*Constructions* 16 (2024)
doi 10.24338/cons-684

# Constructional language processing and learning starting from unsegmented linguistic input

Veronica Juliana Schmalz[1,2], Lara Verheyen[3], Jens Nevens[3]
[1]Department of Linguistics, Faculty of Arts, KU Leuven
[2]Itec, imec research group at KU Leuven
[3]Artificial Intelligence Laboratory, Vrije Universiteit Brussel

**Abstract**

Constructionist theories of language acquisition claim that all linguistic knowledge can be captured in form-meaning mappings, which can vary in size and degree of abstraction and can contain information from all levels of linguistic analysis. Evidence from child language acquisition suggests that this linguistic knowledge is learned from situated, communicative interactions, where children observe unsegmented, continuous speech acts. However, current computational operationalisations of construction grammar do not fully corroborate these theoretical claims. Specifically, it is difficult to combine constructions of varying levels of granularity in the same constructional analysis and it is not feasible to learn computational construction grammars from unsegmented input. Both issues result from treating segmentation and language processing as separate steps, while in children these processes are intertwined. In response, we introduce two novel algorithms in this paper: one for the processing of constructions in which the linguistic forms are represented as unsegmented character sequences, and one for the learning of constructions starting from unsegmented linguistic forms. The novelty lies in operating directly on character sequences, thereby removing the need for a pre-segmentation step prior to linguistic processing. We operationalise these two algorithms within the framework of Fluid Construction Grammar and illustrate their application through several examples. Through these novel algorithms, we aim to offer greater flexibility to construction grammarians for implementing constructional analyses as well as to pave the way for experiments where the segmentation of continuous input is learned jointly with the constructions. In this way, we bring the computational implementation of construction grammar closer to its theoretical foundations.

## 1 Introduction

Children demonstrate a remarkable ability to acquire language. They are able to construct a "fluid and dynamic" (Bybee & Beckner 2009: p. 854) language system by organizing their experiences with language use through domain-general cognitive

**Corresponding author(s):** Veronica Schmalz, KU Leuven, veronicajuliana.schmalz@kuleuven.be 

processes (Bybee 2006; Tomasello 2003). The resulting linguistic knowledge is captured in a set of form-meaning mappings that children derive from situated, communicative interactions (Beuls & Van Eecke 2024). In terms of form, children observe unsegmented streams of input (Ambridge & Lieven 2011: p. 47), i.e. utterances produced by their caregivers, while the meaning is reconstructed from the environment in which the utterances are observed and from hypotheses about the underlying intentions of their caregivers. For example, the child's own name, *mommy* or *daddy* might be meaningful patterns referring to the child itself, its mother or its father, respectively. Larger patterns such as *you want more milk?* can also be meaningful when paired with the expectation of receiving more milk. In order to acquire such meaningful patterns, children require mechanisms that allow them to identify statistical regularities (Saffran et al. 1996; Aslin et al. 1996), as well as commonalities and differences in both the form and the meaning (Lieven et al. 1997; Tomasello 2003, 2006; Abbot-Smith & Tomasello 2006; Ambridge & Lieven 2015). The former helps children to segment the continuous input, while the latter allows them to generalise over observations, resulting in a more abstract and efficient linguistic framework. Indeed, when the child is in the beginning stages of language learning, they will not know the meaning of the word *milk*, nor that *milk* is a word. Instead, they will first store an observation such as *you want more milk?* and its hypothesised meaning holistically. By learning how to segment these utterances, and by generalising over similar observations, the child will eventually discover that *milk* is a separate word and its meaning is a nutritious white liquid. These mechanisms for linguistic segmentation and processing are highly intertwined and more experiments investigating the interaction between them are needed (Jones et al. 2010).

The patterns that children learn are mappings between form and a hypothesised meaning. According to constructionist theories as introduced by, among others, Fillmore (1968, 1988), Croft (1991), and Goldberg (1995, 2003), such mappings constitute the basic units of language. Constructions can consist in any combination of morphological, lexical and grammatical information on the form side, paired with semantic and pragmatic information on the meaning side (Fillmore 1968; Goldberg 2003, 2006). Moreover, there is no distinction between constructions that capture lexical or grammatical patterns, since "lexicon, morphology and syntax form a continuum of symbolic structures" (Langacker 1987: p. 3). Therefore, all constructions can be of "arbitrary size and degree of abstraction" (Beuls & Van Eecke 2023: p. 43). From this, it becomes apparent that constructions are not constrained at the word level. As a matter of fact, as Goldberg (2003: p. 219) states, constructions range from morphemes to words all the way up to "general linguistic patterns". This can be exemplified by morphological constructions, such as *pre-N* or *V-ing* (Goldberg 2006: p. 52), as well as idiomatic expressions such as the *let-alone* construction (Fillmore et al. 1988: p. 511) or *the X-er the Y-er* construction (Goldberg 2003: p. 55). Even though construction grammar mainly focuses on larger grammatical patterns (Ungerer & Hartmann 2023), construction morphology (Booij 2010; Audring et al. 2013; Booij & Audring 2017) investigates also the smaller patterns in language.

Computational models of construction grammar (Bergen & Chang 2005; Gaspers et al. 2011; Steels 2011b; Boas & Sag 2012; Dunn 2017) focus on operationalising the fundamentals of Construction Grammar in computational systems. They can be

used to test and implement construction grammar theories (Beuls & Van Eecke 2023). However, to date, the view that constructions can range from the morphological level all the way to the syntactical level is not truly reflected in computational construction grammar frameworks. Additionally, several of these frameworks separate the segmentation of linguistic utterances from the linguistic analysis and learning of constructions. This leads to the analyses and learned constructions being restricted to the specificity of the chosen segments (e.g. words or morphemes). Integrating analyses from different levels of granularity, for example combining morphological and idiomatic constructions, still represents a challenge in the computational implementation of linguistic theories. Take for example the sentence *He never went swimming let alone diving* showing, amongst others, morphological constructions (*V-ing*) and an idiomatic construction (*X let-alone Y*). Choosing a segmentation on the morphological level will allow to capture the *V-ing* construction, but will not treat *let alone* as a single segment. Vice-versa, choosing a less fine-grained segmentation might allow to capture *let alone*, but will not separate the verb stem from the suffix. In many computational approaches, the level of segmentation needs to be chosen on beforehand, and is typically based on whitespaces. This capturing neither the *-ing* nor the *let alone* as separate segments.

Not only computational construction grammar frameworks, but also many other models in computational linguistics and NLP at large, rely on a pre-segmentation step using various types of tokenisers (Sennrich et al. 2016; Wu et al. 2016; Kudo & Richardson 2018). However, reliance thereon has been argued to have several drawbacks. For one, any errors in the segmentation process might propagate throughout the subsequent linguistic analysis or influence the learning system that relies on it. Additionally, many of these tokenisers often lose fine-grained morphosyntactic information in low-resource languages and languages with more complex morphology than English (Kudo & Richardson 2018; Oudah et al. 2019; Clark et al. 2022) and have also been shown to introduce biases (Gaido et al. 2021; Petrov et al. 2023).

In this paper, we aim to bridge the gap between construction grammar theory and its computational implementation by introducing novel algorithms for representing, processing and learning computational construction grammars that work on unsegmented input.[1] Indeed, while construction grammar theory, inspired by language acquisition processes, states that constructions can contain information from all levels of linguistic analysis, computational implementations typically rely on a pre-segmentation step that is separate from processing. Similarly, constructional learning experiments begin by segmenting the input before learning constructions (see e.g. Nevens et al. 2022 and Doumen et al. 2023, 2024). This ultimately limits the constructional analysis and the learned constructions to the granularity of the segments that is chosen on beforehand. Therefore, we introduce a novel way of representing the form side of constructions, as well as processing mechanisms for both language comprehension and production. Instead of relying on separate segments, the form side of constructions can now be represented as continuous sequences of characters. These can in turn be used in constructional language processing through a novel mechanism that relies on a regular expression search of the character sequence. In this way, we can

[1] The algorithms are fully implemented and integrated in the open-source FCG framework, which can be found at: https://fcg-net.org

avoid the need for a pre-segmentation process, evading the percolation of segmentation errors and bearing the promise of being more language-independent. Moreover, we present a proof of concept of how this new representation can be used in learning constructions from semantically annotated corpora extending the work by Doumen et al. (2023, 2024). Together, the new representation, processing and learning mechanisms will enable construction grammarians (i) to have more flexibility in analysing and processing languages that have richer and more complex morphological systems, (ii) to analyse and process languages at varying degree of abstraction simultaneously in a more straightforward way (i.e. combining constructions for morphemes, words, multi-word expressions, idioms, grammatical patterns, etc.) and (iii) to automatically learn these distinctions in the form of constructions from unsegmented input.

We start the paper by describing how the form side of constructions is represented, processed and learned in different computational construction grammar approaches (Section 2). Then, we provide a high-level overview of constructional language processing in FCG (Section 3). Afterwards, we introduce our new method for processing and learning unsegmented linguistic forms in FCG (Section 4). Finally, we discuss the implications of our novel contribution and draw our conclusions (Section 5).

