TuLiPA: an extension for Frames

David Arps, Simon Petitjean

TreeGraSP Meeting #1

Introduction

- Resources using LTAG + frames, developed with XMG-2
- No parser for these resources
- TuLiPA: parser for LTAG/TTMCTAG
- Our goal: extending TuLiPA for this new parsing task
- Available online:
  `https://github.com/spetitjean/TuLiPA-frames`

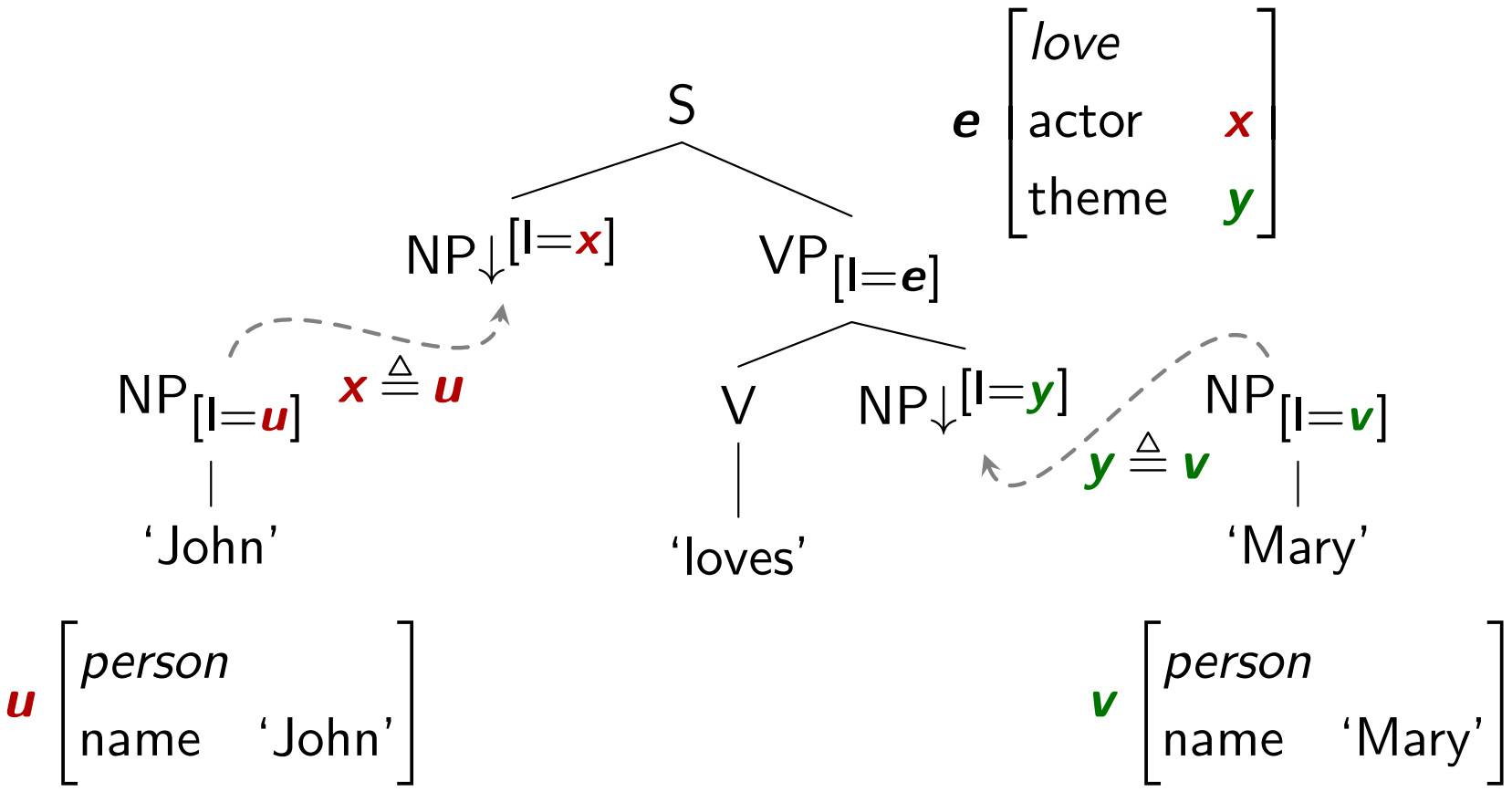
Plan

LTAG/frames parsing

- Grammar: Pairs of LTAG trees and semantic frames [Kallmeyer and Osswald, 2013]
- Linking syntactic structure (elementary trees) and semantic roles (frames)
- Linking variables in feature structures at tree nodes and in frames
- Unify variables to build semantic representation

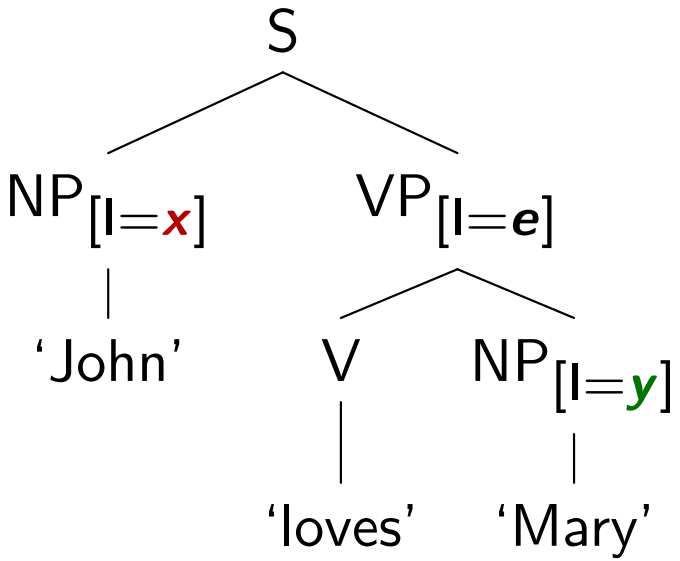
LTAG/frames parsing

Example: 'John loves Mary'



LTAG/frames parsing

Example: 'John loves Mary'



$$e\begin{bmatrix} love & & \\ \text{actor} & x & \begin{bmatrix} person & \\ \text{name} & \text{'John'} \end{bmatrix} \\ \text{theme} & y & \begin{bmatrix} person & \\ \text{name} & \text{'Mary'} \end{bmatrix} \end{bmatrix}$$

TuLiPA: before

- Parser for (TTMC)TAG
- RCG conversion, polarity filtering [Kallmeyer et al., 2008]
- Predicate semantics and untyped feature structures
  [Gardent and Kallmeyer, 2003], [Kallmeyer and Romero, 2008]
- Written in Java, Graphical user interface
- 3 levels lexicon:
  - tree templates, generated by XMG
  - lemmas generated by LexConverter
  - inflected forms generated by LexConverter

TuLiPA: now

- CYK for TAG parser
- Frame semantics
  - Implemented type hierarchy
  - Separate frame and tree input
- 3-4 levels lexicon, generated by XMG-2

Implementation of the semantic parsing

- Every tree comes with a list of frames
- Adjunction/substitution $\rightarrow$ concatenate the lists of frames and unify

The architecture

- Elementary trees, distributed in families
- Lemma lexicon
- Morphological lexicon

The architecture

- Elementary trees, distributed in families
- Lemma lexicon
- Morphological lexicon
- Semantic frames

The architecture

- Elementary trees, distributed in families (Tree description language)
- Lemma lexicon
- Morphological lexicon
- Semantic frames

