Handling abstract constructions in a dictionary-based constructicon

Bálint Sass[1], Éva Dömötör[1], Balázs Indig[2], Mátyás Lagos Cortes[3], Veronika Lipp[1], Márton Makrai[4], Gergely Pethő[5]

[1]ELTE Research Centre for Linguistics, Institute for Lexicology
[2] ELTE University, Faculty of Informatics
[3] ELTE Research Centre for Linguistics, Institute for General and Hungarian Linguistics
[4] HUN-REN Research Centre for Natural Sciences, Institute of Cognitive Neuroscience and Psychology
[5] University of Debrecen
E-mail: sass.balint@nytud.hu, domotor.eva@nytud.hu, indig.balazs@inf.elte.hu, lagos.matyas@nytud.hu, lipp.veronika@nytud.hu, makrai.hlt@gmail.com, pagstudium@gmail.com

Abstract

Taking seriously the common construction grammar statement that "it's constructions all the way down" (Goldberg, 2006: 18), the Hungarian Constructicon aims to encompass the widest possible range of constructions. As it is a dictionary-based constructicon, it naturally contains what a dictionary can provide — from morphemes to words, and to partially schematic multiword constructions containing open slots. What had been missing were the more schematic abstract constructions. In this paper, we have added some important constructions of this kind to the database of the constructicon as an experiment, and have enhanced the integrated analyzer tool to handle them appropriately. Now, the system has the machinery to recognize all types of constructions in text and display them to the user. Thanks to the integration of abstract constructions, it does not present constructions in isolation; it reveals the intertwined nature of them, their connections and interactions instead. This results in a fundamentally extended functionality compared to a dictionary. A case study in Section 5 demonstrates the capabilities of the system. The list of the integrated abstract constructions is far from complete, expanding it remains future work.

Keywords: construction; constructicon; abstract construction; constructional schema; lexicon-grammar continuum

1. Introduction

Imagine a lexical resource which covers the language as a whole on the one hand, and has an analyzer tool built upon its content, its entries on the other, which processes text and uncovers linguistic phenomena from it.

This paper describes our efforts to get one important step closer to the above goal. We supplement a dictionary-like database with a small set of abstract constructions and develop procedures to handle these, together with the original more concrete, fixed constructions, in a uniform manner at both the database and the analyzer tool levels.

Our theoretical background is the construction grammar framework (Goldberg, 2006). According to the classical definition, constructions are pairings of form and meaning (or function) that cannot be inferred but must be learned. They cover all levels of language from morphemes even to abstract schemas, as Hilpert (2014) puts it "a person's knowledge of language consists of nothing but constructions". This principle has particular importance for us as our goal is to handle preferably all linguistic phenomena uniformly.

Since, according to the above, all linguistic phenomena are constructions, the database mentioned above can be called a *constructicon*. This is the inventory of all constructions, i.e. everything that is part of a language. Dictionaries contain words, constructicons contain constructions. As words are constructions themselves, they are in the constructicon by default. Therefore, a constructicon can be considered an enhanced dictionary or a superset of a dictionary. Nowadays, an evolving field of lexicography called constructicography studies the topics of constructicon building and their properties (Lyngfelt et al., 2018a). Several constructicons are currently being built for different languages (cf. Section 3).

The language which we deal with is Hungarian, and we largely build on the work described by Sass (2024). The approach is dictionary-based, meaning that in order to collect constructions, it starts from the set of linguistic elements found in a particular dictionary. Beyond words, it obtains morphemes, multiword units, and partially schematic constructions with fixed and variable elements both as well this way. What has been missing, are the more schematic abstract constructions — especially those that consist of no fixed elements at all. Our aim here is to supplement the data from the dictionary with some such abstract constructions in order to achieve a more complete coverage of all types of constructions.

The resulting lexical resource is traditional in one sense: it has entries with descriptions and examples and does not use recent technologies such as LLMs for any subtasks. At the same time, however, it is novel and unconventional in two respects — in terms of its content and its operation. Concerning content, by taking the principles of construction grammar strictly seriously and targeting all types of constructions, it could serve as a practical test of the construction grammar approach in its ability to describe language. We can view abstract constructions as grammatical rules. From this perspective, our work aligns with a classic principle of construction grammar — namely, that the lexicon and the grammar form a continuum (Boas, 2010). It should be noted that our approach is rather unique among constructicons in its intended comprehensiveness. Other constructicons tend to focus on certain specific types of constructions, most often partially schematic ones (e.g. Janda et al., 2020). Concerning operation, the main point is that the resulting lexical resource has an integrated analyzer tool. This tool does not do just some kind of dictionary lookup, but it takes an arbitrary short text excerpt as input and dynamically extracts both concrete and abstract interacting constructions from it.