## 2 Related Work

In what follows, we provide an overview of computational construction grammar approaches and how they deal with the level of segmentation on the form side of the analysis. For each approach, we focus on answering two questions: (i) is there a pre-segmentation step that is separate from processing or learning? and (ii) is it possible to have a construction spanning the entire continuum, i.e. from the morphological level to the syntactic level? In this overview, we include approaches that identify themselves as "construction grammar" and that have a computational implementation. In particular, we discuss the Sign-based Construction Grammar (Boas & Sag 2012; Michaelis 2013), Embodied Construction Grammar (Bergen & Chang 2005) and Fluid Construction Grammar (Steels 2011a, 2017; van Trijp et al. 2022; Beuls & Van Eecke 2023) frameworks. We also include the computational approaches to learning construction grammars proposed by Dunn (2017, 2024) as well as by Gaspers et al. (2011, 2017) and Gaspers & Cimiano (2014).

As summarised by Ungerer & Hartmann (2023), Sign-based Construction Grammar (SBCG) (Boas & Sag 2012; Michaelis 2013) has focused on providing constructional analyses of a wide variety of constructions, ranging from morphological and lexical to syntactical constructions. Its focal point is most often on the description of linguistic phenomena (Ungerer & Hartmann 2023). Signs in SBCG are represented as feature structures that include "phonology, (morphological) form, syntax (e.g., a word's syntactic category and combinatorial potential), semantics (e.g., the frames that collectively define the meaning of a word, a word's referential index) and use conditions (e.g., the information-structure articulation of a phrasal type)" (Michaelis 2013: p. 2). In theory, SBCG thus models constructions from different levels of granularity (morphological to syntactical). One cannot really make the distinction between segmentation and processing in SBCG due to the absence of a computational formalism. Although implementations of Head-Driven Phrase Structure Grammar

could be re-used due to their close relatedness, no direct implementations of SBCG are known (van Trijp 2013a).

According to the overview presented by Chang (2008: Ch. 3), the Embodied Construction Grammar (ECG) formalism is capable of accommodating various forms and relations between them. Typically, phonological schemas and their temporal ordering relations are represented (Bergen et al. 2004). Such relations are used to describe concatenative morphology (affixation) as well as syntax (word order) (Chang 2008: p. 65). Within this framework, Bergen (2003) presents various morphological analyses of the English present tense and past tense. He discusses the meaningfulness of the agreement markers, morpho-phonological alternations and in general how the ECG framework may be used to represent and relate morphological forms. These analyses include morphological constructions for the verb root and past tense suffixes. Schneider (2010) extends ECG to include non-concatenative morphology. In particular, he presents a case study for Hebrew verbs, however, with the intention of providing a more general framework that can be extended to other morphological phenomena as well. Moreover, it shows how the morphological constructions can be integrated with syntactic structures, specifically argument structure constructions. The ECG framework is thus capable of accounting for different levels of granularity in the constructions and thereby does not seem to rely on a pre-segmentation process. However, this framework does not combine these different levels of granularity in bidirectional constructional language processing, as it is designed for comprehension only (van Trijp et al. 2022). Also, the full range of constructions (from morphological to syntactical) is not exploited in the learning of constructions in ECG introduced by Chang (2008), as it starts from a predefined lexicon (Doumen et al. 2025).

Dunn (2017, 2024) provides a computational framework for learning construction grammars. Within this framework, he represents constructions from different levels of abstraction as combinations of lexical, syntactical and semantic information. Although the learned constructions can range from lexical and item-based to abstract constructions, their representation does not go all the way down to the morpheme level. Indeed, Dunn (2024) states that the "expansion of computational CxG to include constructional morphology remains a problem for future work" (Dunn 2024: p. 10). The lexical constructions rely on a pre-segmentation based on spaces, while the item-based constructions do not rely on pre-segmentation as they make use of character-based embeddings.

Gaspers et al. (2011) introduce an algorithm for inducing grammars that consist in form-meaning mappings in an unsupervised way. In this algorithm, the first step is to learn a lexicon that is used to make generalisations in a second step. The lexicon is learned through finding statistical regularities based on either tokens or smaller units such as phonemes (Gaspers et al. 2017). In these models, the segmentation into word-like units is thus learned as a first step. Furthermore, Gaspers et al. (2014) provide a model for learning verb-general constructions, however, they do not focus on learning morphology.

Fluid Construction Grammar (Steels 2011a, 2017; van Trijp et al. 2022; Beuls & Van Eecke 2023) is a computational framework for representing, processing and leaning construction grammars. It relies on a pre-segmentation step that can be configured according to the desired linguistic analysis. Typically, utterances are segmented by

splitting on spaces. However, the FCG framework allows to specify custom segmentation algorithms, e.g. tailor-made for a linguistic phenomenon. For instance, FCG has been used to represent and process various morphological phenomena where the analyses start from morpheme-based segmentation processes that are tailored towards German (see e.g. van Trijp 2011b, 2013b and Schmalz & Cornillie 2022), Russian (see e.g. Gerasymova 2012), and Spanish (see e.g. Beuls 2012). One exception to the reliance on a fixed pre-segmentation step is the 'regex operator' introduced in Beuls (2013), which allows for more flexibility on the form side of constructions by using regular expressions and served as inspiration for the novel methodology in this paper. Although very promising, the functioning of this operator could not be fully extended to production[2], whereas computational construction grammar typically aspires to account for bidirectional language processing. Doumen et al. (2023, 2024); Nevens et al. (2022); Beuls & Van Eecke (2023) propose a way of learning computational construction grammars. However, in those experiments, the learning operators again start from pre-segmented linguistic utterances.

In short, the field of computational construction grammar does not fully reflect the view of "constructions all the way down" (Goldberg 2006: p. 18). Indeed, current computational construction grammars either start from a word-level segmentation for the learning process (as is the case with Dunn 2017, 2024, Gaspers et al. 2011, 2014, 2017, Doumen et al. 2023, 2024 and Nevens et al. 2022) or fail to fully reflect the bidirectionality assumed by construction grammarians, i.e. comprehending and formulating language (as is the case with Chang 2008, Dunn 2017, 2024, Gaspers et al. 2011, 2014, 2017 and Beuls 2013).

## 3 Constructional Language Processing in FCG

In this section, we provide a high-level overview of how constructional language processing is operationalised in FCG. This section is mostly aimed at readers that are not familiar with the framework. Here, we introduce concepts and terms that will facilitate the introduction of the novel character-based form representation in Section 4. This overview is based on the work by Steels (2017), Van Eecke (2018) and van Trijp et al. (2022). The interested reader is referred to these works for more detailed descriptions.

Constructional language processing in FCG is the task of finding a sequence of constructions, i.e. form-meaning pairs, that can map linguistic utterances to meaning representations, in the case of language comprehension, or meaning representations to linguistic utterances, in the case of language production. Central in this process are so-called *transient structures*. Transient structures contain all linguistic information known up to a specific point in language processing. The process starts with an initial transient structure that is constructed from the input utterance or meaning through a process called *de-rendering*. During processing, constructions are said to *apply* to transient structures and thereby expand the transient structure with new information. Different constructions can apply to the same transient structure, thereby creating different transient structures that lead to different paths in the constructional analysis.

[2] We obtained this information in communication with the author.

Each time a new transient structure is created, it undergoes a test to check whether it qualifies as a solution. A transient structure is typically considered a solution when the entire input utterance (in comprehension) or meaning (in production) is consumed and no more constructions can apply onto it. When a solution is reached, the construction application process stops and the resulting meaning (in comprehension) or the resulting utterance (in production) is then extracted from the final transient structure through *rendering*. Figure 1 gives a schematic overview of such a construction application process.

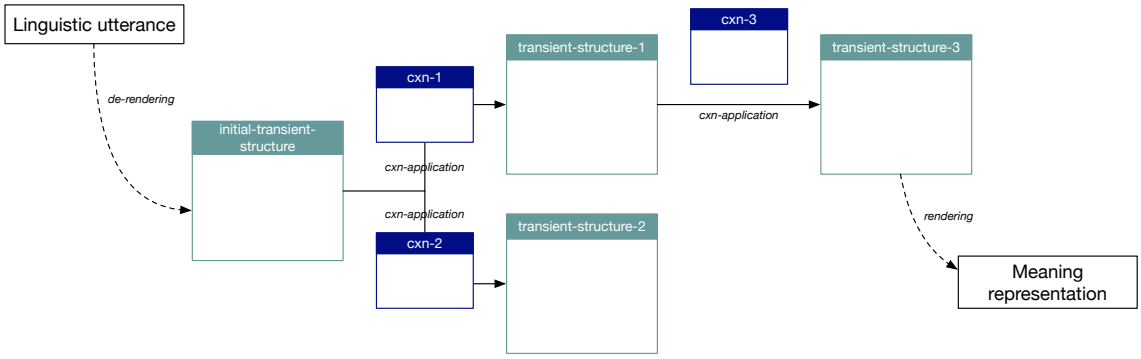



Figure 1: Schematic overview of the construction application process in comprehension. The process starts with de-rendering the linguistic utterance in the initial transient structure, on which two constructions can apply, resulting in two separate transient structures. Only on the first transient structure, another construction can apply leading to the transient structure that is a solution, from which the resulting meaning representation is extracted through a rendering process.

Both constructions and transient structures are represented as feature structures. They can contain linguistic information from all levels of linguistic analysis, such as morphosyntactic, phonetic and phonological information on the form side, and semantic and pragmatic information on the meaning side. FCG does not have a predefined set of features. Instead, the grammar engineer is free to choose the features that best fit the desired analysis. Features are further organised into *units*, grouping together relevant bundles of features. Transient structures thus consist of multiple units. There is a special unit, called ROOT, which is used to store the input of the construction application process, i.e. the form in comprehension or the meaning in production. Constructions, on the other hand, are separated into *pre-condition* units and *post-condition* units. The pre-condition units are further split in a *form* side and a *meaning* side. This facilitates bidirectional processing using the same constructions.

