# Argument linking in LTAG: A constraint-based implementation with XMG

**Laura Kallmeyer** and **Timm Lichte** and **Rainer Osswald** and **Simon Petitjean**
University of Düsseldorf, Germany
{kallmeyer,lichte,osswald,petitjean}@phil.hhu.de

## Abstract

This paper develops a first systematic approach to argument linking in LTAG, building on typologically oriented work in Van Valin (2005). While Van Valin's argument linking mechanism is procedurally defined, we propose a constraint-based implementation. The advantage is that we can separate between the linguistic generalizations to be captured and algorithmic considerations. The implementation is couched into the metagrammar framework eXtensible MetaGrammar (XMG).

## 1 Introduction

The syntax-semantics interface of Lexicalized Tree Adjoining Grammar (LTAG) builds on the assumptions that (i) the elementary tree of a predicate contains slots for the arguments of the predicate, (ii) this elementary tree is paired with a semantic representation with semantic arguments, (iii) there is a linking between syntactic argument slots and semantic arguments that makes sure that the filling of an argument node in the syntax triggers the insertion of a corresponding semantic representation into the linked semantic argument position. This holds for the unification-based approach from Kallmeyer and Joshi (2003), Gardent and Kallmeyer (2003) and Kallmeyer and Romero (2008) using predicate logic, for the approach based on synchronous TAG (Shieber, 1994; Nesson and Shieber, 2006; Nesson and Shieber, 2008) and also for the frame-based approach of Kallmeyer and Osswald (2013). However, none of these approaches has implemented a theory which explains why only certain patterns of argument linking are allowed. In Fig. 1, for instance, the elementary tree for *ate* is paired with the upper frame while the lower frame is not grammatical

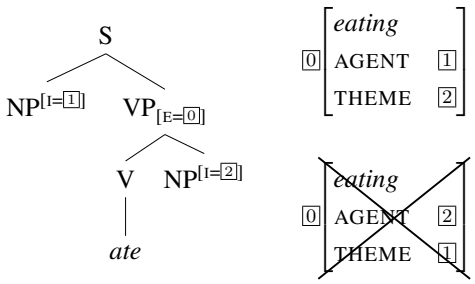



Figure 1: Simple linking example

in combination with this tree because of the incorrect linking. The AGENT has to be contributed by the subject while the THEME slot is filled via the object substitution node.

The principles and constraints underlying linking have been extensively investigated (Levin and Rappaport Hovav, 2005; Wechsler, 2015). Dowty (1991), for instance, introduces proto-AGENT and proto-PATIENT as intermediate roles linking syntax to semantics. Van Valin (2005) presents an elaborated linking algorithm based on the *macroroles* ACTOR and UNDERGOER and on the *actor-undergoer-hierarchy*. This algorithm is not fully formalized, however, and it is formulated in a procedural way that mixes constraints which express linguistic generalizations with algorithmic aspects. The latter point is problematic insofar as Van Valin's linking system is intended to be used both for language understanding and language generation. It is thus desirable for a formalization and implementation of the system to keep the principles and constraints separate from aspects of processing order.

Based on Van Valin's proposal, this paper provides a constraint-based implementation of linking principles that captures the systematic relation between syntactic arguments and semantic roles. These constraints restrict the set of possible ele-

*Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12)*, pages 48–57, Düsseldorf, Germany, June 29 - July 1, 2016.

mentary pairs of tree and semantic representation. In other words, they act on the level of elementary structures in the grammar and are therefore part of the metagrammar, together with syntactic tree fragments and fragments of semantic representations. This is the part of the grammar that provides a systematic constraint-based definition of the set of elementary trees along with the semantic representations they can be paired with (Crabbé and Duchier, 2004; Duchier et al., 2004). We will use the metagrammar compiler XMG (Crabbé et al., 2013; Lichte and Petitjean, 2015) with frames as semantic representations (Kallmeyer and Osswald, 2013). Since argument linking is independent from the choice of the semantic representation, our analysis could also be applied to an LTAG syntax-semantics interface using one of the other frameworks mentioned above.

## 2 The syntax-semantics interface

We assume familiarity with the basic notions of LTAG (Joshi and Schabes, 1997; Abeillé and Rambow, 2000), enriched with syntactic feature structures in the usual way (Vijay-Shanker and Joshi, 1988).

Following Kallmeyer and Osswald (2013), we pair syntactic trees with frame-semantic representations, which instantiate a slightly extended variant of typed feature structure. An example is given in Fig. 2. We use *interface features* on the syntactic nodes that are responsible for triggering semantic composition (i.e., frame unification) via the syntactic feature unifications during substitution and adjunction. These features here are I (for "individual") and E (for "event"), whose values are variables that also occur in the frames. Upon substituting the elementary tree of *John* into the subject NP slot of the elementary tree of *ate*, the variables 1 and 3 get equated, hence triggering the unification of the frame of *John* with the AGENT component of the frame of *ate*.