The architecture

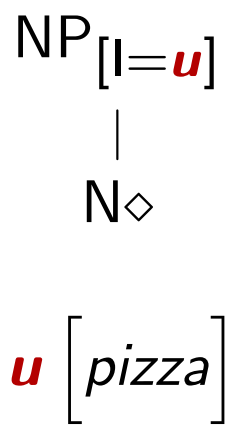
- Elementary trees, distributed in families (Tree description language)
- Lemma lexicon
- Morphological lexicon
- Semantic frames (Typed feature structures description language)

The architecture

- Elementary trees, distributed in families (Tree description language)
- Lemma lexicon (Associate lemmas to families)
- Morphological lexicon (Associate inflected forms to lemmas)
- Semantic frames (Typed feature structures description language)

Describing trees and frames

```
class commonnoun
declare ?NP ?N ?X0 ?X1
{
  <syn>{
    node ?NP [cat=np, i=?X0];
    node ?N (mark=anchor)
             [cat=n, i=?X1];
    ?NP -> ?N
  };
  <frame>{
    ?X0[pizza]
  }
}
```



- Syntax-semantics interface with unification variables
- Problem: we do not really want to link this tree to only this frame

Describing trees and frames separately

```
class commonnoun
declare ?NP ?N ?X0 ?X1
{
  <syn>{
    node ?NP
         [cat=np, i=?X0];
    node ?N (mark=anchor)
         [cat=n, i=?X1];
    ?NP -> ?N
  };
    <iface>{[i=?X0]}
}
```

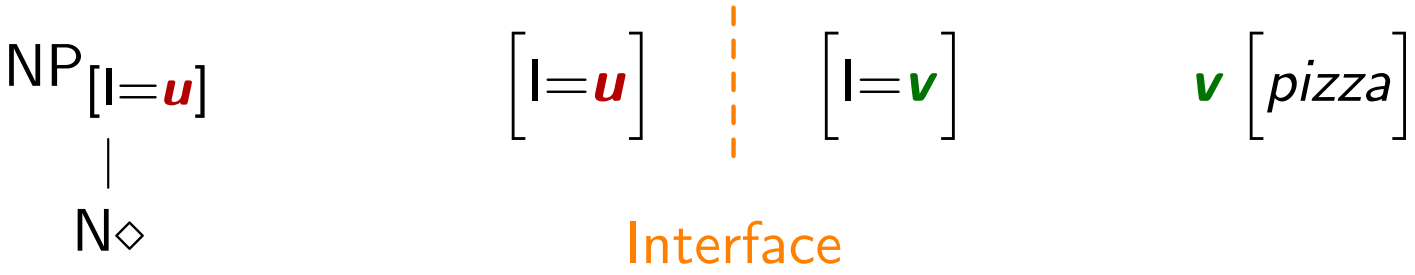
```
class FramePizza
declare ?X0
{
  <frame>{
    ?X0[pizza]
  };
  <iface>{
    [i=?X0]
  }
}
```

- Pairing through the lexicon (coming soon)
- Syntax-semantics interface with the `iface` dimension

Describing trees and frames separately

```
class commonnoun
declare ?NP ?N ?X0 ?X1
{
  <syn>{
    node ?NP
         [cat=np, i=?X0];
    node ?N (mark=anchor)
         [cat=n, i=?X1];
    ?NP -> ?N
  };
    <iface>{[i=?X0]}
}
```

