Chunking Constructions

Kevin Stadler

Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Belgium

Abstract. Compositionality is a core property of human languages that sets them apart from other communication systems found in the animal world. But psycholinguistic evidence indicates that humans do not always decompose complex expressions in language processing. Redundant representations of compositional structure appear to be necessary to account for human linguistic capacities, a fact that should be reflected in any realistic language processing framework. This chapter presents an algorithm for dynamically combining multiple constructions into a single *chunk* in Fluid Construction Grammar. We further investigate where cases of spontaneous combinations of productive constructions occur in natural language, and discuss the relevance of redundant representations for experiments on artificial language evolution.

1 Introduction

Compositionality is one of the defining properties of human languages. Being able to use productive rules to combine words into complex expressions gives rise to the uniquely human capacity to easily communicate previously unexpressed meanings. Consequently, it is these compositional rules which have been at the center of investigations in mainstream linguistic theory for the last few decades, with idiosyncratic properties of particular expressions or lexical items pushed to the periphery of linguistic theory.

But psycholinguistic evidence indicates that humans make use of a significant amount of direct access to compositional structures, even for perfectly productive and transparent combinations which would be analysed as compositional by a linguist [9]. This intriguing finding, which is at odds with the very formal and fully generalising dogma of generative grammar [8], has sparked interest in the question of what the productive units of language really are, and why. Computational models have helped address this question, with grammar learning reframed as a problem of finding the optimal encoding balance between storage and computation (i.e. a tradeoff between machine-level resources [23]), or the ability to predict future novelty versus future reuse in order to best account for the linguistic data observed [11].

What these models still share with the generative paradigm is their reductive stance in linguistic description, also coined the "rule/list fallacy" [7] – the assumption that any kind of linguistic knowledge would have to be *either* stored in an autonomous lexical entry *or* be derived productively via rule, but not both. In a *usage-based* account on the other hand, knowledge about particular instances is never generalised away, specific experiences are stored while at the same time giving rise to more general, productive patterns [1]. Formal and computational models of language should therefore not only require the possibility of having *multiple representations* to account for the same compositional structures [10], but consequently also a theory and mechanisms which can efficiently handle and make use of these 'redundancies'.

In cognitive linguistics, this need for multiple representations has been countered by approaches such as Construction Grammar [5]. Their uniform representation of linguistic knowledge signifies not only that the boundary between pure lexical entries and compositional rules (such as grammatical constructions) is gradual rather than abrupt. The fact that all linguistic items – lexical, idiomatic and syntactic – are stored and retrieved from the same linguistic inventory, the *constructicon*, also lends itself to the idea of co-maintaining representations of the same structure at many different levels of abstraction.

What characterises these multiple representations is that they are not a static immutable representation of our language capacity but that they can be derived and updated dynamically. During interactive dialogue, for example, humans exhibit a strong tendency for *routinisation* [12]: when a compositional phrase is used with a specific meaning in discourse, it is likely to be adopted by the conversation partner and become a *routine*, a strong convention for the duration of a conversation. But such routines are more than just a temporary phenomenon of dialogue. Depending on factors such as frequency or saliency they might also persist beyond the scope of a conversation and leave a permanent trace in a human's linguistic inventory [1].

While such mechanisms have already been proposed in the theoretical literature, we want to bring these models to a next level by describing a computational algorithm of how to dynamically derive holistic routines in Fluid Construction Grammar [17]. The following section presents a formal definition of the algorithm which can be used to *chunk* constructions, with reference to the computational concepts employed by FCG. Section 3 discusses consequences of using multiple representations and handling and exploiting redundant information in the linguistic inventory. Section 4 addresses not only the repercussions of the approach for artificial language evolution experiments but also potential applications in natural language processing tasks, followed by the conclusion.

2 Chunking in Fluid Construction Grammar

For the remainder of this article we will use the simple example phrase "the pretty dog" to illustrate the workings of the algorithm. The example is highly simplified and we make no claim about the best or most realistic linguistic representation of the example phrase. For our purposes, we only require a very simple grammar with the three lexical constructions (`the-cxn`, `pretty-cxn` and `dog-cxn`) as well as two grammatical constructions to link the adjective to the noun (`adjective-noun-cxn`) as well as the article to the noun phrase (`article-np-cxn`).

While we assume familiarity with the basic concepts of Fluid Construction Grammar, we briefly recapitulate the core concepts employed by FCG before presenting the chunking algorithm. For a more detailed account of the construction application process in FCG we kindly refer the interested reader to [17] as well as the individual articles referenced throughout this section.