Constructional language processing is nearly identical for both language comprehension and production. In general, constructions will check their pre-conditions with the transient structure and, if they are met, expand the transient structure with new information. In comprehension, a construction can apply whenever its pre-conditions on the form side are met. In particular, this means that all features on the form sides of the construction precondition units need to be found in some units in the transient structure and *match* with those features. When a construction applies in comprehension, it *merges* its post-condition units, as well as its pre-condition units from the meaning side into the transient structure. Similarly, in production, the meaning sides of the pre-conditions of the construction should be met in the transient structure. When this is the case, the construction can merge its post-conditions, as well as its pre-conditions from the form side into the transient structure. The match and

merge operations are unification-based operations (Steels & De Beule 2006). Figure 2 illustrates this process schematically in the comprehension direction.

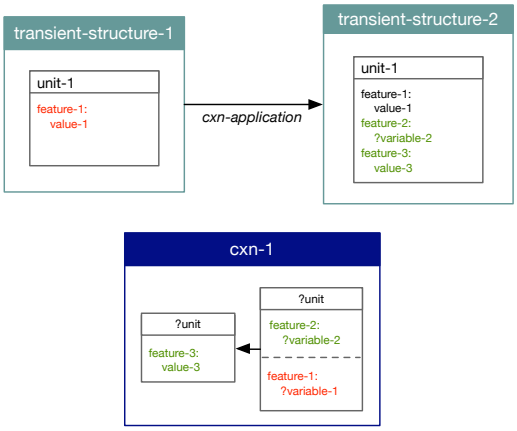



Figure 2: Schematic overview of a single construction application in comprehension. The features on the form side of the construction's pre-condition units match with some features in the transient structure (in red). The construction can apply and merges its post-conditions and pre-conditions on the meaning side into the transient structure (in green).

# 4 Character-based form representation

The novel methodology we introduce here for representing linguistic forms in computational construction grammars is based on sequences of characters. This new representation removes the need for a pre-segmentation step. Along with its accompanying processing and learning mechanisms, it thereby enables linguistic forms of varying degrees of granularity to be represented and processed in the same analysis, in turn closing the gap between the theory of construction grammar and its computational operationalisation in FCG. The main benefit of this representation thus lies in its flexibility, both for grammar engineers operationalising a linguistic analysis in FCG and for the automatic learning of constructions. In the following sections, we introduce the character-based form representation (Section 4.1) and how it is used for constructional language processing (Section 4.2) and learning (Section 4.3). We demonstrate its flexibility by providing several examples.

## 4.1 Representation

First and foremost, we need a way to represent linguistic forms such that they can be processed through construction application in FCG. In this novel methodology, linguistic forms are represented by one or more SEQUENCE predicates. A SEQUENCE predicate consists of three parts: a sequence of characters, followed by its left and right boundary. The boundaries represent the index of the leftmost and the rightmost character of the sequence.

De-rendering an input utterance in comprehension is straightforward, as illustrated in Figure 3. The initial transient structure obtained by de-rendering the utterance *John takes Jill for granted* contains a single SEQUENCE predicate, containing the character sequence `[john takes jill for granted]`, with boundaries 0 and 27. Note that

the boundaries indicate positions *in between* the characters of the sequence. Specifically, the index 0 denotes the left side of the initial `J` while the index 27 denotes the right side of the final `d`. As such, the entire sequence of characters may be annotated with indexes as seen in Example 4.1. The SEQUENCE predicates, however, only capture the leftmost and rightmost index. This convention for indexing and representing boundaries allows to express adjacency between sequences of characters by making their boundaries equal, which will be illustrated later on.

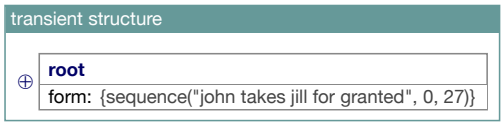



Figure 3: Initial transient structure obtained by de-rendering the utterance *John takes Jill for granted*. De-rendering yields a single SEQUENCE predicate containing the entire utterance and its boundary indexes.

**Example 4.1.**

$$[_0\mathtt{j}_1\mathtt{o}_2\mathtt{h}_3\mathtt{n}_4\ {}_5\mathtt{t}_6\mathtt{a}_7\mathtt{k}_8\mathtt{e}_9\mathtt{s}_{10}\ {}_{11}\mathtt{j}_{12}\mathtt{i}_{13}\mathtt{l}_{14}\mathtt{l}_{15}\ {}_{16}\mathtt{f}_{17}\mathtt{o}_{18}\mathtt{r}_{19}\ {}_{20}\mathtt{g}_{21}\mathtt{r}_{22}\mathtt{a}_{23}\mathtt{n}_{24}\mathtt{t}_{25}\mathtt{e}_{26}\mathtt{d}_{27}]$$

## 4.2 Processing

To use the character-based form representation in constructional language processing, we need to specify (i) how the form side of constructions can be matched with the transient structure, (ii) how the ROOT unit is updated when a construction effectively matches and (iii) how constructions handle the adjacency of character sequences.

(i) Matching is achieved through the use of a straightforward regular expression search that looks up the character sequence of the SEQUENCE predicate in the transient structure. If it is found, the indices representing the position of the matched string are returned and bound to the left and right boundary in the SEQUENCE predicate. (ii) After the matching, the ROOT unit is updated. Concretely, the matched string is removed from the SEQUENCE predicate in the ROOT in which it was found. The ROOT is then updated with one or more new SEQUENCE predicates with updated boundaries. (iii) Adjacency is expressed by variable equality of the boundaries of different SEQUENCE predicates.

We illustrate the processing mechanism using a construction and a transient structure that both contain one single SEQUENCE predicate. For example, consider the JILL-CXN in Figure 4. Its pre-conditions on the form side contain a single predicate: (SEQUENCE "JILL" ?LEFT ?RIGHT). Note that the FORM feature is annotated with +HANDLE-REGEX-SEQUENCES+. This indicates a procedural attachment for this feature (see Van Eecke 2018: p. 44 for an explanation on procedural attachment in FCG), meaning that it will not be matched to the transient structure through regular unification (as indicated in Section 3), but through the regular expression search as explained above.

We will now apply the JILL-CXN from Figure 4 to the transient structure of Figure 3. The resulting transient structure is shown in Figure 5. Concretely, for this construction to apply, the character sequence of the construction `[jill]` is looked up in the character sequence of the ROOT: `[john takes jill for granted]`. This search returns the leftmost and rightmost index of the matching sub-string, in this case 11 and 15. These indexes are used to instantiate the boundaries ?LEFT and ?RIGHT of the

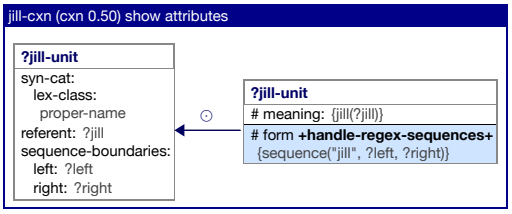



Figure 4: The JILL-CXN maps between the form *jill* and the meaning in the form of the predicate (JILL ?JILL). The form contains a single SEQUENCE predicate with the character sequence `[jill]`.

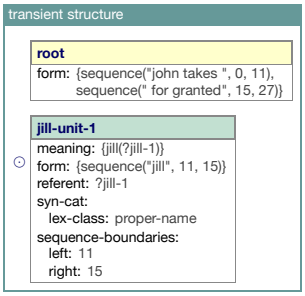



Figure 5: The transient structure after the application of the JILL-CXN from Figure 4 applied on the transient structure from Figure 3. The root is updated and contains two discontinuous SEQUENCE predicates obtained by removing the character sequence "jill" from the SEQUENCE predicate in the root in the initial transient structure.

SEQUENCE predicate of the JILL-CXN, as can be seen in the JILL-UNIT-1 in Figure 5. The SEQUENCE predicate of the ROOT has to be updated now that the JILL-CXN has consumed the character sequence `[jill]`. The updated ROOT can be seen in Figure 5. It now contains two SEQUENCE predicates with updated boundaries. Note the spaces at the end of `[john takes ]` and at the start of `[ for granted]` as these are treated as any other character.