In the rest of the paper, we will be focusing on the properties of single elementary entries of verbs such as *ate*.

## 3 Constraints on elementary entries

Given that in LTAG, the set of composition operations is rather small, the actual domain of linguistic theorizing lies mainly in the way elementary entries like that of *ate* in Fig. 2 are designed, or rather constrained. Hence, constraints pertaining to the tree structure, the frame-semantic representation, and the linking among the two make up an important part of theories expressed within this framework. These constraints on elementary structures constitute what is commonly called a metagrammar, the general outline of which is shown in Fig.3.

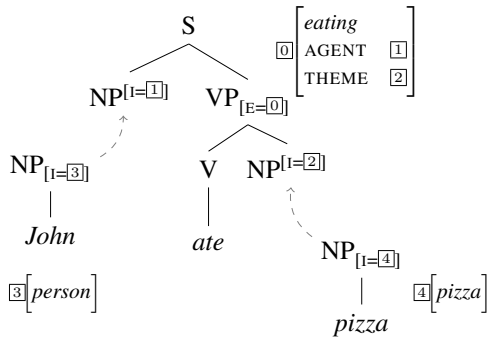



Figure 2: Sample derivation

### 3.1 Constraints on tree structure

The most explicit and systematic proposal to constrain the shape and structure of elementary trees was made in works of Robert Frank (1992; 2002).[1] Based on the Fundamental TAG Hypothesis ("Every syntactic dependency is expressed locally within an elementary tree."), the inner structure (Condition on Elementary Tree Minimality) and the number of non-terminal leaf nodes ($\theta$-Criterion for TAG) are covered, the latter aspect, of course, being the more relevant here. The $\theta$-Criterion for TAG states that there is a bijective mapping between "$\theta$-roles", i.e. semantic arguments, and non-terminal leaf nodes. This is met, for example, in the tree of *ate* based on the shown *eating* frame. Yet there is no particular constraining on what this mapping might look like. In other words, it is unclear how to prevent the deficient linking pattern in Fig. 1 from coming into existence. This is exactly the gap that our contribution is supposed to fill.

One important remaining issue is the status of the $\theta$-Criterion for TAG when dealing with semantic frames. In its original formulation, the $\theta$-Criterion requires the numbers of syntactic and semantic arguments to be equal, whereas modifiers are neglected. However, in a frame, there is no principled distinction between arguments and modifiers anymore – both get represented as ordinary functional attributes. Hence, combining

[1] But see also Abeillé and Rambow (2000).

the $\theta$-Criterion, as is, with frame semantics would have the unwanted effect that modifiers are represented as nonterminal leaves just like arguments. This is unwanted because it would increase the number and size of elementary trees massively, though not ad infinitum due to the functional nature of frame attributes (therefore being reminiscent of the treatment of optional arguments). Possible remedies could be to apply the $\theta$-Criterion to descriptions of lexical frames where modifiers have not yet entered, or to insert a mediating valency layer that helps to abstract away from the full set of semantic roles (cf. Lichte, 2015). However, this discussion touches the issue of argument-modifier distinction more generally, and therefore should be treated elsewhere.

### 3.2 Constraints on frame structure

Following Kallmeyer and Osswald (2013), we formalize semantic frames as typed feature structures with base labels and relations. The frame constraints used in this paper are basically Horn clauses built from atomic attribute-value descriptions of the form $\mathrm{P} : a$ and $\mathrm{P} \doteq \mathrm{Q}$, where P and Q are (possibly empty) feature paths and $a$ is a type (including $\top$ and $\bot$). For example, the constraint that *eating* involves an ACTOR and a THEME but no PATH can be formalized in this logic as follows:

$$\textit{eating} \Rightarrow \textsc{actor} : \top \wedge \textsc{theme} : \top$$
$$\textit{eating} \wedge \textsc{path} : \top \Rightarrow \bot$$

### 3.3 Constraints on argument linking

The linking constraints in the metagrammar basically combine tree constraints with frame constraints. For instance, regarding the elementary entry of *ate* in Fig. 2, one such constraint could be paraphrased as "in an active sentence, the AGENT is realized as the subject, and the THEME is realized as the object". We will learn about a systematic and typologically oriented linking theory in Section 5, which will be the starting point for our implementation in Section 6. The overall structure of the metagrammar, together with the metagrammar compiler, is shown in Fig. 3.

