Constructing a Construction Grammar with LTAG:
Linguistic and Computational Perspectives

Timm Lichte & Laura Kallmeyer

University of Düsseldorf, Germany

ICCG8, September 03, 2014
LTAG (= Lexicalized Tree-Adjoining Grammar)
- one of the major grammar formalisms (Müller, 2014)
- rich history, dates back to 1975 (Joshi et al., 1975)
- originally developed by engineers, further studied by theoretical computer scientists and computational linguists, finally discovered by linguists
- large implemented grammars for several languages (e.g. XTAG at UPenn)
- parsers, implementation tools, grammar induction tools, ...

Construction Grammar?
- not really in the focus of the LTAG community so far
- and that’s surprising given the rather obvious connections!
Aims and overview

Aims of this talk:

- present Lexicalized Tree-Adjoining Grammar (LTAG) as a grammar formalism that shares central ideas with (some versions of) Construction Grammar (CxG):
  1. grammatical constructions
  2. only surface structure: no transformational or derivational component
  3. a network of constructions “which nodes are related by inheritance links” (Goldberg, 2013)

- show that it substantially differs from other explicit implementations of CxG, namely Sign-based Construction Grammar (SBCG), and Fluid Construction Grammar (FCG).
LTAG: basic ingredients

- a set of **elementary trees**
- two combinatorial operations:
  - substitution (replace a leaf node)
  - adjunction (replace an inner node)
a set of **elementary trees**

two combinatorial operations:
  - substitution (replace a leaf node)
  - adjunction (replace an inner node)
LTAG: long distance dependencies

By virtue of adjunction, cases of long-distance dependencies can be immediately captured:

(1) Who does Mary say sometimes walks into the house.

\[
\begin{align*}
& S \\
& \quad \text{AUX} \quad S \\
& \quad \quad \text{does} \\
& \quad \quad \text{NP} \\
& \quad \quad \text{VP} \\
& \quad \quad \quad \text{V} \\
& \quad \quad \quad \quad \text{S*} \\
& \quad \quad \quad \quad \quad \text{say} \\
\end{align*}
\]

\[
\begin{align*}
& S \\
& \quad \text{NP} \quad S \\
& \quad \quad \epsilon \\
& \quad \quad \text{VP} \\
& \quad \quad \quad \text{V} \\
& \quad \quad \quad \quad \text{walks} \\
\end{align*}
\]

only surface structure
LTAG and frames

Kallmeyer & Osswald (2013):

- **lexicon**: **pairs of elementary trees and frames** (= typed feature structures)
- Elementary trees are enriched with **interface features**, which contain base labels from the frame representation.
  - unification of interface features $\leadsto$ unification of frames
- parallel composition of derived trees and larger frames

```
S_{E=\square}
  NP_{I=\square}  VP_{E=\square}
    VP_{E=\square}  PP_{I=\square, E=\square}
      V_{E=\square}
      walked

[bounded-locomotion]
  [ bounded-locomotion ]
  [ ACTOR 1 ]
  [ MOVER 1 ]
  [ GOAL 2 ]
  [ PATH path ]
  [ MANNER walking ]
```


(2) John walked into the house.
(2) John walked into the house.
(2) John walked into the house.

Lichte & Kallmeyer (Düsseldorf)
(2) John walked into the house.
(2) John walked into the house.

Nice, but where are the constructions ???
Constructions in LTAG

Elementary trees:

\[
S_{[E=\square]} \\
NP_{[I=\square]} \\
\quad VP_{[E=\square]} \\
\quad \quad VP_{[E=\square]} \\
\quad \quad \quad PP_{[I=\square, E=\square]} \\
\quad \quad \quad \quad V_{[E=\square]} \\
\quad \quad \quad \quad \quad walked
\]

\[
\begin{bmatrix}
\text{bounded-locomotion} \\
\text{ACTOR} & 1 \\
\text{MOVER} & 1 \\
\text{GOAL} & 2 \\
\text{PATH} & \text{path} \\
\text{MANNER} & \text{walking}
\end{bmatrix}
\]
Constructions in LTAG

Elementary trees with multiple lexical anchors:

\[ S_{[E = 0]} \]
\[ NP_{[I = 1]} \]
\[ VP_{[E = 0]} \]
\[ V \]
\[ NP \]
\[ \textit{kicked} \]
\[ D \]
\[ \textit{the} \]
\[ N_{[E = 0]} \]
\[ \textit{bucket} \]
\[ 0 \text{dying} \]
\[ \text{PATIENT 1} \]
Constructions in LTAG

Lexical anchoring:

\[
\begin{array}{c}
\text{walked} \\
\text{locomotion} \\
\text{bounded-translocation}
\end{array}
\]

\[
\begin{array}{c}
\text{actor} \\
\text{mover} \\
\text{manner}
\end{array}
\]

↑ argument structure construction
Constructions in LTAG

Transitive motion construction:

(3) John rolls the ball into the goal
Constructions in LTAG

Dative alternation: DO and PO construction

(4) John gives/sends Mary the book
Constructions in LTAG

(5) John gives/sends the book to Mary

grammatical constructions

a network of constructions ???
Inheritance hierarchies and metagrammatical factorization

- In order to produce and maintain a consistent LTAG of a considerable coverage, one uses a **metagrammar** (MG, Candito 1996; Crabbé & Duchier 2005).
- An MG contains factorized descriptions of unanchored elementary trees. It defines a set of tree fragments (MG classes) that can be used in other MG classes.
- This way, an unanchored elementary tree family is the denotation of an MG class that makes use of a series of other, smaller tree fragments in the MG.

![S NP VP](NP V\^ NP)

![S NP VP](S VP \^ VP NP)

- Advantage of MGs for TAG from a linguistic point of view: The MG allows to express and implement lexical generalizations.
Inheritance hierarchies and metagrammatical factorization

Class hierarchy in the MG (fragment):
Points of comparison

Fundamental distinction between two classes of grammar frameworks:

- **limited domain of locality (LDL)**
  - list-like valency that is processed stepwise
  - movement, type raising, valency merge
  - examples: CG, (binarized) HPSG, SBCG, MG

- **extended domain of locality (EDL)**
  - set-like valency without predetermined order
  - capability to immediately access arbitrarily distant parts of a sentence within one lexical entry or syntactic rule
  - examples: LTAG, RRG, *some* versions of CxG, Dependency Grammar

Another recently discussed distinction that is orthogonal:

- lexical vs. phrasal (Müller & Wechsler, 2014)
Comparison

Lexicalized Tree-Adjoining Grammar:
- EDL
- tree rewriting + unification of typed feature structures
- inheritance network based on classes of the metagrammar

Sign-based construction grammar:
- LDL
- constraint-based architecture à la HPSG
- inheritance network based on types

Fluid Construction Grammar:
- EDL
- “match” (of conditional parts) and “merge” (of contributitional parts) on non-functional untyped feature structures
- no inheritance, but conditioned unifiability
Summary

LTAG incorporates central ideas of CxG:

- only surface structure ✓
- grammatical constructions ✓
- inheritance network of constructions ✓

LTAG differs substantially from other implementations of CxG.

⇒ different empirical predictions or theoretical ramifications?