2. Formal classification of constructions

In terms of their form, constructions can be considered as combinations of slots and fillers (cf. Diessel, 2023: 3.4), which is in line with the model that we use. Based on this, constructions can be classified into three groups: (1) *fully fixed constructions*, consisting of fixed elements exclusively, (2) *partially schematic constructions*, consisting of both fixed elements and open slots, and (3) *fully schematic constructions*, consisting of open slots exclusively (Lyngfelt et al., 2018b).

Fully fixed constructions are morphemes, words, and fixed multiword expressions. Sayings and even proverbs belongs to this group if they contain no open slots. This group includes, for example, *'-ban'* which is a Hungarian case marker, a bound morpheme meaning something like the preposition *in*; *'avokádó'* (avocado) or any simple words; completely non-mutable MWEs such as *'ad hoc'*; and mutable ones as well, like *'él és virul'* ('live and thrive') or *'üvegbura alatt él'* ('glass.dome under live' = live in a bubble).

Partially schematic constructions can be placed along a scale according to of how many of their slots are filled. For example, *'részt vesz -ban'* ('part.ACC take in' = take part in) has two filled and one open slot, while *'ad -nek -t'* (give IOBJ OBJ) is just the opposite, so the latter is considered more schematic. Beyond these more concrete cases, there are schema-like constructions in this group as well that do not consist solely of a content-word head and its complements, but instead embody more general patterns, in which the formal head can be an open slot itself. For example, *'ha VP, akkor VP'* (if VP, then VP) which is the general schema for if-then sentences. Or, to take an example that truly lies on the border of lexicon and grammar, demonstrating the potentials of construction grammar: *'ADJ volt N-tól, hogy VP'* ('ADJ was from N that VP' = it was ADJ of N to VP) (see Figure 1). We will call the above two constructions *'_if-then'* and *'_adj-of-to'* respectively. Following from the above, we divide the group of partially schematic constructions into two, termed *head-complement constructions* and *cheese-with-holes constructions*. The latter term refers to the fact that open slots can appear in any position within these constructions (for further details on this see Section 4).

Kedves volt Anyától, hogy elmosogatott.			
Kedves	volt	Anyá·tól	, hogy elmosogatott.
nice	was	Mom.FROM	that wash.the.dishes.PAST.S3
It was nice of Mom to wash the dishes.			

Figure 1: A realization of the *'_adj-of-to'* construction

Fully schematic constructions, or, as Diessel (2023: 3.1) calls them, constructional schemas, form the last group. An example of this group is the attributive adjective construction, denoted by *'_adj-n'* in our resource. The second example of this group is the possessive construction: *'N_1 N_2.POSS'* (N_1's N_2) denoted by *'_poss1'*. While in English the possessor is followed by *'s*, in Hungarian the possessive marker appears on the possessed noun (see Figure 2).

	Péter könyve
okos gyerek	Péter könyv·e
clever kid	Peter book.POSS
	Peter's book

Figure 2: Realizations of the *'_adj-n'* and *'_poss1'* constructions respectively

As a result of the above, we separate the set of constructions into four groups instead of the conventional three (Table 1). We define the term *abstract constructions* as including

both cheese-with-holes constructions and fully schematic constructions. They do share a formal property, namely the possible locations of open slots within the construction.

Lyngfelt et al. (2018b)	fully fixed	partially schematic		fully schematic
refined classification	fully fixed	head-complement	cheese-with-holes	fully schematic
defining *abstract* constructions	concrete		abstract	

Table 1: A refined classification of constructions and terminology used in this paper

The focus of this paper is on abstract constructions. These are the constructions that are "more interesting": unlike concrete constructions, they tend not to appear in a dictionary, at least not as a headword, so the constructicon needs to be supplemented by them. By convention, we use an '_' mark at the beginning of all abstract constructions. Abstract constructions mentioned in this paper are summarized in Section 7.

3. Comparison with other constructicons and our approach

There are several ongoing constructicon-building projects, as well as a number of well-developed, mature constructicons built over the last decade. The overall basic structure of constructicons is the following: there is a database of constructions, often supplemented by a user interface. The basic functionality of such user interfaces is browsing. That is, the user can search for certain constructions in a list or a drop-down. Clicking on a construction reveals its properties which usually include its name, a more or less formalized description of its form, description of its meaning and examples. We look at some constructicons in the following paragraphs.

