Argument linking in RRG: A constraint-based implementation

Laura Kallmeyer, Timm Lichte & Rainer Osswald

Heinrich-Heine-Universität Düsseldorf

International Conference on Role and Reference Grammar (RRG), Düsseldorf, 02.08.2015
Introduction

- Overall aim: A thorough formalization and a computational implementation of Role and Reference Grammar (RRG)
- Focus of this talk: Argument linking in simple cores.

Question: How can we formalize and implement a principled linking theory?
Background

We make use of the following formal and computational elements (along the lines of Kallmeyer et al., 2013; Osswald & Kallmeyer, to appear):

1. A formal characterization of syntactic composition consisting of substitution, sister adjunction, and wrapping substitution. For this talk, only substitution is relevant.

```
CLAUSE
  |
  |
  CORE
  |
  RP  NUC  RP
  |      |
  V[PRED=+]
  |
  RP
  |
  Kim
  |
  V[PRED=+]
  |
  smashes
  |
  DEF
  |
  the
  |
  NUC
  |
  N
  |
  glass
```
Background

(2) A formal specification of the elementary syntactic templates to which these composition operations apply. Assumption: argument structure templates are clause templates with slots for each of the arguments occurring in the clause. These syntactic templates are specified via a metagrammar.

(3) A formalization of the logical structure in terms of decompositional frames (= typed feature structures).

\[
\text{causation} \\
\begin{cases}
\text{cause} & [\text{activity}] \\
\text{effector} & [1\text{Kim}] \\
0 & \text{change-of-state} \\
\text{effect} & [\text{result}] \\
\text{patient} & [2\text{glass}] \\
\text{result} & \text{smashed-state} \\
\text{patient} & [2]
\end{cases}
\]

\[
\text{do}(\text{Kim}, \emptyset) \text{ CAUSE } \text{BECOME smashed}(\text{glass})
\]
Background

Syntactic and semantic composition:

```
CLAUSE
  | CORE
    | RP[1=1] NUC[1=0] RP[1=2]
        | V[pred=+]
            | smashes
    | RP[1=3]
        | Kim

  | DEF CORE
    | the NUC
        | glass

causation
  | cause
    | effect
        | activity effector [1]
            | change-of-state
                | patient
                    | result
                        | smashed-state
                            | patient
```

5 / 21
(4) A specification of linking constraints that capture the way syntactic arguments and semantic roles are systematically related. These constraints are part of the metagrammar, together with syntactic tree fragments and fragments of semantic frames.

- Constraint-based formulation of the principles underlying the linking algorithm from Van Valin (2005) instead of a procedural specification in the form of an algorithm.
- Advantage: we can separate between the linguistic generalizations to be captured and algorithmic considerations.
- We use the XMG grammar development system (Lichte & Petitjean, 2015) for the implementation of (2)–(4).
Constraints on event frames

- First element of our linking system: **Universal constraints on semantic roles and macroroles.**
- Constraints can be of the form (Kallmeyer & Osswald, 2013)
  - \( path : type \) (the value of \( path \) is of type \( type \))
  - \( path_1 \equiv path_2 \) (\( path_1 \) and \( path_2 \) yield the same value)
  - \( constraint_1 \leq constraint_2 \) (\( constraint_1 \) implies \( constraint_2 \))

Some examples:

- **MOVER** : \( \top \leq \text{MOVER} \equiv \text{EFFECTOR} \)
  (a mover is always an effector)
- **CAUSE EFFECTOR** : \( \top \leq \text{EFFECTOR} \equiv \text{CAUSE EFFECTOR} \)
  (if the causing event of a causation has an effector, then this effector is also the effector of the embedding causation event)
- **EFFECTOR** : \( \top \leq \text{EFFECTOR} \equiv \text{ACTOR} \)
  (if an effector is given, then the effector is also the actor)
- **PATIENT** : \( \top \leq \text{PATIENT} \equiv \text{UNDERGOER} \)
  (a patient is always an undergoer)
MG classes for semantic arguments

Second element of our linking system:
Classes for semantic arguments of rank 1, 2, 3 or 4 respectively

(Fig. 2.3 from Van Valin, 2005, p. 58)
MG classes for semantic arguments

class ArgRank1
  \(?e0[\text{effector} \ ?x1]\)

class ArgRank2
  \(?e0[\text{location|wanter|judger}|... \ ?x2]\)

class ArgRank3
  \(?e0[\text{theme|stimulus|content}|... \ ?x3]\)

class ArgRank4
  \(?e0[\text{patient}|... \ ?x4]\)

(?e0, ?x1, ... are metagrammar variables)
Macrorole assignment

- Goal: For combinations of these argument classes, we want to obtain the correct macrorole assignment.

- Due to the universal constraints, we already have that 1) if a rank 1 argument is present, it is the actor and 2) if a rank 4 argument is present, it is the undergoer.

Two possible solutions:

1. List all possible argument combinations and specify the corresponding macrorole assignment (see abstract).

2. Further factorization of these combinations using a variable for the highest rank and binary variables giving information about the rank of the undergoer.
Macrorole assignment

First solution yields for instance the classes

class *Pred_Rank2*

\( \text{ArgRank2, } ?x2 = ?\text{promi}, \)

\[ ?e0[\text{actor } ?x2] \]

class *Pred_Rank1_2*

\( \text{ArgRank1, ArgRank2,} \)

\( ?x1 = ?\text{promi}, ?x2 = ?b\text{-promi} \)

\[ ?e0[\text{undergoer } ?x2] \]

?promi and ?b-promi are interface variables:

- ?promi is the argument with the highest rank,
- ?b-promi the one with the lowest, provided there is a higher one.
Macrorole assignment

Second solution: We use

- a variable $?\text{highest}$ for the highest argument rank,
- boolean variables $?\text{und\_lower\_2}$ and $?\text{und\_lower\_3}$, indicating whether the undergoer has a rank lower than 2 (resp. 3), and
- variables $?r1, \ldots, ?r4$ giving the argument of rank 1, $\ldots$, 4 respectively.