Figure 1 shows the `adjective-noun-cxn` from the example phrase which represents the coupling of a specific semantic structure on the left with its syntactic representation on the right. The construction expresses the association of two sibling units (an adjective and a noun) under one overarching unit. On the semantic pole, the construction establishes variable equality between the respective referents of the units' semantic predicates by using the same `?ref` variable. On the syntactic pole on the right, this operation is expressed by a `form` feature which specifies that the form content of the adjective-unit has to directly precede the form content of the noun-unit.

Of the units visible in the example construction, we will refer to the first three (above the dashed line) as the *match units* of the construction, since they have to unify with the *transient feature structure* during the matching phase of FCG construction application [18]. The transient feature structure, which captures FCG's intermediate processing structure, has to fulfill the constraints expressed within the match units in order for the construction to apply. In the case of the example construction they assert that the lexical categories (`lex-cat`) of the two units which are combined are `adjective` and `noun`, respectively. They stand in contrast to the *J-units* (below the dashed line) which are the core component of FCG responsible for building hierarchy [3]. Their contents are ignored during the initial matching phase but are merged in during the second phase of construction application, which allows the creation of new units on both poles.

2.1 Terminology

The *chunking* algorithm presented here can be used to combine a collection of co-occurring constructions into one holistic *chunked construction* which has as its main property that its application results in exactly the same changes to the FCG feature structure as if all of its constituent constructions had applied consecutively. To determine which constructions are candidates for such a chunking operation, the algorithm builds on the networks of application dependencies which can be tracked by FCG [22]. An example of such a dependency network for the example phrase "the pretty dog" can be seen in Figure 2. The network captures the dependencies between the constructions, i.e. it shows which constructions could only apply because of material that was merged in by earlier constructions. Note that the conditions are not explicitly expressed in the networks, and usually much more general than visible in the specific examples. The `adjective-noun-cxn` for example can take as its constituents any units with the lexical categories `adjective` and `noun`, respectively.

The relationships captured by the dependency networks provide the basis for the chunking algorithm, since it only makes sense to chunk together construc-

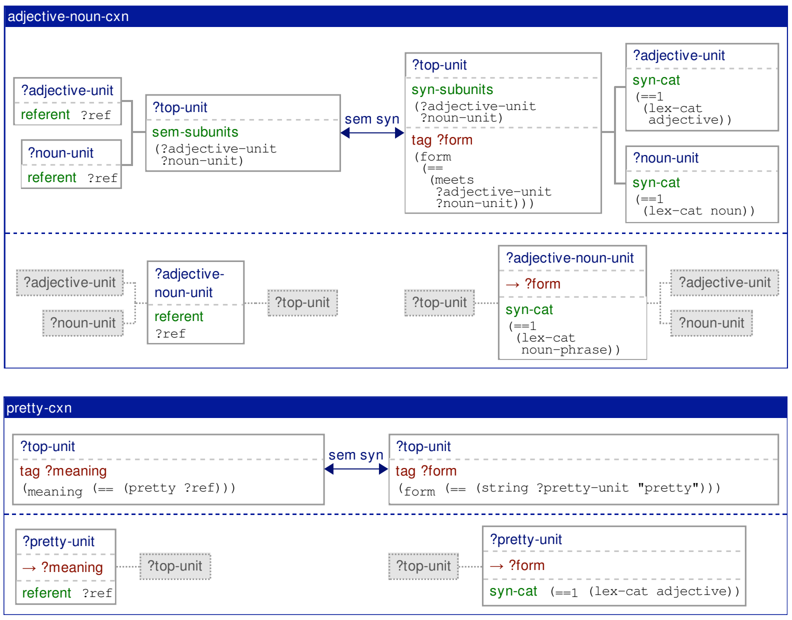

Fig. 1. Two of the original constructions required to parse or produce the example phrase. Above: the phrasal `adjective-noun-cxn` which consists of three match units (above the dotted line) and one J-unit (below) on each pole. The match units match onto the adjective and noun units as well as their parent unit, while the J-unit is used to introduce hierarchy into the transient feature structure. Below: the simple lexical `pretty-cxn` with just one match unit matching on the simple semantic predicate `pretty` and its corresponding string representation, and one J-unit to manipulate the feature structure accordingly.

