This paper is the author's draft and has now been officially published as:

Nevens, J., Van Eecke, P., & Beuls, K. (2019). Computational Construction Grammar for Visual Question Answering. Linguistics Vanguard 5(1).

Computational Construction Grammar for Visual Question Answering

Jens Nevens, Paul Van Eecke, and Katrien Beuls

Artificial Intelligence Laboratory - Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium

In order to be able to answer a natural language question, a computational system needs three main capabilities. First, the system needs to be able to analyse the question into a structured query, revealing its component parts and how these are combined. Second, it needs to have access to relevant knowledge sources, such as databases, texts or images. Third, it needs to be able to execute the query on these knowledge sources. This paper focuses on the first capability, presenting a novel approach to semantically parsing questions expressed in natural language. The method makes use of a computational construction grammar model for mapping questions onto their executable semantic representations. We demonstrate and evaluate the methodology on the CLEVR visual question answering benchmark task. Our system achieves a 100% accuracy, effectively solving the language understanding part of the benchmark task. Additionally, we demonstrate how this solution can be embedded in a full visual question answering system, in which a question is answered by executing its semantic representation on an image. The main advantages of the approach include (i) its transparent and interpretable properties, (ii) its extensibility, and (iii) the fact that the method does not rely on any annotated training data.

Keywords— Computational Construction Grammar, Fluid Construction Grammar, Natural Language Understanding, Procedural Semantics, Visual Question Answering, CLEVR, Artificial Intelligence

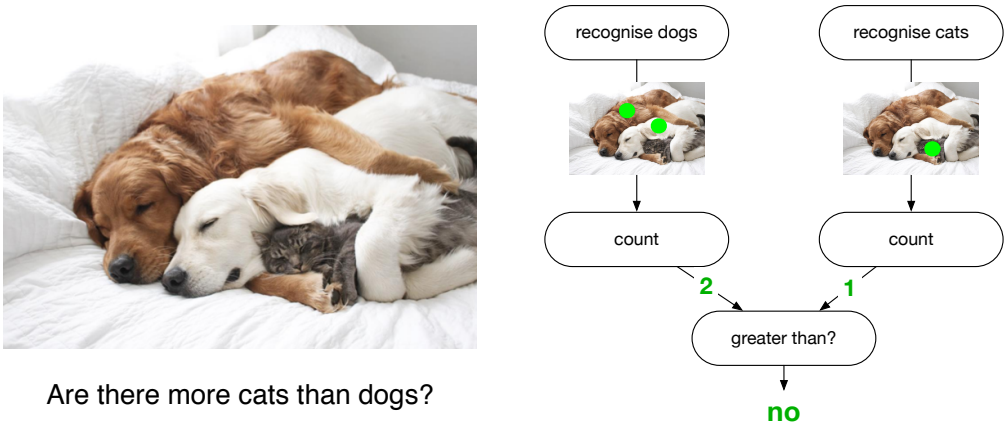


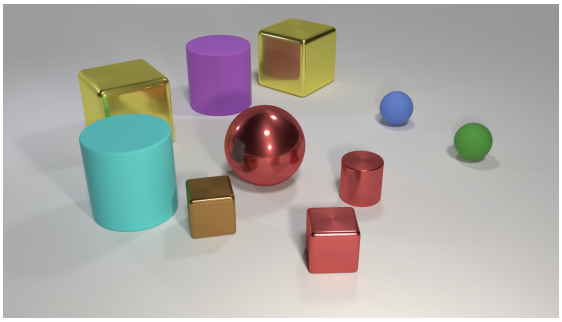
Figure 1: Schematic representation of the processes involved in visual question answering.

1 Introduction

Imagine a scene like the one on the left side of Figure 1, where three pets are sleeping on a bed. When you would ask a child whether there are more cats than dogs on the bed, how would he or she get to the answer "no"? In order to answer the question, the child needs to be able to recognise cats and dogs, to count them, and to compare these counts. Correspondingly, if you would want a computational system to answer this question, you would need to endow it with three main capabilities. First, upon receiving the question, the system should retrieve its logical structure in the form of a sequence of steps that need to be taken in order to find the answer, as shown on the right side of the figure. Second, it needs to have access to the scene, in this case in the form of a bitmap image. Third, the steps specified by the logical structure of the question need to be executed on the image. This involves identifying the dogs and cats that are present (as indicated by the green dots), counting them (2 dogs and 1 cat), and comparing these numbers ($2 > 1$).

In this paper, we focus on the first capability, namely analysing natural language questions into a series of actions that need to be taken in order to retrieve their answers. We present a novel approach, based on computational construction grammar, that semantically parses questions into their executable meaning representations. The methodology is demonstrated and evaluated using CLEVR (Johnson et al. 2017), a community-wide benchmark task that evaluates how well computational systems perform at answering logically complex questions about images of simple scenes. Our approach has three main advantages with respect to the state of the art. First, the models are fully transparent and explainable in human-understandable categories. Second, the models are extensible, as their coverage can be broadened without the need to rebuild the existing system. Finally, the methodology does not require any annotated training data, as it allows capturing expert linguistic knowledge in the model.

The paper is structured as follows. Section 2 introduces the CLEVR benchmark task, which is used for demonstrating and evaluating the methodology. Section 3 discusses prior work on



Q: Are there an equal number of large things and metal cubes?

Q: What size is the cylinder that is left of the brown metal thing that is in front of of the big sphere?

Q: There is a cylinder with the same size as the brown metal cube; is it made of the same material as the small green ball?

Q: How many objects are either small cylinders or red things?

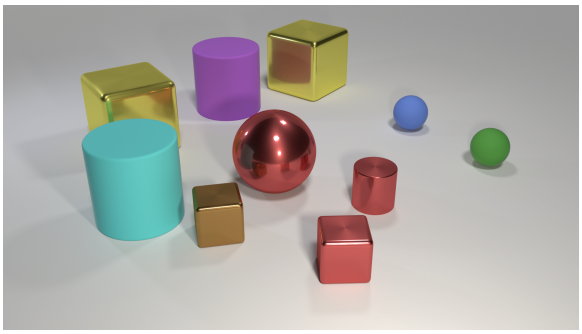
Figure 2: An example image from the CLEVR dataset, together with questions displaying the reasoning skills required by the visual question answering model: counting, comparison, attribute identification, spatial relations and logical operations. Examples from Johnson et al. (2017).

mapping natural language questions to structured queries. Our computational construction grammar solution to this challenge is presented in Section 4, and subsequently evaluated in Section 5. Section 6 and 7 go beyond the benchmark task by presenting how our approach can be embedded in a full visual question answering system, and how the model that was used for semantically parsing questions can also be used for producing questions expressing structured queries. Finally, Section 8 discusses the contributions of this paper.

2 The CLEVR benchmark

The CLEVR benchmark task (Johnson et al. 2017) was designed to facilitate the development and evaluation of computational techniques for answering natural language questions that involve logical reasoning. CLEVR is set up as a visual question answering task. This means that each question is accompanied by an image, in which the answer needs to be found. The images, of which an example is shown on the left side of Figure 2, depict scenes that consist of 3D geometrical objects (*spheres*, *cylinders* and *cubes*). These objects differ in color (*blue*, *brown*, *cyan*, *grey*, *green*, *purple*, *red* or *yellow*), size (*small* or *large*), material (*metal* or *rubber*) and position relative to each other (*behind*, *left of, in front of* or *right of*).

