

A Metagrammar of Causatives in Morphologically Rich Languages

Valeria Generalova¹, Simon Petitjean¹

¹Heinrich Heine University of Dusseldorf, Institute for Language and Information, Department of Computational Linguistics, TreeGraSP project, Univesitaetstr. 1, 40225 Dusseldorf, Germany

Abstract

Agglutinative languages are known to encode (almost) each grammatical category with a separate morpheme. It results in longer words that might be challenging for some NLP methods. However, the one-to-one correspondence between word segments and their meanings makes agglutinative languages especially suitable for descriptions with help of rule-based precision grammar formalisms. The present paper demonstrates a solution for describing agglutinative languages with help of eXtensible MetaGrammar (XMG). The main innovation of the present paper consists in a careful representation of agglutinative morphology, offering a clear distinction between derivational and inflectional affixes. In the metagrammar, morphology and syntax are represented in a single tree structure, where each morpheme is regarded as a leaf. This paper presents data from three genetically non-related languages: Finnish, Bashkir and Kannada. More languages can be easily added since linguistic generalizations are described separately from individual language properties. The paper focuses on a single type of linguistic constructions, namely, morphological causatives derived from transitive verbs.

Keywords

causative, verbal derivation, morphologically rich languages, syntactic trees, metagrammar, precision grammar,

1. Introduction

Currently, the most widespread approach for processing natural language comprises large corpora and machine-learning algorithms that automatically detect features relevant to the analysis. It performs best on widely spoken languages with little morphology, which constitute a minor fraction of all world languages. However, morphologically-rich languages encode information about their structure overtly, making the machine search of syntactic and semantic rules less relevant. Moreover, since the features of particular phenomena can be induced from the morphology, one does not need to collect large amounts of training data for these languages.

In this paper, we present a rule-based approach for creating a thorough analysis of a specific linguistic phenomenon. Namely, we present a formalized description of three-argument causative constructions in three genetically non-related languages, linking three language levels in the single output representation: morphology, syntax, and semantics.


This solution is based on the creation of a metagrammar [1], which is a factorized description

The International Conference and Workshop on Agglutinative Language Technologies as a challenge of Natural Language Processing (ALTNLP), June 7-8, 2022, Koper, Slovenia

✉ generalova@hhu.de (V. Generalova); petitjean@phil.uni-duesseldorf.de (S. Petitjean)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

to modify the feature structure of constituents but not the structure of the sentence itself. This fundamental difference proves it important to make a clear distinction between derivational and inflectional elements in the final solution.

The language sample of our project is controlled for having a single explicit derivational CAUS marker on a single verb. Thus, the suggested solution currently does not cover lexical or periphrastic causatives (e.g., like in English and French).

One of the goals of this paper is to present some challenges in handling inflectional morphology. For instance, some languages allow more than one noun to have an unmarked case (although respective case markers exist in the paradigm). This situation is called differential object marking [10] and is often observed in Turkic languages. Example (2) from Bashkir demonstrates this phenomenon.

- (2) min bala-nan xat jað-ðər-a-m BASHKIR
1.SG child-ABL letter write-CAUS-IPFV-1.SG
'I make the child write a letter.' [11, p. 225]

The nominative case is normally unmarked in Bashkir. Therefore the lack of an overt case marker on the word 'min' is well expected. In contrast, the accusative case has a corresponding segmental marker. It is omitted in sentence (2) because the object is non-specific (for a detailed analysis of this phenomenon see [12]).

Another trait of morphologically-rich languages is stacking several inflectional morphemes on the verb. Example (3) shows a sentence from Kannada (Dravidian), where tense and person are expressed with two different verbal suffixes.

- (3) Avanu-∅ nann-inda tīy-annu kud-is-id-anu KANNADA
3SG-NOM 1SG-INS tea-ACC drink-CAUS-PST-3SG.M
'He had/let me drink tea.' Cole 1983:121 cited in [13, p. 384, (vi)]

This situation is not uncommon in the world's languages and thus requires special attention.