3.1 Constructicons for individual languages

The Swedish Constructicon (Lyngfelt et al., 2018b) matches the description above. Additionally, it has free-text search, which in this case means that the user can search for arbitrary text in the definition, meaning and examples fields of the database.

The German project calls itself the German FrameNet-Constructicon (Ziem et al., 2019), emphasizing that they follow the original approach of Fillmore et al. (2012), namely, building a constructicon on top of the German FrameNet. The user interface is similar to the Swedish one: the user can browse constructions in the so called "Construction Index", but here all the frame elements are annotated in the examples in a very detailed way. Additional tools are provided for investigating the constructions and the frames.

The Russian Constructicon (Bast et al., 2021) has a similar user interface as well, amended by an advanced search interface that allows searching for different morphological and syntactical properties, as well as language proficiency level. The database is freely downloadable.

The Estonian Constructicon (Vainik et al., 2024) is not being built on top of a FrameNet, but it is closely connected to a large Estonian dictionary. Currently, constructional information is being integrated into the lexicographical database. As the project is still a work in

progress, the database cannot yet be searched by constructions, only by individual words, as it is common in an online dictionary interface.

As for now, the Italian Constructicon (Pannitto et al., 2024) is primarily a github repository for collecting constructions, accompanied by a guide for annotating constructions in dependency-parsed corpora. Currently, it has only the database component.

3.2 Discussion

As we mentioned in the introduction, two aspects are particularly important for us. We aim to cover all types of constructions, and we would like to make our integrated analyzer tool capable of recognizing all these construction types in arbitrary texts where they interact with each other.

Constructicons which originate from a FrameNet, like the German one, obviously contain many schematic constructions. However, they usually do not include the simplest constructions: morphemes and words. They are often interconnected with a dictionary, which provides information of these units, but this does not imply full integration, and does not imply a unified treatment of all constructions and all connections between them either. In contrast, constructicons which originate from a dictionary, like the Estonian one, inherently contain words. The challenge is how to integrate schematic constructions into the resource, and, again, working out the unified treatment. For us, the challenge is similar: beyond the fully fixed ones, partially schematic head-complement constructions were already part of the resource. We now expand it by adding abstract constructions, to cover the entire range of construction types. It is the common representation of all constructions what allows us to handle them uniformly (see Section 4).

As we saw, the main functionality of constructicons' user interfaces is browsing. We can say that they "start from the constructions". Our approach is different, we start from the text. Browsing alone is not sufficient for us, we need the analyzer mentioned above. We believe that our approach could be useful in L2 teaching and in L1 grammar classes as well, as it presents the constructions in practice, it shows how they are used in real language, what connections they have, how they fit together and how they cooperate with each other. For example, the user can modify parts of the input text and immediately see how the identified constructions change. While most constructicons, including the MoCCA project (Lorenzi et al., 2024), can be considered primarily theoretical, ours is more practically oriented and can serve as a tool to demonstrate the usefulness of construction grammar at work.

Janda et al. (2020: 163) state that "The constructicon of a language is an open-class inventory that is potentially limitless. Therefore it would be unrealistic to expect to produce a comprehensive constructicon resource." She then notes that, as there are different kinds of lexical and grammatical resources that cover parts of the lexicon-grammar continuum, all that is left for constructicographers is to focus on the parts traditionally not covered by dictionaries, i.e. partially schematic constructions, or as she puts it "entrenched multi-word expressions that contain at least one open (not fixed) slot".

To put our point yet another way: we clearly disagree with the above reasoning. Firstly, this reasoning forgets about the fundamental principle of construction grammar, which collects all form–meaning pairs into a single set, and secondly, it forgets about the importance

of revealing the connections between constructions, inside this set, regardless of their complexity. That is why we are attempting to at least partially address this "unrealistic" task.

4. Representation and algorithm

The common representation of constructions is based on the fact that filler-slot structures can be directly represented as trees: slots correspond to edges and fillers to nodes (Sass, 2024). We use slightly modified UD (de Marneffe et al., 2021) dependency trees, with one modification: the oblique dependency relations are broken down into specific relations according to Hungarian cases (`ACC` for *'-t'* (object), `INE` for *'-ban'* (in), etc.).

