Tree-Adjoining Grammars: Theory and implementation

Day 5: Parsing TAG

Kata Balogh & Simon Petitjean
Heinrich-Heine-Universität Düsseldorf, Carl von Ossietzky Universität Oldenburg

NASSLLI 2025

June 23 – 27, 2025

University of Washington, Seattle

Outline

Overview

Last sessions

Mon: Motivation and the basic TAG

Tue: Linguistic applications and using LTAG: syntax

Wed: Linguistic applications and using LTAG: semantics

The following sessions

Wed: Introduction to grammar engineering and XMG

Thu: Grammar implementation with XMG

Fri: Parsing TAG

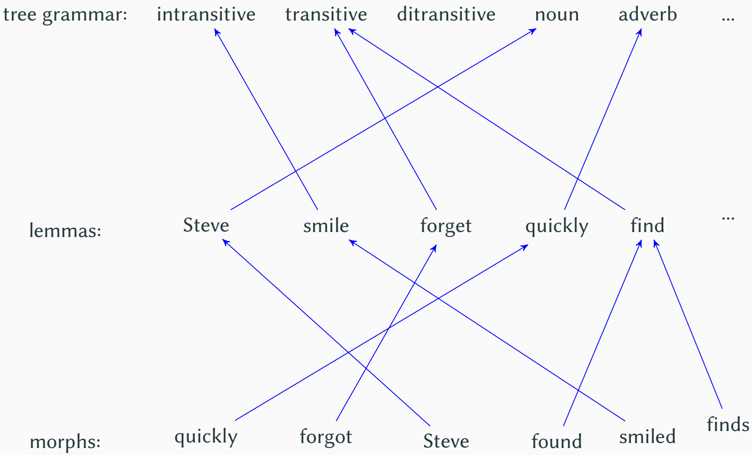
Parsing with TuLiPA

- TuLiPA Arps & Petitjean [1] and Parmentier et al. [2]: parser for Tree Adjoining Grammars and Tree Wrapping Grammars
- Syntax + (frame-based) semantics
- Input: grammar and sentence to parse
- Output: all derivations that can be derived by combining the elementary trees
- Standard CYK algorithm
- Bottom-up, left-to-right traversion of the derived tree
- Available on Github (`https://github.com/spetitjean/TuLiPA-frames`)

Lexicalization

- Lexicalized grammar: all rules contain one lexical item
- Computational interest: parsing with limited number of trees
- Lexical anchoring: separate lexical items from the tree templates
- Trees contain a lexical anchor leaf marked with $\diamond$

- Architecture: inspired by the XTAG grammar XTAG Research Group [3]
- Tree family: all syntactic environments where an anchor can appear
- 2-level lexicon:
    - Lemmas, mapped to a tree family
    - Inflected forms, mapped to a lemma


tree grammar:
intransitive
transitive
ditransitive
noun
adverb
...
lemmas:
Steve
smile
forget
quickly
find
...
morphs:
quickly
forgot
Steve
found
smiled
finds

Structure of the lexicon

- Parsing with XMG → TuLiPA

- 3 levels to describe the syntax-lexicon interface
  - Grammar: set of tree templates organized in families
  - Lexicon of lemmas, mapped to tree families
  - Lexicon of inflected forms, mapped to lemmas

- First step of parsing: lexical anchoring → select all the trees of the grammar which could be used for parsing the current input

- For each word of the sentence, find the corresponding lemma(s). Add the tree families bound to these lemmas to the set of trees to consider.

XMG-2: several compilers

- Build a set of syntactic trees (using the `<syn>` dimension), possibly paired with semantic frames (`<frame>` dimension) → `synframe` compiler
- Describe the mapping from lemmas to tree families (using the `<lemma>` dimension) → `lex` compiler
- Describe the mapping from inflected forms to lemmas (using the `<morpho>` dimension) → `mph` compiler
- Once the 3 files are written, compile them to produce the lexicalized grammar
- Input of TuLiPA: grammar (3 XML files), axiom (root of the expected derived tree(s)), input sentence
- Also available online: `http://xmg.phil.hhu.de/index.php/upload/tulipa`

The <lemma> dimension

Every entry is a feature structure with specific features:

- `entry`: the lemma (as a string)
- `fam`: the tree family it is mapped to
- (`sem`: the semantic frame corresponding to the lemma)

And possibly standard morphosyntactic features (`cat`, etc.)

```
class LemmaBig
  {
  <lemma>{
    entry <- "big";
    sem <- FrameBig;
    cat <- a;
    fam <- Adj
  }
}
```

The <lemma> dimension: example with abstractions

```
class Determiner
{
  <lemma>{
    cat   <- d;
    fam   <- Det
  }
}

class LemmaThe
import Determiner[]
{
  <lemma> {entry <- "the"}
}

class LemmaA
import Determiner[]
{
  <lemma> {entry <- "a"}
}
```

The <morpho> dimension

Specific features:

- `morph`: the inflected form (string)
- `lemma`: the lemma this form is mapped to (string)

Other features can be defined:

- `cat`
- `wh`, `rel` (`bool`)
- `agr` $\longrightarrow$ `thirdsg` (`bool`), `num` (`sg`, `pl`)
- `mode` (`base`, `ind`)
- `tnsval` (`past`, etc.)