We add $?r_i = ?x_i$ to each of the previous argument classes.

For instance: class $Arg\text{Rank1}$

```
?r1 = ?x1, ?e0[effector  ?x1]
```

For each rank, we introduce new classes for arguments of this rank a) being an actor, b) being an undergoer, c) without macrorole and d) not realized (if applicable).
Macrorole assignment

class Rank2_actor
ArgRank2, ?highest = 2, ?promi = ?x2, ?e0[actor ?x2]

class Rank2_undergoer
ArgRank2, ?highest = 1, ?b-promi = ?x2, ?und_lower_2 = false,
?e0[undergoer ?x2]

class Rank2_no_macrorole
ArgRank2, ?highest = 1, ?und_lower_2 = true

class Rank2_no_argument
?r2 = nil
Macrorole assignment

Class comprising all these alternatives:

class Rank2
   Rank2_actor
   OR Rank2_undergoer
   OR Rank2_no_macrorole
   OR Rank2_no_argument

Final class describing all possible semantic frames:

class Event_Frame
   Rank1, Rank2, Rank3, Rank4

XMG compiles this into all possible combinations, creating thereby all possible event frames.
Linking syntax and semantics

- Within the metagrammar, elementary syntactic templates are decomposed.
- The variables ?promi, ?b-promi, ?r1, ?r2, ?r3, ?r4 can be used as interface variables.

Language-specific classes for English:

```
class Privileged_syn_arg

```

```
<table>
<thead>
<tr>
<th>CORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP[1=?arg1]</td>
</tr>
<tr>
<td>NUC[1=?e0]</td>
</tr>
</tbody>
</table>

\[ V_{\text{PRED}=+, \text{VOICE}=?\text{voice}, i=?e0} \]

(?voice = active, ?promi = ?arg1) OR
(?voice = passive, ?b-promi=arg1)

```
class Obj_RP

```

```
<table>
<thead>
<tr>
<th>CORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUC[1=?e0]</td>
</tr>
<tr>
<td>RP[1=?arg2]</td>
</tr>
</tbody>
</table>

\[ ?r2=?arg2 \]
OR \[ ?r3=?arg2 \]
OR \[ ?r4=?arg2 \]
Putting together these two classes and the class Event Frame yields among others:

```
<table>
<thead>
<tr>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>undergoer</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>effector</td>
</tr>
<tr>
<td>patient</td>
</tr>
</tbody>
</table>
```

Interface variables:
\[?\text{promi} = 1, \ ?\text{b-promi} = 2, \ ?r1 = 1, \ ?r2 = \text{nil}, \ ?r3 = \text{nil}, \ ?r4 = 2\]

This can then be combined with the lexical entry of the verb.
Linking syntax and semantics

Lexical entry of _smashed_:

\[ V_{[\text{pred}=+, \text{voice}=\text{active}, i=3]} \]

- **causation**
  - cause
  - effect

- **activity**
  - effector 4
  - patient
  - result

- **change-of-state**
  - patient

- **smashed-state**
  - patient

Interface variables: \(?r1=4, ?r2 = \text{nil}, ?r3 = \text{nil}, ?r4=5\)

The universal constraints on event frames yield:

\[
\begin{align*}
\text{causation} & \\
\text{cause} & [\ldots] \\
\text{effect} & [\ldots] \\
\text{effector} & 4 \\
\text{patient} & 5 \\
\text{actor} & 4 \\
\text{undergoer} & 5 \\
\end{align*}
\]
Linking syntax and semantics

Combining the template with the lexical predicate:

```
CORE
  RP[i=1]  NUC[i=0]  RP[i=2]
  |    |    |
  V[VOICE=active,i=0]  V[VOICE=active,i=3]
  |    |    |
  smashed  smashed

[ event
  actor 1
  0 undergoer 2
  effector 1
  patient 2 ]

[ causation
  cause
    [ effector 4
      ... ]
  effect
    [ patient 5
      ... ]
  effector 4
  patient 5
  actor 4
  undergoer 5 ]
```
Linking syntax and semantics

Result:

[Diagram showing syntactic and semantic roles]

- **causation**
  - actor: 1
  - undergoer: 2
  - effector: 1
  - patient: 2

- **activity**
  - effector: 1
  - actor: 1

- **change-of-state**
  - patient: 2
  - undergoer: 2

- **smashed-state**
  - result
    - patient: 2
    - undergoer: 2
Summary

- We propose a constraint-based implementation of argument linking in RRG using a metagrammar approach.

- Universal constraints on event frames and semantic roles.

- Constraints (MG classes) capturing the macrorole possibilities for each argument rank. The possible combinations of these classes are controlled by MG variables.

- The syntactic tree is decomposed into single fragments for each argument that carry constraints about the ranking of this argument, expressed via interface variables.

- The lexical predicates substitute into the argument structure templates, which triggers a unification of the lexical frame and the template frame.