The processing of a larger pattern where boundaries are made equal is demonstrated in Figure 6. The transient structure on the left contains two noun-phrase units, one for `[jill]` and one for `[john]` and also the TAKES-UNIT-2 representing the verb form `[takes]`. The ROOT unit still contains two spaces from the input utterance, as well as the character sequence `[ for granted]`. In all units, the sequence boundaries are instantiated with their indexes. On the bottom of the figure, there is the X-TAKES-Y-FOR-GRANTED-CXN. This construction matches on some noun phrase for the X slot (?X-UNIT), on some verb with lemma 'take' (?VERB-UNIT), on another noun phrase for the Y slot (?Y-UNIT), on the fixed character sequence `[ for granted]` in the ?FOR-GRANTED-UNIT and on two spaces in the ?CLAUSE-UNIT. The SEQUENCE predicates of the construction and their matching SEQUENCE predicates in the ROOT of the transient structure are highlighted in the same colour. The X-TAKES-Y-FOR-GRANTED-CXN handles word order by making boundary variables of SEQUENCE predicates equal. Specifically, the left boundary of `[ for granted]` is the same as the right boundary for the Y slot, indicated with the variable ?RIGHT-Y. This means that the Y slot should immediately precede the character sequence `[ for granted]`. Similarly, the first space (highlighted in green) should immediately follow the X slot (?RIGHT-X) and immediately be followed by the verb (?LEFT-VERB). Finally, the second space (highlighted in yellow) should immediately follow the verb (?RIGHT-VERB) and precede the Y slot (?LEFT-Y). All of these conditions are indeed satisfied when we consult the instantiated boundaries in the transient structure on the left. Hence, the construction can apply, resulting in the transient structure on the right, which has now consumed all characters from the ROOT and attributes the meaning (DOES-NOT-VALUE ?JOHN-6 ?JILL-3) to the utterance *John takes Jill for granted*. Note that in the case of a construction containing

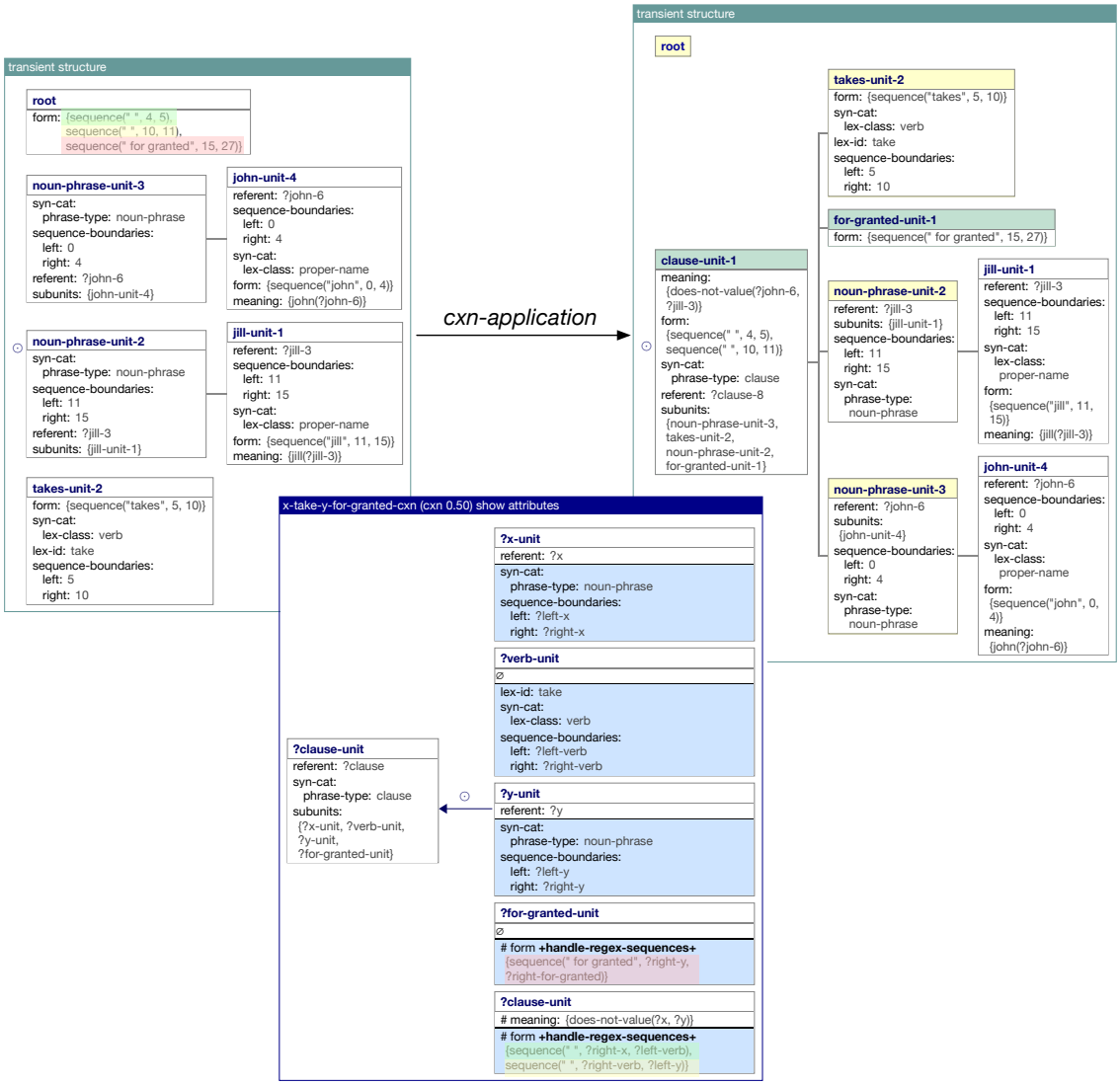



Figure 6: The application of the X-TAKE-Y-FOR-GRANTED-CXN. The construction applies on a transient structure (shown on the left) resulting in an updated transient structure (shown on the right). Sequence predicates in the form feature that match on the root are highlighted.

multiple SEQUENCE predicates, the match is only successful whenever the instantiated boundaries are non-overlapping.

Next to larger patterns, the character-based form representation can be easily used for constructions that represent morphemes. This is illustrated using the Italian verbs *mangiare* and *guidare* from the first conjugation in Figure 7. This figure shows the sequence of transient structures (top) obtained by consecutively applying the constructions on the bottom. In the transient structures, only the ROOT unit is shown. Similarly, in the constructions, only units containing SEQUENCE predicates are shown. Starting with the verb form `[guidavo]` (imperfect form of to drive, i.e. *I was driving*), first a construction for the stem `[guid]` applies, afterwards a construction for the 1st person singular marker `[o]` applies and finally, a construction for the first conjugation imperfect tense marker `[av]` applies. Each time, the SEQUENCE predicate in the ROOT is updated, until it is empty in the rightmost transient structure.

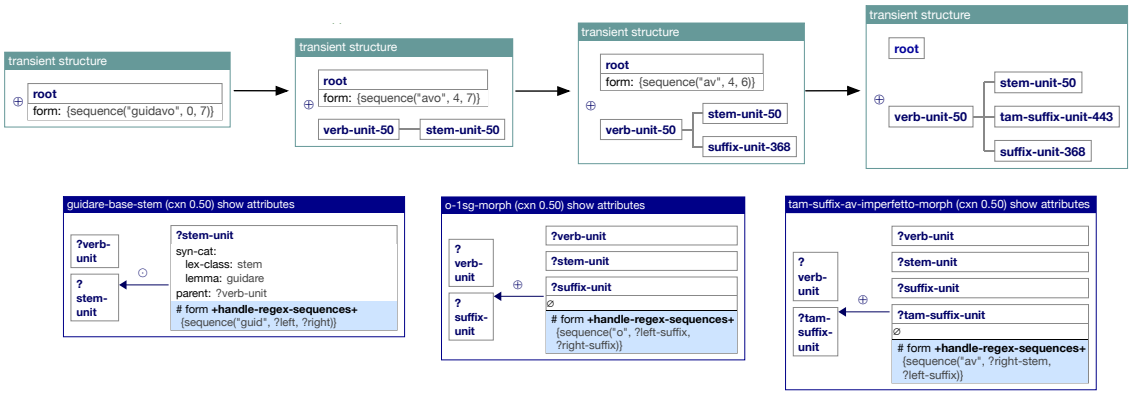



Figure 7: Application of several constructions to analyse the verb `[guidavo]`. The three constructions that match on the form in the ROOT are shown. They match on the stem (`[guid]`), the marker for 1st person singular (`[o]`) and the marker for the first conjugation imperfect tense (`[av]`). Only units that contain a SEQUENCE predicate are expanded, the others are collapsed.

In the above examples, we focused on comprehension. In production, the form side is handled by merging SEQUENCE predicates with uninstantiated boundaries in the corresponding units in the transient structure. In rendering, the final transient structure is scanned for all SEQUENCE predicates and the character sequences of these predicates are combined, taking into account the boundaries, to obtain a linguistic utterance. For example, if the resulting transient structure contains the SEQUENCE predicates (SEQUENCE "COMPR" ?L1 ?R1), (SEQUENCE "AV" ?R1 ?R2) and (SEQUENCE "O" ?R2 ?R3), the rendering combines the strings in the predicates based on the variable equalities and return the string "compravo". In the case that not all SEQUENCE predicates are sequential due to boundaries that are not made equal, the rendering will choose an order.

## 4.3 Learning

In FCG, constructions can be learned from semantically annotated corpora by looking for *syntactico-semantic generalisations* (Doumen et al. 2023, 2024). These are generalisations over form-meaning mappings that capture differences and similarities in both the form and the meaning. This process is inspired by the way in which children

acquire language, namely through pattern finding mechanisms with which they find generalisations in the linguistic input that they perceive (Tomasello 2003; Goldberg 2006; Behrens 2009). These claims are supported by empirical evidence (Tomasello 2003; Behrens 2009; Hartmann et al. 2021; Koch et al. 2022) as well as computational evidence (Steels 2004; Spranger 2017; Doumen et al. 2023, 2024).

The learning happens by processing the semantically annotated corpus in sequence. Given a form-meaning pairing, the learning algorithm will first try to process it with the already acquired constructions. If this fails, the learning algorithm makes new constructions by generalising over the observed form-meaning pairing and the acquired constructions. The best generalisation, i.e. the one capturing the most similarities in both form and meaning, is used. The type of utterances or speech acts observed by the learning algorithm thus depend on the semantically annotated corpus. Corpus entries may consist of well-formed sentences, e.g. in synthetic corpora such as CLEVR (Johnson et al. 2017), or contain incomplete sentences and sentences with disfluencies, e.g. in the transcribed corpora available via CHILDES (MacWhinney 1996).