Answering the questions, of which a few examples are shown on the right side of Figure 2, requires logical reasoning skills, such as counting (e.g. *"how many spheres are there?"*), comparison (e.g. *"are there more spheres than squares?"*), attribute identification (e.g. *"what is the color of the cube?"*), spatial relations (e.g. *"what color is the sphere left of the cube?"*) and logical relations (e.g. *"how many things are either red spheres or blue cubes?"*). The images and questions are automatically generated with uniform probabilities in order to control for biases in the dataset. For example, the probability that a *large, blue, metal sphere* is located *left of a small, green, rubber cube* is exactly the same as the probability that a *small, red, rubber cylinder* is located *right of a small, grey, metal ball*. This ensures that answering the questions requires actual logical reasoning, and cannot be done by exploiting statistical biases of the dataset (for



What size is the red sphere?

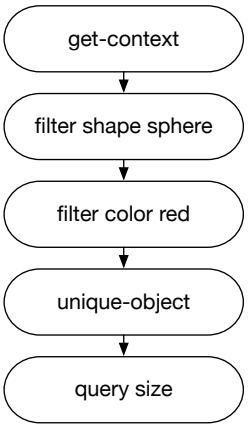


Figure 3: Example of a scene and a question from the CLEVR dataset, annotated with the semantic representation of the question in the form of a sequence of steps that need to be executed to find the answer.

example if spheres would always be red or if cubes would always be located on the right of cylinders). As the questions are automatically generated, they do not reflect actual language use. However, all questions follow patterns that are typical for English question sentences. They also feature a challenging amount of lexical and syntactic variation. For instance, a question that queries the material of a large blue ball could, among other possibilities, be expressed as *"what material is the big blue ball?"*, *"what is the large blue sphere made of?"* or *"there is a large blue ball; what is its material?"*.

The CLEVR dataset consists of more than 860,000 questions about 100,000 different scenes. Each question is annotated not only with its answer, but also with its semantic representation in the form of a sequence of steps that need to be executed on the image to find the answer. The answers to the questions can be used to evaluate the full visual question answering task, while the semantic representations allow for a separate evaluation of the language understanding subtask. An example of a scene and a question that is annotated with its semantic representation is shown in Figure 3. Here, the question *"what size is the red sphere?"* is annotated with a semantic program that consists of 5 steps: GET-CONTEXT, which reads in the image, FILTER SHAPE SPHERE, which returns all spheres in the image, FILTER COLOR RED, which filters the set of spheres that was found by red objects, and returns thus the set of red spheres, UNIQUE-OBJECT, which asserts that only a single red sphere was found, and QUERY SIZE, which qualifies this red sphere in terms of size.

3 State of the art

State-of-the-art approaches to (visual) question answering can be divided into two main groups, based on whether they perform an explicit semantic analysis of the input question or not. The first group of approaches does not perform any such analysis, and tries to model a direct mapping

between a question and its answer. This is typically done using large end-to-end neural networks, which are trained on huge amounts of annotated data (Ren et al. 2015, Malinowski et al. 2015, Zhou et al. 2015, Yang et al. 2016, Xu & Saenko 2016, Noh et al. 2016). While these systems perform well on a large variety of tasks, they have a number of major shortcomings (Agrawal et al. 2016, Zhang et al. 2016). They fall into the trap of exploiting the statistical biases that are present in their training data, thus failing to perform any explicit reasoning about the question or to even look into the image. For example, in training data, the question "*what covers the ground?*" will almost always be accompanied with an image of a snowy landscape and the answer "*snow*". The association between this question and the answer "*snow*" will be so strong, that this answer will always be given regardless of the actual input image. Furthermore, it has been shown that these kinds of visual question answering models poorly generalise to substantially novel questions and images, and that they only consider the first few words of the input question to formulate an answer (Agrawal et al. 2016).

The second group of approaches tries to overcome these shortcomings by dividing the question answering process into two parts. First, a semantic analysis of the input question is performed. The semantic representation of a question specifies a sequence of actions that need to be performed to retrieve the answer. In other terms, the semantic representation takes the form of a query that can be executed by a computational system on one or more sources of knowledge. Second, the semantic representation is executed on a computational system on which the query language is implemented, so that the answer can be found. Different ways of semantically parsing questions have been explored in the literature. A first group of models is grammar-based. These models use linguistically-informed computational grammars to retrieve the semantic representation of a question. This representation can either be a query-like structure that can be directly executed (McFetridge et al. 1996), or a more abstract meaning representation that is later mapped to an actual query (Frank et al. 2007, Zettlemoyer & Collins 2012, Li & Liao 2012). Question answering systems with a grammar backbone have used many different grammar formalisms, including Head-Driven Phrase Structure Grammar (HPSG) (McFetridge et al. 1996, Abou-Assaleh et al. 2005, Frank et al. 2007), Lexical-Functional Grammar (LFG) (Yarmohammadi et al. 2008, Shamsfard & Yarmohammadi 2010), Combinatorial Categorial Grammar (CCG) (Li & Liao 2012, Zettlemoyer & Collins 2012), and dependency parsing (Andreas et al. 2016). The main advantages of the grammar-based approaches are their precise nature, in the sense that an analysis is always linguistically motivated, as well as the fact that (usually) no annotated training data is required. A disadvantage of most systems is that only an abstract meaning representation can be built, rather than an actual query. Mapping this representation to an executable query is then still a highly non-trivial task. A second disadvantage of these approaches is that their coverage is typically limited to a closed domain.

Alternatively, semantic parsing of questions can be performed without any explicit grammar representation. Instead, the mapping between the sequence of words in a question and the sequence of actions to perform is modelled using statistical or neural approaches, typically some form of Recurrent Neural Networks (Hu et al. 2017, Johnson et al. 2017, Yi et al. 2018, Mao et al. 2019). These models implement thus a direct mapping between questions and queries, and are

well-suited for broad-coverage open-domain parsing. They rely on large amounts of training data, which is not always available. Moreover, the decisions made by these models are not transparent, as it is not clear how and why they come up with a particular analysis.

The language understanding part of the CLEVR benchmark task introduced in the previous section has been virtually solved using non-grammar based approaches (Mascharka et al. 2018, Yi et al. 2018). In this paper, we revisit the benchmark using a grammar-based approach, obtaining the same level of accuracy, while offering a transparent model that does not rely on annotated training data.

4 A computational construction grammar approach

In this paper, we present for the first time a computational construction grammar approach to semantically parsing questions into directly executable queries. The query language is used as the meaning representation of the grammar, which allows mapping from questions to queries in a single step, hence remediating one of the major shortcomings of earlier grammar-based approaches. This section first introduces computational construction grammar and its implementation called Fluid Construction Grammar (4.1) and then presents a grammar that solves the CLEVR visual question answering task (4.2).