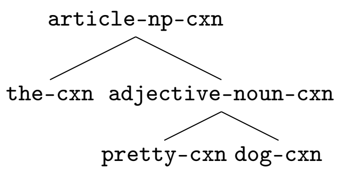

Fig. 2. The construction dependency network [22] for parsing the example phrase “the pretty dog”, using three lexical constructions as well as two grammatical ones (`adjective-noun-cxn` and `article-np-cxn`). Note that the ordering of constructions is only for clarification, the exact ordering of constituents is not dependent on the ordering of construction applications. The direction of processing is bottom-up (i.e. the `adjective-noun-cxn` could only apply because both `pretty-cxn` and `dog-cxn` applied before and supplied material for the `adjective-noun-cxn` to match on, although in no particular order). The exact matching conditions are also not shown, but since the model is general they can be arbitrarily complex (or simple).

tions which actually interact in construction application, i.e. they merge in or match on the same material of the transient feature structure. Consequently, it is only possible to chunk together connected subsets of a dependency network. Example chunks from the given network could be `the-cxn` together with `dog-cxn`, `adjective-noun-cxn` and `article-np-cxn`, which would leave an open slot for any adjective and thus represent a general `the-<adjective>-dog-cxn`. The maximal case is when the entirety of the network is chunked together, which would signify a direct representation of the entire phrase "the pretty dog" including linking operations [13]. We call the (sub-)hierarchy of a dependency network that a chunked construction is based on its underlying *construction hierarchy* or also the *chunked hierarchy*, and the constructions which contribute to a chunked construction its *constituent constructions*.

2.2 The algorithm

In order for a chunked construction to have exactly the same impact on the transient feature structure as if all of its constituent constructions had applied consecutively, this construction has to meet the following requirements:

– it combines all the match conditions expressed by its constituent constructions (it is consequently applicable in exactly the same contexts as its underlying construction hierarchy)
– it merges in all the same units and unit-content as all the constructions of the original hierarchy

Starting from an initially empty construction with no units on either pole, the chunked construction is created through the following steps which are applied to both the semantic and the syntactic poles separately:

– Given a hierarchy of dependent constructions, represented by a subtree of a construction dependency network, go through the constructions bottom-up[1] and, for every construction:
  1. match units which are matched on parts of the feature structure which were already there before the first construction of the chunked hierarchy applied are copied over to the chunked construction as they are or, if the unit is already part of the chunked hierarchy, the match constraints are merged into the already existing unit.
  2. match units which are matched on structure which was only merged in by one of the previous constructions which are part of the chunked hierarchy are added to the chunked construction as J-units. The reasoning behind this is as follows: if the chunked construction applies then all the match conditions for the leaves of the dependency network are met (since the

[1] The exact ordering doesn't matter as long as a construction's priming constructions (i.e. its children in the dependency network, which means that they provided some content for the construction to match on) are processed first. This can easily be achieved by processing the constructions in the order of their original application.

relevant match conditions were taken over as they are). Consequently, the applicability of all inner constructions is met since all conditions of its dependent constructions are met. While the match conditions of all inner constructions of the dependency hierarchy are thus not relevant, the information is still required for merging, making the transformation to J-units an ideal solution.

3. J-units are copied over as they are, or merged with the current unit content if the J-unit is already part of the chunked construction.

– Return the chunked construction

Additional issues have to be taken into account every time a unit is added to the chunked construction:

– *Merging units*: when the same unit is referred to from more than one construction of the hierarchy, the contents of the respective match units have to be merged together into one unit. This operation is not to be confused with the merge applied during construction application, where content from a match pattern is merged into the transient feature structure [2]. Rather, we are talking about merging multiple match patterns into one all-encompassing match pattern. What this means becomes clear when looking at the example of a chunked construction in Figure 3. All five constructions underlying this chunk matched on different subparts of the top unit's `meaning` and `form` features – in the chunked construction all these constraints are brought together in one unit, with all special operators (in this case only `==`) being preserved. The example case is trivial since the constraints are non-overlapping, but more complex handling is required when different special operators applying on the same feature have to be combined.
– *Tracking of transitive variable equalities*: during step-wise construction application, variable linking (such as equating the referents of two previously unrelated units) is carried out incrementally [19]. Individual variables are added one by one and get linked during the unification step of construction application. In a chunked construction however there are no intermediate unification steps, therefore all variable equalities have to be expressed explicitly. This can also be seen in Figure 4, where only one variable is introduced on the semantic pole, in contrast to three in the original construction application process which only get equated later on by the two grammatical linking constructions. To take care of this, the variable bindings established during construction application have to be inspected every time a new construction is processed during the build-up of a chunked construction. Whenever a binding between two variables is detected, the algorithm selects one of them to become the new unique representation of that variable, and greedily replaces all occurrences of the other variable. This approach makes the linking of variables explicit and guarantees that all variable equalities are expressed in only one step whenever the chunked construction is applied.