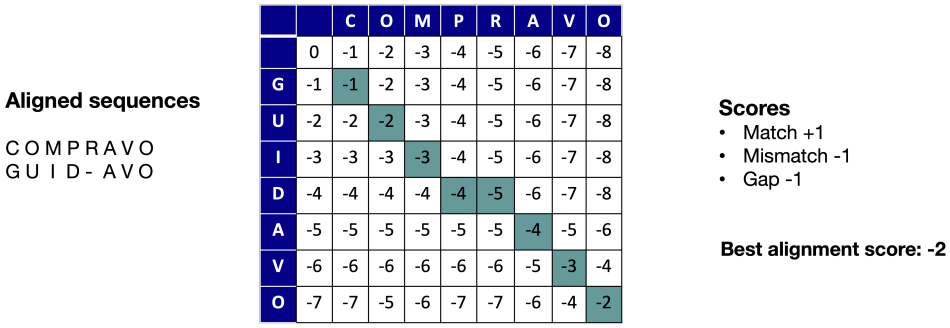
Here, we introduce a novel algorithm for one aspect of the learning mechanisms of Doumen et al. (2023, 2024), namely for finding generalisations on the form side using the newly introduced SEQUENCE predicates. This removes the need for pre-segmenting the form, also in construction learning experiments. The algorithm combines string alignment (cf. Section 4.3.1) and pattern matching (cf. Section 4.3.2) in order to compute differences and similarities given two sets of SEQUENCE predicates. Concretely, we first compute a maximal alignment between the two character sequences. From the aligned character sequences, we extract the generalisation through pattern matching. At the end of this section, we demonstrate how this algorithm can be used in learning constructions (cf. Section 4.3.3).

To introduce the generalisation algorithm, we work out the example of generalising over (SEQUENCE "COMPRAVO" ?L1 ?R1) and (SEQUENCE "GUIDAVO" ?L2 ?R2). The resulting generalisation corresponds to the SEQUENCE predicate that captures the common characters in both sequences, namely (SEQUENCE "AVO" ?G1 ?G2). From the differences, the SEQUENCE predicates (SEQUENCE "COMPR" ?L1 ?X5) and (SEQUENCE "GUID" ?L2 ?X4) are constructed. In terms of syntactico-semantic generalisation, it is thus possible to learn a construction that captures the suffix *-avo*. This construction has a single slot, which can be filled by constructions that capture the verb stems *compr* and *guid*.

In what follows, we will refer to the first input predicate as $S_1$ and to the second input predicate as $S_2$. For clarity reasons, we illustrate the algorithm by generalising over two sets containing just a single predicate. However, the algorithm can be applied in a similar way to generalise over sets of multiple SEQUENCE predicates.

### 4.3.1 String alignment

The first step in the generalisation algorithm is to find a maximal alignment between two sequences of characters. Here, we use the most canonical sequence alignment algorithm, being the Needleman-Wunsch algorithm (Needleman & Wunsch 1970). Other sequence alignment algorithms could also be used.




| | | C | O | M | P | R | A | V | O |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| G | -1 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| U | -2 | -2 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| I | -3 | -3 | -3 | -3 | -4 | -5 | -6 | -7 | -8 |
| D | -4 | -4 | -4 | -4 | -4 | -5 | -6 | -7 | -8 |
| A | -5 | -5 | -5 | -5 | -5 | -5 | -4 | -5 | -6 |
| V | -6 | -6 | -6 | -6 | -6 | -6 | -5 | -3 | -4 |
| O | -7 | -7 | -5 | -6 | -7 | -7 | -6 | -4 | -2 |

Figure 8: Optimal sequence alignment between the sequences `[compravo]` and `[guidavo]` obtained with the Needleman-Wunsch algorithm. The optimal alignment of the two sequences is determined by computing all alignments of all subsequences and combining the optimal results (highlighted cells). The optimal alignment consists of mismatches between `[comp]` and `[guid]`, a gap for the `r` of `[compravo]`, and a match for `[avo]`. Using the scoring scheme on the right, this yields an alignment score of -2.

The result of the sequence alignment algorithm for aligning the sequences `[compravo]` and `[guidavo]` is illustrated in Figure 8. The optimal alignment is shown on the left side of the figure. This alignment has a mismatch between the subsequences `[comp]` and `[guid]`, a gap for the character `r` of `[compravo]` and a match for the subsequences `[avo]`. The scores for a match, a mismatch and a gap are set to default values of 1, -1 and -1, respectively. The alignment score of -2 is obtained by adding 1 for every matching character and subtracting 1 for every mismatched character and for every gap. Every other alignment has a worse alignment score. In other words, this alignment maximises the number of matching characters.

The aligned sequences are converted back to SEQUENCE predicates. As an intermediate step, one SEQUENCE predicate per character is used. This is necessary for the pattern matching step which will follow. For the boundaries of the SEQUENCE predicates, we re-use the boundary variables of the original inputs and introduce new variables where necessary. Here, as a convention, we use new variables with ?X for the first input $S_1$ and new variables with ?Y for the second input $S_2$. Concretely, for the aligned sequences `[compravo]` and `[guid_avo]`, this means that we now have the following sets of aligned sequences predicates:

(SEQUENCE "C" ?L1 ?X1) – (SEQUENCE "G" ?L2 ?Y1)
(SEQUENCE "O" ?X1 ?X2) – (SEQUENCE "U" ?Y1 ?Y2)
(SEQUENCE "M" ?X2 ?X3) – (SEQUENCE "I" ?Y2 ?Y3)
... – ...
(SEQUENCE "O" ?X7 ?R1) – (SEQUENCE "O" ?Y7 ?R2)

### 4.3.2 Pattern Matching

The pattern matching part of the algorithm orderly traverses the aligned SEQUENCE predicates, compares each of the aligned characters and divides the predicates into three groups:

1. The generalisation - the SEQUENCE predicates that $S_1$ and $S_2$ have in common

2. The $S_1$ delta - the SEQUENCE predicates of $S_1$ that are not part of the generalisation
3. The $S_2$ delta - the SEQUENCE predicates of $S_2$ that are not part of the generalisation

Whenever two aligned characters are the same, one of the two SEQUENCE predicates is added to the generalisation. Conversely, when the characters are different, the SEQUENCE predicates are added to the $S_1$ delta and the $S_2$ delta, respectively. Gap characters are never added to the generalisation nor to the deltas. Predicates that are added to the generalisation are again given new boundary variables. However, a list of variable mappings is kept for both $S_1$ and $S_2$ such that the new variables in the generalisation can be translated back to the original variables in $S_1$ and $S_2$. This will be necessary when using the output of the generalisation algorithm for learning constructions. Finally, when the SEQUENCE predicates are divided into groups, the adjacent single-character SEQUENCE predicates are merged into SEQUENCE predicates with multiple character sequences. Two predicates are adjacent when the right boundary variable of one predicate is the same as the left boundary variable of the other predicate or vice-versa.

The complete output of the generalisation algorithm, including the generalisation, the deltas and the variable mappings, for the predicates (SEQUENCE "COMPRAVO" ?L1 ?R1) and (SEQUENCE "GUIDAVO" ?L2 ?R2) consists of the following parts:

- Generalisation: (SEQUENCE "AVO" ?G1 ?G2)
- $S_1$ delta: (SEQUENCE "COMPR" ?L1 ?X5)
- $S_2$ delta: (SEQUENCE "GUID" ?L2 ?Y4)
- $S_1$ mappings: ?X5 $\leftrightarrow$ ?G1, ?R1 $\leftrightarrow$ ?G2
- $S_2$ mappings: ?Y4 $\leftrightarrow$ ?G1, ?R2 $\leftrightarrow$ ?G2

Pseudocode for the complete generalisation algorithm, including both the string alignment and pattern matching phases, is provided in Algorithm 1.

### 4.3.3 Learning Constructions

The generalisation, the deltas and the variable mappings obtained through pattern matching can now be used for building constructions from the syntactico-semantic generalisation. This is illustrated in Figure 9, continuing the example of generalising over *guidavo* and *compravo*. On the semantic side, the verb forms are provided with a meaning representation that captures the activity of the verb (i.e. *drive* and *buy*), the person (i.e. *me* to indicate first person) and the imperfect tense (i.e. the event is simultaneous with some time point in the past). These meanings are represented through the predicates (ACTIVITY ?VERB ?EVENT), (PERSON ?PERSON ?AGENT), (SIMULTANEOUS ?TIME-POINT ?EVENT) and (TIME-POINT ?RECALLED-POINT ?TIME-POINT).