These three languages – Finnish, Bashkir and Kannada – have been selected for presentation as they clearly exemplify the focus areas of this paper. The same architecture can be applied not only to other Uralic, Turkic or Dravidian languages but to a wide range of morphologically rich idioms (cf. [4]).

3. Background

3.1. Role and Reference Grammar

The underlying theoretical background of this paper is Role and Reference Grammar (RRG; Van Valin and LaPolla [14], Van Valin [5]) in its formalized version developed by [15].

The trees in this paper follow the RRG concept of the "layered structure of the clause" [5, 3-8]. The upper level is SENTENCE which contains a CLAUSE, or several CLAUSES. These notions are used in the common linguistic sense and play a minor role in the scope of this paper. The noteworthy layer in RRG is CORE which comprises a predicate with all its arguments. All the structures examined in this paper are COREs.

Adjuncts can be added to any layer of the as PERIPHERY. The difference between CORE elements and PERIPHERY in syntax is semantically motivated: what is included in the semantic

representation of the verb is considered CORE; everything else is PERIPHERY (see discussion of some apparent discrepancies in [5, 8]). Syntactic peripheral elements are not shown in the present paper, but the same mechanism is used for morphological elements.

In this respect, our solution follows one of the recent advances in RRG ([16], [17]): the layered structure of the word, i. e., the representation of a single graphical word with the principles of the syntactic layering. Most attention in this respect has so far been on head-marking languages. We claim that the internal structure of words in morphologically rich dependent-marking languages is vital for correct syntax-to-semantics linking. In our solution, morphological elements not contributing to syntactic changes are considered adjuncts (see below).

The RRG concept of the layered structure also applies to phrases (NPs, PPs and others). Thus, an NP has its CORE_N and NUC_N positions, where modifiers can attach. These syntactic layers are included in the architecture presented in this paper, although no language examples with modifiers are considered for clarity.

3.2. RRG Formalization

The formalization of RRG as Tree Wrapping Grammars, or TWG [15], inherits most principles from (lexicalized) Tree Adjoining Grammars [18, 19]. A TWG is a set of elementary trees that can be composed using different operations. Fig. 1 shows an example set of elementary trees and their composition, while the resulting analysis is shown in Fig.2. Feature structures can be attached to the nodes to carry morphosyntactic information (boxed numbers indicate shared values between features of a same tree). While lexicalization allows reducing the number of trees considered during parsing, the use of *lexical anchors* reduces the size of the grammar itself: each tree contains an anchor node (marked with \diamond). Anchoring is the insertion of a compatible lexical item below an anchor node.

The second crucial operation for building syntactic trees is substitution, i. e., the ability to replace a node with another tree. Trees can be inserted only to specifically selected nodes (marked with \downarrow) and must root in the nodes with the same category. The processing of a sentence is done only when each of the substitution nodes is provided with a tree.

In Fig. 1, the leftmost tree (rooted in CORE) has three substitution nodes (NPs). Each of them can take the rightmost tree below them. Note that this tree has NP as its root (highest node); therefore, it is eligible for each NP \downarrow node, but not for the AFF_CAUS \downarrow . Substitution (as lexical anchoring) triggers the unification of the feature structures attached to the nodes, leading the variables $\boxed{1}$ and $\boxed{2}$ to receive the value *ade*. This allows to propagate the value of some features across trees and, in our example, ensures that the case of the last argument of the verb (which comes from the nominal affix) is the expected one.

To handle adjuncts and operators, [15] following [18] suggested using sister adjunction. This operation requires specific trees, called auxiliary trees, whose root is marked with $*$. This operation is optional and unlimited, i. e. a tree can have any number of adjuncts, including zero. An auxiliary tree can attach only to nodes having the same category as its root. For instance, in Fig. 1, the small tree below, to which the morpheme *-i* anchors, is an auxiliary tree attaching to any node of the category VERB³. The third operation defined in [15], wrapping substitution,

³The position of the adjoined tree relatively to the other daughters of the adjunction site can be controlled with features attached to the edges of the tree. This is not shown in our solution.