## 4 eXtensible MetaGrammar

The framework of eXtensible MetaGrammar (XMG, Crabbé et al., 2013) provides description languages and dedicated compilers for generating a wide range of linguistic resources.[2] Descriptions

[2] https://sourcesup.cru.fr/xmg/

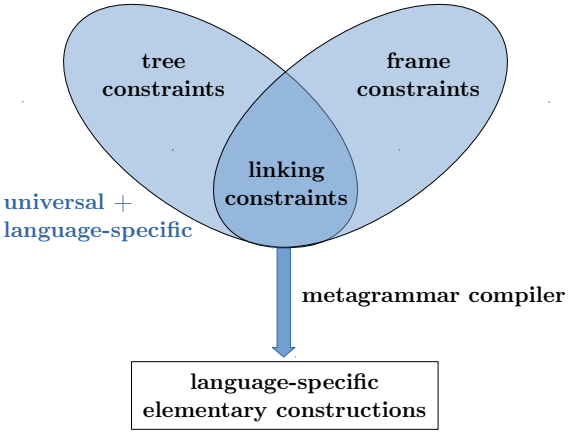



Figure 3: Metagrammar components

are organized into *classes*, alluding to the class concept in object-oriented programming. Similarly, classes have encapsulated name spaces and inheritance relations may hold between them. The crucial elements of a class are *dimensions*. They can be equipped with specific description languages and are compiled independently, thereby enabling the grammar writer to treat the levels of linguistic information separately.

As our focus is on argument linking, the following three class dimensions, which contain the three different sorts of constraints mentioned in Fig. 3, are most relevant to us: **<syn>** holds tree constraints that express dominance and precedence relations among nodes. Moreover, the nodes may carry (untyped) feature structures. **<frame>** holds frame descriptions, i.e. descriptions of typed feature structures. Feature structure *constraints*, on the other hand, are specified globally outside classes. Finally, **<iface>** is an interface dimension where (non-typed) feature structures are used to share information between other dimensions and classes. This dimension will obviously serve as a place for expressing linking constraints. The general structure of a class then looks as follows:

```
class classname
import someOtherClass[]
export ?someVariable
declare ?someVariable
{
  <syn>{ ... };
  <frame>{ ... };
  <iface> { ... }
}
```

Note that **import**, **export**, and **declare** behave similarly to the corresponding constructs in object-oriented programming. The operator **;** ex-

presses the conjunction of contents. One can also express disjunction with the operator **|**.

The description language used inside **<frame>** is a simple recursive bracket notation with the following ingredients:

```
?variable1 [
  type1, type2 , ... ,
  feature2:?variable2,
  feature3:?variable3[ ... ]
]
```

Variables here correspond to the boxed variables in Fig. 1 and 2 and are optional as well. Also note that there can be more than one atomic type specified (making up conjunctive types). Features and their values are separated with the colon (**:**), and values can be either variables, or feature structures. See Lichte and Petitjean (2015) for a detailed definition. The descriptions inside the **<iface>** dimension look very similar, the main difference being that the feature-value separator is the equal sign (**=**) for historical reasons. As far as the description language in **<syn>** is concerned, we will ignore it for now and use graphical representations in the further presentation.

As mentioned before, feature structure constraints (and type constraints), which hold globally for all frame structures inside the classes, are specified externally (see, again, Lichte and Petitjean, 2015). They are constructed in the following way, building on the description language from (Kallmeyer and Osswald, 2013) as sketched in Section 3.2:

*Constraint* ::= *Description* **->** *Description*
*Description* ::= `type ... type` |
`feature ... feature:type` |
`feature ... feature` **=** `feature ... feature`

**->** is the implication operator, `feature ... feature` stands for a path in the feature structure, and `feature ... feature` **=** `feature ... feature` for a path equation. The atomic types which can be used in the constraints are defined by the user. To this set are added the predefined types $+$ and $-$, standing for $\top$ (the most generic type) and $\bot$ (the "false" type).

Note that constraints are treated differently in the compiler: type constraints (with only types on the left-hand side) are precompiled into a type hierarchy, that is, the constraints on types are used to compute the set of valid conjunctive types. The result is a maximal model of the type constraints, meaning that all combinations of atomic types which are not explicitly prohibited are authorized. The remaining set of constraints is checked at runtime.

Based on a set of constraints, XMG compiles full models (i.e., trees and frames such as in Fig. 2), which then enter into parsing. Hence, under this perspective, the metagrammar constraints act as lexical constraints proper. Note, however, that metagrammar constraints could also be used in parsing more directly (de la Clergerie, 2013).