4.1 Computational construction grammar

Computational construction grammar (CCxG) is a branch of linguistics that aims to operationalise insights and analyses from construction grammar into concrete processing models. CCxG implements the basic tenets of construction grammar, in particular the assumptions that (i) all linguistic knowledge can be captured as form-meaning parings, called constructions, (ii) there exists a lexicon-grammar continuum rather than a lexicon-grammar split, and (iii) the same grammar can be used for both comprehending and producing natural language utterances. As constructions can capture structures that range in complexity from simple words and meanings to whole phrases and semantic structures, CCxG provides an elegant and effective way to deal with the non-compositional nature of language, both on the form side (e.g. “many a year”) and the meaning side (e.g. “a piece of cake” in the figurative sense).

Fluid Construction Grammar (FCG - `https://www.fcg-net.org`) is a computational construction grammar implementation that supports the representation and bidirectional processing of construction grammars (Steels 2011, 2017). As such, FCG can be thought of as a special-purpose programming language that provides adequate building blocks for operationalising constructionist approaches to language. FCG aims to be an open environment. As a particular linguistic phenomenon can be analysed in many different ways, FCG does not impose a single analysis, but provides the tools to implement all of these, and compare them to each other. The basic building blocks of FCG are introduced in the next sections, which can be skipped by readers who are already familiar with FCG.

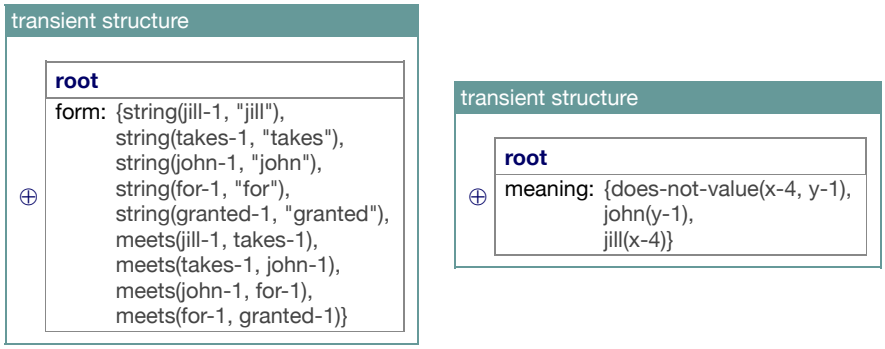


Figure 4: Initial transient structures for the utterance "Jill takes John for granted' in comprehension' (left) and production (right).

4.1.1 The transient structure

In the context of FCG, language processing amounts to mapping between natural language utterances and a representation of their meaning, by means of constructions that capture linguistic knowledge. In comprehension, this process starts from an input utterance. In a stepwise manner, the constructions of a grammar add more and more information until a full analysis is achieved. This analysis includes the meaning of the utterance, which can extracted at the end. In formulation, the process starts from a meaning representation. The same constructions add information until a full analysis is build, from which the utterance can be extracted. All information that is known at a certain point in processing is represented as a feature structure, called the *transient structure*.

Figure 4 shows the transient structure at the start of the comprehension and production processes of the utterance "Jill takes John for granted". The transient structures contain a computational representation of the input utterance in comprehension (left Figure) and of the meaning representation in production (right Figure). In comprehension, the initial transient structure contains a single unit (drawn as a box), called `root`, with a `form` feature that holds a set of form-related predicates. Here, there are two types of predicates. The `string` predicates describe the words that were found in the input and assign them a unique identifier, e.g. `jill-1`. These identifiers allow to unambiguously refer to the tokens in the utterance. The `meets` predicates encode the word order that was observed, e.g. that `for-1` immediately precedes `granted-1`. In formulation, the initial transient structure also contains a single `root` unit, which this time holds a `meaning` feature. The value of this feature is the set of meaning predicates that need to be expressed in an utterance. In this case, the meaning predicates indicate that there exists an entity 'John', that there exists an entity 'Jill', and that the 'Jill' entity does not value the 'John' entity.

4.1.2 Constructions

Constructions have the power to add more information to a transient structure, in the form of units and features. This happens in two stages. First, it is checked whether a construction is

applicable, by matching its preconditions with the transient structure. If the preconditions are satisfied, the linguistic knowledge contained in the construction is added. Consider for example the JILL-CXN shown in Figure 5. The preconditions of this construction are situated on the right side of the arrow. The units on this side, in this case only the `?jill-unit`, are split in two by a horizontal line. The preconditions in production are written above the line, while the preconditions in comprehension are written below. In this case, the construction looks in the transient structure for a `form` feature that contains a predicate `string(?jill-unit, ''Jill'')` in comprehension and for a `meaning` feature with a predicate `jill(?x)` in production. Note that the symbols preceded by a question mark are logical variables, which means that they can match any symbol, and that the features preceded by a # need to found in the `root` unit of the transient structure.

When the preconditions of the construction in the direction of processing (comprehension or production) are satisfied, the features on the left side of the arrow, as well as the preconditions in the other direction are added to the transient structure. For example, when the JILL-CXN applies in comprehension, a new unit is created, to which the features on the left side are added, as well as the meaning predicate from the preconditions in production. The novel information that has been added to the transient structure, can then satisfy the preconditions of other constructions. One construction that relies on information that was contributed by other constructions, such as the JILL-CXN, is the X-TAKE-Y-FOR-GRANTED-CXN, as discussed in Hoffmann & Trousdale (2013: p.2) (see Figure 6). This construction combines (i) a `?subject-unit` with features indicating that it is an animate noun phrase, (ii) a form of the verb "take" that agrees with the `?subject-unit`, (iii) an `?object-unit` with features indicating that it is a noun phrase, and the strings (iv) "for" and (v) "granted". The `meets` constraints in the `?clause-unit` reflect the order in which these elements need to appear. The `meaning` feature in the same unit contains the meaning predicate `does-not-value(?x,?y)`, where `?x` and `?y` are bound to the referent of the `?subject-unit` and `?object-unit` respectively. The construction can be applied both in comprehension and production, depending on the activated preconditions. Note that FCG is entirely open-ended when it comes the the features and values that are used, and that they do not need to be declared anywhere outside the constructions.

4.1.3 Constructional processing

Constructional processing boils down to the problem of finding a sequence of constructions that collaboratively build up a transient structure until it contains a full constructional analysis. This process starts from a transient structure that initially only contains an utterance (in comprehension) or a meaning representation (in production), as shown in Figure 4. Then, if their preconditions are satisfied, constructions expand the transient structure with more information. Finally, when all relevant constructions have applied, the form predicates (in production) or meaning predicates (in comprehension) are extracted from the transient structure as a solution. Figure 7 shows a network representation of the meaning predicates extracted from the transient structure after processing the utterance "Jill takes John for granted" in comprehension. Note that the predicate

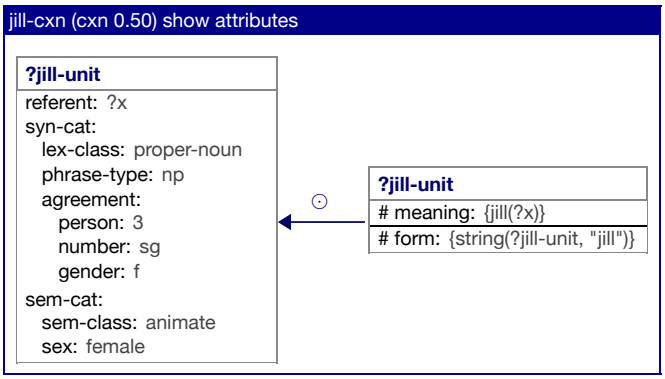


Figure 5: A construction that matches in comprehension on the string "Jill" and in production on the meaning predicate `(jill ?x)`.

