Derivation trees (1)

TAG derivations are described by derivation trees. For each derivation in a TAG there is a corresponding derivation tree. This tree contains:

- nodes for all elementary trees used in the derivation, and
- edges for all adjunctions and substitutions performed throughout the derivation.

Whenever an elementary tree $\gamma$ was attached to the node at address $p$ in the elementary tree $\gamma'$, there is an edge from $\gamma'$ to $\gamma$ labeled with $p$.

(For the tree addresses, Gorn addresses are used: The root has address $\epsilon$ (or 0), and the $i$th daughter of the node with address $p$ has address $pi$.)

Derivation trees (2)

Example:

derivation tree for the derivation of John sometimes laughs

$\Rightarrow$ Semantic dependency graph?

Derivation trees

- are context-free, and
- uniquely determine the derived tree.

$\Rightarrow$ TAG is a linear context-free rewriting system, LCFRS
Linguistic analyses with LTAG

- Linguistic role of the adjunction/substitution dualism
- Design principles for elementary trees
  - Lexicalization
  - Extended domain of locality
  - Factoring of recursion
  - Elementary tree minimality

Lexicalization

Each elementary tree has at least one non-empty lexical item, its lexical anchor.

$\Rightarrow$ All widely used grammar formalisms support some kind of lexicalization!

Reasons for lexicalization:
- The properties of a constituent depend on the lexical items occurring in the constituent: the structure of a VP depends on the subcategorization properties of its verb.

Syntax in LTAG

Elementary tree minimality

- Elementary trees contain slots only for the complements of their lexical head.
- Elementary trees are projections of lexical items.
- Factoring of recursion

Example: to-infinitives

Example:

(1) John gives a book to Mary

Extended domain of locality

- All dependencies are represented in elementary trees.
- Locality constraints (constraints for UDC, island constraints) $\rightarrow$ adjunction constraints

(2) a. who$_i$ did John tell Sam that Bill likes $t_i$
 b. who$_i$ did John tell Sam that Mary said that Bill likes $t_i$

Syntax in LTAG
Principles related to semantics

Predicate argument cooccurrence:
Each elementary tree associated with a predicate contains substitution nodes for each of its arguments.

Semantic anchoring:
Elementary trees are not semantically void (to, that.)

Compositional principle:
An elementary tree corresponds to a single semantic unit.

Functional elements

Besides lexical predicates, there are functional elements (complementizers, determiners, auxiliaries, negation) whose treatment in LTAG is less clear. They can be
- either in separate elementary trees (e.g., XTAG grammar)
- or in the elementary tree of the lexical item they are associated with.

Sample derivations

Complements: NPs, PPs, adjectives, clauses (raising, controlling)
Adjuncts: adjectives, particles, relative clauses

Complements ⇒ Substitution, Adjuncts ⇒ Adjunction ?

Sample derivations - Complements (1)

(3) John buys Bill a book

Elementary trees:

Derivation tree:

1 22 23
John Bill a book
(4) Bill hopes that John wins

(5) John seems to like Bill

(6) John expects [ Bill to win ]

(7) John expected [Mary to make a comment]

(8) whom does John expect to come?
to make a comment: make and comment in the same elementary tree since they form a light verb construction:

Sample derivations - Complements (3)

to make a comment: make and comment in the same elementary tree since they form a light verb construction:

Sample derivations - Adjuncts (1)

(9) the good student participated in every course during the semester

(10) the dog [who ate the cake]

Extraposed relative clauses:

(11) Somebody; lives nearby [who; has a CD-burner].

Derivation trees = Semantic dependency structure?

The derivation tree is not always the semantic dependency structure:

(12) John claims Bill is likely to win
• TAG derivations are described by derivation trees.
• In LTAG, elementary trees for lexical predicates contain slots for all arguments of these predicates, for nothing else. Recursion is factored away.
• The derived tree describes the constituent structure while the derivation tree is close to a semantic dependency graph.