## 5 Argument linking in Van Valin (2005)

In this section, we describe the core generalizations on argument linking proposed in Van Valin (2005), which will serve as a basis for our constraint-based implementation of argument linking in elementary trees. The basic idea is to assign the macroroles ACTOR (AC) and UNDERGOER (UG) to certain semantic arguments, which are then syntactically realized in a specific way that depends on the language and the voice type expressed in the elementary tree, among others. For the purposes of this paper, we put aside the linking of a possible third, non-macrorole argument, which is typically realized with oblique case or by a prepositional phrase.

The relation between semantic roles and macroroles is captured in the *actor-undergoer hierarchy* shown in Fig. 4. Since our approach employs frame-semantic representations, we do not make use of the positional encoding in logical structures sketched in the top line of the figure but build directly on the associated semantic roles. Depending on its semantic role, an event participant obtains a higher or a lower rank with respect to the hierarchy. For example, Fig. 4 implies that a STIMULUS argument gets assigned a lower rank than an EXPERIENCER argument. (Note that rank 1 is considered the highest rank while rank 4 is the lowest rank.)

The relation between the actor-undergoer hierarchy and the macrorole assignment is then as follows: If a rank 1 argument is present, it is the actor. If a rank 4 argument is present, it is the undergoer. A rank 2 argument is the actor if there is no rank 1 argument. A rank 2 argument is the undergoer if there is a rank 1 argument and no rank 3 or rank 4 undergoer. A rank 3 argument is the undergoer if it is not the actor and if there is no rank 4 argument. A rank 3 argument cannot be an actor unless it is at the same time a rank 1 argument. In this latter case, it counts as rank 1 in our constraints.

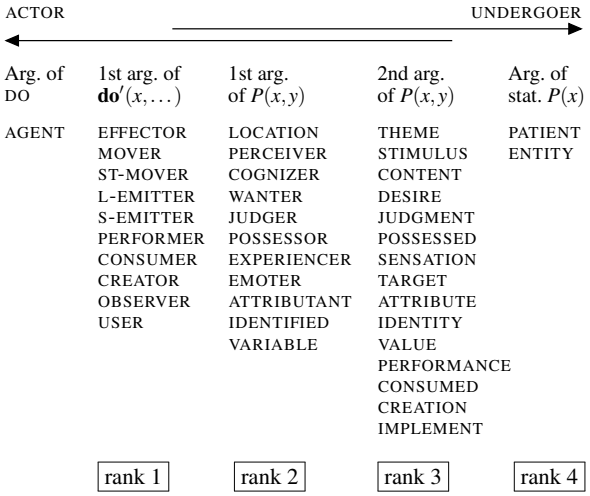



Figure 4: Actor-undergoer hierarchy with semantic roles; adapted from Van Valin (2005, p. 58)

Macrorole assignment is only one of the steps of the linking algorithm laid out in Van Valin (2005); the subsequent step is concerned with the morpho-syntactic realization of arguments based on the assigned marcroroles. This step is dependent on the language type and it is also sensitive to voice-modulation, among others. In the case of English, for instance, if a transitive verb is anchored to an active-voice elementary tree, which can be seen as the default anchoring, then the highest ranking argument, i.e. the actor, is realized as the *privileged syntactic argument*, i.e. in subject position, while the undergoer is realized as the direct object. In passive voice, on the other hand, the undergoer becomes the privileged syntactic argument while the actor may be optionally realized by a *by*-clause. The general observation is that in languages with an accusative syntactic system such as German and English, the highest ranking argument is by default realized as the privileged syntactic argument. In particular, the macrorole assignment does not matter for intransitive verbs (since there is only one argument); the single argument becomes the privileged syntactic argument, which receives nominative case in German (1-b).

(1) a. Der Junge zerbrach
the.NOM boy[AC].NOM broke
den Teller.
the.ACC plate[UG].ACC

b. Der Teller zerbrach.
the.NOM plate[UG].NOM broke

In ergative languages, by contrast, the *lowest* argument with respect to the hierarchy is by default selected as the privileged syntactic argument, which appears in the absolutive case. If the language is syntactically accusative but morphologically ergative then the highest ranking argument becomes the privileged syntactic argument while the lowest ranking argument receives absolutive case. This is illustrated in the following Warlpiri examples (Van Valin, 2005, p. 109; taken from Hale 1973): the actor in (2-a) and the undergoer in (2-b) both receive absolutive case.

(2) a. Ngaju-∅ ka-rna purla-mi.
1SG[AC]-ABS PRES-1SG shout-NPAST
'I am shouting.'

b. Ngaju-rlu ka-rna-∅
1SG[AC]-ERG PRES-1SG-3SG
wawiri-∅ pura-mi.
kangaroo[UG]-ABS cook-NPAST
'I am cooking the kangaroo.'