The result of the syntactico-semantic generalisation is shown on the bottom half of Figure 9. Concretely, the similarities on the form side and the meaning side result in the AVO-CXN, capturing the suffix *-avo* mapped to the meaning of some event that is viewed from the 1st person and occurs simultaneously with some recalled time point in the past. The differences on the form side and the meaning side result in the GUID-CXN and the COMPR-CXN. The former associates the stem *guid* with the agent of

**Algorithm 1** Algorithm for the generalisation of two sequence predicates $S_1$ and $S_2$

**function** GENERALISE-SEQUENCES($S_1$, $S_2$)
  ▷ *Compute all alignments A using Needleman-Wunsch* ◁
  $A \leftarrow$ Sequence-Alignment$(S_1, S_2)$
  $R \leftarrow \emptyset$ ▷ *Resulting generalisations R*
  **for all** $a \in A$ **do**
    $Gen, \delta S_1, \delta S_2 \leftarrow \emptyset$
    **for** $(S_1^c, S_1^{lb}, S_1^{rb}), (S_2^c, S_2^{lb}, S_2^{rb}) \in a$ **do**
      ▷ $S^c$ *are characters from the alignment* ◁
      ▷ $S^{lb}$ *and* $S^{rb}$ *are left and right boundaries from the alignment* ◁
      **if** $S_1^c =$ **_ then** ▷ *Gap in* $S_1$
        push $(sequence\ S_2^c\ S_2^{lb}\ S_2^{rb})$ to $\delta S_2$
      **else if** $S_2^c =$ **_ then** ▷ *Gap in* $S_2$
        push $(sequence\ S_1^c\ S_1^{lb}\ S_1^{rb})$ to $\delta S_1$
      **else if** $S_1^c = S_2^c$ **then** ▷ *Match*
        push $(sequence\ S_1^c\ S_1^{lb}\ S_1^{rb})$ to $Gen$
      **else** ▷ *Mismatch*
        push $(sequence\ S_1^c\ S_1^{lb}\ S_1^{rb})$ to $\delta S_1$
        push $(sequence\ S_2^c\ S_2^{lb}\ S_2^{rb})$ to $\delta S_2$
    $Gen \leftarrow$ merge-adjacent-sequences$(Gen)$
    $\delta S_1 \leftarrow$ merge-adjacent-sequences$(\delta S_1)$
    $\delta S_2 \leftarrow$ merge-adjacent-sequences$(\delta S_2)$
    push $(Gen, \delta S_1, \delta S_2)$ to $R$
  **return** $R$

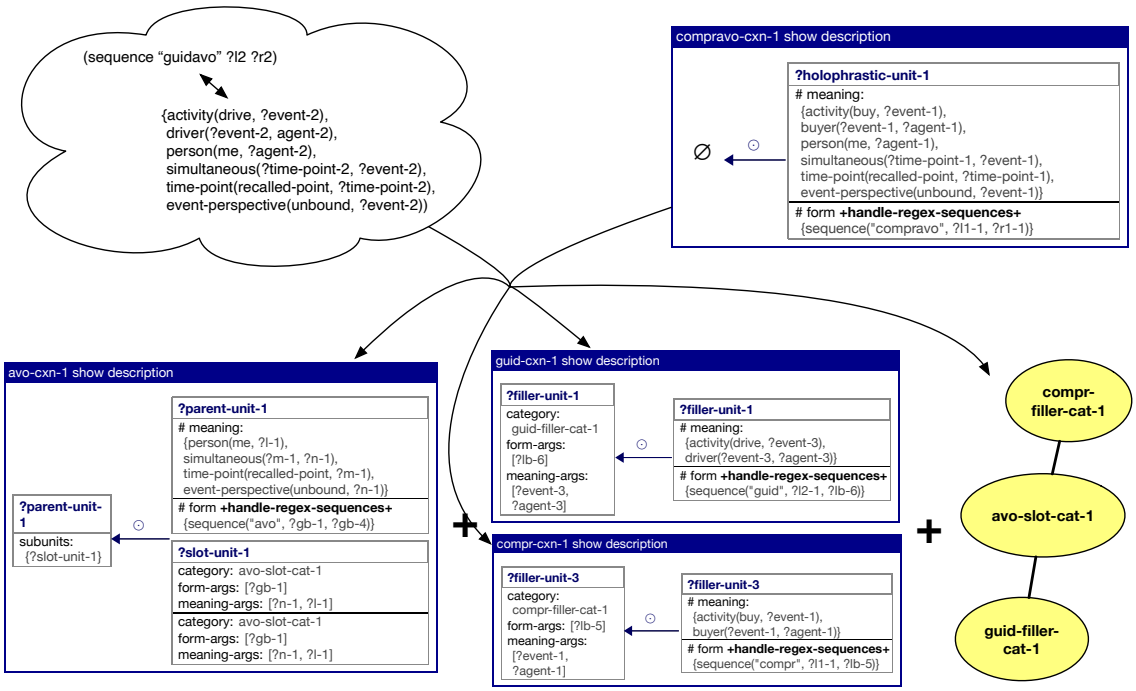



Figure 9: Syntactico-semantic generalisation of *guidavo* and *compravo*. The AVO-CXN maps between the suffix *avo* and the meaning of some event from the first person perspective in the imperfect tense. The GUID-CXN and COMPR-CXN map the verb stems to their respective types of events, i.e. driving and buying.

a driving event, while the latter associates the stem *compr* with the agent of a buying event. Finally, slot-and-filler relations between the constructions are also learned, such that both the GUID-CXN and the COMPR-CXN can fill the slot provided by the AVO-CXN.

With this syntactico-semantic generalisation, the AVO-CXN captures both the person and the tense in the first conjugation. Subsequent generalisations with other verb forms will allow to learn even more fine-grained constructions that separate the first conjugation tense suffix (here *av*) from the person suffix (here *o*).

With the examples throughout this section, we aim to show the flexibility of the new representation of the form side of constructions based on character sequences. Concretely, by working with raw character sequences instead of using a specific level of segmentation, it becomes easier to combine constructions of varying levels of granularity (in terms of the form). Throughout the examples, we have seen how character sequences could be used to capture words (specifically, the JILL-CXN), a larger pattern with a multi-word segment (specifically, the X-TAKES-Y-FOR-GRANTED-CXN), and morphology (specifically constructions for the Italian verb stem and suffixes). By extending the learning of constructions through syntactico-semantic generalisations with the new SEQUENCE predicates and algorithms for generalising over them, it becomes possible to automatically learn distinctions on different levels of granularity.

While the examples presented in this section were chosen for didactic and illustrative purposes, we highlight that the processing of character sequences as well as the generalisation over character sequences is fully operational and integrated in the open-

source Fluid Construction Grammar formalism (see https://fcg-net.org). The presented algorithms do not rely on any other information apart from the assumption that the input utterance is captured as a sequence of characters. For example, sentence boundaries have no effect as spaces and punctuation marks are treated like any other character.

## 5 Discussion

By introducing character sequences, we remove the distinction between the segmentation and processing of linguistic forms, which leads to a range of positive implications. In the following paragraphs, we discuss (i) practical advantages, (ii) the improved flexibility for operationalising constructionist analyses and for learning construction grammars, (iii) how this closes the gap between construction grammar theory and computational models, and (iv) the facilitation of future experiments on the learning, emergence and evolution of languages.

From a practical perspective, removing the pre-segmentation process makes the constructional analysis independent from the quality and the specificity of this process. Indeed, errors in the segmentation process can no longer percolate throughout the analysis. Moreover, the analysis is no longer limited to one specific level of segmentation, which often does not suffice for operationalising constructional analyses. This was illustrated through the inability to combine for example morphological patterns (e.g. verb-stem + suffix constructions) with word-order patterns (e.g. X-take-Y-for-granted-cxn). While these constructions could combine in theory, on a practical level, this is hindered by the choice of pre-segmentation algorithm. Specifically, a segmentation based on whitespaces would not separate the verb stem from the suffix, while a morphology-based segmentation would not capture *for granted* as a single segment. A key contribution of the character-based form representation is that it eliminates the need to specify these distinctions. In general, such segmentation-free representations bear the promise of being more language independent (Petrov et al. 2023). It is therefore argued that they provide a better fit for morphologically complex and low-resource languages (Kudo & Richardson 2018; Oudah et al. 2019).

The segmentation-free representation of linguistic forms offers greater flexibility for operationalising constructionist analyses and for learning computational construction grammars. Indeed, this approach sets a grammar engineer performing a linguistic analysis free from having to determine the level of segmentation they will use at the start of the analysis. Instead, the novel representation allows to unrestrictedly combine constructions of varying degree of granularity, ranging from morphological patterns to larger word-order patterns and abstract patterns. The same holds true for the learning of computational construction grammars. Before, the most fine-grained distinctions that could be learned were determined by the pre-segmentation process. Now, discovering the segmentation becomes part of the learning process. We envision that through this approach, a system can learn constructions of any size and level of abstraction (i.e. morphemes, words, chunks, grammatical patterns, ...), as long as they map between some kind of form and meaning and they can be used successfully in communication. In other words, meaningful segmentation emerges out of the observation of speech acts and the syntactico-semantic patterns detected therein.

This increased flexibility can be particularly useful for operationalising constructionist analyses or learning in morphologically rich languages, the latter of which has been proven to be an open challenge (Schmalz & Cornillie 2022).

From a theoretical perspective, removing the pre-segmentation process closes a gap that existed between construction grammar theory and its computational operationalisation. Indeed, this allows us to more effectively operationalise the notion of "constructions all the way down" (Goldberg 2006: p. 18) as the constructions are no longer constrained by the level of segmentation. Now, constructions spanning the entire continuum can be represented and processed. Moreover, adapting the generalisation algorithm to work directly character sequences also brings the learning of construction grammars closer to its theoretical foundations in first language acquisition, given that children are able to learn language from continuous, unsegmented input (Ambridge & Lieven 2011: p. 47).