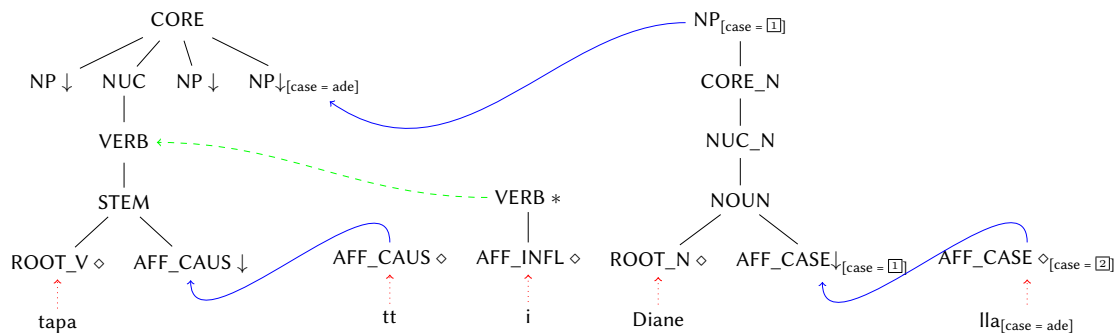


Figure 1: Partial composition of elementary trees for the sentence (1). Arrows, dashed arrows and dotted arrows indicate respectively substitution, sister adjunction and lexical anchoring.

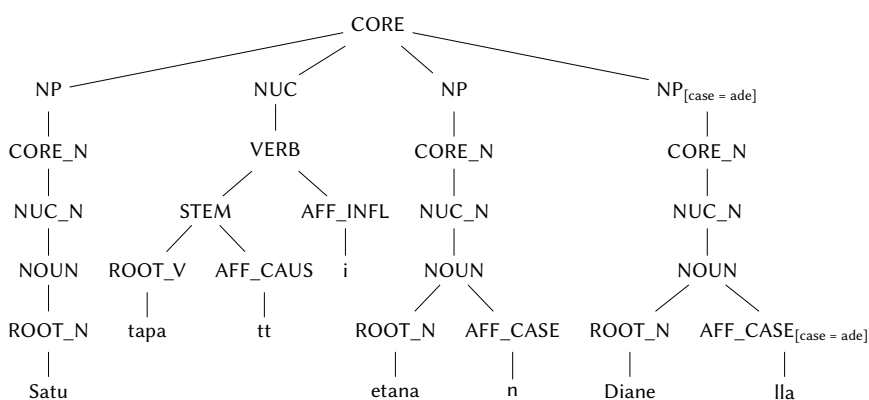


Figure 2: Derived tree for the sentence (1)

allows for dealing with long-distance dependencies and is not used in the present paper.

3.3. Metagrammars and parsing

The basic elements of an XMG metagrammar are called classes; they contain partial linguistic descriptions, which can be combined and reused. Working at this more abstract level helps avoid redundancies in the rules and express generalizations. In this work, the generalizations are partly multilingual and ease the integration of new languages into the resource.

Descriptions are made into dimensions, which allow separating the different levels of linguistic description (in our work, syntax, morphology and lexical information).

The principal part of our implementation effort, namely the description of the syntax and morphology, is done in the syn dimension of XMG, in which trees can be described using dominance and linear precedence between nodes. Nodes can be given morphosyntactic information (feature structures) or receive marks specific to the target formalism (anchor, sister adjunction or substitution nodes in our case). Unification variables can be used in all dimensions to refer

to nodes, values, imported classes, or feature structures.

Classes are organized in a hierarchy thanks to an import mechanism: a class importing another receives the description of the latter getting access to a subset of the variables defined inside of it. Descriptions contained in the high-level classes can therefore combine descriptions from lower-level classes and extend them (define constraints between nodes of different classes, add features). As in a logic program, constraints, instructions, and imports of classes can be combined in a conjunctive or disjunctive way to express alternatives.