As mentioned at the beginning of this section, the linking rules just sketched capture only the very core of Van Valin's system, namely the case of intransitive and simple transitive verbs where all arguments are macrorole arguments. In addition, the lexicon may contain verbs with two arguments of which only one is a macrorole argument, which means that the other argument is realized by an oblique case or a prepositional phrase according to certain rules. Similarly, there are rules, which are partly language-specific, for assigning dative, instrumental or another oblique case to the non-macrorole argument of verbs with three arguments. Moreover, these verbs often permit variable undergoer choice, which is then reflected in alternations such as the dative and the locative alternation. Finally, languages differ with respect to whether the privileged syntactic argument of a construction is restricted to macrorole arguments (as in German or Italian), or whether also non-macrorole arguments can serve as the privileged syntactic argument (as, e.g., in Icelandic). While the implementation presented in the following is restricted to the basic linking of macroroles described above, the long-term goal is to successively extend the implementation towards a full coverage of Van Valin's linking system.

## 6 Constraint-based implementation of Van Valin (2005)

### 6.1 Universal constraints on semantic frames

As a first element of our linking system, we define universal constraints on semantic roles and on macroroles. Macroroles are taken to be attributes in our semantic frames, just like semantic roles.

A selection of these constraints is given in Fig. 5. The constraints are notated with the XMG syntax. They correspond to formulas in the frame logic introduced in Kallmeyer and Osswald (2013), the last constraint in Fig. 5 for instance corresponds to *change-of-state* $\Rightarrow$ PATIENT $\doteq$ RESULT PATIENT in this logic. The first two constraints express the fact that rank 1 arguments (= EFFECTOR) are actors while rank 4 arguments are undergoers. The next two constraints are two of many constraints on hierarchical relations between roles. The roles listed in Fig. 4 under a specific rank, for instance rank 2, are hierarchically ordered. The last three constraints in this list are constraints for lifting semantic roles from the recursive structure of an event to a higher level. Take for instance the frame at the top of Fig. 6 for a *smashing* event, which is analyzed as a causation involving an activity as the causing event. The 5th constraint in Fig. 5 tells us that in a *causation* event, the EFFECTOR of the embedded CAUSE is also the EFFECTOR of the entire *causation*.

To support these constraints, XMG's frame dimension has been extended, compared to Lichte and Petitjean (2015), such that it allows not only single attributes but also sequences of attributes on the left-hand sides of the $\phi_1 -> \phi_2$ implications.

The example in Fig. 6 shows how these constraints enrich the frame structure by adding new attributes to it. At the top, we have the frame representing lexical semantics of *smash*. A smashing event is a causation where an effector performs some activity and, as a result of this, a patient changes its state into the state of being smashed. The second frame in Fig. 6 shows the enriched frame we obtain when applying the constraints from Fig. 5 to this first frame.

### 6.2 Semantic argument classes

The universal constraints introduced so far take care of the fact that rank 1 arguments are actors and rank 4 arguments are undergoers. But for rank 2 and rank 3 arguments, macrorole assignment is more complicated. In the following, we introduce

$$
\boxed{0}\begin{bmatrix}
\textit{causation} & \\
\text{CAUSE} & \begin{bmatrix} \textit{activity} & \\ \text{EFFECTOR} & \boxed{1} \end{bmatrix} \\
\text{EFFECT} & \begin{bmatrix} \textit{change-of-state} & \\ \text{PATIENT} & \boxed{2} \\ \text{RESULT} & \begin{bmatrix} \textit{smashed-state} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

Applying the constraints from Fig. 5 yields

$$
\boxed{0}\begin{bmatrix}
\textit{causation} & \\
\text{EFFECTOR} & \boxed{1} \\
\text{ACTOR} & \boxed{1} \\
\text{PATIENT} & \boxed{2} \\
\text{UNDERGOER} & \boxed{2} \\
\text{CAUSE} & \begin{bmatrix} \textit{activity} & \\ \text{EFFECTOR} & \boxed{1} \\ \text{ACTOR} & \boxed{1} \end{bmatrix} \\
\text{EFFECT} & \begin{bmatrix} \textit{change-of-state} & \\ \text{PATIENT} & \boxed{2} \\ \text{UNDERGOER} & \boxed{2} \\ \text{RESULT} & \begin{bmatrix} \textit{smashed-state} & \\ \text{PATIENT} & \boxed{2} \\ \text{UNDERGOER} & \boxed{2} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

Figure 6: Frame for *smashing*

XMG classes for these arguments and for the different ways of linking them to the macroroles. In contrast to the universal constraints, these classes describe possible frame fragments, i.e., they are used as being existentially quantified whenever they are integrated into some class that gets compiled in the MG. Fig. 7 gives the classes for the arguments with a disjunction of the possible semantic roles. In all the XMG classes given in this section, `?e` and `?x` are XMG variables that are always exported.