`jill(?x-12)` was contributed by the JILL-CXN in Figure 5 and the `does-not-value(?x-12 ?y-4)` predicate was added by the X-TAKE-Y-FOR-GRANTED-CXN in Figure 6. It is also this construction that has linked the arguments of the two predicates.

4.2 A grammar for the CLEVR benchmark task

This section presents a CCxG grammar implemented in FCG that solves the language part of the CLEVR benchmark task, i.e. mapping between questions and their executable semantic representations. In total, the grammar consists of 170 constructions, of which 55 are morphological or lexical constructions. The remaining 115 constructions collaboratively capture the grammatical structures that are used in the dataset (e.g. noun phrases, prepositional phrases, and a wide range of interrogative structures). In this section, we first focus on the semantic representation that is used by the grammar, and then show how the constructions of the grammar collaborate to analyse questions into directly executable semantic programs.

4.2.1 Executable semantic representations

The FCG platform is completely open to the kind of semantic representation that is used (e.g. frame semantics, predicate calculus or AMR). In this case, the grammar employs a procedural semantic representation that is inspired by the annotations in the CLEVR dataset. Each semantic representation consists of a number of predicates that correspond to function calls that can be executed by the computational system on an image. For example, the predicate `(filter ?output ?input ?attribute)` calls the `filter` function with as arguments `?output`, `?input` and `?attribute`. Some of these arguments can be variables, in which case the name of the argument will be preceded by a question mark, as in `?input`. In general, the arguments are ordered in such a way that the output argument comes first, followed by the input argument and optionally by additional input arguments. An overview of the predicates that are used is given in Figure 8.

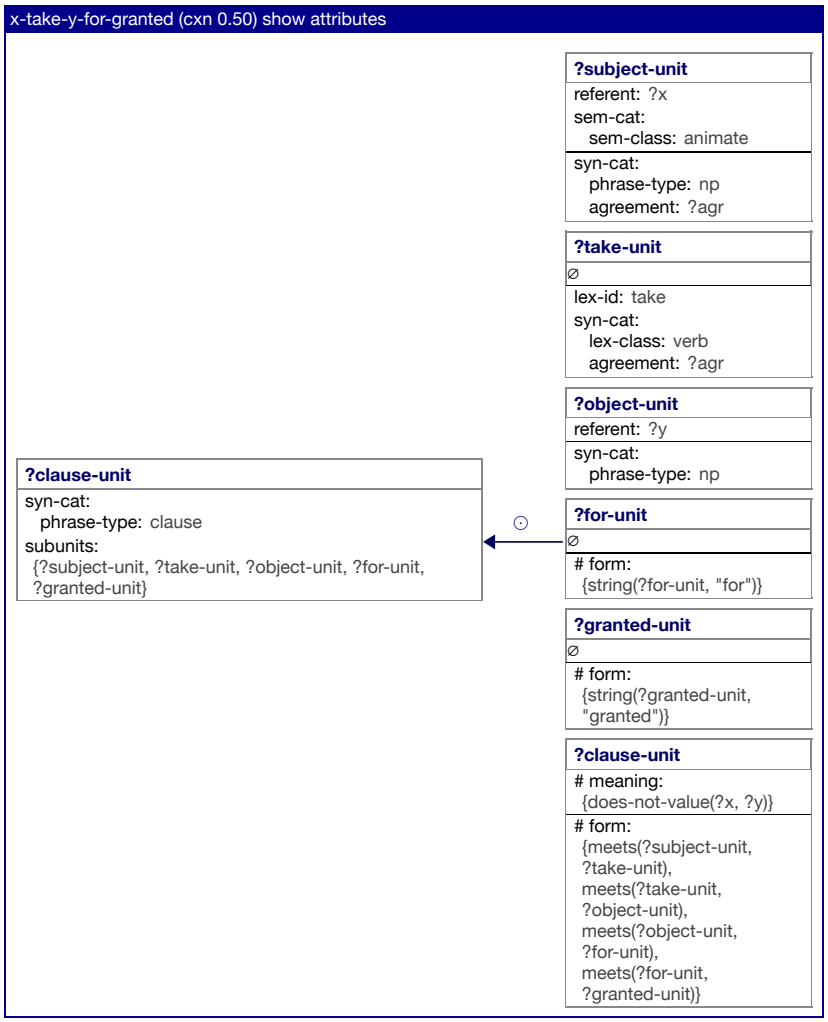


Figure 6: The X-TAKE-Y-FOR-GRANTED-CXN combines an animate subject noun phrase, a form of the verb "take" that agrees with this noun phrase, an object noun phrase, and the words "for" and "granted", and indicates that the utterance expresses the meaning that the subject does not value the object.

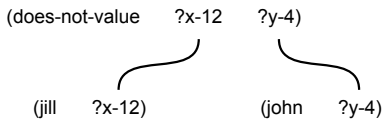


Figure 7: The meaning network resulting from the comprehension process of the utterance "Jill takes John for granted".

(get-context ?context)

(filter ?output-set ?input-set ?attribute)

(query ?output-value ?input-object ?attribute)

(same ?output-set ?input-object ?attribute)

(equal ?boolean ?input-value-1 ?input-value-2 ?attribute)

(count ?number ?input-set)

(exist ?boolean ?input-set)

(unique ?output-object ?input-set)

(relate ?output-set ?input-object ?spatial-relation)

(intersect ?output-set ?input-set-1 ?input-set-2)

(union ?output-set ?input-set-1 ?input-set-2)

(equal-integer ?boolean ?input-count-1 ?input-count-2)

(less-than ?boolean ?input-count-1 ?input-count-2)

(greater-than ?boolean ?input-count-1 ?input-count-2)

(bind type ?variable entity)

Figure 8: The predicates used by the CLEVR grammar.

Using these predicates, complex semantic networks can be built by repeatedly unifying the output argument of one predicate to the input argument of another predicate, linking them together. This is done through the process of construction application, as we will discuss in the following section. An example semantic network is shown in Figure 9. This network is the semantic representation of the question *"what material is the red cube?"*. Note that the topmost predicate, `get-context`, is a special predicate. It has a single output argument and no input arguments. This operation binds the set of all objects in the current scene to its output argument. The set of objects is filtered for cubes, and the resulting set is filtered for red objects. The next predicate checks if a single object remains (the red cube), and finally, its material is queried.

4.2.2 From CLEVR questions to executable queries

The grammar that we present allows parsing natural language questions into directly executable queries, which are build up using the predicates introduced above. This means that during processing, constructions have the crucial task of contributing meaning predicates to the transient structure, and linking the arguments of these predicates. We will now demonstrate this process in detail by means of the question *"what material is the red cube?"*, which is parsed into the semantic representation shown in Figure 9. An overview of the constructions that apply is shown in Figure 10, together with the part of the semantic network that has been built up after each construction application.