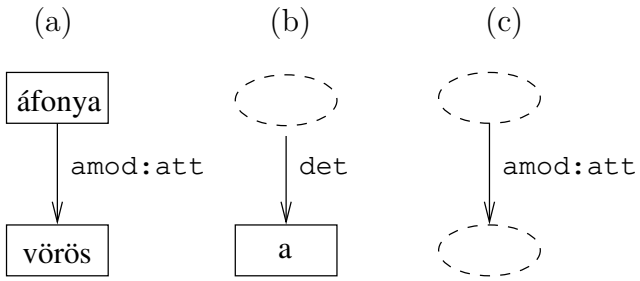

Figure 3: Representation of three constructions from the database

Similarly to headwords in a dictionary, we use the term head-construction for the constructions as they are in the constructicon database. Figure 3 shows the representation of three head-constructions, one from each class described in Section 2. The first one is fully fixed, while the other two are abstract. The second is partially schematic having an open slot at root node (depicted as a dashed ellipse), and the third is fully schematic, consisting of open slots exclusiely. Concerning their meaning: the last one is the above mentioned *'_adj-n'* construction (see Figure 2). The middle one is the definite article construction, *'_det'*. It is formalized as a partially schematic cheese-with-holes construction in the database: its open slot is for the noun, and its filled slot contains the concrete definite article form. The first one is *'vörös áfonya'* ('red berry' = lingonberry). This is a full-fledged construction with its own meaning: it is not just some berry that happens to have a certain color, it is the name of a specific type of berry. It is formally an adjective and a noun, but this inner structure is obscured by the fact that it is a construction on its own (cf. Section 6).

Abstract constructions are added to the database manually. Their entries have a structure similar to that of other constructions: form, meaning, examples, and representation. The difference is that they have a specific identifier (beginning with a '_'), and their formal representation is manually crafted and verified. Entries for newly added abstract constructions naturally integrate into the system. They are just new items in the set of constructions.

The algorithm which recognizes the head-constructions in an input text operates as followins: it analyzes the input and creates the same kind of tree representation for it as the head-constructions have in the database, searches for matching head-constructions at the root node, chooses the longest one as the recognized construction, removes it from the

representation of the input text, and then applies this method recursively to the remaining parts of the tree.

The basic algorithm was available (Sass, 2024), the important functionality what we added is that we adapted it for the treatment of abstract constructions, namely, constructions which can have open slots not just in the leaves of the tree but anywhere—i.e. at the root node or at any other intermediate node. For example, see Figure 3(b) and the left side of Figure 6(b) below. This was the crucial step that made the system work for all classes of constructions.

The length of a construction is measured this way: an open slot is worth one and a filled slot is worth two. It is important that after removing a head-construction from the input tree, the nodes which correspond to open slots are retained in their original form, in order to allow other constructions to match there. We use `udpipe` (Straka et al., 2016) inside our analysis for dependency parsing. We note that for our approach it is not problematic if the analysis that produces the representation is imperfect in any way. What matters for us that the analysis used to create formal representations in the database is the same as the one used for analyzing input. For this reason, if the head-construction in question is present in the input text, they will match.

All of the above will be illustrated in the next section.

The custom-made XML representation of abstract constructions may be of interest to the reader, so it is shown in Figure 4, however, it should be noted that this internal format may change in the future. The lemma itself is just an identifier beginning with '_'. The most important part is the tree representation of the construction in the `depana` attribute. It is just a custom-made textual representation of the tree, `//`, `@@` and `_` representing edges, edge-label-node-label separators and open slots, respectively. For now, part of speech (`<pos>`) simply indicates that this entry is an abstract construction. The definition of the construction (`<def>`) and an example (`<eg>`) are provided in the `<sensevar>` tag. This construction can be explored in action at https://bit.ly/szerkezettar-okos-gyerek.

```
<entry id="id-x00001">
  <lemma depana="@@_ //amod:att@@_">_adj-n</lemma>
  <gramgrp><pos>abstr-cxn</pos></gramgrp>
  <sensevar>
    <def>
      jelző + főnév ("jelzős főnév"):
      a <mention>főnév</mention> a <mention>jelző</mention>
      által megadott tulajdonsággal bír.
    </def>
    <eg>magyar ember, hosszú idő.</eg>
  </sensevar>
</entry>
```

Figure 4: Internal XML representation of *'_adj-n'* (see Figure 3(c)) in the database of the constructicon

5. Case study

In this section, we demonstrate the operation of the algorithm. Figure 6 shows how the algorithm for recognizing interacting constructions works on a complex example. This example features a Hungarian sentence containing several interesting constructions of different types (Figure 5).

Ha részt vesz a munkában, akkor szép fizetést kap.