Once the metagrammar is written, it needs to be processed by XMG to generate the resulting grammar, which the parser will use. This processing step, called compilation, consists in producing all the trees which match the description of a selected set of classes. Because of the disjunction and partial descriptions with constraints, the execution of a class can lead to the production of any finite number of trees, as long as no tree constraint is violated and all feature unifications are successful. The set of trees generated by a class is called a tree family, and is named after the class. The compiled descriptions are then given to the TuLiPA [20, 21] parser, which will use them for analyzing input sentences. For the analysis of each sentence, the features ascribed to each morpheme are used to select the relevant trees from the grammar and restrict the set of operations (described in Sec. 3.2) that can be applied between them.

In order to perform lexical anchoring, we separate the lexicon from the trees: we generate on one side an inventory of syntactic structures (in our case, syntactic and morphological), and on the other side a lexicon of segmental morphemes. This (language-specific) lexicon simply lists a set of morphosyntactic features and indicates in which tree family it can be used as a lexical anchor. For instance, the lexicon maps the Finnish morpheme *tapa* to the tree family `IntersectionFIN`, which lists all possible syntactic/morphological configurations for Finnish verbal roots. The unanchored trees (where the leaf receiving the lexical item is marked with \diamond) are described in the previously introduced *syn* dimension.

3.4. Situating our research

After describing key theoretical and technical concepts behind our research, we would like to situate it in a broader context of NLP practices and grammar engineering initiatives.

The most recent survey of approaches towards morphological processing of low-resource languages [22] summarizes existing tasks in computational morphology. However, they all have to do with morphological level only, without relating it to syntax or semantics. This is also applicable to the finite-state approaches to morphology like [23] or [24]. Therefore, our approach falls out of the main trend and thus requires additional description.

One of the closest comparable alternatives to the system we work with is the LinGO Grammar Matrix [25]. From the theoretical point of view, our system is different as it relies on Role and Reference Grammar (RRG), while the LinGO project is built with HPSG. Both theories make extensive use of features, which helps to use them in a formalized description. The Grammar Matrix features a special library on valence modifications [26]. Although this module is multilingual, it does not pay sufficient attention to the inner structure of words in morphologically-rich languages, in contrast to the approach presented in this paper.

The best developed RRG-based account for derivational morphology is presented in [27]. Although it relies on an early version of RRG (before the formalization by [15]), it describes

rules of lexical derivation as logical structures. The section dedicated to causatives pays specific attention to causative-forming affixes but does not deal with the question of handling the inflection of newly derived verbs. Also, this proposal does not offer any link between morphology and syntax, i. e., does not show argument structures allowed by causative verbs.

There are also XMG-based projects focused on representing morphological processes. Namely, [28] explores the semantics of the English suffix *-al*. However, we are dealing with a different kind of task, trying to tie together morphology and syntax, taking into account the structure of different parts of speech. A study of a morphologically rich language Ikota (Bantu family) [29] offers a syntax-compatible treatment of a rather complex morphology. The difference from our study is that it presents morphology in a flat way, while we establish a hierarchy of morphological operations. Not to mention that all these projects deal with one language each, while we set one of our main goals in encoding cross-linguistic generalizations applicable to genetically non-related languages.

4. Solution

The solution we present comprises two main parts: construction classes that capture cross-linguistic generalizations and language plugins that encode language-specific information as a feature structure. Our metagrammar decomposes the description of trees such as the ones shown in Fig. 1 into a set of classes, which are then combined thanks to the import mechanism. These classes separate the syntactic and morphologic parts of the linguistic analysis and can be reused to build trees for other types of constructions. The classes for three-argument causative constructions define a set of trees in which verbal roots can be anchored. The classes for arguments define a set of trees for nominal roots with slots for all types of adjuncts. These trees can fill the substitution slots of a verbal tree. The construction classes are built in a language-independent way. In order to generate the trees specific to a given language, the construction class must simply be combined with the language plugin class. The language-specific features of the nodes used in the construction (for instance, the case values of the arguments) receive through this process the values specified in the plugin. The plugin information also allows discarding the configurations that are not allowed, for example, because of the word order.