The algorithms introduced in this paper facilitate a range of new constructional analyses, experiments on construction grammar learning, and studies on the emergence and evolution of language. Specifically, computational constructional analyses can move from focusing on modelling either morphological linguistic phenomena (such as van Trijp 2011b, Gerasymova 2012 and Beuls 2012) or syntactic linguistic phenomena (such as Rădulescu & Beuls 2016, van Trijp 2011a and Van Eecke 2017) to analyses that more easily combine both. In terms of learning construction grammars, we aim to facilitate further experiments that contribute to the growing body of literature that provides empirical and computational evidence (Lieven et al. 2003; Tomasello 2003; Steels 2004; Behrens 2009; Spranger 2017; Hartmann et al. 2021; Koch et al. 2022; Nevens et al. 2022; Doumen et al. 2023, 2024) for the theory according to which children learn language in a constructivist and usage-based way (Lieven et al. 1997; Tomasello 2003, 2006; Ambridge & Lieven 2015; Behrens 2021). Crucially, it now becomes possible to investigate the interplay between segmenting the continuous input and pairing the obtained segments with some sort of meaning in order to acquire constructions. Finally, the novel algorithms could be used to facilitate experiments on the emergence and evolution of language. Such experiments have so far focused on smaller morpho-syntactical phenomena, such as agreement systems (Beuls & Steels 2013), case grammar (van Trijp 2016), definite articles (van Trijp 2013b) and inflection (Pijpops et al. 2015), or on larger syntactic structures, such as phrase structure (Steels & Garcia Casademont 2015) and word order (Van Eecke 2018). Future work could focus on more general experiments that investigate how these different aspects of linguistic analysis can emerge and evolve together. Indeed, as stated by Dunn (2024: p. 97), "we would expect a symmetry between the emergence of constructions in syntactic structure and in morphological structure".

## 6 Conclusion

In this paper, we have introduced two novel algorithms that allow computational construction grammars to operate on unsegmented linguistic forms. The introduction of these algorithms eliminates the traditional separation between segmenting linguistic forms and processing them. Both algorithms rely on representing the form side of

constructions as raw character sequences. The first algorithm allows constructional language processing to accommodate such character sequences through the use of regular expressions (cf. Section 4.2). The second algorithm extends the learning of computational construction grammars introduced by Nevens et al. (2022), Doumen et al. (2023) and Doumen et al. (2024) with these new representation and processing mechanisms. Specifically, we have designed a two-step process in which the character sequences are first aligned using the Needleman-Wunsch algorithm, followed by the computation of a generalisation using a pattern matching process (cf. Section 4.3).

The algorithms introduced in this paper have been fully implemented and integrated in the open-source implementation of Fluid Construction Grammar as well as in the FCG Editor (see https://fcg-net.org). We have illustrated them and discussed their advantages through a range of examples. The incorporation of these novel algorithms has allowed us to remove the separation between segmentation and processing. Apart from practical advantages, such as being more language independent, this offers greater flexibility to construction grammarians for operationalising constructionist analyses and for learning computational construction grammars. Moreover, this brings the computational operationalisation of construction grammars closer to its theoretical foundations and paves the way towards more advanced experiments on the learning, emergence and evolution of construction grammars.

### Acknowledgements

The authors would like to thank Katrien Beuls, Jérôme Botoko Ekila, Piet Desmet, Liesbet De Vos, Paul Van Eecke and Remi van Trijp for their insightful comments and feedback to the paper. We would like to express our gratitude to Katrien Beuls and Paul Van Eecke for their contributions to the implementation of the algorithms presented in this work. The research presented in this paper was funded by the Research Foundation Flanders (FWO) through a doctoral fellowship awarded to Veronica Juliana Schmalz (1108725N) and the Horizon Europe VALAWAI project through a grant agreement (101070930) attributed to Lara Verheyen, who carried out part of this work while at Sony Computer Science Laboratories Paris.

## References

Abbot-Smith, Kirsten & Michael Tomasello. 2006. Exemplar-learning and schematization in a usage-based account of syntactic acquisition. *The Linguistic Review* 23(3). 275–290. https://doi.org/10.1515/TLR.2006.011.

Ambridge, Ben & Elena Lieven. 2015. A constructivist account of child language acquisition. In Brian MacWhinney & William O'Grady (eds.), *The handbook of language emergence*, 478–510. Hoboken, NJ, USA: John Wiley and Sons.

Ambridge, Ben & Elena V. M. Lieven. 2011. *Child language acquisition: Contrasting theoretical approaches*. Cambridge University Press.

Aslin, Richard N, Julide Z Woodward, Nicholas P LaMendola, Thomas G Bever et al. 1996. Models of word segmentation in fluent maternal speech to infants. *Signal to syntax: Bootstrapping from speech to grammar in early acquisition* 117–134.

Audring, Jenny, Francesca Masini & Geert Booij. 2013. Construction morphology: A welcome. In *Bologna seminar on construction morphology*, .

Behrens, Heike. 2009. Usage-based and emergentist approaches to language acquisition. *Linguistics* 47(2). 383–411. doi:10.1515/LING.2009.014.

Behrens, Heike. 2021. Constructivist approaches to first language acquisition. *Journal of Child Language* 48(5). 959–983. doi:10.1017/S0305000921000556.

Bergen, Ben. 2003. Towards morphology and agreement in embodied construction grammar www2.hawaii.edu/bergen/papers/ECGmorph.pdf.

Bergen, Benjamin, Nancy Chang & Shweta Narayan. 2004. Simulated action in an embodied construction grammar. In *Proceedings of the annual meeting of the cognitive science society*, vol. 26 26, .

Bergen, Benjamin K. & Nancy Chang. 2005. Embodied Construction Grammar in simulation-based language understanding. In Mirjam Fried & Jan-Ola Östman (eds.), *Construction Grammars: Cognitive grounding and theoretical extensions*, 147–190. Amsterdam, Netherlands: John Benjamins.

Beuls, Katrien. 2012. Handling scope in Fluid Construction Grammar: A case study for Spanish modals. In Luc Steels (ed.), *Computational issues in Fluid Construction Grammar*, 123–142. Berlin, Germany: Springer. doi:10.1007/978-3-642-34120-5_6.

Beuls, Katrien. 2013. *Towards an agent-based tutoring system for spanish verb conjugation*. Brussels: VUB Press: Vrije Universiteit Brussel dissertation.

Beuls, Katrien & Luc Steels. 2013. Agent-based models of strategies for the emergence and evolution of grammatical agreement. *PLOS ONE* 8(3). e58960.

Beuls, Katrien & Paul Van Eecke. 2023. Fluid Construction Grammar: State of the art and future outlook. In Claire Bonial & Harish Tayyar Madabushi (eds.), *Proceedings of the first international workshop on construction grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023)*, 41–50. Association for Computational Linguistics.

Beuls, Katrien & Paul Van Eecke. 2024. Humans learn language from situated communicative interactions. what about machines? *Computational Linguistics* 1–35.

Boas, Hans C. & Ivan A. Sag. 2012. *Sign-based construction grammar*. Stanford, CA, USA: CSLI Publications/Center for the Study of Language and Information.

Booij, Geert. 2010. Construction morphology. *Language and linguistics compass* 4(7). 543–555.

Booij, Geert & Jenny Audring. 2017. Construction morphology and the parallel architecture of grammar. *Cognitive science* 41. 277–302.

Bybee, Joan. 2006. From usage to grammar: The mind's response to repetition. *Language* 82(4). 711–733.

Bybee, Joan L. & Clay Beckner. 2009. Usage-Based Theory. In *The Oxford Handbook of Linguistic Analysis*, Oxford University Press. doi:10.1093/oxfordhb/9780199544004.013.0032.

Chang, Nancy. 2008. *Constructing grammar: A computational model of the emergence of early constructions*. Berkeley, CA, USA: University of California, Berkeley dissertation.

Clark, Jonathan H., Dan Garrette, Iulia Turc & John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics* 10. 73–91. doi:10.1162/tacl_

a_00448.
Croft, William. 1991. *Syntactic categories and grammatical relations: The cognitive organization of information*. Chicago, IL, USA: University of Chicago Press.
Doumen, Jonas, Katrien Beuls & Paul Van Eecke. 2023. Modelling language acquisition through syntactico-semantic pattern finding. In Andreas Vlachos & Isabelle Augenstein (eds.), *Findings of the association for computational linguistics: EACL 2023*, 1317–1327. Association for Computational Linguistics.
Doumen, Jonas, Katrien Beuls & Paul Van Eecke. 2024. Modelling constructivist language acquisition through syntactico-semantic pattern finding. *Royal Society Open Science* 11(7). 231998. doi:10.1098/rsos.231998.
Doumen, Jonas, Veronica J. Schmalz, Katrien Beuls & Paul Van Eecke. 2025. The computational learning of construction grammars: State of the art and prospective roadmap. *Constructions and Frames* 17.
Dunn, Jonathan. 2017. Computational learning of construction grammars. *Language and Cognition* 9(2). 254–292.
Dunn, Jonathan. 2024. *Computational construction grammar: A usage-based approach* Elements in Cognitive Linguistics. Cambridge University Press.
Fillmore, Charles. 1968. The case for case. In Emmon W. Bach & Robert T. Harms (eds.), *Universals in linguistic theory*, 1–88. New York, NY, USA: Holt, Rinehart & Winston.
Fillmore, Charles, Paul Kay & Mary O'connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of let alone. *Language* 64(3). 501–538.
Fillmore, Charles J. 1988. The mechanisms of "construction grammar". In *Annual meeting of the berkeley linguistics society*, vol. 14, 35–55.
Gaido, Marco, Beatrice Savoldi, Luisa Bentivogli, Matteo Negri & Marco Turchi. 2021. How to split: the effect of word segmentation on gender bias in speech translation. In *Findings of the association for computational linguistics: Acl-ijcnlp 2021*, 3576–3589.
Gaspers, Judith & Philipp Cimiano. 2014. A computational model for the item-based induction of construction networks. *Cognitive Science* 38 3. 439–88.
Gaspers, Judith, Philipp Cimiano, Sascha S. Griffiths & Britta Wrede. 2011. An unsupervised algorithm for the induction of constructions. In *Proceedings of the 2011 IEEE international conference on development and learning (icdl)*, vol. 2, 1–6. IEEE.
Gaspers, Judith, Philipp Cimiano, Katharina Rohlfing & Britta Wrede. 2017. Constructing a language from scratch: Combining bottom–up and top–down learning processes in a computational model of language acquisition. *IEEE Transactions on Cognitive and Developmental Systems* 9(2). 183–196.
Gaspers, Judith, Anouschka Foltz & Philipp Cimiano. 2014. Towards the emergence of verb-general constructions and early representations for verb entries: Insights from a computational model. In *Proceedings of the annual meeting of the cognitive science society*, vol. 36 36, .
Gerasymova, Kateryna. 2012. Expressing grammatical meaning with morphology: A case study for russian aspect. In Luc Steels (ed.), *Computational issues in Fluid Construction Grammar*, vol. 7249 Lecture Notes in Computer Science, 91–122. Berlin, Germany: Springer. doi:10.1007/978-3-642-34120-5_5.
Goldberg, Adele E. 1995. *Constructions: A construction grammar approach to argument*