Ha	rész·t	vesz	a	munká·ban	,	akkor	szép	fizetés·t	kap.
if	part.ACC	take.SG3	the	work.IN		then	nice	salary.ACC	receive.SG3

If he takes part in the work, then he receives a nice salary.

Figure 5: Input text for the main example

What follows is a step-by-step overview of how the sentence in Figure 5 is processed, using the illustrations in Figure 6. The tree representation of the example sentence is shown in Figure 6(a). While two of the relevant head-constructions are depicted on the left side of the figure, the right side of the figure shows the tree representation of the input text being dismantled as more and more head-constructions are identified and removed. Constructions being recognized are marked with a green circle, recognized open slots belonging to them are marked with green underlining. It should be emphasized that these open slots are inherent parts of these constructions. Serial numbers of constructions in the figure correspond to their serial number in the following overview.

79. The first recognized construction is an *'_if-then'*. This is an abstract construction with two open slots for the two verbs (see the left side of Figure 6(b)). It matches at the root node (*kap*) of the tree representation of the text to be processed: edges and filled slots are the same, and open slots just match any fillers. Of course, the single-node *'kap'* (receive) construction would match here as well, but as we mentioned, the algorithm always chooses the longest matching construction. This principle is crucial for the correct identification of constructions. Figure 6(c) shows the state after removing this construction from the tree. The result is smaller trees that are to be the subjects of further recognition of constructions recursively.
80. Proceeding in a depth-first manner, the next recognized construction is the single-node *'kap'* (receive). It now not only does match, but is also the longest one considering the left-hand tree in Figure 6(c).
81. The next construction to handle is an *'_adj-n'* which is a fully schematic construction consisting of two open slots (see the left side of Figure 6(d)). It is the attributive adjective construction, consisting of an adjective and a noun in this word order. Its function is to express that the noun has the property described by the adjective.
82. Then we finish the processing of the left tree by recognizing the single-node *'fizetés'* (salary) ...
83. ... and *'szép'* (nice).
84. Moving further to the right tree, a complex, partially schematic verbal head-complement construction matches at the root node: *'részt vesz -ban'* (take part in) which has two filled slots and one open. Similarly to what we mentioned in item 1, the single-node *'vesz'* (take) construction would also match here, but the algorithm chooses the longest one from the set of matching constructions.