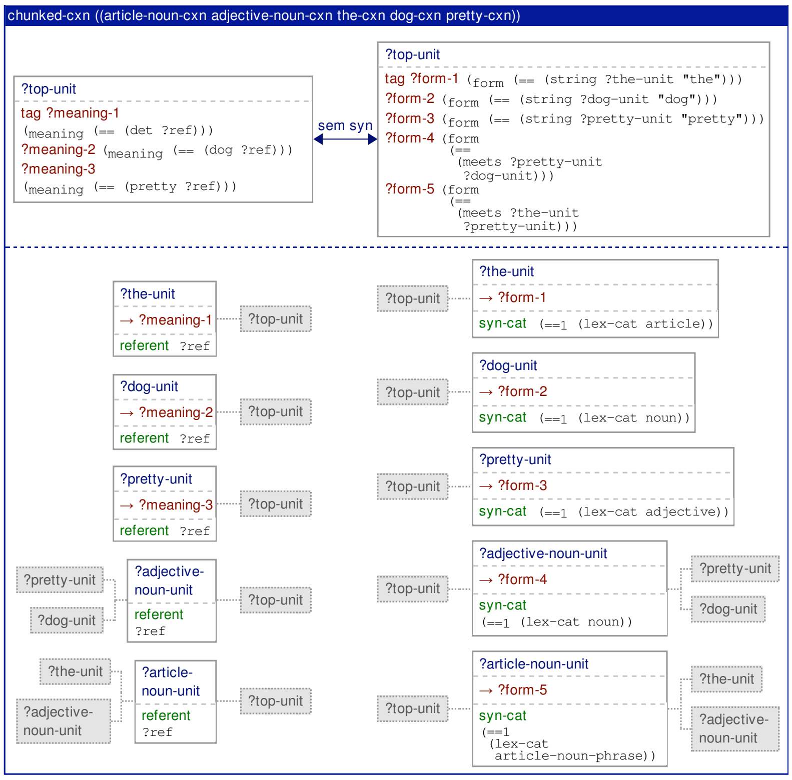
chunked-cxn ((article-noun-cxn adjective-noun-cxn the-cxn dog-cxn pretty-cxn))
?top-unit
tag ?meaning-1 (meaning (== (det ?ref)))
?meaning-2 (meaning (== (dog ?ref)))
?meaning-3 (meaning (== (pretty ?ref)))
sem syn
?top-unit
tag ?form-1 (form (== (string ?the-unit "the")))
?form-2 (form (== (string ?dog-unit "dog")))
?form-3 (form (== (string ?pretty-unit "pretty")))
?form-4 (form (== (meets ?pretty-unit ?dog-unit)))
?form-5 (form (== (meets ?the-unit ?pretty-unit)))
?the-unit
→ ?meaning-1
referent ?ref
?top-unit
?dog-unit
→ ?meaning-2
referent ?ref
?pretty-unit
→ ?meaning-3
referent ?ref
?adjective-noun-unit
referent ?ref
?article-noun-unit
referent ?ref
?the-unit
→ ?form-1
syn-cat (==1 (lex-cat article))
?dog-unit
→ ?form-2
syn-cat (==1 (lex-cat noun))
?pretty-unit
→ ?form-3
syn-cat (==1 (lex-cat adjective))
?adjective-noun-unit
→ ?form-4
syn-cat (==1 (lex-cat noun))
?article-noun-unit
→ ?form-5
syn-cat (==1 (lex-cat article-noun-phrase))
?pretty-unit
?dog-unit
?the-unit
?adjective-noun-unit

Fig. 3. FCG representation of the chunked construction derived from the full original construction hierarchy for processing "the pretty dog". The construction encompasses the match conditions and operations of all three lexical constructions as well as the two linking constructions.

