TuLiPA –

Parsing Mildly Context-Sensitive Languages

with Range Concatenation Grammars

Laura Kallmeyer
University of Tübingen

Stuttgart, 14 February 2008

Overview

Introduction (1)

- The project: Emmy-Noether Project "A Lexicalized Tree-Adjoining Grammar for a Fragment of German Focussing on Syntax and Semantics"
- Members: Laura Kallmeyer, Timm Lichte, Wolfgang Maier, Yannick Parmentier
- Implementation: TuLiPA, a parser for mildly context-sensitive (MCS) grammar formalisms.

Introduction (2)

Mildly context-sensitive formalisms (Joshi 87) have the following properties:

1. They can describe a limited amount of cross-serial dependencies.
2. They are polynomially parsable.
3. Their string languages are of constant growth.

Introduction (3)

Closest characterization of MCS via a formalisms: linear context-free rewriting systems (LCFRS, Weir 88), equivalent to set-local MCTAG and to simple RCG.

Idea of TuLiPA: instead of providing different parsers for a range of formalisms, we use RCG as a pivot formalism, i.e.,

- a given MCS grammar is first transformed into an equivalent simple RCG,
- then this RCG is used for parsing,
- and then the output is transformed into an analysis in the original MCS formalism.

Formalisms: TAG (1)

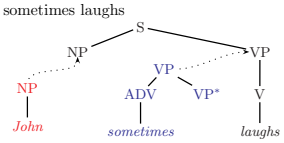
Tree Adjoining Grammars (TAG): Tree-rewriting system: set of *elementary* trees with two operations:

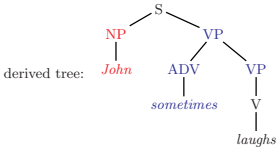
- adjunction: replacing an internal node with a new tree. The new tree is an auxiliary tree and has a special leaf, the foot node.
- substitution: replacing a leaf with a new tree. The new tree is an initial tree

A good LTAG introduction: Joshi & Schabes (1997)

Formalisms: TAG (2)

(1) John sometimes laughs



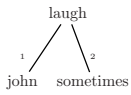


Formalisms: TAG (3)

TAG derivations are described by derivation trees that contain

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

Example: derivation tree for the derivation of (1) *John sometimes laughs*



Formalisms: TAG (4)

Important features of LTAG:

- Grammar is lexicalized
- Recursive parts are put into separate elementary trees that can be adjoined (Factoring of recursion, FR)
- Elementary trees can be arbitrarily large, in particular (because of FR) they can contain elements that are far apart in the final derived tree (Extended domain of locality)

Formalisms: TAG (5)

Elementary trees for natural languages:

- The elementary tree of a lexical predicate contains slots for all arguments of the predicate, for nothing more.
- 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.

Formalisms: TT-MCTAG (1)

Multicomponent TAGs (MCTAGs, Weir 1988):

- consist of sets of elementary trees, so-called multicomponents;
- If a multicomponent is used in a derivation, all its members must be used.
- Linguistic motivation: desire to split the contribution of a single lexical item (e.g., a verb and its arguments) into several elementary trees.

Tree-tuple MCTAG with shared nodes (TT-MCTAG, Lichte 2007):

Each set has one lexicalized tree, the head and a set of auxiliary trees, the arguments. We write it as a tuple $\langle \gamma, \{\beta_1, \ldots, \beta_n\}\rangle$.

Formalisms: TT-MCTAG (2)

Derivation in TT-MCTAG:

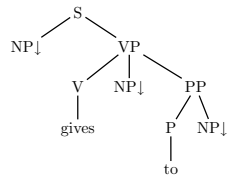
For each tuple $\langle \gamma, \{\beta_1, \ldots, \beta_n\}\rangle$ used during the derivation, each argument tree β_i must

- either adjoin directly to its head γ, or
- be linked by a chain of adjunctions at root nodes to a tree that adjoins to γ.