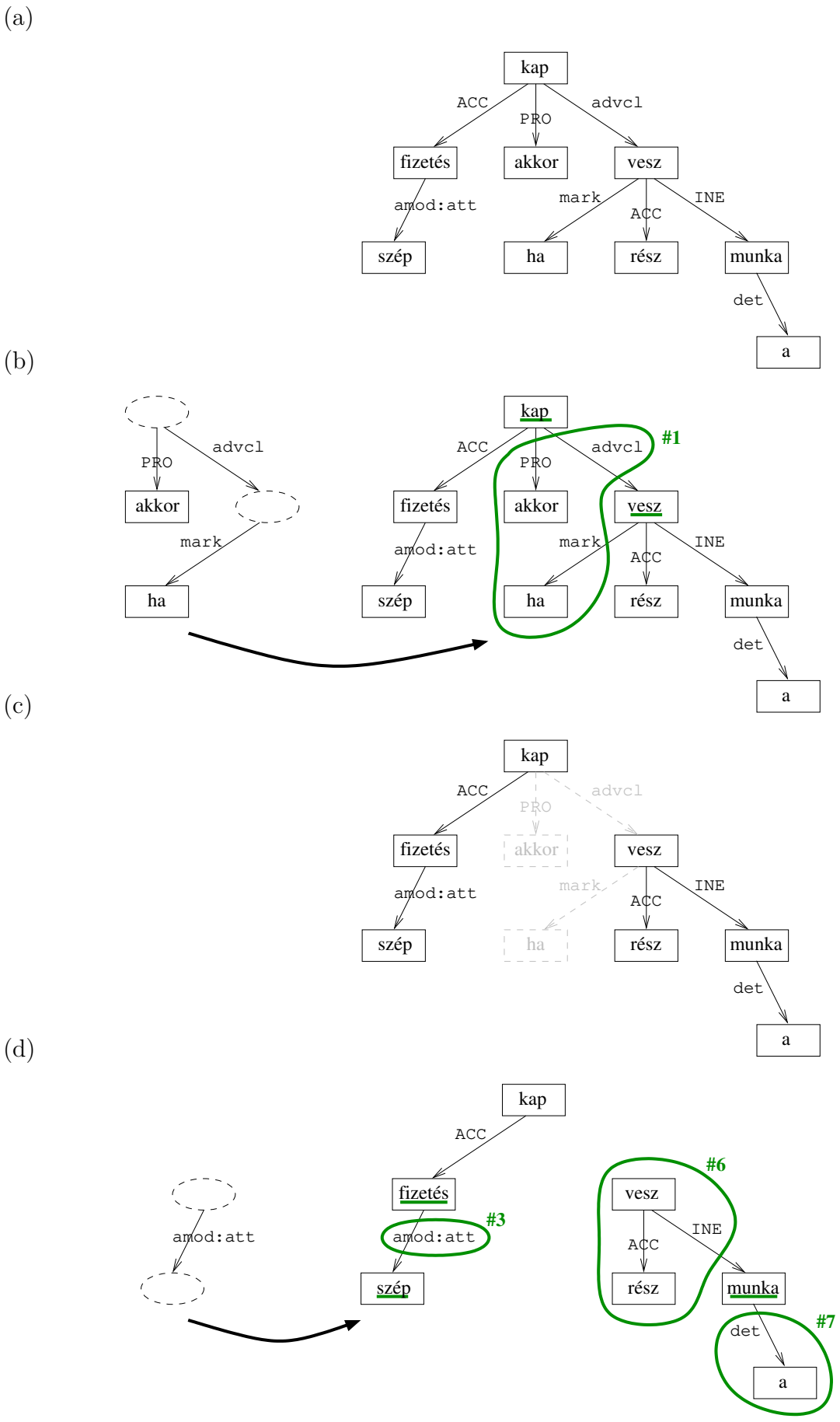

Figure 6: Demonstration of how the algorithm works. (a) The analyzed representation of the main example. (b) The first recognized construction is an *'_if-then'*. It is shown as in the database with its open slots on the left and as matched in the text representation on the right (circled and marked with #1). (c) The state after removing this construction. (d) Recognizing further constructions inside the remaining smaller trees. *'_adj-n'* is shown on the left as an example. Recognized complex (i.e. not single-node) constructions are

85. After removing all the recognized constructions above, we are left with *'munka'* (work), *'a'* (the) and the `det` (determiner) edge between the two, i.e. the rightmost part of the right tree in Figure 6(d). The *'_det'* (definite article) construction matches here (cf. Figure 3).
86. As it normally the case, although opens slots are considered during matching, they are not considered during removal and therefore they remain. Note that this occurs twice concerning the *'munka'* (work) node: it is the filler for the open slot at a leaf of construction **#6** and, at the same time, it is the filler for the open slot of construction **#7** at its root node. This is the only node that finally remains, so it is recognized as a single-node construction: *'munka'* (work).

Eight different constructions are there in this text, and, as we see, all of them are correctly recognized by the Hungarian Constructicon. Four of them are single-node constructions, they come from the original dictionary as is (together with their entries). One of them (**#6**) is a partially schematic verbal head-complement construction, also coming from the dictionary. The remaining three are abstract constructions, newly added to the database of the constructicon: **#1** and **#7** are cheese-with-holes constructions, and **#3** is a fully schematic construction.

Note that the system is able to appropriately handle the cheese-with-holes constructions. These are the constructions which have open slots not at the leaves but somewhere inside. The best example for this the *'_if-then'* (**#1**) construction depicted on the left side of Figure 6(b).

All of the above can be seen during practical operation online at https://szerkezettar.hu (username: `szerkezettar`, password: `belepes`). Direct link to the sentence what we discussed above is the following: https://bit.ly/szerkezettar-ha-reszt-vesz. The constructicon processes the input text, reveals all eight interacting constructions from it properly, and displays them to the user. The example demonstrates the fundamental guiding principle of our approach, which is in line with the principles of construction grammar: all constructions has equal status and are processed uniformly, regardless of their complexity. The depth-first manner of the processing is reflected in the order in which constructions appear on the user interface.

All examples mentioned in the paper are available on the site as clickable examples, and arbitrary short Hungarian texts can also be entered and tested.

6. Absorption

In Section 4, we mentioned that the inner structure of a construction is obscured by the fact that it is a construction on its own, with its own meaning, which must be learned. We call this phenomenon absorption.

Since the large construction has its own meaning, which can not be inferred from the meanings of its parts, it is no longer important what kinds of constructions can be discovered inside it just by their form. The meaning of these smaller forms is no longer relevant in terms of the meaning of the large construction. If it were relevant, the meaning of the large entity could be inferred, and thus the large entity would not be a construction by definition. Since these smaller units have lost their meaning, they are no longer considered

constructions—they are absorbed. They can not be investigated from a construction grammar point of view, but, of course they can still be examined from a purely syntactic point of view.