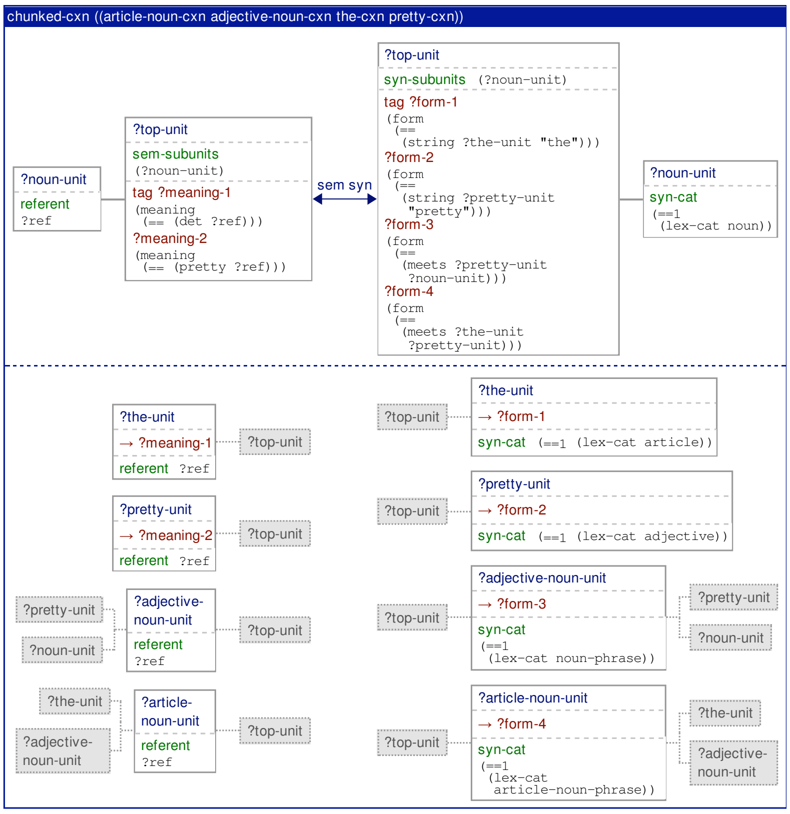
chunked-cxn ((article-noun-cxn adjective-noun-cxn the-cxn pretty-cxn))
?noun-unit
referent ?ref
?top-unit
sem-subunits (?noun-unit)
tag ?meaning-1 (meaning (== (det ?ref)))
?meaning-2 (meaning (== (pretty ?ref)))
sem syn
?top-unit
syn-subunits (?noun-unit)
tag ?form-1 (form (== (string ?the-unit "the")))
?form-2 (form (== (string ?pretty-unit "pretty")))
?form-3 (form (== (meets ?pretty-unit ?noun-unit)))
?form-4 (form (== (meets ?the-unit ?pretty-unit)))
?noun-unit
syn-cat (==1 (lex-cat noun))
?the-unit
→ ?meaning-1
referent ?ref
?top-unit
?pretty-unit
→ ?meaning-2
referent ?ref
?top-unit
?pretty-unit
?noun-unit
?adjective-noun-unit
referent ?ref
?top-unit
?the-unit
?adjective-noun-unit
?article-noun-unit
referent ?ref
?top-unit
?top-unit
?the-unit
→ ?form-1
syn-cat (==1 (lex-cat article))
?top-unit
?pretty-unit
→ ?form-2
syn-cat (==1 (lex-cat adjective))
?top-unit
?adjective-noun-unit
→ ?form-3
syn-cat (==1 (lex-cat noun-phrase))
?pretty-unit
?noun-unit
?top-unit
?article-noun-unit
→ ?form-4
syn-cat (==1 (lex-cat article-noun-phrase))
?the-unit
?adjective-noun-unit

Fig. 4. FCG representation of a chunked construction based on four out of the five constructions required for processing “the pretty dog”. This example, which could be rephrased as a *the-pretty-<noun>-cxn* shows how chunking can be used to create chunks with slots, and potentially even chunks comprising multiple grammatical constructions without any lexical material.

3 Computational and conceptual considerations

The immediate use of chunked constructions is evident: since they apply in the same contexts as their underlying construction hierarchy and affect the feature structure in exactly the same way, they can be used in place of these compositional construction applications. It is useful to look into some of the consequences and challenges of the approach.

– From a processing point of view, chunking introduces the possibility of trade-off between grammar size and complexity of search. The flexibility of full compositionality is sacrificed for a reduction in combinatorial complexity of the search space in combination with a decrease of processing cost for the application of individual constructions. The sacrifice for this is, however, potentially huge: adding all potential subhierarchies of all utterances that are encountered by a speaker in parsing or production to the constructicon would lead to the holistic storage of all compositional (sub-)structures ever encountered, as exemplified in Figure 5. This explosion in grammar size would have to be dampened, retaining only those constructions which are actually relevant and useful to the language user.