4.1. Standard case

In this section, we describe in detail all the steps in describing and parsing sentence (1).

Affixes for nominal cases are introduced in a class describing nouns, which generates two trees. The first one (4th tree of Fig.1) has a substitution node for a case affix. In (1), this is how ACC and ADESS noun phrases are built. The second variant is just the nominal root without overt case markers. Cases eligible for zero marking are listed in language plugins (for Finnish, it is just NOM). Variable sharing between the plugin and the tree will set the case feature of the NP node to NOM in the second variant of the Finnish noun tree.

Derivational verbal affixes (for the moment, only CAUS) are described as substitution nodes in the trees for verbal roots (1st tree of Fig. 1 for instance). Namely, we introduced a class *CausativeVerbalStem* for a causative stem, which adds a substitution node for an affix next to the verbal root. The verbal root is defined in a separate class which is also used to build

non-causative constructions. Inflectional verbal affixes are realized as auxiliary trees, such as the 2nd tree of Fig. 1, adjoining at the Verb level (corresponding to the graphical word of the verb). This decision allows for an economical yet precise representation of multi-morpheme words. In (1), there is a single synthetic marker *-i* for tense and person; more agglutination is described in the next section.

Once the verb is combined from morphemes, it starts functioning as a syntactic unit. The class describing the verbal morphology is imported in every class defining the argument structure of the verb, to make it lexicalized (with the verbal root as a lexical anchor). In our example, the upper-level class creates a causative-of-transitive construction. Importantly, it corresponds to the cross-linguistic notion of a causative construction. Therefore, in this construction, we encode the necessity of three NPs and one NUC with a causative verb inside but do not specify either the morphological information or the word order.⁴

The intersection of the verbal constructions class with the Finnish language plugin (by importing both classes in a new class *CausativeFinnish*) reveals that the PSA (an RRG term for the syntactic subject) is nominative and unmarked, the direct object is accusative, etc. The class *CausativeFinnish* is, in fact, the uppermost class where the verbal root is used. It generates the family of trees where the verbal roots of the lexicon can be anchored.

After compilation of the metagrammar, the parser considers all the relevant trees for the morphemes given in the input. For sentence (1), it produces the analysis shown in Fig. 2.

4.2. Several unmarked cases

Some languages happen to have several (in most cases, two) NPs without overt case marking in a single sentence. If a single case value could have been represented as a zero feature value, the task becomes problematic if two zeros have to encode different case values. To handle this issue, we introduce a special feature to the language plugin called *UnmarkedCases* and specify its value using a so-called *atomic disjunction*, which allows assigning a set of possible values instead of a single one.

In Bashkir, two cases can be unmarked: NOM and ACC. Using a standard disjunction would have generated two trees for unmarked nouns. In the parsing of sentence (2), *min* and *xat* could have been anchored to both these trees, while in our solution, they are both anchored to instances of the same tree (with an underspecified case value). The correct case value is selected when the lexical anchoring triggers the unification of the atomic disjunction with the case value coming from the lexical entry of the morpheme. The anchoring of the noun fails its case value is not listed in the atomic disjunction. In other words, we determine the case of the words *min* and *xat* to an extent, but not completely: we know that it must be either NOM or ACC, but nothing else.

So, once an unmarked word is encountered, it anchors to a nominal tree without the case marker node. When intersected with the Bashkir language plugins, the case feature on this NP receives the underspecified value, *nom or acc*⁵. Afterwards, when the substitution takes place and the NP tree is inserted into the CORE tree, the construction class sets the expected case value of all arguments depending on the construction requirements and the word order

⁴The code for a crucial subclass is given in Fig. 4.

⁵See Fig. 3 for a graphical view of the classes.