The first construction that applies is the CUBE-MORPH-CXN. This construction does not add

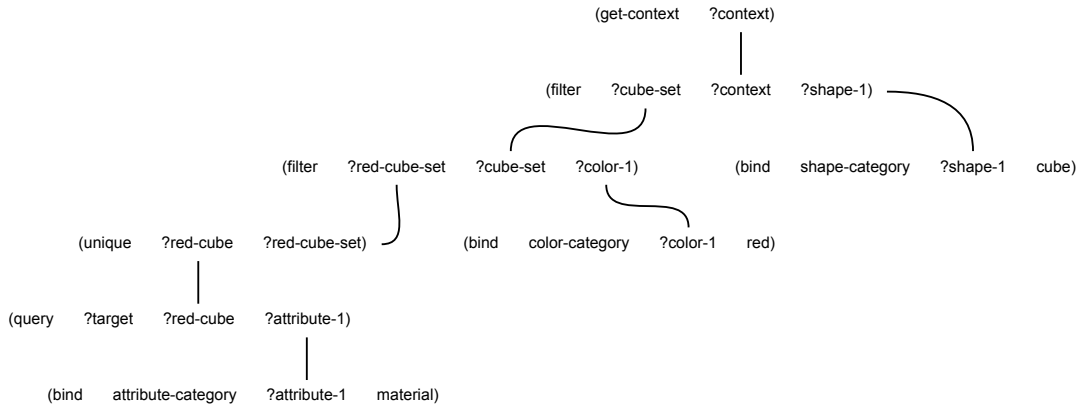


Figure 9: A complete semantic network for the question *"what material is the red cube?"*. Read from top right to bottom left.

any meaning predicates, but contributes morphological features which are later used to satisfy the preconditions of the CUBE-LEX-CXN. Then, three lexical constructions apply, which add meaning predicates that reflect the meaning of the the words "material", "red" and "cube". After this, the BASE-NOMINAL-CXN adds a `filter` predicate of which the last argument is unified with the output argument of the meaning predicate for "cube", indicating that a set of objects, which is still unbound at this moment, needs to be filtered for cubes. The NOMINAL-CXN adds a second `filter` predicate that specifies that the output of the filter-for-cube operation (i.e. a set of cubes) serves as the input of a filter operation for the color red. The UNIQUE-DETERMINED-CXN adds a predicate that checks whether the output of the filter-for-color operation yields a single object. The WHAT-T-IS-CXN adds a predicate that queries the material of an object that is for now unbound. Finally, the HOP-QUERY-PROPERTY-CXN binds the input variable of the `query` predicate to the output variable of the `unique` predicate. This construction also adds a `get-context` predicate, which binds the input of the filter-for-cube operation to the set of objects in the scene. The complete semantic network specifies thus that (i) a set of objects in the scene needs to be considered, (ii) this set needs to be filtered for cubes, (iii) the resulting set needs to be filtered for red objects, (iv) the resulting set needs to consist of a single object, and (v) the material of this object needs to be queried.

Let us have a look at two constructions that were used in the comprehension process, namely the CUBE-LEX-CXN shown in Figure 11 and the BASE-NOMINAL-CXN shown in Figure 12. The meaning predicates that are added when these constructions apply are listed under their `meaning` features. The variables in these predicates are also explicitly represented using a feature called `args`, for easier access by other constructions. The constructions clearly show how these variables are shared across units (e.g. the variable `?shape` both in the `?nominal-unit` and the `?noun-unit` of the BASE-NOMINAL-CXN), allowing them to create links in the meaning network through unification. Note that also morpho-syntactic features are necessary for finding a correct semantic representation, as reflected for example by the `number` feature in both constructions.

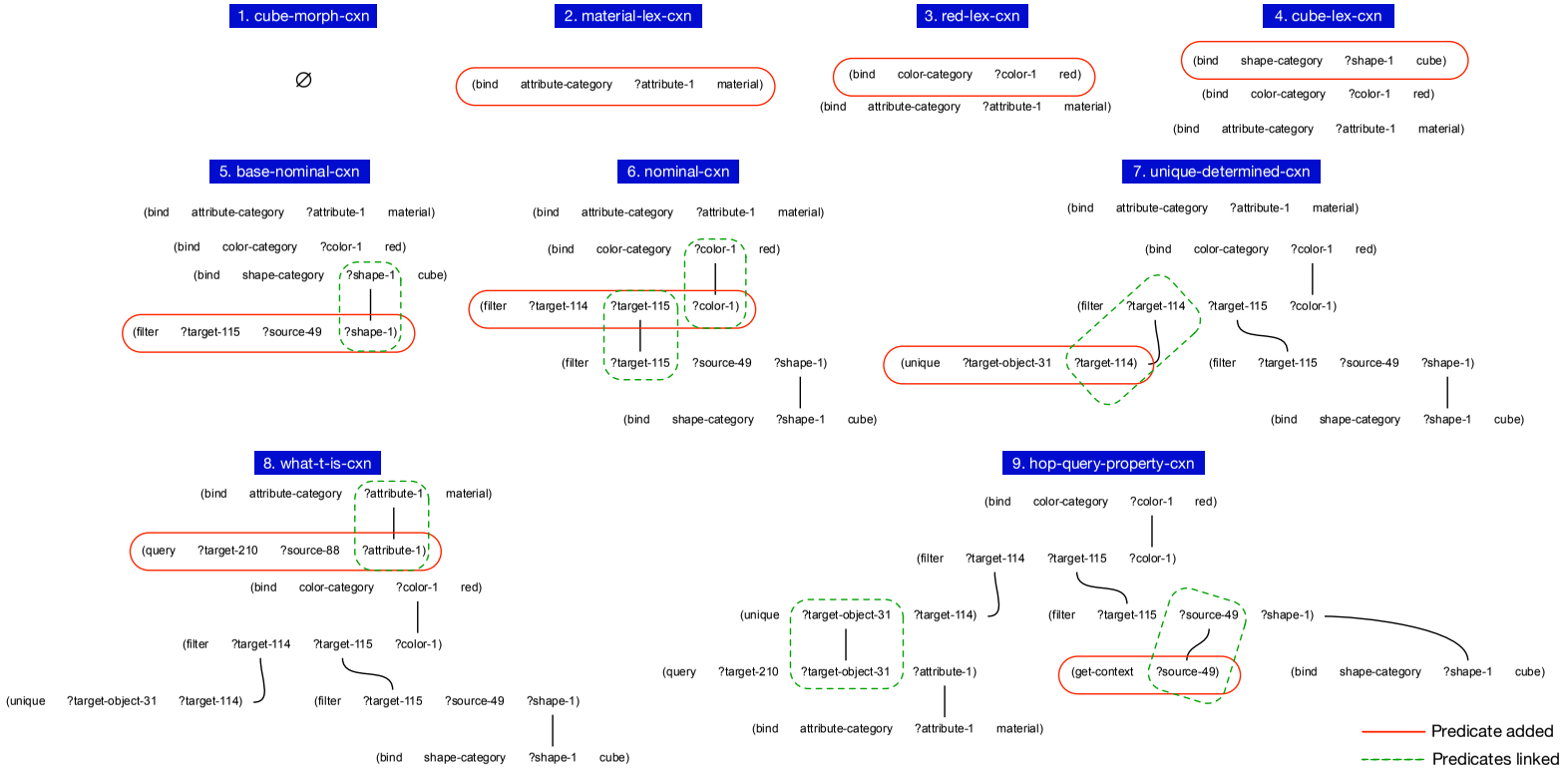


Figure 10: Overview of the constructions that apply when comprehending the question *"what material is the red cube?"*, visualising how the constructions collaboratively build up a semantic representation. For each construction application, predicates that were added are encircled using a red solid line. Arguments of these predicates that were bound through the construction application are encircled using a green dashed line.