- In some ways the optimisations that chunked constructions reproduce functionality of construction dependency networks, particularly the *priming* networks already implemented in FCG [22]. But unlike in dependency networks, chunked constructions are *autonomous* from their original constituent parts. While entrenchment through frequent co-occurrence can lead to a strong preference for co-activation in dependency networks as well, the underlying representations of the constituent constructions do not change, and are individually activated and processed every time the compositional structure is encountered. In the chunking approach, the content of the constructions is copied and the new constructicon entries are not explicitly coupled to their constituent parts. The only connection betweem them is in fact indirect, through direct competition in search. This corresponds nicely to the duality observed in human language processing: while a human might use a chunked version of a construction in active parsing and production, the compositional route is still present and very much accessible to the language user when required. He can potentially be aware of the compositionality of his or her utterance since it is still represented in the linguistic inventory, but in most cases of processing this representation is not regarded due to a preference for the holistic analysis.
- Another important feature which sets the chunking approach apart from optimisations using construction networks is that it is possible to chunk together structures in which a single construction is used *more than once.* This is relevant for any compositional idiomatic expression in which the same construction occurs at least twice, and particularly when a construction can be used recursively, as is the case in natural language. The easiest example for this is a noun phrase which is embedded in another (more complex) noun phrase, such as “the cat on the mat”. In a priming network, trying to capture the entrenched and thus preferred parsing of such a phrase will result in a (potentially indirect) circle. Instead of representing a particular instance of chunking, construction networks constitute a separate layer which generalises eagerly to all (co-)occurrences of constructions. The interesting fact that chunked constructions do not introduce an additional layer of representation into the formalism will be discussed in more detail in the next section.

No matter to what extent chunked constructions are used in practice, the approach brings additional challenges for search. Creating redundant representations is only half the job, more importantly this redundancy has to be handled and exploited efficiently during actual parsing and production. The decision of

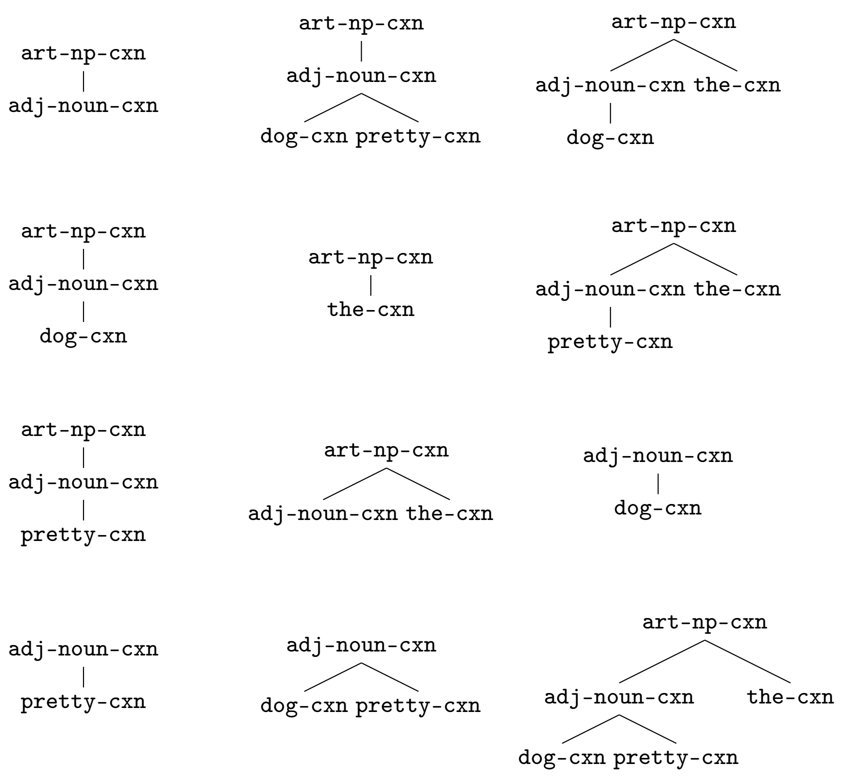

Fig. 5. The twelve different chunkable sub-hierarchies for the dependency network of the example phrase. The minimal case with just two grammatical constructions can be found at the top left, the full hierarchy which also corresponds to the complete dependency network of the parse at the bottom right. The number of possible combinations for a dependency hierarchy is a function of the tree structure of the dependencies. Note that with the exception of the last two, none of these hierarchies can be represented by their surface forms alone, since the resulting collapsed constructions contain both semantic as well as syntactic slots (such as the chunked construction in Figure 4). The hierarchy in the middle of the second line for example could be coined a general `the-<noun-phrase>-cxn`, the one above it a somewhat more specialised `<article>-pretty-dog-cxn`.

which chunked constructions to retain or even reinforce and which ones to remove from the linguistic inventory are closely coupled to their utility for the language user, which is highly correlated to their frequency of activation during search. Intelligent models of construction inventory self-organisation should not only capture the autonomy of frequently accessed chunked constructions, but also dampen the productivity of its constituent constructions when they are only infrequently activated on their own.