(e.g., the first NP must encode the causer and bear NOM case). At this moment, the value of the case required by the construction unifies with one of the values from the disjunction, and the substituted NP receives one single determined case. If the morphological case required by the construction does not match any of the values available in the disjunction, the substitution is not possible. For this reason, nouns without case marking cannot take the place of the causer in Bashkir.

4.3. Several inflectional affixes

Contrarily to Finnish, some languages express tense and person (as well as other verbal inflectional categories) in separate morphemes, like Bashkir in (2) and Kannada in (3). Our solution can handle any number of inflectional affixes as long as they are placed after the stem or before it, but not inside. Currently, there are no ordering constraints, i. e. the parser tries to adjoin the morphemes in all possible orders. This shortcut is feasible for parsing well-formed sentences, which is our current goal. The addition of such constraints is technically possible through the mechanism of *edge features* in TWG and constitutes one of the next steps of our work.

5. Conclusion

In this paper, we showed how a factorized description could help create a multilingual resource for analyzing a type of morphological constructions. We presented an in-depth approach that combines information from separate morphemes to create a detailed syntactic representation without analyzing any previously collected corpora. We also demonstrated that our architecture complies with linguistic evidence and theoretical assumptions. In future work, we would like to keep extending the resource developed in this study to more languages to account for more syntactic and morphological configurations. Namely, we aim to cover the second kind of valency-increasing constructions, i.e., applicatives.

References

- [1] M. Candito, A principle-based hierarchical representation of ltags, in: COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics, 1996.
- [2] B. Crabbé, D. Duchier, C. Gardent, J. L. Roux, Y. Parmentier, Xmg: extensible metagrammar, *Computational Linguistics* 39 (2013) 591–629.
- [3] S. Petitjean, D. Duchier, Y. Parmentier, Xmg 2: describing description languages, in: *International Conference on Logical Aspects of Computational Linguistics*, Springer, 2016, pp. 255–272.
- [4] V. Generalova, S. Petitjean, A prototype of a metagrammar describing three-argument constructions with a morphological causative, *Typology of Morphosyntactic Parameters* 3 (2020) 29–51.
- [5] R. D. Van Valin, Jr., *Exploring the syntax-semantics interface*, Cambridge University Press, 2005.