For example, *'vörös áfonya'* ('red berry' = lingonberry) (taken from Figure 3(a)) is one construction that has its own meaning. In contrast, *'okos gyerek'* (clever kid), which has exactly the same inner syntactic structure, i.e. an adjective plus a noun, is a combination of three constructions: *'_adj-n'*, *'gyerek'* (kid) and *'okos'* (clever). The meaning of each of these three constructions must be learned separately, and the overall meaning is simply derived from the sum of their individual meanings. In other words, constructions have non-compositional meaning. This statement is not surprising at all, it is just a straightforward consequence of the definition of constructions. The two constructions above can be studied at https://bit.ly/szerkezettar-voros-afonya and https://bit.ly/szerkezettar-okos-gyerek on the online user interface.

In line with the concept of absorption, the absorbed units are ignored by our system: they are not identified as constructions and they are not displayed, for example *'hogy'* (that) in *'_adj-of-to'*, or *'rész'* (part) or transitive *'vesz'* (take) in *'részt vesz -ban'* (take part in). This behavior is realized by the algorithm, primarily through its longest-match principle, which ensures that when multiple candidate constructions exist, the one spanning the most edges and nodes is selected.

7. List of new abstract constructions

Abstract constructions which has been added to the constructicon database and mentioned in this paper are listed here. Some of them are very frequent constructions, while others are more complex, more interesting ones.

87. *'_adj-n'* attributive adjective construction (fully schematic)
    Example: *'okos gyerek'* (clever kid).
    See Figure 2, Figure 3(c) and also Figure 4.
88. *'_poss1'* possessive construction (fully schematic)
    Example: *'Péter könyve'* ('Peter book.POSS' = Peter's book)
    See Figure 2.
89. *'_poss2'* prepositional possessive construction (cheese-with-holes)
    Example: *'Péternek a könyve'* ('Peter.DAT the book.POSS' = the book of Peter)
    In Hungarian, a dative/genitive case marker is used.
90. *'_det'* definite article construction (cheese-with-holes)
    Example: *'a munka'* (the work)
    See Figure 3(b).
91. *'_adj-of-to'* (a kind of evaluative construction) (cheese-with-holes)
    Example: *'Kedves volt Anyától, hogy elmosogatott.'* (It was kind of Mom to wash the dishes.)
    See Figure 1.
92. *'_if-then'* conditional construction (cheese-with-holes)
    Example: *'Ha főz, akkor eszik.'* (If he cooks, then he eats.)
    See Figure 6(b).
93. *'_this'* demonstrative construction (cheese-with-holes)
    Example: *'erre az évre'* ('this.SUB the year.SUB' = for this year)

8. Conclusion

In this paper, we supplemented the database of the Hungarian Constructicon with some abstract constructions and made the integrated analyzer tool capable of handling these kind of constructions appropriately. Being dictionary-based, the constructicon had so far lacked these types of constructions. Now, the system has the technical capability to handle all kinds of constructions: morphemes, words, fixed multiword expressions, expressions with open slots, and fully schematic constructions included. This extension is demonstrated using seven abstract constructions currently included in the database. Constructions extracted dynamically from input texts are displayed on the user interface, showing how they interact and cooperate with each other. This functionality represents a fundamental extension beyond what online dictionaries provide and foreshadows how constructicons could be some kind of enhanced dictionaries and grammar resources at the same time in the future, covering the full lexicon-grammar continuum. Future work includes adding more abstract constructions to the Hungarian Constructicon. The operation of the system can be explored online at https://szerkezettar.hu (username: `szerkezettar`, password: `belepes`), where all examples discussed in this paper can be tested.

Acknowledgements

The research described in this paper was supported by the OTKA (K 147452) grant of the National Research, Development and Innovation Office (NKFIH), Hungary. Project K 147452 is being implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the K_23 funding scheme.

This paper was also supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences (BO/00260/25/1).

For polishing the English of this paper, generative AI was used.