*structure*. Chicago, IL, USA: University of Chicago Press.

Goldberg, Adele E. 2003. Constructions: A new theoretical approach to language. *Trends in Cognitive Sciences* 7(5). 219–224.

Goldberg, Adele E. 2006. *Constructions at work: The nature of generalization in language*. Oxford, United Kingdom: Oxford University Press.

Hartmann, Stefan, Nikolas Koch & Antje Endesfelder Quick. 2021. The traceback method in child language acquisition research: Identifying patterns in early speech. *Language and Cognition* 13(2). 227–253. doi:10.1017/langcog.2021.1.

Johnson, Justin, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick & Ross Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, 2901–2910. IEEE Computer Society.

Jones, Bevan, Mark Johnson & Michael C Frank. 2010. Learning words and their meanings from unsegmented child-directed speech. In *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*, 501–509.

Koch, Nikolas, Stefan Hartmann & Antje Endesfelder Quick. 2022. The traceback method and the early constructicon: theoretical and methodological considerations. *Corpus Linguistics and Linguistic Theory* 18(3). 477–504. doi:10.1515/cllt-2020-0045.

Kudo, Taku & John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco & Wei Lu Lu (eds.), *Proceedings of the 2018 conference on empirical methods in natural language processing: System demonstrations*, 66–71. Association for Computational Linguistics. doi:10.18653/v1/D18-2012.

Langacker, Ronald W. 1987. *Foundations of cognitive grammar: Theoretical prerequisites*, vol. 1. Stanford, CA, USA: Stanford University Press.

Lieven, Elena, Heike Behrens, Jennifer Speares & Michael Tomasello. 2003. Early syntactic creativity: A usage-based approach. *Journal of child language* 30(2). 333–370.

Lieven, Elena VM, Julian M Pine & Gillian Baldwin. 1997. Lexically-based learning and early grammatical development. *Journal of child language* 24(1). 187–219.

MacWhinney, Brian. 1996. *The childes project: Tools for analyzing talk*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates.

Michaelis, Laura A. 2013. Sign-based construction grammar, 133–152. Oxford, United Kingdom: Oxford University Press.

Needleman, Saul B. & Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48(3). 443–453.

Nevens, Jens, Jonas Doumen, Paul Van Eecke & Katrien Beuls. 2022. Language acquisition through intention reading and pattern finding. In Nicoletta Calzolari & Chu-Ren Huang (eds.), *Proceedings of the 29th international conference on computational linguistics*, 15–25. International Committee on Computational Linguistics.

Oudah, Mai, Amjad Almahairi & Nizar Habash. 2019. The impact of preprocessing on arabic-english statistical and neural machine translation. *arXiv preprint*

*arXiv:1906.11751* .

Petrov, Aleksandar, Emanuele La Malfa, Philip Torr & Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt & S. Levine (eds.), *Advances in neural information processing systems*, vol. 36, 36963–36990. Curran Associates, Inc.

Pijpops, Dirk, Katrien Beuls & Freek Van de Velde. 2015. The rise of the verbal weak inflection in germanic an agent-based model. *Computational linguistics in the Netherlands Journal 5*. 81–102.

Rădulescu, Roxana & Katrien Beuls. 2016. Modelling pronominal gender agreement in dutch: From a syntactic to a semantic strategy. *Belgian Journal of Linguistics* 30(1). 219–250. doi:10.1075/bjl.30.10rad.

Saffran, Jenny R, Richard N Aslin & Elissa L Newport. 1996. Statistical learning by 8-month-old infants. *Science* 274(5294). 1926–1928.

Schmalz, Veronica Juliana & Frederik Cornillie. 2022. Towards truly intelligent and personalized icall systems using fluid construction grammar. In Jozef Colpaert, Yijen Wang & Glenn Stockwell (eds.), *Proceedings of the xxist international call research conference*, 169–179. Melbourne, Australia: Castledown Publishers. doi:10.29140/9781914291050.

Schneider, Nathan. 2010. Computational cognitive morphosemantics: Modeling morphological compositionality in hebrew verbs with embodied construction grammar. In *Annual meeting of the berkeley linguistics society*, 353–367.

Sennrich, Rico, Barry Haddow & Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Katrin Erk & Noah A. Smith (eds.), *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 1715–1725. Association for Computational Linguistics. doi:10.18653/v1/P16-1162.

Spranger, Michael. 2017. Usage-based grounded construction learning: A computational model. In *The 2017 AAAI Spring symposium series*, 245–250. AAAI Press.

Steels, Luc. 2004. Constructivist development of grounded construction grammar. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, 9–16.

Steels, Luc (ed.). 2011a. *Design patterns in Fluid Construction Grammar*. Amsterdam, Netherlands: John Benjamins. doi:10.1075/cal.11.

Steels, Luc. 2011b. Introducing Fluid Construction Grammar. In Luc Steels (ed.), *Design patterns in Fluid Construction Grammar*, 3–30. Amsterdam, Netherlands: John Benjamins. doi:10.1075/cal.11.03ste.

Steels, Luc. 2017. Basics of Fluid Construction Grammar. *Constructions and Frames* 9(2). 178–225. doi:10.1075/cf.00002.ste.

Steels, Luc & Joachim De Beule. 2006. Unify and merge in Fluid Construction Grammar. In *International workshop on emergence and evolution of linguistic communication (eelc 2006)*, 197–223.

Steels, Luc & Emília Garcia Casademont. 2015. Ambiguity and the origins of syntax. *The Linguistic Review* 32(1). 37–60.

Tomasello, Michael. 2003. *Constructing a language: A usage-based theory of language acquisition*. Harvard, MA, USA: Harvard University Press.

Tomasello, Michael. 2006. Acquiring linguistic constructions. In William Damon,

Richard M. Lerner, Deanna Kuhn & Robert S. Siegler (eds.), *Handbook of child psychology, volume 2, cognition, perception, and language*, 255–298. Hoboken, NJ, USA: John Wiley & Sons.

Ungerer, Tobias & Stefan Hartmann. 2023. *Constructionist approaches: Past, present, future* Elements in Construction Grammar. Cambridge, United Kingdom: Cambridge University Press. doi:10.1017/9781009308717.

Van Eecke, Paul. 2017. Robust processing of the dutch verb phrase. *Constructions and Frames* 9(2). 226–250. doi:10.1075/cf.00003.van.

Van Eecke, Paul. 2018. *Generalisation and specialisation operators for computational construction grammar and their application in evolutionary linguistics research*. Brussels: VUB Press: Vrije Universiteit Brussel dissertation.

van Trijp, Remi. 2011a. A design pattern for argument structure constructions. In Luc Steels (ed.), *Design patterns in Fluid Construction Grammar*, 115–145. Amsterdam, Netherlands: John Benjamins. doi:10.1075/cal.11.07tri.

van Trijp, Remi. 2011b. Feature matrices and agreement: A case study for German case. In Luc Steels (ed.), *Design patterns in Fluid Construction Grammar*, 205–235. Amsterdam, Netherlands: John Benjamins. doi:10.1075/cal.11.12tri.

van Trijp, Remi. 2013a. A comparison between Fluid Construction Grammar and Sing-Based Construction Grammar. *Constructions and Frames* 5. 88–116.

van Trijp, Remi. 2013b. Linguistic assessment criteria for explaining language change: A case study on syncretism in German definite articles. *Language Dynamics and Change* 3(1). 105–132.

van Trijp, Remi. 2016. *The evolution of case grammar*. Berlin, Germany: Language Science Press.

van Trijp, Remi, Katrien Beuls & Paul Van Eecke. 2022. The FCG Editor: An innovative environment for engineering computational construction grammars. *PLOS ONE* 17(6). e0269708. doi:10.1371/journal.pone.0269708.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey et al. 2016. Bridging the gap between human and machine translation. *Google's neural machine translation system* .