- Find all T for given S in $O(G^2n^3)$
- For lexicalized CFG: $O(n^5)$

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

- 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 $O(n^6)$

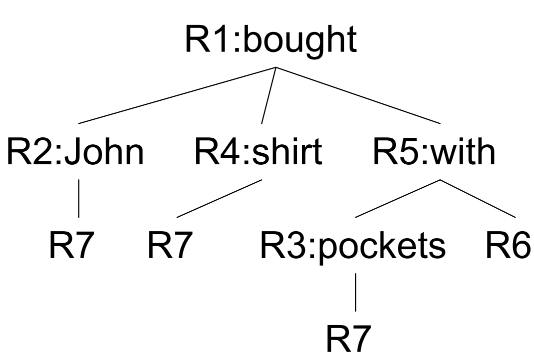
$$P(T,S) = p(i) \times \prod_{i=1}^{s} P(i,t \mid i) \times \prod_{j=1}^{a} P(j,\{t,\textsc{none}\} \mid j)$$

Statistical Parsing with ATGs

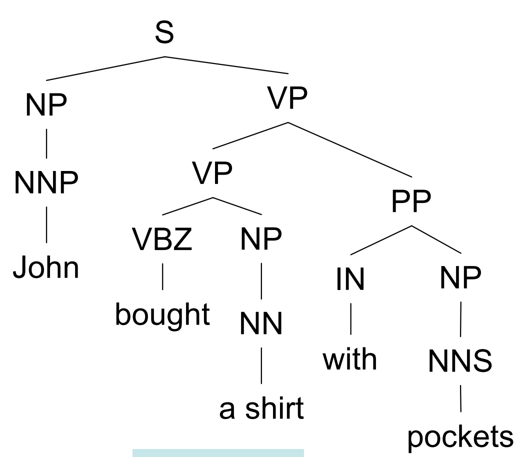


Probabilities are not bi-lexical!

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

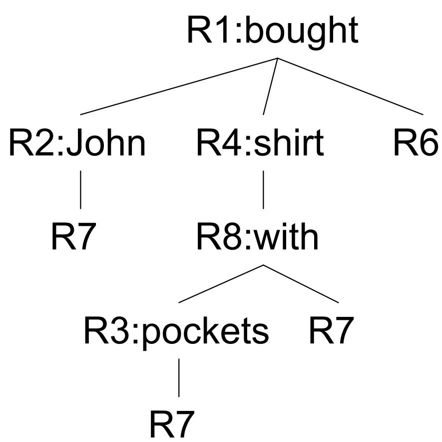


derivation tree

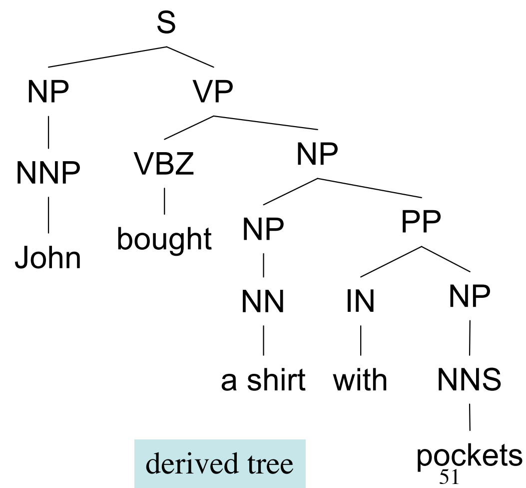


derived tree

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

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

(Brainerd, 1969): W. S. Brainerd, Tree generating regular systems, *Inf. and Contr.* 14 (1969), 217-231

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

(Fujiyoshi, 2005): A. Fujiyoshi. Linearity and nondeletion on monadic context-free tree grammars. *Inf. Proc. Lett.* 93 (2005), 103-107.

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

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

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

Bibliography

(Kepser & Mönnich, 2006): S. Kepser and U. Mönnich. Closure properties of linear CFTLs with an application to optimality theory. *Theor. Comp. Sci.* 354, 82-97. 2006.

(Kepser & Rogers, 2007): S. Kepser and J. Rogers. The equivalence of TAG and Monadic Linear CFTGs. In MOL-10. Jul 28-30, 2007.

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

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

(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.

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

Bibliography

(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.

(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