Concerning macrorole assignment, we have to make sure that for combinations of these arguments, macroroles are assigned in accordance with the constraints stated above. To this end, we use interface features. Interface features are a special dimension in our XMG classes, the `<iface>` dimension. They form an untyped non-recursive feature structure. When combining two classes, the interface features have to unify. In other words, their values (where specified) have to be equal. For our purposes, we need the following linking interface features: A feature `highest-rank` that gives the highest rank in the combination of arguments, and boolean features `und-lower2` and `und-lower3` that indicate whether there is an undergoer of rank lower than 2 (resp. 3).

| constraint | explanation |
|---|---|
| `effector:+ -> effector=actor` | if an effector exists, then it is the actor |
| `patient:+ -> patient=undergoer` | a patient is always an undergoer |
| `mover:+ -> mover=effector` | a mover is always an effector |
| `emoter:+ -> emoter=experiencer` | an emoter is always an experiencer |
| `cause effector:+ -> cause effector=effector` | if the cause of a causation has an effector, then this is also the effector of the entire causation |
| `effect patient:+ -> effect patient=patient` | if the effect of a causation has a patient, this is also the patient of the causation event |
| `change-of-state -> patient=result patient` | the patient of a change-of-state is the patient of the embedded result state |

Figure 5: Universal constraints for semantic roles and macroroles

```
class argRank1
<frame>{?e[event, effector:?x] |
  ?e[event, mover:?x] |
  ?e[event, st-mover:?x] | ...}

class argRank2
  <frame>{?e[event, location:?x] |
  ?e[event, perceiver:?x] | ...}

class argRank3
  <frame>{?e[event, theme:?x] |
  ?e[event, stimulus:?x] | ...}

class argRank4
  <frame>{?e[event, patient:?x] |
  ?e[event, entity:?x] }
```

Figure 7: XMG classes for rank $i$ arguments

```
class Rank2_actor
import argRank2[]
<frame>{?e[event, actor:?x]};
<iface>{[highest-rank=r2,highest-arg=?x,
    rank2=?x]}

class Rank2_undergoer
import argRank2[]
<frame>{?e[event,undergoer:?x]};
<iface>{[highest-rank=r1,rank2=?x,
  und-lower2=false,lowest-arg=?x]}

class Rank2_no_macrorole
import argRank2[]
<iface>{[highest-rank=r1,
  und-lower2=true,rank2=?x]}

class Rank2_no_arg
<frame>{?e[event]}
```

Figure 8: Rank-specific XMG classes for the different macrorole assignments: classes for rank 2

These features are needed for macrorole assignment. In addition, we also need the following interface features for linking the resulting semantic frame with a syntactic tree: features `rank`$i$ that give the frame node of the argument with rank $i$, if it is present, and features `highest-arg` and `lowest-arg` where `highest-arg` gives the frame node of the argument with the highest rank while `lowest-arg` gives the one with the lowest rank provided there exists a higher ranked argument.

As a further step of factorizing possible argument realizations in the metagrammar, for each rank, we then 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). Fig. 8 gives these classes for rank 2, where all four cases are possible. For each possibility, the interface features are set accordingly. For instance, if a rank 2 argument is an undergoer (see the second class in Fig. 8), there has to be a higher argument (`highest-rank=r1`) and there cannot be any other lower undergoer (`und-lower2=false`, `und-lower3=false`).

The classes for rank 3 arguments are given in Fig. 9. Such an argument can either be the undergoer (no rank 4 argument is present) or without macrorole (there is a rank 4 argument) or not realized. In the latter case, captured in the class `class Rank3_no_arg`, the interface features state that either there is no undergoer lower than rank 2, which allows for combinations with the first two rank 2 classes in Fig. 8. Or there is a rank 4 undergoer. In this case, a rank 2 undergoer is excluded via the assignment `und-lower2=true`.