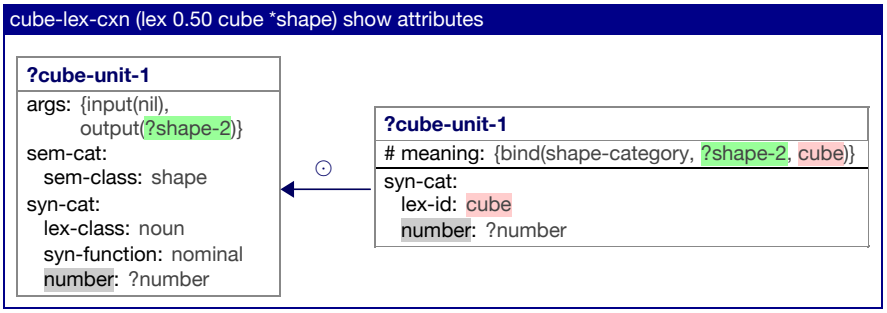


Figure 11: The `cube-lex-cxn`. Note that the variable in the `meaning` feature (`?shape-2`) is also explicitly represented using the `args` feature, for easier access by other constructions.

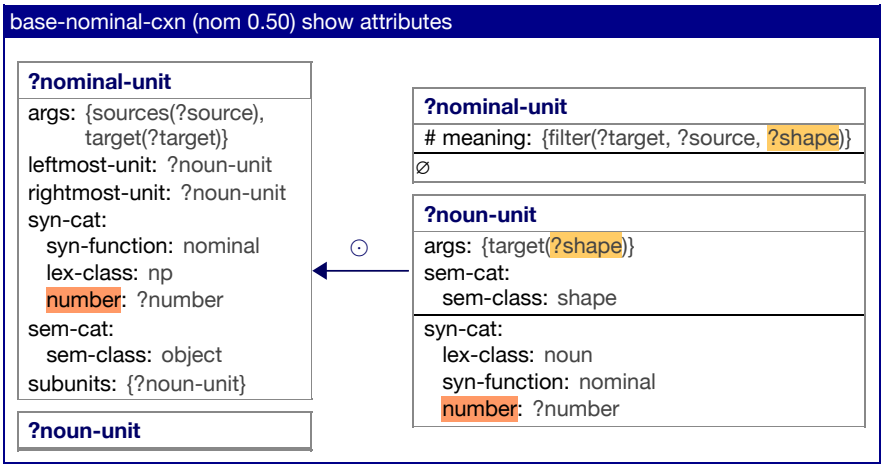


Figure 12: The `base-nominal-cxn` matches on a noun unit and creates a new nominal unit with an additional meaning predicate.

Interactive demonstrations of the comprehension process of different utterances, as well as a complete specification of all constructions in the grammar can be found at `https://ehai.ai.vub.ac.be/demos/clevr-grammar`.

5 Evaluation

The CLEVR dataset provides ground-truth annotations for more than 800,000 questions, in the form of so-called "functional programs". Very much like our semantic representation, these functional programs consist of a sequence of operations that need to be executed in order to answer the question, albeit in a different format. In order to compare these ground-truth annotations with our semantic networks, both representations were first transformed into a common format, in this case a tree data structure. Every node in this tree represents an operation with its argument(s). The root of the tree is the last function of the program that needs to be called. Each tree node thus points to the other function node(s) that provide its input. The comparison of the trees is done by traversing them simultaneously, starting from the root. At every node, both the function name and its arguments are compared. If these are equal across all nodes, we conclude that the tree data structures are equal and the analysis performed by our grammar is correct. An additional difficulty arises when a node in the tree has multiple children, as is for example the case for a 'union' or 'compare' operation. Since both trees are build up by different procedures (one by reading from the ground-truth functional program, the other by transforming a meaning network), the order of the children of a certain node is not necessarily equal in both trees. For example, the left child in one tree might be the right child in the other tree. Despite this difference, the trees are still functionally equivalent. In order to accommodate such cases, all orders of traversals are tried when a tree-node with multiple children is encountered.

When evaluated on the more than 800,000 annotated questions in the CLEVR dataset, an accuracy of 100% was achieved. This result proves that the grammar effectively solves the language understanding part of the CLEVR benchmark task.

6 Towards an operational VQA system

The grammar described in the previous sections allows parsing natural language questions into structured queries. These queries can then be executed on knowledge sources to retrieve the answers to the questions. As explained above, the queries consist of a network of function calls that can take as input the output of other calls. These functions, which form the query language, can be implemented using a wide variety of techniques. Certain functions, such as filtering images for particular objects, are for example well suited to be implemented as neural networks. Others, like comparing two integers, can more precisely be implemented as symbolic operations.

In this section, we demonstrate how the grammar can be integrated into a full visual question answering system, in which all functions are implemented as symbolic computations. It is important to note that the functions do not operate on the actual bitmap images, but on a

structured representation of their content, which is included as meta-data in the CLEVR dataset. For each object in an image, the annotation specifies the object's properties (colour, size, material and shape) and additionally states the spatial relations between the objects. These annotations can be thought of as the result of a segmentation and feature extraction process. As the system works with these annotations and not with the images themselves, it is not meant to compete on the benchmark test, but to demonstrate that the meaning representation used by the grammar is directly executable, and can thus be used in a task where answers to the questions need to be formulated.

For executing the semantic networks, we use a procedural semantics framework called Incremental Recruitment Language (IRL) (Spranger et al. 2012). IRL provides an interface for defining cognitive operations and semantic entities (as introduced by Steels (2007)). We use the former to provide an implementation for the various meaning predicates such as `count` and `filter` and the latter to represent the symbolic annotation of the input image. Furthermore, IRL provides mechanisms for executing the meaning networks through a search process. At every processing step, the system will search for the next meaning predicate it can execute. This is possible when enough variables are instantiated. The execution of a meaning predicate can then in turn instantiate new variables in the meaning network. This is repeated until all variables are bound, the output variable of the semantic network being bound to the answer to the question.

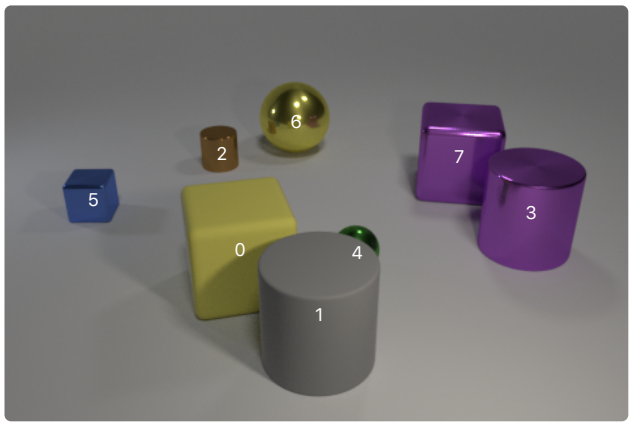
A live demonstration of this system is provided at `https://ehai.ai.vub.ac.be/demos/visual-question-answering/`. A snapshot is provided in Figure 13, for the question *"how many things are cubes or spheres?"*. The objects in the picture on the left are annotated with an id, and the semantic network on the right is executed on the meta-data describing this image. The meaning predicates that are executed are shown in circles, and their answers in boxes. We can see that the context consists of 8 objects (objects 0 to 7). Objects 4 and 5 are spheres, and objects 0, 5, 7 are cubes. The union of these two sets is taken, and the cardinality of the resulting set is 5. The yellow color indicates that this box contains the final answer to the question.