List of abbreviations

Universal Dependencies (de Marneffe et al., 2021) codes for dependency relations:

- `advcl` – adverbial clause modifier
- `amod:att` – attributive adjectival modifier
- `det` – determiner
- `mark` – subordinate marker

As mentioned in Section 4, we break down the `obl` (oblique) relation of UD into specific relations according to Hungarian cases, which are the following:

- `ACC` – case marker *-t* for object
- `DAT` – case marker *-nak/-nek* for *for*
- `INE` – case marker *-ban/-ben* for *in*
- `PRO` – case marker *-kor* for *at*
- `SUB` – case marker *-ra/-re* for *for*

Other abbreviations used in glosses:

- FROM – a counterpart for Hungarian case marker *-tól/-től*
- IN – a counterpart for Hungarian case marker *-ban/-ben*
- IOBJ – indirect object
- OBJ – object
- PAST – past tense suffix
- POSS – possessive marker
- SG3 – 3rd person singular

Software

Bast, R., Endresen, A., Janda, L.A., Lund, M., Lyashevskaya, O., Mordashova, D., Nesset, T., Rakhilina, E., Tyers, F.M. & Zhukova, V. (2021). The Russian Constructicon. An electronic database of the Russian grammatical constructions. URL https://constructicon.github.io/russian.

Boas, H.C. (2010). The syntax–lexicon continuum in Construction Grammar: A case study of English communication verbs. *Belgian Journal of Linguistics*, 24(1), pp. 54–82.

de Marneffe, M.C., Manning, C.D., Nivre, J. & Zeman, D. (2021). Universal Dependencies. *Computational Linguistics*, 47(2), pp. 255–308. URL https://aclanthology.org/2021.cl-2.11/.

Diessel, H. (2023). *The Constructicon*. Cambridge University Press.

Fillmore, C.J., Lee-Goldman, R. & Rhomieux, R. (2012). The FrameNet Constructicon. In H.C.B..I.A. Sag (ed.) *Sign-Based Construction Grammar*. Stanford: CSLI Publications, pp. 309–372.

Goldberg, A.E. (2006). *Constructions at work: The nature of generalization in language*. Oxford University Press.

Hilpert, M. (2014). *Construction Grammar and Its Application to English*. Edinburgh University Press.

Janda, L., Endresen, A., Zhukova, V., Mordashova, D. & Rakhilina, E. (2020). How to build a constructicon in five years: The Russian example. *Belgian Journal of Linguistics*, 34, pp. 162–175.

Lorenzi, A., Ljunglöf, P., Lyngfelt, B., Timponi Torrent, T., Croft, W., Ziem, A., Böbel, N., Bäckström, L., Uhrig, P. & Matos, E.E. (2024). MoCCA: A Model of Comparative Concepts for Aligning Constructicons. In H. Bunt, N. Ide, K. Lee, V. Petukhova, J. Pustejovsky & L. Romary (eds.) *Proceedings of the 20th Joint ACL - ISO Workshop on Interoperable Semantic Annotation @ LREC-COLING 2024*. Torino, Italia: ELRA and ICCL, pp. 93–98. URL https://aclanthology.org/2024.isa-1.12/.

Lyngfelt, B., Borin, L., Ohara, K. & Torrent, T.T. (eds.) (2018a). *Constructicography: Constructicon development across languages*. Amsterdam: John Benjamins.

Lyngfelt, B., Bäckström, L., Borin, L., Ehrlemark, A. & Rydstedt, R. (2018b). Constructicography at work: Theory meets practice in the Swedish Constructicon. In Lyngfelt et al. (2018a), pp. 41–106.

Pannitto, L., Bernasconi, B., Busso, L., Pisciotta, F., Rambelli, G. & Masini, F. (2024). Annotating Constructions with UD: the experience of the Italian Constructicon.

Sass, B. (2024). The "Dependency Tree Fragments" Model for Querying a Constructicon. In K. Š. Despot, A. Ostroški Anić & I. Brač (eds.) *Lexicography and Semantics. Proceedings of the XXI EURALEX International Congress*. Cavtat: Institut za hrvatski jezik, pp. 275–283.

Straka, M., Hajič, J. & Straková, J. (2016). UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk & S. Piperidis (eds.) *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), pp. 4290–4297. URL https://aclanthology.org/L16-1680.

Vainik, E., Paulsen, G., Sahkai, H., Kallas, J., Tavast, A. & Koppel, K. (2024). From a Dictionary to a Constructicon: Putting the Basics on the Map. In K. Š. Despot, A. Ostroški Anić & I. Brač (eds.) *Lexicography and Semantics. Proceedings of the XXI EURALEX International Congress*. Institute for the Croatian Language, pp. 209–216.

Ziem, A., Flick, J. & Sandkühler, P. (2019). The German Constructicon Project: Framework, methodology, resources. *Lexicographica*, 35(2019), pp. 15–40.

This work is licensed under the Creative Commons Attribution ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-sa/4.0/