Formalisms: TAG (6)

Example: elementary tree for *gives to*

(2) John gives a book to Mary

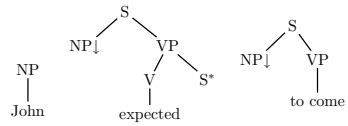


Formalisms: TAG (7)

Example: elementary trees for ECM verbs

(3) John expected Mary to come

expected selects for a subject NP and an infinitival sentence:



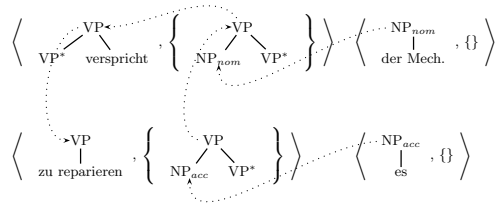
The sentential object is realised as a foot node in order to allow extractions:

(4) whom does John expect to come?

Formalisms: TT-MCTAG (3)

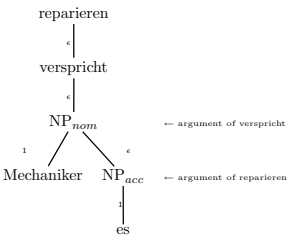
(5) ... dass es der Mechaniker zu reparieren verspricht

TT-MCTAG analysis:



Formalisms: TT-MCTAG (4)

Derivation tree:



Formalisms: TT-MCTAG (5)

- The recognition problem for TT-MCTAG is NP-hard (Søgaard et al. 2007).
- Therefore we define an additional limitation:
  TT-MCTAG are of rank k if, at any time during the derivation, at most k argument trees depending on higher head trees in the derivation tree are still waiting for adjunction.

RCG: Definition (1)

A Range Concatenation Grammar (RCG, Boullier 2000) consists of

- a finite set of predicates N, each with a fixed arity,
- finite sets of terminals T and variables V,
- a start predicate $S \in N$ of arity 1, and
- a set P of productions, so-called *clauses*, of the form
  $A_0(x_{01}, \ldots, x_{0a_0}) \rightarrow \epsilon$
  or $A_0(x_{01}, \ldots, x_{0a_0}) \rightarrow A_1(x_{11}, \ldots, x_{1a_1}) \ldots A_n(x_{n1}, \ldots, x_{na_n})$
  with $x_{ij} \in (T \cup V)^*$.

An RCG with maximal predicate arity n is called *of arity n*.

RCG: Definition (2)

- When applying a clause with respect to a string $w = t_1 \ldots t_n$, the arguments of the predicates in the clause are instantiated with substrings of w, more precisely with the corresponding ranges $\langle i, j \rangle$.
- An instantiation of a clause maps variables and terminals to ranges such that each terminal t is mapped to a range containing an occurrence of t.
- In each derivation step, the left-hand side of an instantiation of some clause is replaced by its right-hand side.
- The language of an RCG G is the set of strings that can be reduced to the empty word, i.e., $\{w \mid S(\langle 0, |w| \rangle) \stackrel{*}{\Rightarrow} \epsilon$ with respect to $w\}$.

RCG: Definition (3)

Example: RCG with string language: $\{a^n b^k a^n \mid k, n \in \mathbb{N}\}$: Clauses

$S(X\,Y\,Z) \rightarrow A(X, Z)\,B(Y)$ $\quad A(a\,X, a\,Y) \rightarrow A(X, Y)$ $\quad A(\epsilon, \epsilon) \rightarrow \epsilon$

$B(b\,X) \rightarrow B(X)$ $\quad B(\epsilon) \rightarrow \epsilon$

Sample derivation for $w = aabaa$:

$$\begin{array}{ccccccccc} S(X & & Y & & Z) & \rightarrow & A(X, & Z) & B(Y) \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & \downarrow & \downarrow \\ \langle 0,2\rangle & & \langle 2,3\rangle & & \langle 3,5\rangle & & \langle 0,2\rangle & \langle 3,5\rangle & \langle 2,3\rangle \\ aa & & b & & aa & & aa & aa & b \end{array}$$

With this instantiation, $S(\langle 0, 5\rangle) \Rightarrow A(\langle 0, 2\rangle, \langle 3, 5\rangle)B(\langle 2, 3\rangle)$. Then