7 Bi-directional language processing

Up until this point, we have focussed on parsing natural language questions into structured queries, a language understanding task that is of great importance to (visual) question answering. Here, we shift our focus to producing questions that express such queries in natural language. Language production is typically not part of question answering tasks, but it is easy to see its relevance for interactive intelligent systems such as chatbots or personal assistants.

As discussed in Section 4.1 above, the FCG system can use the same grammar and processing mechanisms for comprehending and producing utterances. The only difference is that a construction now matches its production preconditions against the transient structure to decide whether it is applicable (instead of its comprehension preconditions), and if so, that its comprehension preconditions are added to the transient structure (instead of its production preconditions).

In production, the grammar has the task of adding word forms and word order constraints to the transient structure. Word forms are added in the form of `string` predicates, while word order



How many things are cubes or spheres?

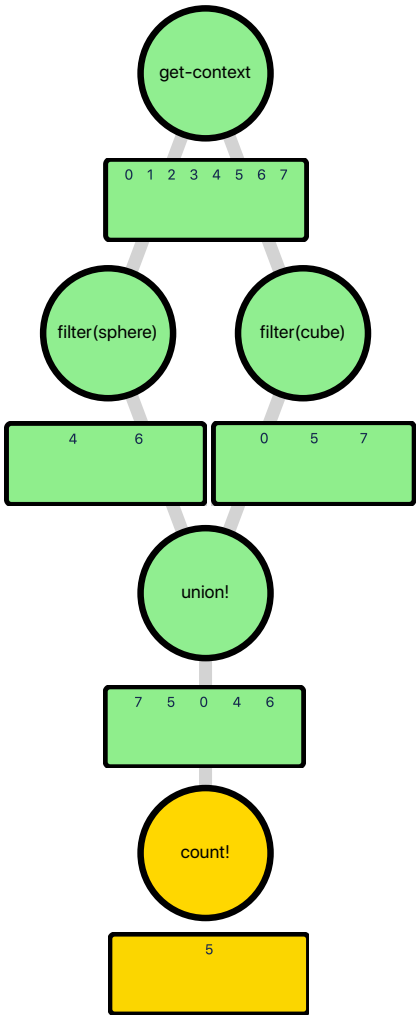


Figure 13: Integrating the grammar into a full visual question answering system. The meaning predicates that are executed are shown in circles, the intermediate answers in boxes. The last step and final answer are shown in yellow.

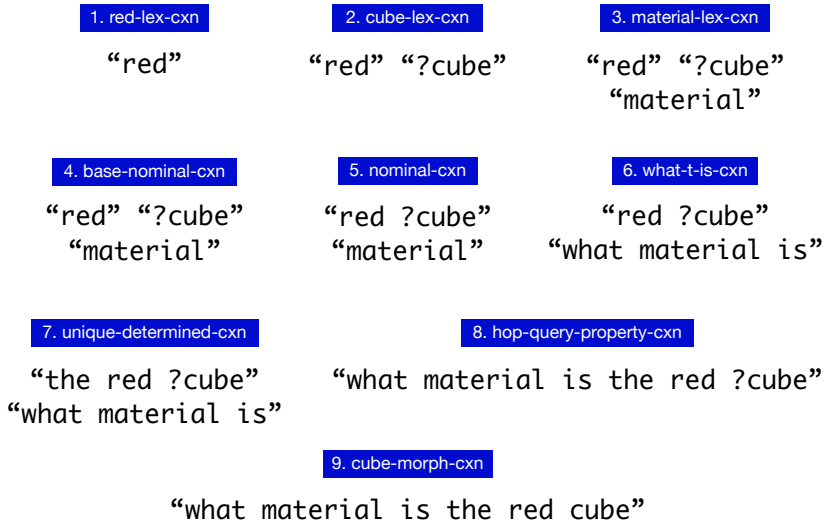


Figure 14: Overview of the constructions that apply when formulating the question *“what material is the red cube?”*, and the part of the utterance that has been built up after each construction application. Quotation marks group parts of the utterance that are subject to word order constraints. Variables hold the place of words of which the concrete morphological realisation is only decided on later in the construction application process.

constraints are added in the form of `meets` and `precedes` predicates, indicating adjacency and precedence relations respectively. At the end of processing, the output utterance is rendered based on these strings and the word order constraints between them. Figure 14 illustrates the production process of the question *“what material is the red cube?”*, based on the meaning representation that was shown in Figure 9. The blue boxes show the constructions that have applied, together with the part of the utterance that has been constructed after each construction application. If word order constraints have been imposed on two or more elements, they are written within a single pair of quotation marks. The first three (lexical) constructions add word forms to the transient structure. Note that `?cube` is still a variable at this point, as its final morphological form (“cube” or “cubes”) is only decided on later on in the construction application process, based on number constraints added by other constructions. The NOMINAL-CXN imposes a constraint that “red” and “?cube” need to be adjacent to each other. The WHAT-T-IS-CXN adds the word forms “what” and “is” and specifies the word order of “what material is”. The UNIQUE-DETERMINED-CXN adds the word form “the” and imposes the word order of “the red ?cube”. It is also this construction that imposes a constraint that `?cube` must be singular. The HOP-QUERY-PROPERTY-CXN constrains that “what material is” immediately precedes “the red ?cube”. Finally, the CUBE-MORPH-CXN instantiates the `?cube` variable with the singular word form “cube”.

The grammar includes a considerable amount of lexical and syntactic variation, as the same meaning network can be expressed in many different ways. The lexical variation is due to (exact)

Utterances:

- **"what is the material of the red block"**
- **"what material is the red cube"**
- **"what material is the red cube made of"**
- **"the red block is made of what material"**
- **"there is a red cube ; what is its material"**
- **"there is a red block ; what material is it"**

Figure 15: Syntactic variation leads to multiple possible utterances when formulating the meaning network in Figure 9.

synonyms in the dataset, such as "cube"-"block" and "shiny"-"metal". The grammar handles this by including these word forms as morphological instantiations of the same lexical items. The grammatical variation in the dataset is more interesting, and is handled by having different combinations of constructions of which the application leads to the the same meaning network. For example, when asking for all syntactically different solutions, the meaning network in Figure 9 is mapped by the grammar to the utterances in Figure 15. When comprehending a question and using the resulting meaning network for producing more questions, the grammar and FCG system can thus readily be used as a paraphrasing tool.