Finally, for each rank $i$, we have a class `Rank`$i$ that is just a disjunction of the different possibilities captured in the `Rank`$i$`_...` classes, and, furthermore, there is a general class `event` that com-

```
class Rank3_undergoer
import argRank3[]
{<frame>{?e0[event, undergoer:?x]};
 <iface>{[und-lower3=false,
      und-lower2=true, rank3=?x,
      lowest-rank=r3, lowest-arg=?x]}}

class Rank3_no_macrorole
import argRank3[]
{<iface>{[und-lower3=true, rank3=?x]}}

class Rank3_no_arg
<frame>{?e[event]};
{<iface>{[und-lower2=false]} |
   <iface>{[und-lower2=true,
      und-lower3=true]}}
```

Figure 9: Classes for different macrorole assignments for arguments of rank 3

bines these `Rank`$i$ classes (see Fig. 10).

```
class Rank2
  {?arg=Rank2_actor[] |
   ?arg=Rank2_no_arg[] |
   ?arg=Rank2_undergoer[] |
   ?arg=Rank2_no_macrorole[]};
  ?e=?arg.?e

class event
import Rank1[] Rank2[] Rank3[] Rank4[]
```

Figure 10: XMG classes `Rank2` and `event`

XMG compiles the class `event` into all possible semantic role combinations while computing the correct macrorole assignments, creating thereby all possible event frames.

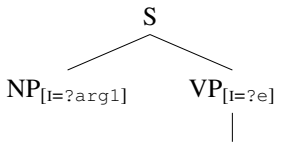
### 6.3 Linking syntax and semantics

The part of the MG described in the previous two sections is language-independent. Depending on the language, the arguments with the highest and the lowest rank are realized differently in the syntax (see section 5). In the following, we will only introduce the XMG classes for syntax-semantics linking for English.

Recall that the interface feature `highest-arg` gives the argument with the highest rank and that `lowest-arg` gives the one with the lowest, provided there is a higher one. Besides these two, we also use the interface features `rank`$i$ for linking.

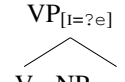
A simplified version of the syntactic XMG classes implemented for English is given in Fig. 11. The two classes for subject and object combine into the class `n0Vn1` for transitive verbs. Note that in our actual implementation, `Subject` if further decomposed and, furthermore, a range of

```
class Subject
<syn>{
```




```
};
{{?Voice=active;
  <iface>{[highest-arg=?arg1]}} |
{?Voice=passive;
  <iface>{[lowest-arg=?arg1]}}}

class Object
<syn>{
```




```
};
{<iface>{[rank2=?arg2]} |
  <iface>{[rank3=?arg2]} |
  <iface>{[rank4=?arg2]}}

class n0Vn1
import Subject[] Object[] event[]
```

Figure 11: Language-specific XMG classes for English (simplified)

additional boolean interface features is used in order to constrain the combinations of arguments of different ranks in the two syntactic argument slots. The `Subject` class expresses that in active sentences, the `highest-arg` fills the subject slot while in passive sentences, this slot is filled by the `lowest-arg`. Further boolean interface features check that in these cases, `highest-arg` and `lowest-arg` are actually given.

When limiting the semantic argument classes `argRank`$i$ to just one single semantic role, the XMG compiler yields 16 different frames when compiling the class `event` (each argument rank can be present or absent and the rank combination determines the macrorole assignments) and a total of 25 tree frame pairs when compiling `n0Vn1`, which corresponds to the correct 25 linking patterns that we expect here: In active sentences, either a) the subject is of rank 1, then there are 3 possible object ranks and the other two ranks can be each present or absent in the frame or b) the subject is of rank 2, then the object has rank 3 or 4 and the rank among $\{3, 4\}$ that is not object can be present or not in the frame or c) the subject has rank 3 and the object rank 4. In passive sentences, either a) the subject is of rank 4, then the object can be 3 or 2 and the remaining two ranks can be present or absent in the frame, or b) the subject is of rank 3, the object of rank 2 and rank 1 is present or absent in the frame. Fur-

thermore, a rank 2 argument requires a rank 3 argument that could, however, be promoted to the rank 1. The implementation yields a total of 23 linking patterns. Note that this simplified implementation of `n0Vn1` leaves out a lot of possibilities for argument realizations. It considers only the case of canonical subject and object positions and does not take the possibility into account that the highest argument in passive constructions can be realized as a *by*-PP. All these other cases would of course lead to more tree frame pairs.

Our analysis allows for tree frame pairs where only some of the arguments listed in the frame have corresponding syntactic slots. In other words, we implement a relaxation of the $\theta$-criterion for LTAG mentioned in section 3 that requires that each syntactic argument slot in the elementary tree of a predicate corresponds to a semantic argument slot in the frame but not necessarily vice versa. To what extend the stronger version of the $\theta$-criterion, requiring a bijection between syntactic and semantic arguments, should be applied, is an open question (see section 7).