4 Applications

The study of the emergence and self-organisation of linguistic inventories has been at the core of experiments carried out in Fluid Construction Grammar. While the distributed development of a shared lexicon [14] and shared ontologies of perceptually grounded categories [15] have been investigated and successfully characterised early, the extension of these principles to compositional structures is not that straightforward. The additional problem arising with compositional structures is that of *multi-level selection* [21]. For competition on one level (i.e. holistic names) consolidation strategies based on lateral inhibition constitute an adequate model for selecting linguistic conventions. But once such entries are themselves re-used as parts of larger compositional structures it is not clear on which units the lateral inhibition dynamics should apply. So far only the explicit linking of constructions in a separate network layer has been proposed as a solution [20], but a model building on chunked constructions offers an alternative approach to the same problem. Instead of explicitly coupling individual co-occurring constructions together, chunked constructions added to the constructicon could *implicitly* compete with their compositional constituents during search. One of the central tenets of the multiple representation model is that there is more than just one productive unit at play in producing or understanding any compositional structure, as exemplified by dual-route models of lexical access in which direct and compositional access explicitly compete with each other [6]. Given the similarity of the phenomena in these two domains, it is likely that a cognitively plausible computational model of capturing redundancy and productivity in human language will also provide answers to open research questions in the field of self-organising communication systems.

Conversely, a better understanding of language coordination dynamics based on computational models would also lead to improvements in Natural Language Processing systems. The phenomenon of *routinisation* in humans happens automatically and leads to strong lexical and syntactic alignment processes between interlocutors [12]. Such alignment processes and their importance for the emergence of shared communication systems is well-known [16], but they are hardly exploited in natural language applications. For example, the rigid nature of most current dialogue systems keeps them far from human-like performance, but the importance and potential benefits of reciprocal learning in language coordination between humans and interactive agents is receiving more and more attention [4].

The chunking mechanism presented here provides a cognitively plausible basis for the computational modelling of routinisation effects. But interactive alignment does not just lead to more natural discourse, it can also aid in optimising and guiding computational processing through a unified theory of dialogue. Tracking of discourse referents and anaphora resolution are often treated as modular problems which are not handled by core parsing components but using extra-linguistic systems. Since routines are derived from more specific cases of use than their individual constituents, humans use them naturally to express association to specific discourse referents, a fact easily exploited by systems making use of dynamically-derived redundant representations.

5 Conclusion

In this article we argued for the relevance of redundant representations in the linguistic inventory and presented an algorithm for dynamically deriving holistic constructions from sets of dependent constructions which are used to build compositional structure. The model crucially relies on some of the properties of Construction Grammar in general and some features of Fluid Construction Grammar in particular. Construction Grammar makes use of only a single representation for all kinds of linguistic structure which is also capable of handling the additional complexity that characterises chunked constructions. Lexical and grammatical constructions can be combined just as easily, and parts of a chunked phrase can also be left unexpressed. Constructions derived by chunking can themselves become part of even larger chunks using exactly the same algorithm. The fact that FCG is unification-based enables the combination and composition of chunked constructions to be carried out relatively straightforward.

Another important feature is the bidirectional applicability of constructions in both parsing and production. Routinisation has been shown to cover both language understanding as well as generation, and a homogeneous linguistic representation for both tasks allows to support this aspect and enable positive feedback loops for alignment processes between interlocutors. Most natural language processing frameworks on the other hand are optimised for one task (most prominently parsing), and this one-sidedness is often reflected in the representations and data structures they employ. Consequently, these approaches can not easily capture and take advantage of cognitive mechanisms such as routinisation.

Although we pointed out some potential applications in language modelling and processing, the mechanism is by far not limited to these cases. Chunking is a theory-neutral operation and can also be used for other purposes. Since chunked constructions form autonomous units, their content is immediately amenable to modifications, be it capturing semantic idiosyncracies or simplifying syntactic structure. The algorithm presented is thus not only useful for questions of optimising linguistic processing, but also extensible to any area of linguistics research in which entrenchment processes play a role.

Acknowledgements

This research was carried out at the Artificial Intelligence Laboratory at Vrije Universiteit Brussel with funding from the Vrije Universiteit Brussel. I would like to thank Luc Steels, Pieter Wellens and Joachim De Beule for their help with the work and Remi van Trijp for useful comments on an earlier draft.