8 Discussion and conclusion

This paper has introduced a novel methodology for semantically parsing natural language questions into structured queries. The methodology adopts a computational construction grammar approach, in which the query language is used as the meaning representation of the grammar. This way, the meaning representation can be directly executed on knowledge sources, without requiring any intermediate transformation steps. We have demonstrated the approach in the domain of visual question answering using the CLEVR benchmark task, on which the grammar yielded a perfect accuracy score. Additionally, we have shown how the grammar can be used as a component in a full visual question answering system, and how the same grammar that is used for comprehending questions can also be used for producing questions expressing structured queries.

The main contribution of our methodology to the state of the art in (visual) question answering, in particular with respect to the CLEVR benchmark task, is that it combines the main strengths of grammar-based approaches with a level of accuracy that was previously only achieved using deep learning techniques (Mascharka et al. 2018, Yi et al. 2018). One of these strengths is that no annotated training data is required, as expert linguistic knowledge is captured in the grammar. A second strength is that the analyses are transparent and interpretable, as the linguistic motivation of each step in the semantic parsing process can be inspected. A final strength of the methodology is its open-endedness, in the sense that the coverage of a grammar can be extended with new constructions, e.g. covering new interrogative structures, without having to retrain or rebuild the existing model. An additional advantage is that the same model cannot only be used for

comprehending questions, but also for producing questions based on their meaning representation.

Compared to other grammar-based approaches, one of the main advantages of computational construction grammar is that it provides an elegant and effective way to deal with the non-compositional nature of linguistic expressions, both when it comes to their form and to their semantics. This results from the operationalisation of basic construction grammar insights, such as the lexicon-grammar continuum, and the tight integration of morpho-syntax with semantics. For instance, a HOW-MANY-X-ARE-Y construction can capture at once all relevant linguistic information that is associated with utterances instantiating the construction. This includes, among other aspects, the fact that its meaning (filtering x by y and counting the result) is only parametrised on the specific values of x and y, that x needs to be a bare noun phrase in the plural form, that y needs to be a predicative expression that agrees with x, and that the word 'many' cannot be replaced by a different determiner or adjective (such as 'few' or 'numerous'). Apart from making grammar design easier, this way of dealing with non-compositionality in language also facilitates the use of an actual query language as the meaning representation of the grammar, which avoids the highly non-trivial step to transform more abstract meaning representations into executable queries.

In sum, we have introduced computational construction grammar as a methodology for tackling challenges that involve mapping between natural language expressions and their executable meaning representations, and have demonstrated its potential for the first time on a community-wide benchmark task.

References

Abou-Assaleh, Tony, Nick Cercone & Vlado Keselj. 2005. Question-answering with relaxed unification. In *Proceedings of the pacling*, vol. 5.

Agrawal, Aishwarya, Dhruv Batra & Devi Parikh. 2016. Analyzing the behavior of visual question answering models. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, 1955–1960.

Andreas, Jacob, Marcus Rohrbach, Trevor Darrell & Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, 1545–1554.

Andreas, Jacob, Marcus Rohrbach, Trevor Darrell & Dan Klein. 2016b. Neural module networks. In *Proceedings of the 2016 ieee conference on computer vision and pattern recognition*, 39–48.

Frank, Anette, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg & Ulrich Schäfer. 2007. Question answering from structured knowledge sources. *Journal of Applied Logic* 5(1). 20–48.

Hoffmann, Thomas & Graeme Trousdale. 2013. Construction Grammar: Introduction. In *The Oxford Handbook of Construction Grammar*. Oxford: Oxford University Press.

Hu, Ronghang, Jacob Andreas, Marcus Rohrbach, Trevor Darrell & Kate Saenko. 2017. Learning to reason: end-to-end module networks for visual question answering. In *Proceedings of the 2017 ieee international conference on computer vision*, 804–813.

Johnson, Justin, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick & Ross Girshick. 2017a. Inferring and executing programs for visual reasoning. In *Proceedings of the 2017 ieee international conference on computer vision*, 3008–3017.

Johnson, Justin, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick & Ross Girshick. 2017b. Clevr: a diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the 2017 ieee conference on computer vision and pattern recognition*, 1988–1997.

Li, Peng & Lejian Liao. 2012. Web question answering based on ccg parsing and dl ontology. In *2012 8th international conference on information science and digital content technology (icidt2012)*, vol. 1, 212–217.

Malinowski, Mateusz, Marcus Rohrbach & Mario Fritz. 2015. Ask your neurons: a neural-based approach to answering questions about images. In *Proceedings of the 2015 ieee international conference on computer vision*, 1–9.

Mao, Jiayuan, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum & Jiajun Wu. 2019. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International conference on learning representations*. `https://openreview.net/forum?id=rJgMlhRctm`.

Mascharka, David, Philip Tran, Ryan Soklaski & Arjun Majumdar. 2018. Transparency by design: closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the 2018 ieee conference on computer vision and pattern recognition*, 4942–4950.

McFetridge, Paul, Fred Popowich & Dan Fass. 1996. An analysis of compounds in hpsg (head-driven phrase structure grammar) for database queries. *Data & Knowledge Engineering* 20(2). 195–209.

Noh, Hyeonwoo, Paul Hongsuck Seo & Bohyung Han. 2016. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the 2016 ieee conference on computer vision and pattern recognition*, 30–38.

Ren, Mengye, Ryan Kiros & Richard S Zemel. 2015. Exploring models and data for image question answering. In *Proceedings of the 28th international conference on neural information processing systems-volume 2*, 2953–2961.

Shamsfard, Mehrnoush & Mahsa Arab Yarmohammadi. 2010. A semantic approach to extract the final answer in sbuqa question answering system. *International Journal of Digital Content Technology and its Applications* 4(7).

Spranger, Michael, Simon Pauw, Martin Loetzsch & Luc Steels. 2012. Open-ended procedural semantics. In Luc Steeks & Manfred Hild (eds.), *Language grounding in robots*, 159–178. New York: Springer.

Steels, Luc. 2007. The recruitment theory of language origins. In C. Lyon, C. L. Nehaniv & A. Cangelosi (eds.), *Emergence of language and communication*, 129–151. Berlin: Springer.

Steels, Luc (ed.). 2011. *Design patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, Luc. 2017. Basics of fluid construction grammar. *Constructions and frames* 9(2). 178–225.

Xu, Huijuan & Kate Saenko. 2016. Ask, attend and answer: exploring question-guided spatial attention for visual question answering. In *European conference on computer vision*, 451–466.

Yang, Zichao, Xiaodong He, Jianfeng Gao, Li Deng & Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the 2016 ieee conference on computer vision and pattern recognition*, 21–29.

Yarmohammadi, Mahsa A, Mehrnoush Shamsfard, Mahshid A Yarmohammadi & Masoud Rouhizadeh. 2008. Sbuqa question answering system. In *Computer society of iran computer conference*, 316–323.

Yi, Kexin, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli & Josh Tenenbaum. 2018. Neural-symbolic vqa: disentangling reasoning from vision and language understanding. In *Advances in neural information processing systems*, 1031–1042.

Zettlemoyer, Luke S & Michael Collins. 2012. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

Zhang, Peng, Yash Goyal, Douglas Summers-Stay, Dhruv Batra & Devi Parikh. 2016. Yin and yang: balancing and answering binary visual questions. In *Proceedings of the 2016 ieee conference on computer vision and pattern recognition*, 5014–5022.

Zhou, Bolei, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam & Rob Fergus. 2015. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*.