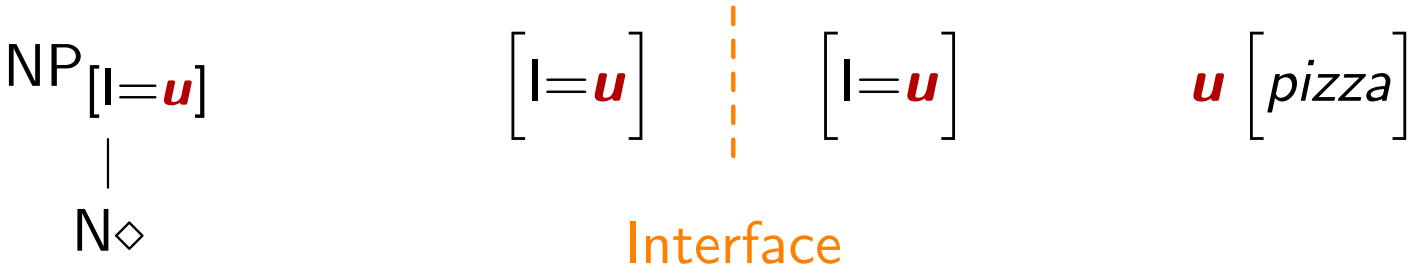
```
class FramePizza
declare ?X0
{
  <frame>{
    ?X0[pizza]
  };
  <iface>{
    [i=?X0]
  }
}
```



Describing trees and frames separately

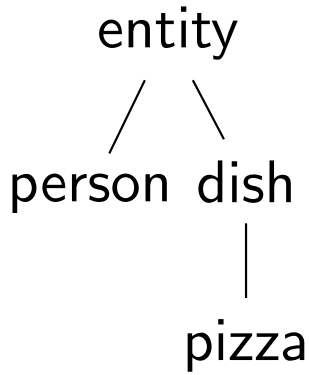
```
class commonnoun
declare ?NP ?N ?X0 ?X1
{
  <syn>{
    node ?NP
         [cat=np, i=?X0];
    node ?N (mark=anchor)
         [cat=n, i=?X1];
    ?NP -> ?N
  };
    <iface>{[i=?X0]}
}
```

```
class FramePizza
declare ?X0
{
  <frame>{
    ?X0[pizza]
  };
  <iface>{
    [i=?X0]
  }
}
```



Describing a type hierarchy

```
frame-types = {event,
  activity, eat, entity,
  person, dish, pizza}
frame-constraints = {
  activity -> event,
  entity event -> -,
  eat -> activity,
  person -> entity,
  dish -> entity,
  dish person -> -,
  pizza -> dish }
```



- Type hierarchy used by XMG to assemble frames
- Exported to be used by TuLiPA

Describing lexical entries (lemmas)

```
class LemmaPizza
{
  <lemma> {
    entry <- "pizza";
    sem   <- FramePizza;
    cat   <- n;
    fam   <- commonnoun
   }
}
```

- The XMG compiler `lex` replaces the historical LexConverter
- `entry` is the lemma
- `sem` is the frame
- `fam` is the family of trees which can be anchored

Describing lexical entries (inflected forms)

```
class MorphPizza
{
  <morpho> {
    morph <- "pizza";
    lemma <- "pizza";
    cat   <- n;
    num   <- sg
   }
}
```

- The XMG compiler `mph` replaces the historical LexConverter
- `lemma` is the lemma
- `morph` is the inflected form

Conclusion

- We extended an existing parser for TAG and XMG-style grammars
- Semantic part: Typed feature structures instead of predicate semantics
- Build semantic representation by unification of typed feature structures
- 2 parsing modes: TAG to RCG conversion or CYK
- Future work
  - Extend TuLiPA for RRG
  - XML output

References I

Gardent, C. and Kallmeyer, L. (2003).
Semantic construction in feature-based tree adjoining grammar.
In *10th conference of the European Chapter of the Association for Computational Linguistics*.

Kallmeyer, L., Lichte, T., Maier, W., Parmentier, Y., Dellert, J., and Evang, K. (2008).
Tulipa: Towards a multi-formalism parsing environment for grammar engineering.
In *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, pages 1–8, Manchester, England.

Kallmeyer, L. and Osswald, R. (2013).
Syntax-driven semantic frame composition in lexicalized tree adjoining grammars.
Journal of Language Modelling, 1(2):267–330.

References II


Kallmeyer, L. and Romero, M. (2008).
Scope and Situation Binding in LTAG using Semantic Unification.
Research on Language and Computation, 6(1):3–52.