- [6] L. Pyllkkänen, The linking of event structure and grammatical functions in finnish, in: M. Butt, T. H. King (Eds.), *Proceedings of the LFG97 Conference*, CSLI Publications, University of California, San Diego, 1997, pp. 1–15.
- [7] B. Comrie, The syntax of causative constructions: cross-language similarities and divergences, in: M. Shibatani (Ed.), *Syntax and semantics: The grammar of causative constructions*, Academic Press, 1976, pp. 261–312.
- [8] R. M. W. Dixon, A. Y. Aikhenvald, Introduction, in: R. M. W. Dixon, A. Y. Aikhenvald (Eds.), *Changing valency: Case studies in transitivity*, Cambridge University Press, Cambridge, 2000, pp. 1–29.
- [9] S. Manninen, D. Nelson, What is a passive? the case of finnish, *Studia linguistica* 58 (2004) 212–251.
- [10] K. Sinnemäki, A typological perspective on differential object marking, *Linguistics* 52 (2014) 281–313.
- [11] N. K. Dmitriev, *Grammatika baškirskogo jazyka [Grammar of Bashkir]*, Nauka, Moscow, 2008 [1948].
- [12] A. R. Garejshina, *Mnogofaktornyj podhod k padezhnomu var'irovaniju (na materiale differencirovannogo padezhnogo markirovanija v bashkirskom jazyke) [Multifactorial approach to differential case marking (on the material of differential case marking in Bashkir)]*, Bachelor thesis, Moscow State University, 2014.
- [13] W. Foley, R. D. Van Valin, Jr., *Functional syntax and universal grammar*, Cambridge University Press, 1984.
- [14] R. D. Van Valin, Jr., R. J. LaPolla, *Syntax: Structure, meaning, and function*, Cambridge University Press, 1997.
- [15] R. Osswald, L. Kallmeyer, Towards a formalization of Role and Reference Grammar, in: R. Kailuweit, E. Staudinger, L. Künkel (Eds.), *Applying and expanding Role and Reference Grammar (NIHIN Studies)*, Freiburg: Albert-Ludwigs-Universität, Universitätsbibliothek, 2018, pp. 355–378.
- [16] D. Bentley, R. Marial Uson, W. Nakamura, R. D. Van Valin, Jr. (Eds.), *The Cambridge Handbook of Role and Reference Grammar*, Cambridge University Press, in press.
- [17] R. D. Van Valin, Jr., Head-marking languages and linguistic theory, in: B. Bickel, L. A. Grenoble, D. A. Peterson, A. Timberlake (Eds.), *Language typology and historical contingency: In honor of Johanna Nichols*, volume 104 of *Typological Studies in Language*, 2013, pp. 91–124.
- [18] L. Kallmeyer, R. Osswald, R. D. Van Valin, Jr., Tree wrapping for role and reference grammar, in: G. Morrill, M.-J. Nederhof (Eds.), *Formal Grammar*, 2013, pp. 175–190.
- [19] D. Arps, T. Bladier, L. Kallmeyer, Chart-based rrg parsing for automatically extracted and hand-crafted rrg grammars, in: *University at Buffalo, Role and Reference Grammar RRG Conference*, 2019.
- [20] L. Kallmeyer, T. Lichte, W. Maier, Y. Parmentier, J. Dellert, K. Evang, Tulipa: towards a multi-formalism parsing environment for grammar engineering, in: *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, 2008, pp. 1–8.
- [21] D. Arps, S. Petitjean, A parser for ltag and frame semantics, in: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
- [22] A. Wiemerslage, M. Silfverberg, C. Yang, A. D. McCarthy, G. Nicolai, E. Colunga, K. Kann,

Morphological processing of low-resource languages: Where we are and what’s next, arXiv preprint arXiv:2203.08909 (2022).

- [23] M. Hulden, Foma: a finite-state compiler and library, in: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2009, pp. 29–32.
- [24] K. Lindén, E. Axelson, S. Hardwick, M. Silfverberg, T. Pirinen, HFST–framework for compiling and applying morphologies (2011) 67–85.
- [25] E. M. Bender, S. Drellishak, A. Fokkens, L. Poulson, S. Saleem, Grammar customization, Research on Language and Computation 8 (2010) 23–72.
- [26] C. M. Curtis, A parametric implementation of valence-changing morphology in the LinGO Grammar Matrix, Master’s thesis, University of Washington, 2018.
- [27] F. J. Cortés Rodríguez, Derivational morphology in role and reference grammar: a new proposal, Revista Española de Lingüística Aplicada 19 (2006) 41–66.
- [28] M. Andreou, S. Petitjean, Describing derivational polysemy with xmg, in: Actes des 24ème Conférence sur le Traitement Automatique des Langues Naturelles. Volume 2-Articles courts, 2017, pp. 94–101.
- [29] D. Duchier, B. M. Ekoukou, Y. Parmentier, S. Petitjean, E. Schang, Describing morphologically-rich languages using metagrammars: a look at verbs in ikota, Language Technology for Normalisation of Less-Resourced Languages (2012) 55–66.