RCG: Definition (4)

$$\begin{array}{ccccc} B(b & & X) & \rightarrow & B(X) \\ \downarrow & & \downarrow & & \downarrow \\ \langle 2,3\rangle & & \langle 3,3\rangle & & \langle 3,3\rangle \\ b & & \epsilon & & \epsilon \end{array}$$ and $B(\epsilon) \rightarrow \epsilon$

lead to $A(\langle 0, 2\rangle, \langle 3, 5\rangle)B(\langle 2, 3\rangle) \Rightarrow A(\langle 0, 2\rangle, \langle 3, 5\rangle)B(\langle 3, 3\rangle) \Rightarrow$ $A(\langle 0, 2\rangle, \langle 3, 5\rangle)$.

$$\begin{array}{ccccccccc} A(a & & X & & a & & Y) & \rightarrow & A(X, & Y) \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \downarrow \\ \langle 0,1\rangle & & \langle 1,2\rangle & & \langle 3,4\rangle & & \langle 4,5\rangle & & \langle 1,2\rangle & \langle 4,5\rangle \\ a & & a & & a & & a & & a & a \end{array}$$

leads to $A(\langle 0, 2\rangle, \langle 3, 5\rangle) \Rightarrow A(\langle 1, 2\rangle, \langle 4, 5\rangle)$. Then

RCG: Definition (5)

$$\begin{array}{ccccccccc} A(a & & X & & a & & Y) & \rightarrow & A(X, & Y) \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & \downarrow \\ \langle 1,2\rangle & & \langle 2,2\rangle & & \langle 4,5\rangle & & \langle 5,5\rangle & & \langle 2,2\rangle & \langle 5,5\rangle \\ a & & \epsilon & & a & & \epsilon & & \epsilon & \epsilon \end{array}$$ and

$A(\epsilon, \epsilon) \rightarrow \epsilon$

lead to $A(\langle 1, 2\rangle, \langle 4, 5\rangle) \Rightarrow A(\langle 2, 2\rangle, \langle 5, 5\rangle) \Rightarrow \epsilon$

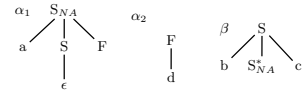
From TAG to RCG (1)

General idea of the transformation (Boullier, 1998):

- The RCG contains predicates $\langle\alpha\rangle(X)$ and $\langle\beta\rangle(L, R)$ for initial and auxiliary trees respectively.
- X covers the yield of α and all trees added to α, while L and R cover those parts of the yield of β (including all trees added to β) that are to the left and the right of the foot node of β.
- The clauses reduce the argument(s) of these predicates by identifying those parts that come from the elementary tree α/β itself and those parts that come from one of the elementary trees added by substitution or adjunction.

From TAG to RCG (2)

TAG:



Equivalent RCG:

$S(X) \rightarrow \langle\alpha_1\rangle(X) \mid \langle\alpha_2\rangle(X)$

$\langle\alpha_1\rangle(aX) \rightarrow \langle\alpha_2\rangle(X)$

$\langle\alpha_1\rangle(aLRX) \rightarrow \langle\beta\rangle(L, R)\langle\alpha_2\rangle(X)$

$\langle\beta\rangle(Lb, cR) \rightarrow \langle\beta\rangle(L, R)$

$\langle\alpha_2\rangle(d) \rightarrow \epsilon \quad \langle\beta\rangle(b, c) \rightarrow \epsilon$

From TT-MCTAG to RCG (1)

Idea: construct the RCG for the underlying TAG and enrich the predicate names.

- There are predicates $\langle\gamma\rangle$ for the elementary trees (not the tree sets).
- The predicates are enriched in a way that allows to keep track of the "still to adjoin" argument trees.
- The yield of a γ-predicate contains not only γ and its arguments but also arguments of predicates that are higher in the derivation tree and that are adjoined below γ.

From TT-MCTAG to RCG (2)

We have three kinds of predicates:

1. $\langle\gamma, LPA\rangle$ with LPA being the list of pending arguments.
   $\langle\gamma, LPA\rangle$-clauses distribute the yield variables among corresponding *adj* and *sub* predicates, pass the LPA to the root-position *adj* predicate and distribute the args of γ among the LPAs of all *adj* predicates.
2. $\langle adj, \gamma, dot, LPA\rangle$ with LPA containing a) the list of pending args if $dot = \epsilon$, and b) arguments of γ.
   $\langle adj, \gamma, dot, LPA\rangle$-clauses adjoin a γ' to the dot in γ. If $\gamma' \in LPA$, then it is removed from the LPA.
3. $\langle sub, \gamma, dot\rangle$ as intermediate predicates.
   $\langle sub, \gamma, dot\rangle$-clauses substitute a γ' into dot in γ.