### 6.4 Lexical insertion

When combining a lexical item, for instance *smashed* paired with the frame from Fig. 6, with an unanchored class such as `n0Vn1`, the two event frames unify. Due to our relaxed implementation of the $\theta$-criterion, this unification can enrich the lexical frame with further semantic roles and there might even be roles coming from the lexical element that do not have a corresponding syntactic argument slot. Sometimes this might be desired. The semantics of *walk* for instance does not necessarily contain a GOAL component. Such a component can however be added by a directed-motion construction as in (3).

(3) John walked into the house.

One way to prevent additional roles from being added is to constrain the frame via the interface features. We have implemented this for the example of *smash* to the effect that when anchoring our `n0Vn1` class with the lemma *smash*, we obtain only the elementary tree frame pair where we have active voice, the EFFECTOR linked to the subject and the PATIENT linked to the object.

## 7 Conclusion

We proposed a constraint-based formulation of the principles underlying the linking algorithm from Van Valin (2005), instead of a procedural specification. The advantage is not only that we can separate between the linguistic generalizations to be captured and algorithmic considerations. There is also a straightforward way to implement this with XMG and, within this metagrammar framework, to connect it with existing implementations of LTAG, allowing for a neat separation between language-specific and language-independent linking constraints. From a less technical perspective, the presented work can be seen as the first attempt to fill an important gap in the theory of the shape of elementary entries that Frank's $\theta$-Criterion left open.

## Acknowledgments

We thank the three anonymous reviewers for helpful comments. The work presented in this paper was financed by the Deutsche Forschungsgemeinschaft (DFG) within the CRC 991.

## References

Anne Abeillé and Owen Rambow. 2000. Tree Adjoining Grammar: An Overview. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*, pages 1–68. CSLI.

Benoit Crabbé and Denys Duchier. 2004. Metagrammar Redux. In *International Workshop on Constraint Solving and Language Processing*, Copenhagen.

Benoit Crabbé, Denys Duchier, Claire Gardent, Joseph Le Roux, and Yannick Parmentier. 2013. XMG: eXtensible MetaGrammar. *Computational Linguistics*, 39(3):1–66.

Éric Villemonte de la Clergerie. 2013. Exploring beam-based shift-reduce dependency parsing with DyALog: Results from the SPMRL 2013 shared task. In *4th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL'2013)*, Seattle.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

D. Duchier, J. Le Roux, and Y. Parmentier. 2004. The Metagrammar Compiler: an NLP application with a Multi-Paradigm Architecture. In *Proceedings of the 2nd international Mozart-Oz Conference MOZ 2004, Lecture Notes in Computer Science, Vol. 3389, Springer*, Charleroi, Belgium.

Robert Frank. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives*. Ph.D. thesis, University of Pennsylvania.

Robert Frank. 2002. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Cambridge, Mass.

Claire Gardent and Laura Kallmeyer. 2003. Semantic Construction in FTAG. In *Proceedings of EACL 2003*, pages 123–130, Budapest.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoning Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–123. Springer, Berlin.

Laura Kallmeyer and Aravind K. Joshi. 2003. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Research on Language and Computation*, 1(1–2):3–58.

Laura Kallmeyer and Rainer Osswald. 2013. Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammar. *Journal of Language Modelling*, 1:267–330.

Laura Kallmeyer and Maribel Romero. 2008. Scope and situation binding in LTAG using semantic unification. *Research on Language and Computation*, 6(1):3–52.

Beth Levin and Malka Rappaport Hovav. 2005. *Argument Realization*. Cambridge University Press, Cambridge.

Timm Lichte and Simon Petitjean. 2015. Implementing semantic frames as typed feature structures with XMG. *Journal of Language Modelling*, 3(1):185–228.

Timm Lichte. 2015. *Syntax und Valenz. Zur Modellierung kohärenter und elliptischer Strukturen mit Baumadjunktionsgrammatiken*. Number 1 in Empirically Oriented Theoretical Morphology and Syntax. Language Science Press, Berlin.

Rebecca Nesson and Stuart M. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*, Malaga, Spain, 29–30 July.

Rebecca Nesson and Stuart M. Shieber. 2008. Synchronous vector tag for syntax and semantics: Control verbs, relative clauses, and inverse linking. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 9)*, Tübingen, Germany.

Stuart M. Shieber. 1994. Restricting the weak-generative capacity of synchronous Tree-Adjoining Grammars. *Computational Intelligence*, 10(4):271–385.

Robert D. Van Valin, Jr. 2005. *Exploring the Syntax-Semantics Interface*. Cambridge University Press.

K. Vijay-Shanker and Aravind K. Joshi. 1988. Feature structures based tree adjoining grammar. In *Proceedings of COLING*, pages 714–719, Budapest.

Stephen Wechsler. 2015. *Word Meaning and Syntax*. Oxford Surveys in Syntax and Morphology. Oxford University Press, Oxford.