A. Classes of the metagrammar

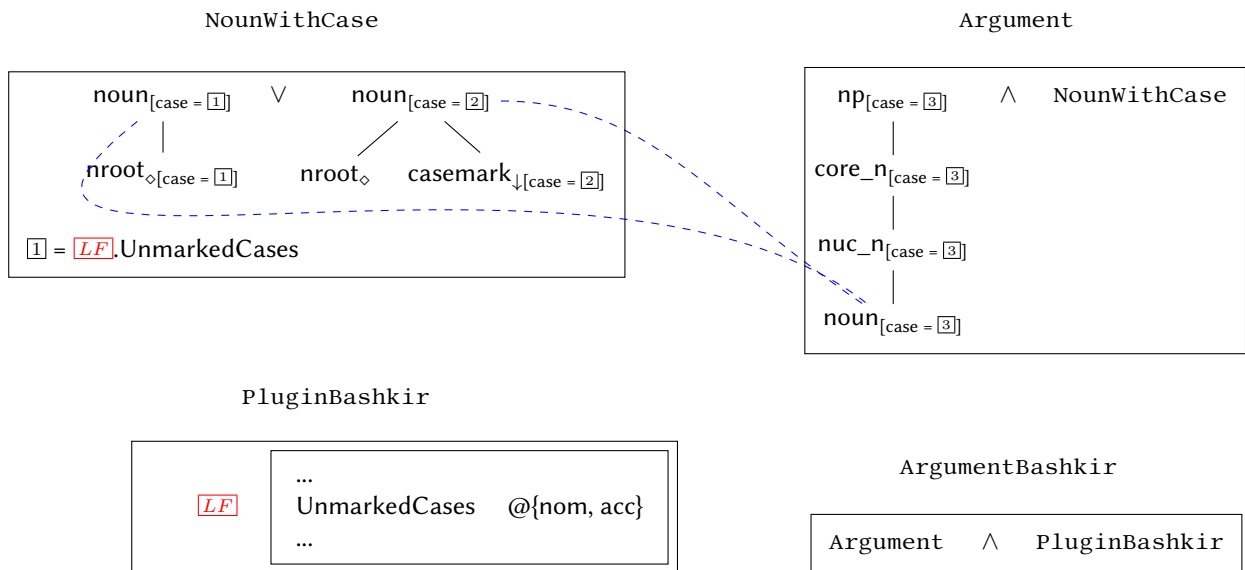


Figure 3: Metagrammatical classes used to build nominal trees for Bashkir. The operators \vee and \wedge respectively correspond to disjunction and conjunction, while `@` introduces an atomic disjunction. Dashed lines represent (node) variable sharing between classes. As an exported variable, `LF` is colored in red. This export allows to set the value of `[1]` according to the information provided by the plugin. Only the simple top level class and the plugin class are language specific.

```

class AllRangeCausConstruction
import VerbalSpine[]
export ?CORE ?LF ?Causer ?Causee ?Theme ?Recipient ?CaseCausee ?CaseTheme
declare ?LF ?Causer ?Causee ?Theme ?Recipient ?CaseCausee ?CaseTheme
{
  <syn>{
    node ?Causer (mark = subst) [cat = np, case = ?LF.MorphPsaCase];
    ?CORE -> ?Causer;
    {
      {
        ?Trans = intr;
        node ?Causee (mark = subst) [cat = np, case = ?LF.MorphUgCase];
        ?CORE -> ?Causee
      } | {
        ?Trans = tr;
        node ?Causee (mark = subst) [cat = np, case = ?CaseCausee];
        node ?Theme (mark = subst) [cat = np, case = ?CaseTheme];
        ?CORE -> ?Causee; ?CORE -> ?Theme
      } | {
        ?Trans = ditr;
        node ?Causee (mark = subst) [cat = np];
        node ?Recipient (mark = subst) [cat = np];
        node ?Theme (mark = subst) [cat = np];
        ?CORE -> ?Causee; ?CORE -> ?Recipient; ?CORE -> ?Theme
      }
    }
  }
}

```

Figure 4: Code for the class AllRangeConstructions. The symbols ; and | are respectively the conjunction and disjunction operators. The keyword **node** is used to declare a syntactic node, followed by its identifier, properties and features. The dominance constraint between nodes is expressed using ->. Linear precedence constraints are not used in this class, as it should account for the word order of any language. Forbidden configurations for a given language can then be filtered out by using the plugin information. All possible trees are generated for the 3 disjunctively separated blocks: for instance, 24 configurations of the arguments are generated for transitive verbs (?Trans = tr).