Bibliography

[1] Bybee, J.: From usage to grammar: The mind's response to repetition. Language 82(4) (2006)
[2] De Beule, J.: A formal deconstruction of Fluid Construction Grammar. In: Steels, L. (ed.) Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[3] De Beule, J., Steels, L.: Hierarchy in Fluid Construction Grammar. In: Furbach, U. (ed.) Proceedings of the 28th Annual German Conference on AI. Lecture Notes in Artificial Intelligence, vol. 3698, pp. 1–15. Springer Verlag (2005)
[4] Fernández, R., Larsson, S., Cooper, R., Ginzburg, J., Schlangen, D.: Reciprocal learning via dialogue interaction: Challenges and prospects. In: Proceedings of the IJCAI 2011 Workshop on Agents Learning Interactively from Human Teachers (ALIHT). Barcelona, Catalonia, Spain (2011)
[5] Goldberg, A.E.: Constructions. A Construction Grammar Approach to Argument Structure. The University of Chicago Press (1995)
[6] Hay, J.: Causes and Consequences of Word Structure. Routledge (2003)
[7] Langacker, R.W.: Foundations of Cognitive Grammar, vol. I: Theoretical Prerequisites. Stanford University Press (1987)
[8] Langacker, R.W.: A usage-based model. In: Rudzka-Ostyn, B. (ed.) Topics in Cognitive Linguistics, Current Issues in Linguistic Theory, vol. 50, pp. 127–161. John Benjamins Publishing Company (1988)
[9] McQueen, J.M., Cutler, A.: Morphology in word recognition. In: Spencer, A., Zwicky, A.M. (eds.) The Handbook of Morphology, pp. 406–427. Blackwell Handbooks in Linguistics, Blackwell, Oxford (1998)
[10] Mos, M.B.J.: Complex Lexical Items. Netherlands Graduate School of Linguistics (2010)
[11] O'Donnell, T.J., Snedeker, J., Tenenbaum, J.B., Goodman, N.D.: Productivity and reuse in language. In: Proceedings of the Thirty-Third Annual Conference of the Cognitive Science Society (2011)
[12] Pickering, M.J., Garrod, S.C.: Toward a mechanistic psychology of dialogue. Behavioral and Brain Sciences 27(2), 169–190 (2004)
[13] Steels, L., De Beule, J., Neubauer, N.: Linking in Fluid Construction Grammars. In: Proceedings of BNAIC. pp. 11–18. Transactions of the Belgian Royal Society of Arts and Sciences, Brussels (2005)
[14] Steels, L.: Emergent adaptive lexicons. In: Proceedings of the Simulation of Adaptive Behavior Conference. The MIT Press, Cambridge MA (1996)

[15] Steels, L.: The origins of ontologies and communication conventions in multi-agent systems. Journal of Agents and Multi-Agent Systems 1(2), 169–194 (1998)

[16] Steels, L.: Language as a complex adaptive system. In: Schoenauer, M. (ed.) Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI). Lectures Notes in Computer Science, vol. 1917, pp. 17–26. Springer-Verlag, Berlin (2000)

[17] Steels, L. (ed.): Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)

[18] Steels, L., De Beule, J.: Unify and Merge in Fluid Construction Grammar. In: Vogt, P., Sugita, Y., Tuci, E., Nehaniv, C. (eds.) Symbol Grounding And Beyond. Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communications, EELC 2006. Lecture Notes in Computer Science, vol. 4211, pp. 197–223. Springer (2006)

[19] Steels, L., De Beule, J., Neubauer, N.: Linking in Fluid Construction Grammar. In: Proceedings of BNAIC. Transactions of the Belgian Royal Society of Arts and Sciences. (2005)

[20] Steels, L., van Trijp, R., Wellens, P.: Multi-level selection in the emergence of language systematicity. In: Proceedings of the 9th European conference on Advances in Artificial Life. pp. 425–434. Springer-Verlag (2007)

[21] van Trijp, R.: Analogy and Multi-level Selection in the Formation of a Case Grammar. A Case Study in Fluid Construction Grammar. Ph.D. thesis, University of Antwerp, Antwerp (2008)

[22] Wellens, P.: Organizing constructions in networks. In: Steels, L. (ed.) Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)

[23] Zuidema, W.: What are the Productive Units of Natural Language Grammar? A DOP Approach to the Automatic Identification of Constructions. In: Proceedings of the Tenth Conference on Computational Natural Language Learning. pp. 29–36. Association for Computational Linguistics, New York City, USA (June 2006)