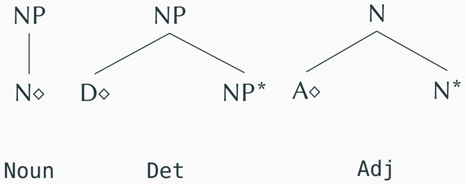
The <morpho> dimension: example

```
class Noun
{
  <morpho>{
    cat   <- N
  }
}

class SingularNoun
import Noun[]
{
  <morpho>{
    agr <- [num = sg,
                thirdsg = +]
  }
}
```

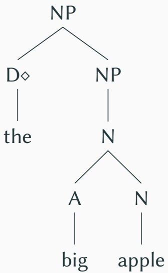
```
class Form[?Morph, ?Lemma]
{
  <morpho>{
    lemma <- ?Lemma;
    morph <- ?Morph
  }
}

class MorphApples
import PluralNoun[]
{
  Form["apples","apple"];
}
```

Example: nouns and determiners

- Minimal grammar: tree for nouns, tree for determiners, tree for adjectives
- Lexicon: 5 nouns (apple, apples, John, who what), 3 determiners (a, an, the), 1 adjective (big)



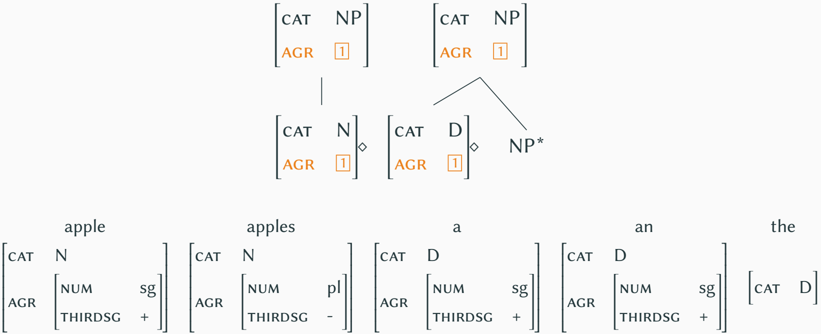
Example: nouns and determiners



Agreement constraints

- Try parsing "a big apples"
- Solution: agr feature
- Example: $\begin{bmatrix} \textsc{agr} & \begin{bmatrix} \textsc{num} & \text{sg} \\ \textsc{thirdsg} & + \end{bmatrix} \end{bmatrix}$

Agreement constraints



Obligatory adjunction constraint

- Try parsing "apple"
- Solution: `det` feature

Obligatory adjunction constraint

- Try parsing “apple”
- Solution: `det` feature
- Idea: a determiner must adjoin at the `NP` node above singular nouns

Obligatory adjunction constraint

- Try parsing “apple”
- Solution: `det` feature
- Idea: a determiner must adjoin at the `NP` node above singular nouns
- What about “apples”?

Obligatory adjunction constraint

- Try parsing "apple"
- Solution: `det` feature
- Idea: a determiner must adjoin at the `NP` node above singular nouns
- What about "apples"?

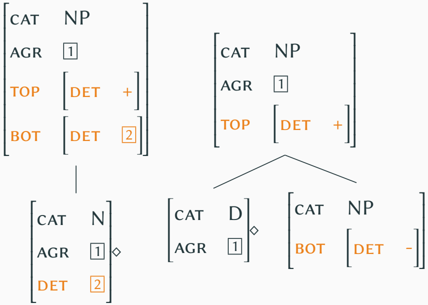
apple

$$\begin{bmatrix} \textsc{cat} & \text{N} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \text{sg} \\ \textsc{thirdsg} & + \end{bmatrix} \\ \textsc{det} & - \end{bmatrix}$$

apples

$$\begin{bmatrix} \textsc{cat} & \text{N} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \text{pl} \\ \textsc{thirdsg} & + \end{bmatrix} \end{bmatrix}$$

Obligatory adjunction constraint



[1] Arps, David & Simon Petitjean. 2018. **A Parser for LTAG and Frame Semantics.** In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis & Takenobu Tokunaga (eds.), *Proceedings of the eleventh international conference on language resources and evaluation (lrec 2018).* Miyazaki, Japan: European Language Resources Association (ELRA).

[2] Parmentier, Yannick, Laura Kallmeyer, Wolfgang Maier, Timm Lichte & Johannes Dellert. 2008. **TuLiPA: A syntax-semantics parsing environment for mildly context-sensitive formalisms.** In *Proceedings of the ninth international workshop on Tree Adjoining Grammars and related formalisms (TAG+9)*, 121–128. Tübingen, Germany.

[3] XTAG Research Group. 2001. ***A Lexicalized Tree Adjoining Grammar for English.*** Tech. rep. Philadelphia, PA: Institute for Research in Cognitive Science, University of Pennsylvania.