From TT-MCTAG to RCG (3)

$\langle$ VP(VP*, verspricht) , { VP(NP$_{nom}$, VP*) } $\rangle$ $\langle$ NP$_{nom}$(der Mech.) , {} $\rangle$

$\langle$ VP(zu reparieren) , { VP(NP$_{acc}$, VP*) } $\rangle$ $\langle$ NP$_{acc}$(es) , {} $\rangle$

$\langle\alpha_r, \emptyset\rangle(L\ zu\ reparieren\ R) \rightarrow \langle adj, \alpha_r, \epsilon, \{\beta_{acc}\}\rangle(L, R)$

$\langle adj, \alpha_r, \epsilon, \{\beta_{acc}\}\rangle(L, R) \rightarrow \langle\beta_{acc}, \emptyset\rangle(L, R) \mid \langle\beta_v, \{\beta_{acc}\}\rangle(L, R)$

$\langle\beta_{acc}, \emptyset\rangle(L\ X, R) \rightarrow \langle adj, \beta_{acc}, \epsilon, \emptyset\rangle(L, R)\langle sub, \beta_{acc}, 1\rangle(X)$

$\langle sub, \beta_{acc}, 1\rangle(X) \rightarrow \langle\alpha_{es}, \emptyset\rangle(X)$ $\quad$ $\langle\alpha_{es}, \emptyset\rangle(es) \rightarrow \epsilon$

$\langle\beta_v, \{\beta_{acc}\}\rangle(L, verspricht\ R) \rightarrow \langle adj, \beta_v, \epsilon, \{\beta_{nom}, \beta_{acc}\}\rangle(L, R)$

...

The parser: general architecture (1)

The TuLiPA parser can be run accepting a TT-MCTAG (as a special case a TAG) or an RCG. In TT-MCTAG-mode,

- the input is a non-anchored grammar (in xml format),
- a lemma file,
- a morphological file,
- and the input sentence.

The parser: general architecture (2)

The parser

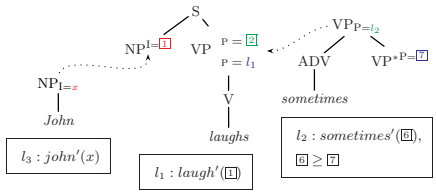
- creates the TT-MCTAG (or TAG) for the input sentence,
- transforms it into an RCG,
- parses the input with the RCG using a non-directional top-down algorithm (Boullier),
- transforms the RCG parse forest into a TT-MCTAG derivation forest, and
- extracts the single derivation trees.

The parser: semantics (1)

The parser allows to process semantics (Gardent & Kallmeyer 2003, Kallmeyer & Romero 2008):

- in the TT-MCTAG each elementary tree can be linked to a semantic representation containing meta-variables for values that will be obtained form other representations during semantic computation;
- identifications of these meta-variables with possible values is done via the feature unifications already implemented in TAG;
- in addition, scope constraints allow to generate underspecified representations.

The parser: semantics (2)



$\rightsquigarrow$ $l_3 : john(x)', l_1 : laugh'(x), l_2 : sometimes'(6)$, $6 \geq l_1$

Conclusion

- TuLiPA is a multiformalism parser that currently supports TAG, TT-MCTAG and RCG.
- Central idea: RCG as pivot formalism.
- Transformations from various other formalisms into RCG have been defined and might be integrated into TuLiPA.

Url of TuLiPA:
`http://www.sfb441.uni-tuebingen.de/emmy-noether-kallmeyer/tulipa`

Future work

Further optimizations planned for the near future:

- POS tagging.
- Use patterns for RCG clauses that get instantiated only during parsing. (Avoids the current explosion in the number of clauses due to the distribution of argument trees among LPAs.)

Research planned for the next years:

- Apply probabilistic techniques to TuLiPA to make it more robust.
- Extract probabilistic RCGs from corpora.
- Transform underspecified semantic representations into normal dominance constraints and disambiguate with utool.